UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

LUCIANO SILVA DA SILVA

# Segmentação de Movimento Coerente Aplicada à Codificação de Vídeos Baseada em Objetos

Tese apresentada como requisito parcial
para a obtenção do grau de
Doutor em Ciência da Computação

Prof. Dr. Jacob Scharcanski
Orientador

Porto Alegre, outubro de 2011

**CIP – CATALOGAÇÃO NA PUBLICAÇÃO**

Silva, Luciano Silva da

Segmentação de Movimento Coerente Aplicada à Codificação de Vídeos Baseada em Objetos / Luciano Silva da Silva. – Porto Alegre: PPGC da UFRGS, 2011.

113 f.: il.

Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2011. Orientador: Jacob Scharcanski.

1. Segmentação de vídeos. 2. Segmentação de movimento. 3. Codificação de vídeos. 4. Codificação baseada em objetos. 5. Codificação progressiva. I. Scharcanski, Jacob. II. Título.

# AGRADECIMENTOS

Ao meu orientador, prof. Jacob Scharcanski, pelos ensinamentos, críticas e experiências transmitidas, e especialmente pelo comprometimento em sempre buscar a melhor qualidade possível em seus trabalhos.

Aos meus colegas de laboratório, pelo companheirismo, apoio e discussões que foram extremamente importantes ao longo de todo o doutorado.

A John, Paul, George e Ringo pela inspiração e companhia onipresente, ajudando a manter meu entusiasmo e minha saúde mental.

Ao CNPq, pelo apoio financeiro, à UFRGS, por disponibilizar a estrutura necessária ao desenvolvimento deste trabalho, e à FURG, pelo apoio institucional.

À minha família, em especial aos meus pais, Eros e Marlene, e à minha irmã, Tatiana, pelos inúmeros ensinamentos e apoio incondicional em todos os aspectos, sem os quais eu jamais teria chegado à conclusão deste trabalho.

# SUMÁRIO

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| CAE | Context-based Arithmetic Encoding |
| CIF | Common Intermediate Format |
| DAG | Directed Acyclic Graph |
| DCT | Discrete Cosine Transform |
| DoG | Difference-of-Gaussian |
| DWT | Discrete Wavelet Transform |
| EM | Expectation-Maximization |
| EZW | Embedded Zerotree Wavelet |
| GOB | Group of B-frames |
| GOI | Group of I-frames |
| GOP | Group of Pictures |
| GPCA | Generalized Principal Component Analysis |
| KLT | Kanade-Lucas-Tomasi |
| LIP | List of Insignificant Pixels |
| LIS | List of Insignificant Sets |
| LSA | Local Subspace Affinity |
| LSP | List of Significant Pixels |
| MRF | Markov Random Fields |
| MSL | Multi-Stage Learning |
| PMHT | Probabilistic Multiple Hypothesis Tracking |
| PSNR | Peak Signal-to-Noise Ratio |
| RLE | Run Length Encoding |
| SA-DCT | Shape-Adaptive Discrete Cosine Transform |
| SA-DWT | Shape-Adaptive Discrete Wavelet Transform |
| SA-EZW | Shape-Adaptive Embedded Zerotree Wavelet |
| SCS | Shape Code Set |

| | |
|---|---|
| SIFT | Scale-Invariant Feature Transform |
| SPIHT | Set Partitioning in Hierarchical Trees |
| ST-SPIHT | Shape and Texture Set Partitioning in Hierarchical Trees |
| VOP | Video Object Plane |
| WWW | World Wide Web |

# LISTA DE FIGURAS

# LISTA DE TABELAS

# RESUMO

A variedade de dispositivos eletrônicos capazes de gravar e reproduzir vídeos digitais vem crescendo rapidamente, aumentando com isso a disponibilidade deste tipo de informação nas mais diferentes plataformas. Com isso, se torna cada vez mais importante o desenvolvimento de formas eficientes de armazenamento, transmissão, e acesso a estes dados. Nesse contexto, a codificação de vídeos tem um papel fundamental ao compactar informação, otimizando o uso de recursos aplicados no armazenamento e na transmissão de vídeos digitais. Não obstante, tarefas que envolvem a análise de vídeos, manipulação e busca baseada em conteúdo também se tornam cada vez mais relevantes, formando uma base para diversas aplicações que exploram a riqueza da informação contida em vídeos digitais. Muitas vezes a solução destes problemas passa pela segmentação de vídeos, que consiste da divisão de um vídeo em regiões que apresentam homogeneidade segundo determinadas características, como por exemplo cor, textura, movimento ou algum aspecto semântico. Nesta tese é proposto um novo método para segmentação de vídeos em objetos constituintes com base na coerência de movimento de regiões. O método de segmentação proposto inicialmente identifica as correspondências entre pontos esparsamente amostrados ao longo de diferentes quadros do vídeo. Logo após, agrupa conjuntos de pontos que apresentam trajetórias semelhantes. Finalmente, uma classificação pixel a pixel é obtida a partir destes grupos de pontos amostrados. O método proposto não assume nenhum modelo de câmera ou de movimento global para a cena e/ou objetos, e possibilita que múltiplos objetos sejam identificados, sem que o número de objetos seja conhecido *a priori*. Para validar o método de segmentação proposto, foi desenvolvida uma abordagem para a codificação de vídeos baseada em objetos. Segundo esta abordagem, o movimento de um objeto é representado através de transformações afins, enquanto a textura e a forma dos objetos são codificadas simultaneamente, de modo progressivo. O método de codificação de vídeos desenvolvido fornece funcionalidades tais como a transmissão progressiva e a escalabilidade a nível de objeto. Resultados experimentais dos métodos de segmentação e codificação de vídeos desenvolvidos são apresentados, e comparados a outros métodos da literatura. Vídeos codificados segundo o método proposto são comparados em termos de PSNR a vídeos codificados pelo *software* de referência JM H.264/AVC, versão 16.0, mostrando a que distância o método proposto está do estado da arte em termos de eficiência de codificação, ao mesmo tempo que provê funcionalidades da codificação baseada em objetos. O método de segmentação proposto no presente trabalho resultou em duas publicações, uma nos anais do SIBGRAPI de 2007 e outra no períodico IEEE Transactions on Image Processing.

**Palavras-chave:** Segmentação de vídeos, segmentação de movimento, codificação de vídeos, codificação baseada em objetos, codificação progressiva.

**Coherent Motion Segmentation Applied to Object-Based Video Coding**

# ABSTRACT

The variety of electronic devices for digital video recording and playback is growing rapidly, thus increasing the availability of such information in many different platforms. So, the development of efficient ways of storing, transmitting and accessing such data becomes increasingly important. In this context, video coding plays a key role in compressing data, optimizing resource usage for storing and transmitting digital video. Nevertheless, tasks involving video analysis, manipulation and content-based search also become increasingly relevant, forming a basis for several applications that exploit the abundance of information in digital video. Often the solution to these problems makes use of video segmentation, which consists of dividing a video into homogeneous regions according to certain characteristics such as color, texture, motion or some semantic aspect. In this thesis, a new method for segmentation of videos in their constituent objects based on motion coherence of regions is proposed. The proposed segmentation method initially identifies the correspondences of sparsely sampled points along different video frames. Then, it performs clustering of point sets that have similar trajectories. Finally, a pixel-wise classification is obtained from these sampled point sets. The proposed method does not assume any camera model or global motion model to the scene and/or objects. Still, it allows the identification of multiple objects, without knowing the number of objects a priori. In order to validate the proposed segmentation method, an object-based video coding approach was developed. According to this approach, the motion of an object is represented by affine transformations, while object texture and shape are simultaneously coded, in a progressive way. The developed video coding method yields functionalities such as progressive transmission and object scalability. Experimental results obtained by the proposed segmentation and coding methods are presented, and compared to other methods from the literature. Videos coded by the proposed method are compared in terms of PSNR to videos coded by the reference software JM H.264/AVC, version 16.0, showing the distance of the proposed method from the sate of the art in terms of coding efficiency, while providing functionalities of object-based video coding. The segmentation method proposed in this work resulted in two publications, one in the proceedings of SIBGRAPI 2007 and another in the journal IEEE Transactions on Image Processing.

**Keywords:** video segmentation, motion segmentation, video coding, object-based coding, progressive coding.

# 1 INTRODUÇÃO

Nos dias atuais, a variedade de dispositivos eletrônicos para a gravação e reprodução de vídeos digitais tem crescido rapidamente. É possível gerar vídeos digitais a partir de dispositivos que vão desde pequenos aparelhos celulares até câmeras de alta-resolução usadas na indústria cinematográfica. Com isso, a disponibilidade de vídeos digitais em diferentes plataformas também vem aumentando constantemente. Além do mais, o rápido desenvolvimento da *internet* e de aplicações multimídia nos permitem acessar grandes quantidades de dados de imagem e vídeos. Este panorama sugere a importância do desenvolvimento de mecanismos eficientes de codificação e transmissão de vídeos.

O grande desafio da codificação de vídeos é reduzir a quantidade de bits necessária para a representação do vídeo, através de mecanismos de compressão, e ao mesmo tempo fornecer facilidades de navegação (como acesso aleatório, sumários, busca baseada em conteúdo, etc), para os mais diversos meios de transmissão e reprodução de vídeos.

Os métodos de codificação de vídeo existentes na literatura, de acordo com a forma com que os dados são organizados, podem ser classificados em dois grupos: os métodos de codificação baseados em blocos, e os métodos de codificação de formas arbitrárias.

Os métodos baseados em blocos dividem quadros de um vídeo em blocos retangulares de tamanhos pré-estabelecidos. O tamanho dos blocos pode ser único ao longo do vídeo ou adaptativo ao conteúdo, e são utilizados para predição de movimento. Os métodos baseados em blocos (ISO/IEC, 2000, 2003) têm recebido grande atenção da comunidade científica e especialmente da indústria, porque são mais adequados para vídeos de propósito geral, e mais fáceis de implementar em *hardware*. Porém, para algumas aplicações, como codificação com altas taxas de compressão, métodos baseados em blocos apresentam desvantagens. A principal desvantagem é que, como a cena e o movimento são constituídos de blocos retangulares, com altas taxas de compressão surgem os chamados **artefatos de blocos**, que são bastante incômodos para um observador humano.

A busca por métodos de codificação de vídeos com altas taxas de compressão que minimizassem os efeitos dos artefatos foi o que motivou inicialmente a pesquisa em codificação de formas arbitrárias. Estas técnicas, inicialmente chamadas de **técnicas de segunda geração** (KUNT; IKONOMOPOULOS; KOCHER, 1985), visavam a composição de modelos mais realísticos com base no conteúdo da cena. Tais modelos permitiram mais tarde a possibilidade de se prover uma melhor interpretação do conteúdo de imagens e vídeos, resultando em outras vantagens, tais como recuperação de conteúdo, escalabilidade e manipulação baseada em conteúdo. Uma alternativa aos métodos baseados em blocos é a codificação de vídeo baseada em objetos, proposta pela primeira vez em (MUSMANN; HOTTER; OSTERMANN, 1990). Nas abordagens baseadas em objetos, os objetos em movimento de uma cena são extraídos, e cada objeto é representado por sua forma, movimento e textura. Parâmetros representando estes três componentes são

codificados e transmitidos, e a reconstrução é executada sintetizando-se cada objeto. A crescente demanda por aplicações que apresentassem todas as vantagens descritas acima levou ao desenvolvimento do padrão MPEG-4, o primeiro padrão de codificação de vídeos baseado em objetos (ISO/IEC, 2001), que tem por objetivo conciliar funcionalidades baseadas em conteúdo com grande eficiência na codificação.

Embora a codificação de vídeos baseada em objetos tenha sido bastante estudada nos últimos anos, alguns obstáculos ainda impedem que estes métodos sejam amplamente utilizados na prática, sendo preteridos por métodos baseados em blocos. Um dos motivos para isso é o fato de que a segmentação de objetos em movimento ainda é um problema mal resolvido. Grande parte dos métodos de segmentação de movimento bem sucedidos restringem sua aplicação a casos bastante específicos (como, por exemplo, modelos de câmeras específicos, classes de objetos restritos, modelos rígidos de movimento, número de objetos da cena, cenas sem oclusão, nível baixo de ruído, ausência de *outliers*, etc). Além do mais, a definição de objeto muitas vezes requer uma ligação a aspectos de alto-nível, que é um problema que ainda está longe de ser resolvido pela comunidade científica, o que resulta em segmentações ruins e, consequentemente, em uma codificação ineficiente.

## 1.1 Contribuições

No presente trabalho, é proposto um novo método para a segmentação de vídeos, bem como uma aplicação deste método na codificação de vídeos baseada em objetos. Segundo o método proposto, objetos são definidos como regiões não-sobrepostas (em nível de pixel) no domínio espaço-temporal. Assim, cada pixel de cada quadro do vídeo é classificado como pertencente a um e apenas um objeto. Considera-se que estas regiões preservam suas características espaciais e fotométricas ao longo do tempo. Como aspectos que diferenciam o método de segmentação proposto em relação à literatura (contribuições), podemos citar:

- O método proposto não impõe nenhuma restrição de movimento global à cena ou aos objetos. Diferente de vários métodos de segmentação encontrados na literatura, o método proposto não diferencia explicitamente *background* de *foreground*. Ao invés disso, o *background* é considerado um objeto qualquer, desde que preserve suas características espaciais e fotométricas ao longo da sequência. E aos objetos não são impostas restrições quanto ao modelo de movimento, bem como não são impostas restrições a modelos de câmera específicos[1].

- A oclusão de objetos é tratada de forma mais geral do que em outros trabalhos na literatura. Pelo método proposto, um mesmo objeto pode sofrer oclusão e ocultar outros objetos ao mesmo tempo, sem que seja necessário um tratamento especial para estes casos. Apesar de tal generalidade, o método proposto potencialmente tem a capacidade de gerar **volumes de oclusão** (regiões que se tornam oclusas) e **volumes de exposição** (regiões que se tornam visíveis), que são novos conceitos propostos por Ristivojevic e Konrad (RISTIVOJEVIC; KONRAD, 2006).

- O método proposto combina vantagens dos métodos baseados em feições e dos métodos diretos (ver Capítulo 2), ao identificar padrões de longa duração ao mesmo

[1]Conforme será visto no capítulo 5, a eficiência do algoritmo de codificação proposto será maior se o movimento dos objetos puderem ser aproximados por transformações afins planares. No entanto, esta não é uma restrição para a segmentação do vídeo.

tempo em que evita a influência de *outliers* e o problema da abertura[2].

- O método proposto identifica padrões de movimento de longa duração combinando informações sobre a segmentação de objetos de diversas partes de um vídeo. Gelgon *et al.* (GELGON; BOUTHEMY; LE CADRE, 2005) também utilizaram uma estratégia semelhante; porém, assumiram a continuidade e a suavidade das trajetórias dos objetos. Já o método proposto permite identificar padrões de movimento temporalmente descontínuos. Como não é imposta nenhuma restrição com relação à continuidade temporal do movimento, o método pode ser utilizado em sequências com movimentos abruptos de câmera, como as obtidas a partir de câmeras de mão.

O método de segmentação proposto tem características que são vantajosas especialmente na codificação de vídeos, embora seja genérico o suficiente para ser utilizado em outras aplicações, tais como rastreamento de objetos, recuperação de informações e análise de vídeos. Uma versão preliminar do método de segmentação de movimento proposto aqui está descrita em (SILVA; SCHARCANSKI, 2007), enquanto uma versão mais completa foi publicada em (SILVA; SCHARCANSKI, 2010).

O método de codificação proposto utiliza dois tipos de quadros, de acordo com a forma com que os dados são codificados: quadros I (*intra-coded*) e quadros B (*bidirectional predicted*). Os quadros B são preditos pela compensação de movimento, descrita por transformações afins, a partir dos dois quadros I adjacentes. Podemos citar as seguintes características que diferenciam o método de codificação proposto dos demais métodos da literatura:

- Segundo o método de codificação proposto, os parâmetros das transformações afins utilizadas na compensação de movimento são obtidos a partir das trajetórias de pontos amostrados que são rastreados durante o processo de segmentação do vídeo. Como estes pontos são rastreados a partir de um método robusto e, após a etapa de segmentação, já estão classificados de acordo com os objetos da cena, as transformações afins obtidas são mais confiáveis. Além do mais, para cada quadro B, duas predições são calculadas e interpoladas (a partir dos dois quadros I adjacentes), aumentando a precisão da predição.

- Ao contrário dos métodos tradicionais, que codificam os dados originais de cada quadro separadamente, segundo o método proposto os objetos são codificados como volumes no domínio espaço-temporal, através de uma abordagem progressiva. Isso significa que cada *bit* recebido pelo decodificador melhora a qualidade global do vídeo (ou de um grupo de quadros), ao invés de um único quadro, resultando em uma abordagem verdadeiramente progressiva, com controle eficiente de taxa de transmissão. Embora alguns outros métodos também codifiquem os objetos através de volumes no domínio espaço-temporal, no método proposto a forma e a textura dos objetos são codificados de forma simultânea e progressiva.

- A abordagem de codificação simultânea da forma e textura dos objetos é feita no domínio espaço-temporal através de uma extensão 3-D do método ST-SPIHT, proposto por Martin *et al.* (MARTIN; LUKAC; PLATANIOTIS, 2006), originalmente

[2]O problema da abertura (*aperture problem*) surge da ambiguidade do movimento uni-dimensional visto por uma pequena abertura. Dessa forma, só é possível detectar localmente o movimento na direção perpendicular ao movimento do contorno do objeto, e não é possível detectar o movimento em regiões homogêneas.

empregado para a codificação de objetos 2-D. Além do mais, o método ST-SPIHT original trata apenas da codificação de um objeto individual; assim, ao se codificar diversos objetos não sobrepostos de um mesmo vídeo, haverá redundância na representação das formas destes objetos. No presente trabalho, ao se codificar diversos objetos não sobrepostos, a inicialização do algoritmo ST-SPIHT é feita de tal forma que esta redundância seja explorada, resultando em uma maior economia de bits.

Tais características tornam o método proposto adequado para aplicações tais como a transmissão de vídeos através da *World Wide Web*: primeiro, *storyboards* podem ser criados a partir da decodificação parcial de quadros I; segundo, tão logo os parâmetros de movimento e a forma dos objetos são decodificados, o vídeo já pode ser visualizado; terceiro, a escalabilidade de qualidade pode ser conseguida no nível dos objetos; quarto, tarefas baseadas em conteúdo podem ser realizadas de forma automática ou semi-automática.

O restante desta tese está organizado em 6 partes. O Capítulo 2 apresenta alguns conceitos fundamentais e uma revisão de trabalhos que constituem o estado da arte em segmentação de vídeos. A codificação de vídeos baseada em objetos, através de seus conceitos e estado da arte, é apresentada no Capítulo 3. No Capítulo 4 é proposta uma nova metodologia para a segmentação de vídeos em regiões de movimento coerente. Utilizando como base a segmentação de vídeos proposta no Capítulo 4, um método de codificação de vídeos baseado em objetos é proposto no Capítulo 5. Os experimentos realizados, assim como uma análise dos resultados obtidos, são apresentados no Capítulo 6. O Capítulo 7, por fim, apresenta as conclusões e um resumo das principais contribuições do presente trabalho.

# 2 SEGMENTAÇÃO DE VÍDEOS: CONCEITOS E ESTADO DA ARTE

A segmentação de vídeos baseada em movimento é uma tarefa de pré-processamento muito importante em diversas aplicações da visão computacional e do processamento de vídeos, tais como sistemas de vigilância, rastreamento de objetos, codificação de vídeos, recuperação de informações, análise de vídeos, etc. Inicialmente, estas aplicações motivaram o desenvolvimento de diversas técnicas de segmentação de movimento 2-D. E aqui, movimento 2-D significa o movimento de objetos em uma cena 3-D projetados no plano de imagem de uma câmera. Tais técnicas têm por objetivo separar cada quadro de um vídeo em diferentes regiões de movimento coerente.

No entanto, a segmentação de movimento 2-D frequentemente ocasiona sobre-segmentação dos vídeos. Por exemplo, um vídeo de uma cena composta por objetos rígidos e estáticos vista por uma câmera em movimento pode ser segmentado em vários movimentos 2-D, devido a descontinuidades de profundidade, oclusões, efeitos da projeção perspectiva, etc. Em diversas aplicações, a cena pode conter diversos objetos em movimento e é necessário identificar cada objeto como uma entidade coerente. Nesses casos, é necessário que a segmentação seja executada assumindo diversos movimentos em um espaço 3-D (largura $x$, altura $y$ e profundidade $z$), e não simplesmente em 2-D. Isso motivou o desenvolvimento de diversos trabalhos de segmentação de movimento 3-D.

A segmentação espaço-temporal é uma abordagem distinta (MITICHE; EL-FEGHALI; MANSOUI, 2003; FEGHALI; MITICHE, 2004; CREMERS; SOATTO, 2003; RISTIVOJEVIC; KONRAD, 2006), onde diferentes objetos em movimento são segmentados em volumes (chamados de túneis) no domínio formado pelas dimensões espaciais (largura $x$ e altura $y$) e a dimensão temporal ($t$), e estes volumes são limitados por bordas de movimento de objetos (ou seja, descontinuidades de movimento).

A definição de objeto em um método de segmentação de vídeos está relacionada ao conceito de homogeneidade, e diferentes aplicações requerem diferentes critérios de homogeneidade. Na codificação de vídeos, por exemplo, a segmentação é frequentemente utilizada para explorar a redundância dos dados ao longo do tempo (TORRES; KUNT; PEREIRA, 1996). Neste contexto, uma região de um objeto que retêm suas características (por exemplo, cor ou textura) ao longo da sequência de quadros pode ser considerada homogêna e redundante. Assim, mesmo que um objeto se mova ao longo da sequência temporal, a representação da região permanece a mesma, ou seja, redundante, dentro das bordas de movimento do objeto.

Na segmentação de movimento 3-D, o conceito de objeto está relacionado aos objetos reais existentes no espaço 3-D que não modificam suas características 3-D ao longo do tempo. O conceito de objeto na segmentação espaço-temporal é diferente, visto que um

objeto é representado por seu túnel espaço-temporal formado por uma sequência de projeções 2-D, sendo cada projeção 2-D obtida em um tempo $t$ de um objeto no espaço 3-D. Dois conjuntos de parâmetros são geralmente utilizados para descrever o movimento paramétrico 3-D: o conjunto de parâmetros globais representando o movimento da câmera e/ou do objeto, e o conjunto de parâmetros locais representando os atributos dos objetos (como por exemplo, forma, cor e textura). No entanto, a estimativa de um grande número de parâmetros é geralmente inábil, particularmente na presença de ruído e *outliers*. Um *outlier* pode ser, por exemplo, a trajetória de um ponto computada incorretamente. Por outro lado, quando a translação de câmera e as variações de profundidade são pequenas se comparadas à distância da câmera aos objetos da cena, modelos de movimento 2-D mais simples se tornam mais atrativos (IRANI; ANANDAN, 2000). Na segmentação paramétrica de movimento 2-D, um pequeno número de parâmetros é necessário para descrever o movimento dos objetos, tornando a segmentação de movimento mais robusta ao ruído. Embora a comunidade da visão computacional venha consistentemente trabalhando para aperfeiçoar a segmentação de movimento 3-D (TRON; VIDAL, 2007; SEKKATI; MITICHE, 2006a; LUBLINERMAN; CAMPS; SZNAIER, 2006; VIDAL; SASTRY; MA, 2005; SEKKATI; MITICHE, 2006b; MITICHE; SEKKATI, 2006; SCHINDLER; U; WANG, 2006; LI et al., 2007), a segmentação de movimento 2-D também tem recebido atenção, já que ainda possui alguns problemas em aberto e é adequada para algumas tarefas de processamento de vídeos, tais como codificação de vídeo, onde uma representação simples é importante, e em geral aspectos semânticos da cena são menos relevantes (TORRES; KUNT; PEREIRA, 1996).

No contexto da estimativa de movimento, a literatura pode ser dividida em duas classes de métodos: os métodos diretos (IRANI; ANANDAN, 2000) e os métodos baseados em feições (TORR; ZISSERMAN, 2000). Métodos de segmentação de movimento também podem ser divididos de acordo com estes dois paradigmas:

**Métodos diretos:** São aqueles que obtêm os parâmetros desconhecidos (ex: movimento e forma dos objetos) diretamente de quantidades mensuráveis em cada pixel da imagem (tais como brilho ou cor). A maioria dos métodos diretos resolvem o problema da segmentação 2-D.

**Métodos baseados em feições:** São aqueles onde primeiramente é extraído um conjunto esparso de feições (*features*) distintas de cada imagem separadamente, e então recuperam e analisam suas correspondências, a fim de determinar os parâmetros desconhecidos. A maioria dos métodos de segmentação 3-D seguem esta abordagem.

Métodos diretos recuperam os parâmetros desconhecidos diretamente de quantidades mensuráveis em cada pixel da imagem, resolvendo dois problemas simultaneamente:

1. O movimento da câmera e/ou dos objetos da cena;

2. A correspondência de cada pixel.

Isto contrasta com os métodos baseados em feições, os quais extraem primeiro um conjunto esparso de feições distintas de cada imagem separadamente, e então recuperam e analisam suas correspondências de forma a determinar o movimento.

Métodos baseados em feições minimizam uma medida de erro que é baseada nas distâncias entre poucas feições correspondentes, enquanto métodos diretos minimizam uma medida de erro global que é baseada na informação da imagem coletada diretamente de

todos os pixels na imagem. Por esta razão, métodos diretos são algumas vezes chamados de **métodos densos** na literatura. No entanto, devemos notar que, de acordo com a filosofia dos métodos baseados em feições, o movimento pode ser estimado utilizando feições esparsas em um primeiro momento, e em um segundo passo o movimento estimado pode guiar a correspondência para os pixels restantes. Assim, neste trabalho nos referimos a qualquer método de estimativa/segmentação de movimento que resulta na correspondência/classificação para cada pixel como sendo "denso", mesmo que o núcleo da estimativa/segmentação de movimento seja guiado apenas por um conjunto esparso de feições.

É importante observar que, com métodos diretos, a correspondência/classificação dos pixels é executada diretamente a partir das quantidades mensuráveis da imagem em cada pixel, enquanto em métodos baseados em feições isto é feito indiretamente, com base em medidas de feições em um conjunto esparso de pixels. Uma importante propriedade dos métodos diretos é que eles podem estimar com sucesso o movimento global mesmo na presença de vários movimentos e/ou *outliers* (IRANI; ANANDAN, 2000). No entanto, tempo de computação é desperdiçado ao incluir na minimização um grande número de pixels onde nenhum movimento pode ser estimado de maneira confiável. Além do mais, os vetores normais de fluxo podem ser combinados apenas através de regiões da imagem que tem alguma forma paramétrica simples (tais como uma representação afim ou quadrática (TORR; ZISSERMAN, 2000)), e erros de estimativa de movimento podem ser acumulados quando quadros estão distantes entre si e os dados não se ajustam muito bem ao modelo. Por outro lado, métodos baseados em feições inicialmente ignoram áreas que contém pouca informação, resultando em um problema com poucos parâmetros a serem estimados, com boa convergência mesmo para sequências longas. Além disso, existem diversas possibilidades de algoritmos para estimar os parâmetros para modelos mais complexos (como por exemplo, geometria epipolar ou trifocal) a partir de feições baseadas em pontos ou linhas. Apesar disso, nestes métodos as correspondências entre feições são computadas de forma independente, sendo mais suscetíveis a *outliers*.

As duas próximas Seções apresentam os conceitos e uma revisão dos métodos diretos (Seção 2.1) e dos métodos baseados em feições (Seção 2.2).

## 2.1 Métodos Diretos

Métodos diretos são métodos que se valem da minimização de uma medida de erro que é baseada em informação da imagem coletada diretamente de todos os pixels da imagem, tal como brilho ou cor. Embora essa medida de erro varie de uma abordagem para outra, o ponto inicial para a maioria dos métodos diretos de segmentação é assumir a constância de brilho de um determinado ponto da cena. Com isso, dadas duas imagens $I(x, y)$ e $J(x, y)$,

$$J(x, y) = I(x + u(x, y), y + v(x, y)),$$

onde $(x, y)$ são as coordenadas espaciais dos pixels, e $(u(x, y), v(x, y))$ representa o deslocamento do pixel $(x, y)$ entre duas imagens. Assumindo pequenos deslocamentos $(u, v)$, e linearizando $I$ em relação a $(x, y)$, podemos obter a seguinte restrição (IRANI; ANANDAN, 2000):

$$I_x u + I_y v + I_t = 0, \tag{2.1}$$

onde $(I_x, I_y)$ são as derivadas espaciais do brilho da imagem, e $I_t = I - J$, a derivada temporal. Todas as quantidades nesta equação são funções da posição $(x, y)$ da imagem e,

consequentemente, cada pixel provê uma equação que restringe o deslocamento daquele pixel. No entanto, como o deslocamento de cada pixel é definido por duas quantidades, $u$ e $v$, apenas a restrição de brilho é insuficiente para determinar o deslocamento de um pixel. Uma segunda restrição normalmente é aplicada com base em um modelo global de movimento, que descreve a variação de movimento ao longo dos objetos ou da imagem inteira.

Um modelo de movimento frequentemente usado em métodos 2-D diretos é o modelo de movimento afim, que é descrito pelas equações:

$$u(x, y) = a_1 + a_2 x + a_3 y$$

$$v(x, y) = a_4 + a_5 x + a_6 y. \tag{2.2}$$

O modelo de movimento afim é uma boa aproximação para a imagem do movimento 3-D projetado em 2-D quando a distância da câmera em relação aos objetos da cena é grande se comparada ao tamanho dos objetos.

Independentemente do modelo de movimento 2-D utilizado, a abordagem de empregar restrição de movimento global é similar. Como exemplo, será mostrado como isso é feito para a transformação afim. Podemos substituir o movimento afim da Eq. (2.2) na restrição de constância de brilho da Eq. (2.1), obtendo:

$$I_x(a_1 + a_2 x + a_3 y) + I_y(a_4 + a_5 x + a_6 y) + I_t = 0. \tag{2.3}$$

Assim, cada pixel provê uma restrição aos seis parâmetros globais desconhecidos ($a_1$, ..., $a_6$). Como estes parâmetros são globais, ou seja, os mesmos parâmetros são compartilhados por todos os pixels do objeto (ou da imagem), teoricamente, seis restrições independentes de seis pixels diferentes são suficientes para recuperar estes parâmetros. Na prática, no entanto, as restrições de todos os pixels na região de análise são combinadas para minimizar o erro:

$$E(a_1, \ldots, a_6) = \sum \left(I_x(a_1 + a_2 x + a_3 y) + I_y(a_4 + a_5 x + a_6 y) + I_t\right)^2.$$

Note que diferentes pixels contribuem diferentemente para esta medida de erro. Pixels em bordas horizontais na imagem terão valores significativos de $I_y$, mas valores muito baixos de $I_x$, e assim irão restringir apenas a estimativa dos parâmetros $(a_4, a_5, a_6)$. Da mesma forma, um pixel em uma borda vertical irá restringir apenas a estimativa dos parâmetros $(a_1, a_2, a_3)$. Por outro lado, em pixels de cantos (*corner-like pixels*) as duas componentes do gradiente serão grandes, e o pixel irá restringir os 6 parâmetros da trasformação afim. Já pixels em regiões homogêneas irão contribuir muito pouco para o erro.

O processo de linearização do brilho da imagem, que resulta na Eq. (2.3) é uma boa aproximação quando os valores de $(u, v)$ são pequenos, ou seja, quando há apenas pequenos deslocamentos entre os quadros. No entanto, isso raramente é satisfeito em vídeos reais. Para sobrepor esta limitação, muitos métodos diretos utilizam uma estratégia hierárquica (do tipo *coarse-to-fine*) visando suportar um maior espectro de movimentos, através de um refinamento iterativo com uma pirâmide multi-resolução. A observação básica por trás dos métodos hierárquicos é que ao serem aplicadas filtragem e sub-amostragem apropriadas, a amplitude do movimento diminui à medida que vamos de resoluções mais altas para resoluções mais baixas. A análise então começa no nível de menor resolução, onde o movimento na imagem é bastante pequeno. Os parâmetros do movimento estimado são então utilizados para deformar (*warping*) uma imagem em relação à outra, fazendo com

que estas imagens se tornem semelhantes. O processo é então repetido entre as imagens deformadas em níveis de resolução mais altos, até que se alcance a resolução das imagens originais. Uma abordagem para a estimativa e a segmentação de movimento utilizando múltiplas escalas foi proposta em (DEMONCEAUX; KACHI-AKKOUCHE, 2004) e se baseia na transformada *wavelet* do fluxo ótico (*optical flow*) e uma modelagem da segmentação baseada em campos aleatórios de Markov (MRF - *Markov Random Fields*). No entanto, os resultados da segmentação apresentam artefatos de blocos.

Um método para o cálculo de fluxo ótico que utiliza uma abordagem de linearização semelhante à descrita acima e uma estratégia hierárquica foi proposto por Sand e Teller (SAND; TELLER, 2006). Os vetores do fluxo ótico são então utilizados como entrada em um método de detecção de movimento híbrido que combina características dos métodos diretos e dos métodos baseados em feições. Este método se baseia no rastreamento de pontos esparsos, chamados de partículas, ao longo de todos os quadros do vídeo. Partículas são eliminadas caso o ponto correspondente da cena torne-se ocluso, e novas partículas são criadas em regiões da cena que se tornam visíveis em cada quadro. A densidade de partículas é adaptativa ao conteúdo da cena, evitando o desperdício de computação em partes da cena que não apresentam informação de movimento relevante (característica dos métodos baseados em feições). Porém, as partículas são tratadas como se fossem espacialmente conectadas, de forma a reduzir a influência do ruído e, consequentemente, a geração de *outliers* (característica dos métodos diretos). O rastreamento das partículas é executado de quadro em quadro, através da minimização de uma função objetivo, com termos que representam o erro de projeção, a dissimilaridade de movimento em relação ao fluxo ótico, e a dissimilaridade de movimento em relação às partículas espacialmente adjacentes. O algoritmo proposto por Sand e Teller é utilizado no presente trabalho, com pequenas modificações, como entrada para o método de segmentação de movimento proposto, e será descrito no Capítulo 4.

Com o objetivo de reduzir a incerteza na detecção e na segmentação de movimento, alguns métodos extraem outros tipos de informação da imagem, de forma a contextualizar o movimento analisado. O exemplo mais difundido é o uso de segmentação com base em cor para reduzir incertezas na segmentação de movimento (XU; KABUKA; YOUNIS, 2004; BRIASSOULI; KOMPATSIARIS; MEZARIS, 2007; JODOIN; ROSENBERGER; MIGNOTTE, 2007). Um método de fusão de rótulos obtidos a partir da segmentação independente de cor e de movimento é proposto em (JODOIN; ROSENBERGER; MIGNOTTE, 2007). Em (XU; KABUKA; YOUNIS, 2004) o algoritmo de *watershed* (BEUCHER; MEYER, 1993) é utilizado para identificar as bordas dos objetos em movimento, bem como para atualizar os modelos dos objetos ao longo do vídeo. Segmentação com base em cor é associada a uma estimativa de fluxo ótico em (BRIASSOULI; KOMPATSIARIS; MEZARIS, 2007) com o objetivo de comparar regiões cromáticas temporalmente. No entanto, os erros da estimativa do fluxo ótico acumulados impedem que se recupere uma segmentação coerente após vários quadros.

Abordagens variacionais têm sido amplamente utilizadas em métodos diretos para a segmentação de movimento (CREMERS, 2006; CREMERS; OSHER; SOATTO, 2006; CREMERS; KOHLBERGER; SCHNÖRR, 2002; MITICHE; EL-FEGHALI; MANSOUI, 2003; FEGHALI; MITICHE, 2004; CREMERS; SOATTO, 2003). De forma a melhorar a qualidade da segmentação de objetos, probabilidades *a priori* das formas dos objetos foram integradas nos métodos variacionais de Cremer *et al*. (CREMERS, 2006; CREMERS; OSHER; SOATTO, 2006; CREMERS; KOHLBERGER; SCHNÖRR, 2002). No entanto, informação *a priori* sobre objetos em um vídeo geralmente não estão disponí-

veis. Então, Mitiche *et al*. (MITICHE; EL-FEGHALI; MANSOUI, 2003) propuseram segmentar objetos em movimento detectando o túnel delimitado por descontinuidades de movimento no domínio espaço-temporal. Feghali e Mitiche depois estenderam esta idéia para também suportar câmeras em movimento (FEGHALI; MITICHE, 2004), e mais tarde Sekatti e Mitiche propuseram um método direto de segmentação 3-D com uma abordagem similar em (SEKKATI; MITICHE, 2006a) e (SEKKATI; MITICHE, 2004). Eles formularam o problema com um classificador Bayesiano de movimento, e abordaram as equações de Euler-Lagrange correspondentes como um problema envolvendo *level sets*. Cremers e Soatto (CREMERS; SOATTO, 2003) propuseram um método multi-fase baseado em *level sets* para segmentar um vídeo utilizando superfícies espaço-temporais (túneis), que separam regiões com movimento localmente constante. Uma limitação dos métodos baseados em descontinuidades de movimento (ou seja, bordas de movimento), é que eles tendem a falhar em quadros onde estas bordas não são evidentes, ou não existem. Por exemplo, um objeto estático em um *background* estático, que se move apenas nos últimos quadros do vídeo, não pode ser corretamente segmentado no início da sequência, já que não existem bordas de movimento e o objeto não estava se movendo com relação ao *background*.

Várias outras abordagens baseadas em *level sets* foram propostas para a segmentação de movimento em vídeos (SEKKATI; MITICHE, 2006a; XU; YU, 2006; VAZQUEZ; LAGANIERE; MITICHE, 2006; MANSOURI; KONRAD, 2003), tratando o problema da segmentação como um problema de evolução de curvas. Basicamente, o que diferencia tais abordagens são os modelos implícitos de movimento representados através das equações dos *level sets*, além da metologia utilizada para encontrar a solução dos *level sets*. Em (MANSOURI; KONRAD; CHOMAUD, 2001) é apresentado um estudo comparativo do desempenho de diferentes métodos de solução a partir de uma equação comum. Em (XU; YU, 2006) a teoria Bayesiana clássica é utilizada na formulação de uma função de energia que descreve a restrição de suavidade de movimento dos objetos, onde nenhuma suposição a respeito das densidades de bordas, cor ou textura são feitas sobre objeto ou *background*. Porém, a formulação é feita assumindo apenas um objeto em movimento com relação ao *background*. Vázquez *et al.* (VAZQUEZ; LAGANIERE; MITICHE, 2006) propuseram uma função de energia contendo três termos que têm como objetivos: 1) propender a uma segmentação com bordas suaves; 2) fazer com que as bordas dos objetos coincidam com bordas de movimento e 3) coincidir a representação paramétrica de movimento de cada região com as variações espaço-temporais. As componentes de movimento em cada região da segmentação são representadas como funções em um espaço gerado por um conjunto de funções base, tais como polinomiais e senos/cossenos. A formulação usada permite a segmentação de um número arbitrário de regiões. Porém, este número deve ser conhecido *a priori*. Mansouri e Konrad (MANSOURI; KONRAD, 2003) propuseram um método semelhante onde a segmentação é puramente baseada em movimento. As bordas de intensidade não são utilizadas como acessório, permitindo estimativas corretas mesmo quando bordas de intensidade são muito brandas. Porém, a eficiência do método depende de uma boa inicialização.

Ristivojevic e Konrad (RISTIVOJEVIC; KONRAD, 2006) também propuseram um método de segmentação espaço-temporal baseado na abordagem dos *level sets*, onde eles definem os conceitos de **volumes de oclusão** (ou seja, regiões do *background* que se tornam oclusas) e os **volumes de exposição** (que são regiões do *background* que se tornam visíveis ao longo do vídeo). Para isso, é proposta uma estrutura variacional que usa apenas informação de movimento (sem bordas de intensidade) aplicada a uma superfície que

particiona o domínio da sequência de quadros em **interior** e **exterior**. O **interior** corresponde a um volume esculpido por um objeto em movimento, chamado de túnel, enquanto o **exterior** corresponde a um *background* estático. O problema é formulado como uma competição de volumes (generalização 3-D da competição de regiões), no sentido de que a superfície é ajustada em resposta à competição de voxels dentro e fora da superfície, e é resolvido utilizando a abordagem de *level sets*. Na formulação do problema, são incluídos modelos explícitos para as áreas do *background* que são oclusas (volumes de oclusão) ou que se tornam visíveis (volumes de exposição) ao longo do vídeo. Os autores sugerem que estes conceitos de volumes de oclusão e volumes de exposição potencialmente podem ser aplicados em uma nova geração de métodos de compressão de vídeos. Como áreas de oclusão e de exposição são difíceis de predizer, novas técnicas eficientes de compressão poderiam ser desenvolvidas estimando-se áreas de oclusão e de exposição *a priori*. No entanto, o conceito de volume de oclusão tem limitações conforme proposto, já que não considera a oclusão de um objeto em movimento pelo *background*, ou por outro objeto em movimento. Conforme será visto no Capítulo 4, o método proposto no presente trabalho gera representações espaço-temporais (túneis) conceitualmente equivalentes àquelas propostas no trabalho de Ristivojevic e Konrad, porém, de forma mais geral, já que o método proposto suporta a segmentação de objetos quaisquer em movimento que geram oclusão uns aos outros, e não é assumido nenhum modelo para o *background*, seja ele estático ou não. Uma característica compartilhada por muitos métodos variacionais é que eles recaem em modelos de movimento definidos *a priori*. Se os dados não se ajustam bem a estes modelos, os métodos tendem a falhar - exceto quando condições especiais podem ser assumidas, como *background* estático, número de objetos conhecido *a priori*, etc.

Outro método que utiliza informação diretamente extraída de vários quadros do vídeo foi proposto por Vidal e Singaraju (VIDAL; SINGARAJU, 2005). Neste trabalho é apresentada uma abordagem algébrica para a segmentação de movimento 2-D direta de cenas estáticas e dinâmicas. Os autores demonstram que é possível estimar o número de modelos de movimento, o fluxo ótico e parâmetros de cada modelo de movimento - modelo translacional e modelo afim - diretamente das intensidades da imagem, sem a necessidade de rastreamento de feições, correspondência de pontos, fluxo ótico ou segmentação computados previamente. No entanto, a derivação algébrica do algoritmo assume dados sem ruído e não considera a presença de *outliers* e oclusão. Isto, somado ao fato de que os modelos são computados individualmente para cada pixel, sem forçar a suavidade espacial dos vetores de movimento, pode acarretar em grandes erros de estimativa. Os autores sugerem que a regularização espacial seja incorporada através de filtros suavizadores. Porém, isto ocasiona borramento da estimativa de movimento, especialmente próximo às bordas de movimento.

Conforme dito anteriormente, a grande maioria dos métodos diretos de segmentação propostos resolvem o problema da segmentação de movimento 2-D. Uma exceção é o método direto proposto por Sekkati e Mitiche em (SEKKATI; MITICHE, 2006a) (e com uma abordagem semelhante em (MITICHE; SEKKATI, 2006)), que extrai informação densa de movimento e da estrutura 3-D de objetos diretamente de uma sequência de imagens em tons de cinza, onde tanto a câmera como os objetos da cena podem estar em movimento. Para isto, o problema da segmentação é abordado a partir de uma formulação variacional, que alterna entre três passos: estimativa de parâmetros de movimento 3-D rígido por mínimos quadrados, estimativa de profundidade por descida de gradiente, e evolução de curvas via *level sets*. No entanto, como todo o método baseado em evo-

lução de curvas, é sensível à inicialização. Outra proposta para a segmentação direta de movimento 3-D foi feita por Pei e Liou (PEI; LIOU, 1997), onde primeiramente o fluxo ótico é particionado em regiões de movimento 2-D coerente (cujas projeções dos pontos no plano formam regiões analíticas). Baseando-se na parametrização destas regiões, o método proposto provê um método para testar a compatibilidade de movimento 3-D entre duas regiões. Porém, assume-se que estas regiões analíticas foram projetadas por regiões planares em movimento 3-D, o que geralmente não acontece em situações reais.

## 2.2 Métodos Baseados em Feições

Os métodos para segmentação de movimento baseados em feições usualmente consistem de dois estágios independentes (TORR; ZISSERMAN, 2000):

1. Seleção de feições e/ou correspondência entre feições;
2. Estimativa dos parâmetros de movimento.

O segundo estágio é geralmente executado através de métodos de fatoração (COSTEIRA; KANADE, 1998; YAN; POLLEFEYS, 2006; ICHIMURA, 2000; LUBLINERMAN; CAMPS; SZNAIER, 2006), embora algumas estratégias simples de agrupamentos podem ser utilizadas (LI; HATZINAKOS; VENETSANOPOULOS, 1999; SMITH; BRADY, 1995). Nos métodos de fatoração, as informações sobre movimento e forma são tratadas separadamente aplicando-se restrições à projeção da cena no plano de formação da imagem, assim como restrições às formas dos objetos e aos tipos de movimentos.

### 2.2.1 Seleção e Correspondência de Feições

O problema da seleção de feições consiste em selecionar pontos (ou regiões) de uma imagem que tenham alto poder de discriminação e tenham um bom potencial de rastreabilidade. Já o problema da correspondência de feições consiste em encontrar a correspondência entre feições selecionadas em quadros diferentes. A dificuldade na solução destes problemas é que não se pode esperar que um mesmo ponto (ou região) de um determinado quadro do vídeo mantenha exatamente as mesmas características em outros quadros do vídeo. Métodos de seleção e correspondência de feições frequentemente entregam ambiguidades e falsas correspondências (MA et al., 2003), e por esta razão os passos seguintes de estimativa dos parâmetros de movimento devem ser robustos para contornar estas limitações.

Diversos métodos têm sido propostos para a seleção e/ou correspondência de feições, e entre os mais populares estão o detector de cantos de Harris (HARRIS; STEPHENS, 1988), o rastreador KLT (*Kanade-Lucas-Tomasi*) (SHI et al., 1994) e o SIFT (*Scale-Invariant Feature Transform*) (LOWE, 2004).

Tanto o detector de Harris quanto o KLT operam na **matriz de estrutura local** $C$, que tem a forma:

$$C = \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} = \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix}^\top \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix}, \tag{2.4}$$

onde $\frac{\partial I}{\partial x}$ e $\frac{\partial I}{\partial y}$ são as derivadas parciais de um quadro $I$ em relação às dimensões espaciais $x$ e $y$. Sejam $\lambda_1 \geq \lambda_2$ os dois auto-valores da matriz $C$. Como $C$ é simétrica e positiva semi-definida, tanto $\lambda_1$ quanto $\lambda_2$ são não-negativos. As seguintes interpretações geométricas podem ser obtidas destes auto-valores:

- Em uma região uniforme e homogênea, $\lambda_1 = \lambda_2 = 0$;
- Na localidade de uma borda, $\lambda_1 > \lambda_2 = 0$, e o auto-vetor correspondente a $\lambda_1$ está associado com a direção que é ortogonal à borda;
- Na localidade de um canto, $\lambda_1 \geq \lambda_2 > 0$, e quanto maiores forem os valores de $\lambda_1$ e $\lambda_2$, maiores são os contrastes das bordas ortogonais às direções dos auto-vetores correspondentes.

Então, os auto-vetores carregam as direções das bordas e os auto-valores carregam as magnitudes das bordas. Isso significa que um canto deve ser marcado em uma localidade onde o menor auto-valor, $\lambda_2$, é grande o suficiente.

Para o detector de Harris, a matriz de estrutura local é suavizada por uma gaussiana:

$$C_{Harris} = G(\sigma) * \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix}, \tag{2.5}$$

onde $G(\sigma)$ é um filtro gaussiano isotrópico com desvio padrão $\sigma$ e a operação $*$ denota convolução. A medida da **resposta de canto** em cada pixel de coordenadas $(x, y)$ é definida então por:

$$r(x, y) = \det(C_{Harris}(x, y)) - \kappa(\text{trace}(C_{Harris}(x, y))^2, \tag{2.6}$$

onde $\kappa$ é uma constante ajustável e $C_{Harris}(x, y)$ é a matriz de estrutura local $2 \times 2$ no pixel de coordenadas $(x, y)$.

Para o detector de cantos do KLT, são utilizados dois parâmetros: um limiar de valor mínimo $\lambda_{min}$, para o segundo auto-valor $\lambda_2$ e um raio $d$ da janela que define a vizinhança em um ponto. O algoritmo pode ser descrito da seguinte forma:

1. Para cada ponto $(x, y)$ da imagem:
   (a) Construir a matriz de estrutura local sobre uma vizinhança $R$ de tamanho $(2d+1) \times (2d+1)$:

$$C_{KLT}(x, y) = \begin{bmatrix} \sum\sum_R \left(\frac{\partial I}{\partial x}\right)^2 & \sum\sum_R \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum\sum_R \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum\sum_R \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix}$$

   (b) Calcular o menor auto-valor $\lambda_2$, da matriz $C_{KLT}(x, y)$
   (c) Se $\lambda_2 > \lambda_{min}$, salvar $(x, y)$ em uma lista $L$ de potenciais cantos
2. Ordenar $L$ em uma ordem decrescente de $\lambda_2$
3. Percorra a lista $L$ ordenada de cima para baixo, e selecione pontos na lista em sequência. Pontos localizados dentro da vizinhança $R$ de qualquer ponto selecionado são removidos.

Já a abordagem de rastreamento do KLT assume que em vídeos o deslocamento dos cantos é pequeno entre quadros adjacentes, e consequentemente os cantos podem ser rastreados. Dadas duas imagens $I$ e $I'$ representando dois quadros adjacentes do vídeo, o rastreador KLT iterativamente procura pela localização de um ponto $(x', y')$ em $I'$, correspondente ao ponto $(x, y)$ em $I$, minimizando a diferença de intensidades entre janelas

de tamanho fixo, $W$ e $W'$, centradas em $(x, y)$ e $(x', y')$, respectivamente. A versão mais simples do rastreador KLT é baseada na aproximação local do movimento pelo modelo translacional, onde o deslocamento $(d_x, d_y)$ de cada ponto é estimado pela minimização do erro:

$$\epsilon = \sum_{(x,y)\in W} [I'(x + d_x, y + d_y) - I(x, y)]^2.$$

O tamanho da janela $W$ depende de duas entidades desconhecidas: o movimento 3-D da câmera e a distância da câmera em relação aos objetos. Geralmente é necessário ajustar o tamanho da janela com base em algum conhecimento prévio do vídeo.

O detector de **pontos-chave** no SIFT é constituído dos seguintes passos:

1. A imagem $I(x, y)$ é convoluída por gaussianas cujos desvios padrões $\{\sigma_1, \sigma_2, \ldots\}$ diferem por um fator de escala fixo. Ou seja, $\sigma_{j+1} = k\sigma_j$ onde $k$ é uma constante e deve ser fixada em $\sqrt{2}$. A convolução resulta em um pequeno número de imagens suavizadas, denotadas por $\{I_G(x, y, \sigma_1), I_G(x, y, \sigma_2), \ldots\}$;

2. As imagens suavizadas adjacentes são então subtraídas, resultando em um pequeno número (3 ou 4) de imagens DoG (*Difference-of-Gaussian*):

$$D(x, y, \sigma_j) = I_G(x, y, \sigma_{j+1}) - I_G(x, y, \sigma_j);$$

3. As imagens suavizadas no passo 1 são sub-amostradas e o procedimento no passo 2 é repetido para as imagens sub-amostradas, resultando em imagens DoG ao longo do espaço-escala;

4. Cada ponto destas imagens DoG são então examinados. Um **ponto-chave** é marcado em uma localidade onde o ponto é um mínimo local ou máximo local de seus 8 vizinhos na mesma escala e seus 9 vizinhos nas escalas imediatamente acima e abaixo.

Ao contrário dos detectores de Harris e do KLT, os **pontos-chave** do SIFT muitas vezes não coincidem com cantos.

Tendo encontrado as localidades dos **pontos-chave**, os gradientes da imagem são computados e classificados em 8 orientações para uma matriz-histograma $4 \times 4$, resultando em um descritor de 128 elementos para cada **ponto-chave**. Após encontrar os **pontos-chave** e calcular os respectivos descritores em diferentes quadros, a correspondência de **pontos-chave** é feita considerando-se as distâncias par-a-par dos descritores em quadros diferentes no espaço 128-D.

Embora as feições utilizadas geralmente se baseiem em pontos selecionados dos quadros do vídeo, tais como no detector de Harris, no rastreador KLT e no SIFT, outros tipos de feições também podem ser utilizados, tais como bordas de imagens (SMITH; CIPOLLA; DRUMMOND, 2004) ou regiões de formas arbitrárias (XU; KABUKA; YOUNIS, 2004). Nestes casos, a etapa de seleção de feições pode ser executada através de um algoritmo para detecção de bordas ou segmentação de imagens.

Conforme mencionado anteriormente, um ponto fraco dos métodos baseados em feições é que as feições correspondentes são computadas de forma independente. Com isso, eles são muito sensíveis a *outliers*, tornando-os suscetíveis a erros na estimativa de parâmetros e, consequentemente, na segmentação de movimento. Ainda, regiões homogêneas de um quadro podem apresentar nenhuma ou poucas feições, tornando a estimativa de movimento difícil (ou mesmo impossível) em grandes áreas dos quadros do vídeo.

### 2.2.2 Estimativa dos Parâmetros de Movimento

Os métodos baseados em feições são comumente utilizados para a segmentação de movimento 3-D. Os métodos de segmentação de movimento 3-D podem ser divididos em duas categorias: 1) métodos baseados no modelo de projeção afim; e 2) métodos baseados no modelo de projeção perspectiva. Em um modelo de projeção afim (que generaliza projeção ortográfica, perspectiva fraca e para-perspectiva), trajetórias de pontos associadas a um determinado objeto em movimento ao longo de diversos quadros recai em um subespaço linear de dimensão no máximo igual a 4. Assim, a segmentação de movimento 3-D pode ser obtida agrupando-se trajetórias de pontos em diferentes subespaços de movimento. Já em modelos de projeção perspectiva, trajetórias de pontos associadas a um objeto em movimento recaem em subespaços multilineares (bilinear para duas vistas, trilinear para três vistas, etc.). Com isso, a segmentação de movimento é obtida ao se agrupar esses diversos subespaços multilineares. Como este problema não é trivial, muitos trabalhos anteriores se limitam a métodos algébricos para a fatoração bilinear (VIDAL et al., 2006) e trilinear (HARTLEY; VIDAL, 2004) e métodos estatísticos para duas (TORR, 1998) e múltiplas vistas (SCHINDLER; U; WANG, 2006). No entanto, atualmente o desempenho dos métodos baseados em projeção perspectiva ainda é muito inferior ao dos métodos baseados em projeção afim e necessitam ser significantemente aperfeiçoados (TRON; VIDAL, 2007). No presente trabalho, o escopo será restrito aos métodos de segmentação baseados em modelos de projeção afim, discutido a seguir.

Sejam $\{x_{fp} \in \mathbb{R}^2\}_{p=1,\ldots,P}^{f=1,\ldots,F}$ as projeções de $P$ pontos 3-D $\{X_p \in \mathbb{P}^3\}$ de um objeto rígido em movimento em $F$ quadros com uma câmera em movimento rígido. Em um modelo de projeção afim, as imagens (quadros do vídeo) satisfazem a equação:

$$x_{fp} = A_f X_p, \tag{2.7}$$

onde:

$$A_f = K_f \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_f & t_f \\ 0^T & 1 \end{bmatrix} \in \mathbb{R}^{2\times 4}$$

é a matriz de câmera afim no quadro $f$, a qual depende dos parâmetros de calibração da câmera $K_f \in \mathbb{R}^{2\times 3}$ e da localização do objeto em relação à câmera $(R_f, t_f) \in SE(3)$.

Seja $W_1 \in \mathbb{R}^{2F\times P}$ a matriz cujas $P$ colunas representam as trajetórias dos pontos $\{x_{fp}\}_{p=1}^{P}$. Deduz-se a partir da Eq. (2.7) que $W_1$ pode ser decomposta em uma matriz de movimento $M_1 \in \mathbb{R}^{2F\times 4}$ e uma matriz de estrutura $S_1 \in \mathbb{R}^{P\times 4}$:

$$W_1 = M_1 S_1^T.$$

$$\begin{bmatrix} x_{11} & \cdots & x_{1P} \\ \vdots & & \vdots \\ x_{F1} & \cdots & x_{FP} \end{bmatrix}_{2F\times P} = \begin{bmatrix} A_1 \\ \vdots \\ A_F \end{bmatrix}_{2F\times 4} \begin{bmatrix} X_1 & \cdots X_P \end{bmatrix}_{4\times P},$$

consequentemente, o posto da matriz $W_1$ é menor ou igual a 4 ($\texttt{posto}(W_1) \leq 4$). Note também que as linhas de cada matriz $A_f$ compõem combinações lineares das duas primeiras linhas da matriz de rotação $R_f$ e, consequentemente, $\texttt{posto}(W_1) \geq \texttt{posto}(A_f) = 2$. Portanto, segundo o modelo de projeção afim, as trajetórias 2-D de um conjunto de pontos 3-D vistos por uma câmera em movimento rígido (ou seja, as colunas de $W_1$) incidem em um subespaço de $\mathbb{R}^{2F}$ de dimensão $d_1 = \texttt{posto}(W_1) = 2, 3$ ou 4.

Assuma agora que as $P$ trajetórias $\{x_{fp}\}_{p=1}^{P}$ correspondem a $n$ objetos submetidos a $n$ movimentos de corpos rígidos em relação a uma câmera em movimento. O problema da segmentação de movimento 3-D é equivalente ao agrupamento (*clustering*) de um conjunto de pontos em $n$ subespaços de $\mathbb{R}^{2F}$ de dimensões desconhecidas $d_i \in \{2, 3, 4\}$ para $i = 1, \ldots, n$.

A matriz de trajetórias pode ser escrita como:

$$W = [W_1, W_2, \ldots, W_n]\,\Gamma \in \mathbb{R}^{2F \times P},$$

onde as colunas de $W_i \in \mathbb{R}^{2F \times P_i}$ são as $P_i$ trajetórias associadas com o $i$-ésimo objeto em movimento, $P = \sum_{i=1}^{n} P_i$, e $\Gamma^{\mathsf{T}} \in \mathbb{R}^{P \times P}$ é uma matriz desconhecida em que as $P$ trajetórias são permutadas de acordo com os $n$ movimentos. Visto que a matriz $W_i$ pode ser fatorada em matrizes $\hat{M}_i \in \mathbb{R}^{2F \times d_i}$ e $\hat{S}_i \in \mathbb{R}^{P_i \times d_i}$, tal que

$$W_i = \hat{M}_i \hat{S}_i^{\mathsf{T}} \quad i = 1, \ldots, n,$$

a matriz associada a todos os objetos pode ser fatorada nas matrizes $M \in \mathbb{R}^{2F \times \sum_{i=1}^{2} d_i}$ e $S \in \mathbb{R}^{P \times \sum_{i=1}^{n} d_i}$ como:

$$\begin{aligned} W &= [W_1, W_2, \ldots, W_n]\,\Gamma \in \mathbb{R}^{2F \times P} \\ &= \left[\hat{M}_1, \hat{M}_2, \ldots, \hat{M}_n\right] \begin{bmatrix} \hat{S}_1^{\mathsf{T}} & & & \\ & \hat{S}_2^{\mathsf{T}} & & \\ & & \ddots & \\ & & & \hat{S}_3^{\mathsf{T}} \end{bmatrix} \Gamma \\ &= MS^{\mathsf{T}}\Gamma. \end{aligned} \tag{2.8}$$

Conclui-se que uma maneira possível de resolver o problema da segmentação de movimento é achar uma matriz de permutação $\Gamma$, tal que a matriz $W\Gamma^{\mathsf{T}}$ possa ser decomposta em uma matriz de movimento $M$ e uma matriz de estrutura diagonal em blocos $S$. Esta idéia foi inicialmente proposta em (BOULT; BROWN, 1991) e (COSTEIRA; KANADE, 1998) e tem sido a base para a maioria dos métodos de segmentação de movimento 3-D existentes até hoje. No entanto, conforme mostrado em (KANATANI, 2001), a fim de que $W$ possa ser fatorada de acordo com a Eq. (2.8), os correspondentes subespaços de movimento $\left\{\mathcal{W}_i \subset \mathbb{R}^{2F}\right\}_{i=1}^{n}$ devem ser independentes. Ou seja, para todo $i \neq j = 1, \ldots, n$, devemos ter $\mathtt{dim}(\mathcal{W}_i \cap \mathcal{W}_j) = 0$, tal que $\mathtt{posto}(W) = \sum_{i=1}^{i} d_i$, onde $d_i$ é a dimensionalidade de $\mathcal{W}_i$.

Infelizmente, a maioria das sequências reais de movimento exibem movimentos parcialmente dependentes, ou seja, existem $i, j \in \{1, \ldots, n\}$ tais que $0 < \mathtt{dim}(\mathcal{W}_i \cap \mathcal{W}_j) < \mathtt{min}\{d_i, d_j\}$. Por exemplo, quando dois objetos têm o mesmo movimento rotacional mas diferentes translações em relação à câmera, ou em movimentos articulados. Isto tem motivado o desenvolvimento de diversos algoritmos que tratam de movimentos parcialmente dependentes, incluindo métodos estatísticos (GRUBER; WEISS, 2004), espectrais (YAN; POLLEFEYS, 2006) e algébricos (VIDAL; HARTLEY, 2004).

O algoritmo MSL (*Multi-Stage Learning*) é uma abordagem estatística proposta por Sugaya e Kanatani (SUGAYA; KANATANI, 2004). Este algoritmo se baseia no método de fatorização de Costeira e Kanade (COSTEIRA; KANADE, 1998) e no método de separação de sub-espaços de Kanatami (KANATANI, 2001). Enquanto estes métodos se aplicam a movimentos independentes e sub-espaços não degenerados, o MSL suporta algumas classes de movimento degenerado refinando a solução do método de separação

de sub-espaços utilizando o algoritmo EM (*Expectation-Maximization*) (DEMPSTER; LAIRD; RUBIN, 1977). Porém, como todo o algoritmo baseado em EM, o MSL sofre do problema de convergência para mínimos locais. Outra desvantagem é que o algoritmo não foi projetado para movimentos parcialmente dependentes, então sua performance não é a ideal nesses casos.

O método GPCA (*Generalized Principal Component Analysis*) (VIDAL; SASTRY; MA, 2005) constitui uma abordagem algébrica para o agrupamento de dados que recaem em múltiplos subespaços. A idéia principal por trás do GPCA é de que é possível ajustar uma união de $n$ subespaços com um conjunto de funções polinomiais de grau $n$, onde a derivada em um ponto resulta em um vetor normal ao subespaço contendo aquele ponto. A segmentação dos dados é então obtida agrupando-se esses vetores normais, o que pode ser feito utilizando diversas técnicas. No contexto de segmentação de movimento, o GPCA opera em três etapas: 1) projeção das trajetórias em um sub-espaço de dimensão 5 para obter a matriz de dados projetada; 2) ajuste de uma função polinomial que representa todos os sub-espaços de movimento aos dados projetados; 3) agrupamento espectral (*spectral clustering*) de feições através de diferenciação polinomial. Como cada sub-espaço é representado por um hiper-plano contendo o sub-espaço, interseções entre os sub-espaços são automaticamente permitidas, e com isso o algoritmo pode lidar tanto com movimentos independentes como com movimentos parcialmente dependentes. A principal desvantagem da GPCA é que o vetor de coeficientes da função polinomial tem dimensão O($n^4$), enquanto existem apenas $4n$ incógnitas nos $n$ vetores normais. Como o vetor de coeficientes é computado usando um método de mínimos quadrados, isso causa uma grande deterioração da performance da GPCA à medida que $n$ aumenta. Além do mais, a estimativa do vetor de coeficientes é sensível a *outliers*.

O método LSA (*Local Subspace Affinity*) proposto por Yan e Pollefeys (YAN; POLLEFEYS, 2006) também é baseado em uma projeção linear e agrupamento espectral. A principal diferença é que o LSA ajusta um subespaço localmente em torno de cada ponto projetado, com base nos $k$ vizinhos mais próximos ao ponto projetado, enquanto o GPCA usa os gradientes de uma polinomial que é globalmente ajustada aos dados projetados. A desvantagem da LSA é que vizinhos de um ponto podem pertencer a um subespaço diferente - e isto é mais provável de acontecer em torno da intersecção de dois sub-espaços. Ainda, os vizinhos selecionados podem não ser suficientes para representar o sub-espaço subjacente. Estes casos são uma fonte de potenciais erros de classificação.

O método proposto em (LUBLINERMAN; CAMPS; SZNAIER, 2006) trata do problema da dependência de movimento considerando restrições temporais (além das restrições geométricas dos métodos de fatoração). Segundo esta proposta, a segmentação é executada agrupando-se pontos de acordo com a complexidade do modelo necessário para explicar o movimento relativo. Esta abordagem resulta em uma maior robustez a *outliers*; porém, não suporta oclusão, pois necessita que todos os pontos sejam definidos em todos os quadros da sequência.

Também com a finalidade de contornar o problema da sensibilidade dos métodos de fatoração a ruído e *outliers*, Wang e Li (WANG; LIN, 2003) propuseram um método baseado em agrupamento espectral para a segmentação de múltiplos objetos em movimento. Ao invés da matriz de interação de forma proposta por Costeira e Kanade (COSTEIRA; KANADE, 1998), os autores utilizam uma nova matriz de afinidade baseada na trajetória dos pontos. A seguir, após mapear as feições para um sub-espaço de baixa-dimensionalidade, são computadas as sensibilidades dos maiores auto-valores em relação a mudanças na matriz de afinidade. Isto torna a tarefa de agrupamento (*clustering*) mais

confiável. No entanto, o procedimento de construção da matriz de afinidade pressupõe que todas as feições sejam conhecidas em todos os quadros e, com isso, despreza os efeitos da oclusão, limitando sua aplicação prática.

Também são encontradas na literatura diversas propostas de segmentação de movimento 2-D que são baseadas em feições. O rastreamento de bordas é utilizado em (SMITH; CIPOLLA; DRUMMOND, 2004) para guiar o processo de segmentação. Para isso é utilizado o detector de bordas de Canny (CANNY, 1986) e um método de amostragem de bordas que permite o rastreamento das bordas em quadros diferentes, com base em estimativas de movimento obtidas pelo algoritmo EM (DEMPSTER; LAIRD; RUBIN, 1977). Apesar de não recuperar informação a respeito da forma 3-D dos objetos da cena, este método retorna uma hierarquia da profundidade dos objetos da cena. No entanto, quando há mais de dois movimentos na cena (ou seja, o *background* e um único objeto em movimento), o método se mostra muito sensível à inicialização do algoritmo EM, tornando a convergência crítica.

Em (GAO; JUNG; THAKOOR, 2004) é proposto um método de segmentação baseado no cálculo do fluxo ótico em que primeiramente são selecionados pontos para os quais o movimento pode ser calculado com um alto grau de confiança (com base em texturas). Após o movimento deste conjunto esparso de pontos ser estimado, é efetuado o agrupamento dos mesmos com o algoritmo *k-means* (LLOYD, 1982). Com base na estimativa de movimento e segmentação deste conjunto inicial de pontos, bem como em uma segmentação a cores de todos os pixels da imagem, os valores dos vetores de movimento desconhecidos são iterativamente calculados e ajustados a modelos afins para cada região. Porém, o uso de informação de cor no processo de segmentação do fluxo ótico poder gerar falsas bordas de movimento (sobre-segmentação), ou mesmo forçar regiões de movimentos distintos a caírem no mesmo grupo (sub-segmentação) por apresentarem similaridade de cor. Além do mais, o método *k-means* busca grupos (*clusters*) esféricos com dispersões semelhantes no espaço de feições, o que restringe bastante o tipo de movimento detectado. Uma alternativa ao *k-means* amplamente utilizada em segmentação de imagens e rastreamento de objetos é o *mean-shift* (COMANICIU; MEER, 2002). O *mean-shift* é um procedimento para localizar máximos de uma função densidade a partir de dados discretos amostrados desta função. É um método iterativo que começa com uma estimativa inicial $x$, utiliza um *kernel* (tipicamente gaussiano) em torno deste ponto $x$ e, com base na resposta dos pontos vizinhos amostrados a esse *kernel*, re-calcula a média dos pontos. O processo é repetido até que se alcance a estabilidade, ou seja, a média calculada não mude de uma iteração para outra. O *mean-shift* pode ser utilizado para o agrupamento de pontos ao se repetir o processo descrito acima utilizando cada um dos pontos do conjunto como estimativa inicial, e com isso obtendo todos os modos da função. A utilização do *mean-shift* no agrupamento de pontos apresenta a vantagem de buscar grupos de pontos com formas arbitrárias no espaço de feições, além de não necessitar definir o número de grupos *a priori*. Conforme será visto no Capítulo 4.3, o *mean-shift* é empregado no presente trabalho para obter grupos de pontos espacialmente amostrados de um vídeo a partir de seus vetores de deslocamento.

Eventualmente, podemos obter segmentações consistentes de objetos combinando várias informações parciais sobre um objeto. A idéia de fundir informação de segmentação de objetos de diversas partes de um vídeo foi proposta por Gelgon *et al.* (GELGON; BOUTHEMY; LE CADRE, 2005), através de uma abordagem de rastreamento probabilístico de múltiplas hipóteses (PMHT - *Probabilistic Multiple Hypothesis Tracking*). Os autores propuseram rastrear um objeto ao longo de toda uma sequência de quadros, com-

binando segmentações parciais de objetos previamente computados em diferentes partes da sequência. Isto é feito a partir da modelagem do movimento e da geometria dos objetos, e estes modelos são combinados assumindo trajetórias suaves, e são utilizados para eliminar ambiguidades causadas pelas oclusões e detecções incorretas. No entanto, a precisão da modelagem do movimento e/ou geometria do objeto depende do quão bem o modelo se ajusta aos dados, além de que as restrições de trajetória impossibilitam a aplicação desta abordagem em vídeos com objetos que contém trajetórias descontínuas, as quais são comuns em vídeos obtidos por câmeras de mão, por exemplo. Conforme será visto no Capítulo 4, no presente trabalho é proposto um método de segmentação que combina informação de segmentações de objetos obtidas ao longo de vários quadros do vídeo, sem as restrições descritas acima.

## 2.3 Conclusões

Neste Capítulo, foi apresentado o problema da segmentação de vídeos em objetos constituintes. Conforme mencionado anteriormente, os métodos de segmentação de vídeos existentes podem ser divididos em duas classes: os métodos diretos e os métodos baseados em feições. Métodos diretos recuperam os parâmetros desconhecidos (como o movimento ou forma dos objetos) diretamente de quantidades mensuráveis em cada pixel da imagem. Pressupondo constância de certas propriedades fotométricas em pontos da cena, cada pixel provê uma equação que restringe o movimento naquele pixel. Assim, os métodos diretos resolvem dois problemas simultaneamente: a identificação do movimento da câmera e/ou dos objetos da cena e a correpondência de cada pixel. Já os métodos baseados em feições resolvem primeiro o problema da correspondência entre um conjunto esparso de pontos e, a partir dessa informação, identificam o movimento dos objetos.

Métodos diretos podem estimar com sucesso o movimento global, seja da cena como um todo ou de um objeto em particular, mesmo na presença de ruído e *outliers*. Porém, desperdiçam esforço computacional ao incluir no processo de minimização diversos pixels que não contém informação de movimento relevante. Além do mais, por trabalharem apenas com a estimativa de movimento na grade de pixels, e assumir constância de propriedades fotométricas, quando essa suposição não é verificada, erros de estimativa tendem a se acumular e degradar rapidamente a qualidade da segmentação ao longo do tempo, dificultando a identificação de padrões de movimento longos.

Já os métodos baseados em feições ignoram áreas que contém pouca informação durante o estágio de correspondência de pontos, resultando em um problema com poucos parâmetros e boa convergência mesmo para sequências longas. No entanto, as correspondências entre pontos em diferentes quadros são computadas de forma independente, se tornando mais suscetíveis à influência de *outliers*.

De modo geral, conclui-se que muitos métodos de segmentação foram desenvolvidos assumindo determinadas propriedades, tais como modelos de câmera específicos, objetos rígidos, número de objetos conhecidos *a priori* (normalmente apenas um objeto em movimento), *background* estático, etc. Portanto, o desenvolvimento de métodos de segmentação de vídeos que sejam ao mesmo tempo robustos e generalistas continua sendo um desafio para a comunidade científica.

# 3 CODIFICAÇÃO DE VÍDEOS BASEADA EM OBJETOS: CONCEITOS E ESTADO DA ARTE

A forma mais utilizada de se representar imagens e vídeos no domínio digital é através de pixels. Isto se deve principalmente ao fato de que a aquisição e a reprodução de informação visual digital através de pixels utiliza tecnologias já maduras e relativamente baratas. Na metade da década de 80, pela primeira vez, estudos dos mecanismos do sistema visual humano motivaram o desenvolvimento de outras técnicas de representação (KUNT; IKONOMOPOULOS; KOCHER, 1985). Como o sistema visual humano é a última etapa da cadeia em inúmeras tarefas de processamento de imagens, então uma representação que combinasse com aquela do sistema visual humano seria mais eficiente no projeto de sistemas de processamento e codificação de imagens.

O primeiro método de codificação de vídeos baseado em objetos foi proposto por Musmann *et al.* (MUSMANN; HOTTER; OSTERMANN, 1990), onde objetos são representados por parâmetros que descrevem movimento, forma e cor. Outras abordagens foram propostas mais tarde para a codificação de vídeo como coleções de imagens 2-D delimitadas por formas arbitrárias (TALLURI et al., 1997; KAUFF et al., 1997; SHAMIM; ROBINSON, 2002; SUN; AHMAD, 2004).

Do ponto de vista de codificação, dada a segmentação de um vídeo em objetos constituintes, existem basicamente três aspectos que diferenciam as abordagens existentes:

**Representação e codificação das formas dos objetos** (Seção 3.1): Diz respeito ao tipo de codificação (com ou sem perdas, implícita ou explícita, etc.) utilizado na representação das formas dos objetos do vídeo, bem como as estratégias utilizadas na predição destas;

**Representação e codificação do movimento dos objetos** (Seção 3.2): Diz respeito aos modelos de movimento utilizados na representação dos objetos, bem como aspectos da codificação dos parâmetros destes modelos;

**Representação e codificação da textura dos objetos** (Seção 3.3): Diz respeito aos métodos de codificação das texturas (informação de cor dos objetos) normalmente sob a forma de resíduo preditivo.

Todos estes aspectos estão profundamente relacionados, e o desempenho das diferentes abordagens está ligado diretamente ao tipo de segmentação utilizada como base para a codificação. Nas próximas Seções, estes aspectos serão discutidos em detalhes separadamente.

## 3.1 Representação e Codificação das Formas dos Objetos

A representação da forma de um determinado objeto, no contexto da codificação de vídeos, diz respeito à informação necessária para se determinar o conjunto de pixels ao longo da sequência de quadros que estão associados a este objeto. Isto normalmente é feito caracterizando diretamente as regiões de diferentes quadros que contém os pixels de um determinado objeto, ou então caracterizando o contorno do objeto, ou seja, a linha que separa os pixels que pertencem ao objeto daqueles que não pertencem ao objeto. Geralmente a transmissão e/ou decodificação da forma do objeto é a primeira etapa na reconstrução do vídeo pelo decodificador. Isto porque a compensação de movimento e a codificação das texturas estão atreladas à essa informação. Nesta Seção serão apresentadas as quatro abordagens representativas empregadas na representação e codificação das formas dos objetos em métodos de codificação de vídeos.

Como a forma mais utilizada de se representar vídeos no domínio digital é através de pixels, uma maneira direta de se representar formas de objetos contidos em um vídeo é através de imagens binárias, com o mesmo tamanho dos quadros do vídeo, em que um pixel tem o valor '1' caso o pixel correspondente do quadro do vídeo pertença ao objeto, e o valor '0' caso não pertença. Assim, métodos de codificação de imagens binárias podem ser utilizados para codificar estas máscaras que representam a forma de um objeto.

O método de codificação de comprimento de corrida (RLE - *Run Length Encoding*) é uma forma bastante simples e bastante utilizada na codificação de imagens binárias, em que "corridas" de pixels (ou seja, sequências contíguas de pixels de um mesmo valor) são representadas por um único valor, correspondente à contagem do número de elementos da "corrida". Dessa forma, a máscara binária pode ser percorrida linha a linha, sendo armazenadas apenas as quantidades de pixels consecutivos de mesmo valor ('0' ou '1'). Um método de codificação com comprimento de palavra variável (como o código de Huffman ou a codificação aritmética) pode ser aplicado às corridas. Porém, o RLE não é eficiente na representação de formas complexas e fragmentadas, que resultam em corridas de pequeno comprimento. Nestes casos, a codificação por RLE pode ser mais ineficiente do que a representação direta do mapa binário de pixels.

No método proposto em (BRADY; BOSSEN; MURPHY, 1997) e empregado no MPEG-4 (ISO/IEC, 2001), a codificação de forma dos objetos é feita através de codificação aritmética baseada em contexto (CAE - *Context-based Arithmetic Encoding*). Para isso, a imagem é dividida em blocos de $16 \times 16$ pixels, e cada bloco pode ser codificado de um modo diferente. Cinco modos são definidos:

**Transparente:** todos os pixels do bloco estão fora da máscara do objeto;

**Opaco:** todos os pixels do bloco estão dentro da máscara do objeto;

**Intra:** bloco codificado no modo **intra**;

**Inter:** bloco codificado no modo **inter**, com compensação de movimento;

**Sem atualização:** nenhuma correção é aplicada ao bloco após a compensação de movimento.

Nos modos **transparente** e **opaco**, nenhuma informação adicional necessita ser enviada ao decodificador. Para os modos **inter** e **sem atualização**, um vetor de movimento é transmitido. Este vetor é obtido através de uma técnica de *block matching* baseada apenas em informação de forma, e é diferente do vetor de movimento utilizado para a codificação

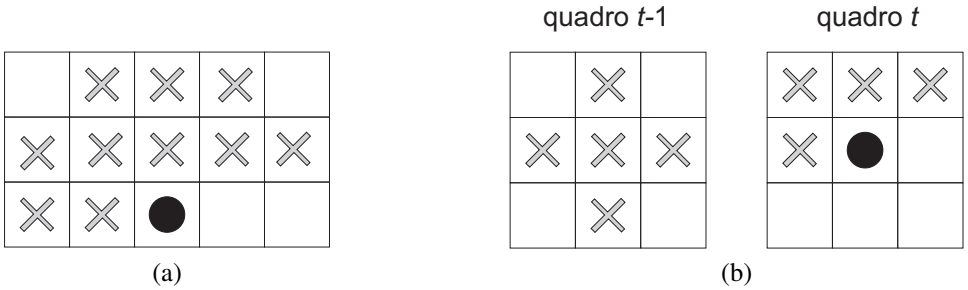



Figura 3.1: *Templates* utilizados no algoritmo CAE no MPEG-4: modo **intra** (a) e modo **inter** (b). Os círculos identificam os pixels de referência e as cruzes identificam os pixels dos contextos.

de textura. Para os blocos nos modos **intra** e **inter**, ainda são enviados bits da codificação aritmética baseada em contexto. Os blocos codificados no modo **intra** utilizam o *template* de 10 pixels ilustrado na Figura 3.1(a), totalizando 1024 contextos, enquanto os blocos codificados no modo **inter** utilizam o *template* de 9 pixels ilustrado na Figura 3.1(b), totalizando 512 contextos. As probabilidades de cada contexto foram obtidas em um conjunto pré-definido de vídeos para treinamento.

Uma *quad-tree* é uma árvore em que cada nodo tem quatro filhos, e representam uma partição de uma determinada região 2-D retangular em quatro quadrantes. *Quad-trees* particionam recursivamente regiões até que um determinado critério de homogeneidade seja satisfeito. No contexto da representação de máscaras 2-D de objetos, um critério de homogeneidade pode ser o de que uma sub-região (bloco) deve conter apenas pixels com um único valor (‘0’ ou ‘1’). Assim, o processo de construção da *quad-tree* pode ser inicializado a partir de uma região que contém todos os pixels da máscara do objeto (*bounding box*). Caso esta região contenha pixels internos e externos à máscara, a região é dividida em quatro sub-regiões, e cada uma destas sub-regiões é testada segundo este critério. O processo termina quando todas as regiões da *quad-tree* contiverem apenas um valor de máscara. Um exemplo de divisão espacial de uma máscara de objeto por uma *quad-tree* utilizando este critério é ilustrado na Figura 3.2: a máscara original é mostrada na Figura 3.2(a), e os quatro primeiros níveis de divisão são mostrados na Figura 3.2 de (b) a (e), e a *quad-tree* final, com 6 níveis, é mostrada na Figura 3.2(f).

O nodo raiz da árvore representa a região original inteira, com todos os pixels internos e externos à máscara, enquanto nodos em outros níveis representam sub-regiões da região original. Um nodo que representa uma região que não satisfaz o critério de homogeneidade terá quatro filhos. Nodos que representam regiões que satisfazem o critério de homogeneidade não terão nenhum filho. Uma *quad-tree* que descreve a máscara binária de um objeto pode ser representada por seus nodos, através de três símbolos. Suponha que uma região homogênea apenas com pixels externos à máscara do objeto seja representada por ‘0’, uma região homogênea apenas com pixels internos à máscara seja representada por ‘1’, e uma região heterogênea (com pixels internos e externos) seja representada por ‘2’. Os primeiros quatro níveis da *quad-tree* que descreve a máscara da Figura 3.2 estão ilustrados na Figura 3.3. A *quad-tree* pode ser representada pela sequência de símbolos dos nodos encontrados ao percorrer a árvore segundo uma ordem pré-estabelecida.

Kassim e Zhao (KASSIM; ZHAO, 2000) utilizaram codificação implícita de *quad-trees* através de uma extensão do algoritmo EZW (*Embedded Zerotree Wavelet*), chamado de SA-EZW (*Shape-Adaptive Embedded Zerotree Wavelet*) para codificar formas

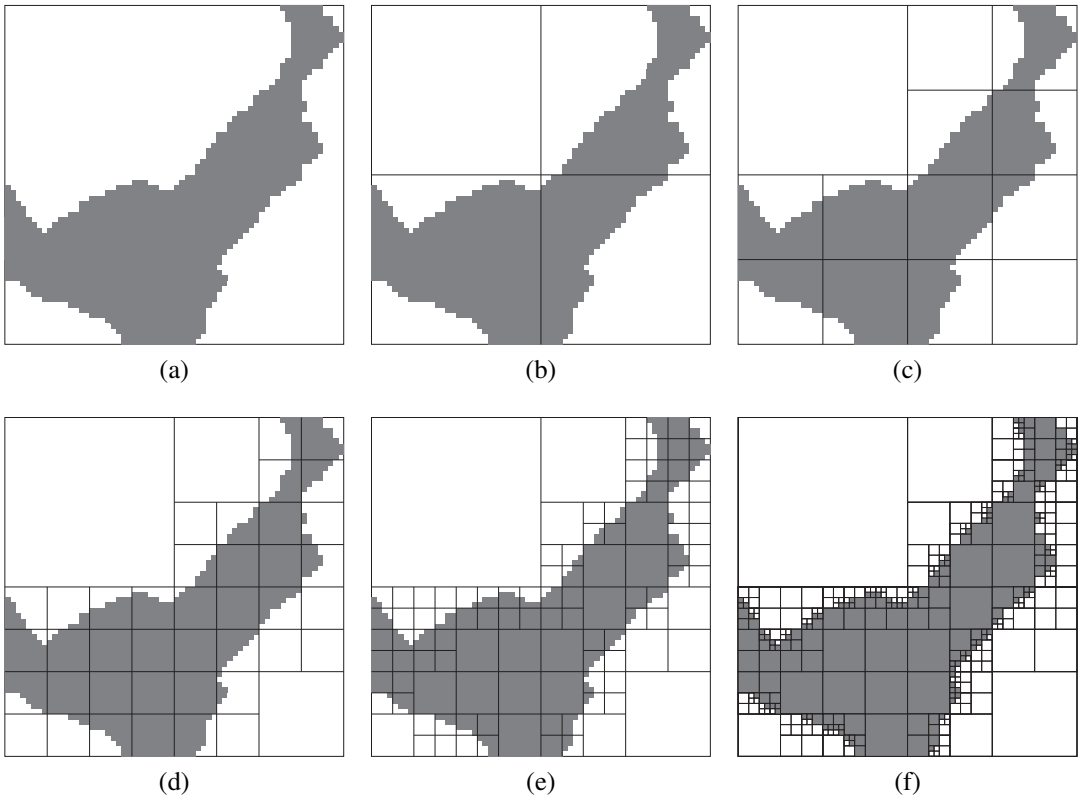



Figura 3.2: Exemplo de uma *quad-tree* que descreve a forma de um objeto: máscara original (a), primeiro (b), segundo (c), terceiro (d) e quarto (e) níveis da *quad-tree* e a *quad-tree* completa com seis níveis (f).

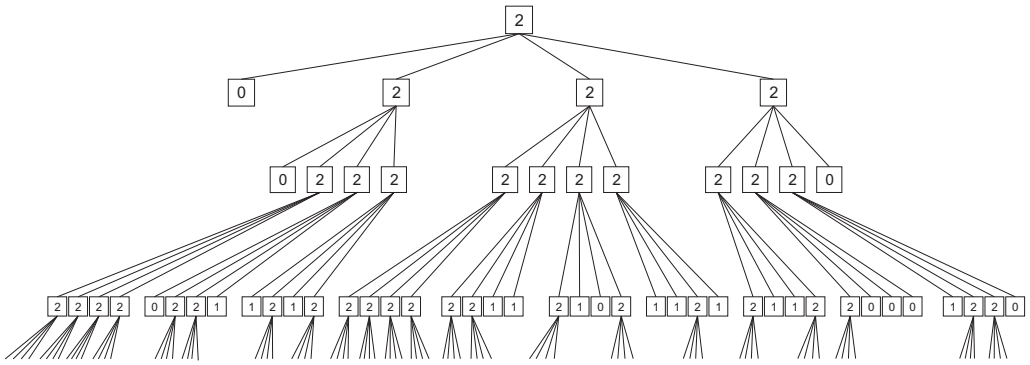


Figura 3.3: Quatro primeiros níveis da *quad-tree* do exemplo da Figura 3.2.

de objetos, sendo essa codificação feita no domínio da transformada. O EZW codifica coeficientes de uma transformada *wavelet* através de uma estrutura hierárquica (árvore) com conjuntos de sub-bandas de resoluções variáveis, e utiliza símbolos que indicam se um determinado nodo ou sub-árvore contém coeficientes de valores significativos para um dado nível de quantização, e se estes valores são positivos ou negativos. O SA-EZW inclui ainda um símbolo que indica se um determinado nodo ou sub-árvore contém coeficientes que se encontram dentro da máscara do objeto ou não. Uma única transformada *wavelet* é computada para toda a imagem, independente do número de objetos do vídeo, e a transmissão de objetos individuais se dá pela seleção dos coeficientes. Porém, alguns coeficientes podem estar relacionados a mais de um objeto, já que resultam de um processo de filtragem. Ainda, para um único objeto, devem ser codificados mais coeficientes do que a quantidade de pixels da máscara.

Assim como é feito no MPEG-4 (ISO/IEC, 2001) para a codificação de formas, o método proposto por Shen *et al.* (SHEN; FRATER; ARNOLD, 2008) divide a máscara binária em blocos de $16 \times 16$ pixels, e os classifica como blocos transparentes (composto apenas de pixels externos ao objeto), opacos (composto apenas de pixels internos ao objeto) e de fronteira (composto por pixels internos e externos ao objeto). Porém, ao contrário do MPEG-4 que codifica os blocos de fronteira diretamente através do algoritmo CAE, o método proposto por Shen *et al.* constrói uma *quad-tree* para cada bloco de fronteira. Depois, para cada nodo da *quad-tree*, o método verifica se o custo de codificar a região que este nodo representa através de sua sub-árvore é melhor ou pior que a codificação através do algoritmo CAE. Os contextos utilizados pelo algoritmo CAE são formados utilizando sub-blocos vizinhos e também de níveis maiores na hierarquia da *quad-tree*. Para cada bloco, são escolhidos os pontos de poda ótimos da *quad-tree*, de forma que as regiões representadas pelas sub-árvores podadas são codificadas através do algoritmo CAE.

Shamim e Robinson (SHAMIM; ROBINSON, 2002) propuseram um método para a codificação de bordas através de árvores binárias assimétricas. Estas árvores são utilizadas para a representação do interior dos objetos em um único quadro, e codifica as formas de modo hierárquico. A idéia básica do algoritmo é a de que blocos retangulares da imagem são recursivamente subdivididos até que eles contenham apenas pixels de borda ou apenas pixels de interior. A cada passo, o método divide um bloco em dois sub-blocos se este possuir tanto pixels de borda como pixels de interior. A posição em que o bloco é cortado é ajustada de forma a maximizar o tamanho do sub-bloco sem pixels de borda. Esta árvore pode ser codificada de forma eficiente e proporcionar uma representação simples para os contornos de objetos em um determinado quadro da sequência. Porém, como os próprios autores ressaltam, isto ainda demanda uma quantidade de dados considerável, e um esquema baseado em múltiplos quadros (se valendo da redundância temporal) pode resultar em códigos mais eficientes.

Código de cadeia (*chain code*) é uma abordagem para representação de contornos em imagens. O princípio básico de um código de cadeia na representação da forma de um objeto é codificar separadamente cada componente conexo através de suas bordas (localidades entre os pixels da imagem). Para cada componente conexo, um ponto da borda é selecionado e transmitido. O codificador então se move ao longo da borda e, a cada passo, transmite um símbolo representando a direção deste movimento. O processo continua até que o codificador retorne à posição inicial, de forma que o componente conexo esteja completamente descrito.

Diferentes códigos de cadeia foram propostos, cada um com sua semântica própria.

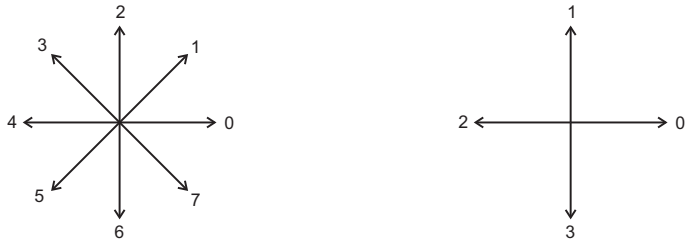



Figura 3.4: Código de cadeia de Freeman em oito (a) e em quatro (b) direções.

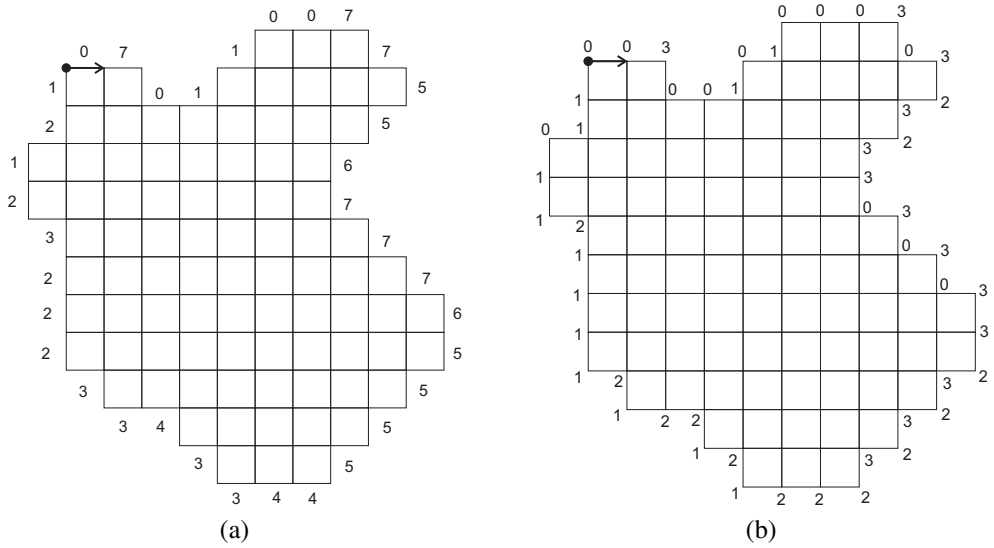



Figura 3.5: Exemplo do uso do código de Freeman com oito (a) e quatro (a) direções, para a máscara de um objeto simples.

O mais popular, e no qual todos os outros se baseiam, é o código de cadeia de Freeman (FREEMAN, 1961). No código de Freeman, à medida que o codificador se move pela borda, símbolos que representam os ângulos dos movimentos são transmitidos a cada pixel. Para a representação mais usual de imagens digitais, que é através de uma grade retangular de pixels, temos oito direções possíveis, conforme ilustrado na Figura 3.4(a). Neste caso, a direção de cada movimento é codificada como um número $\{i|i = 0, 1, 2, \ldots, 7\}$, denotando um ângulo de $i \times 45^\circ$ a partir da direção leste (esquerda) no sentido anti-horário. Como temos oito direções possíveis, cada direção pode ser representada por 3 bits. Uma versão de quatro direções do código de Freeman também pode ser utilizada (Figura 3.4(b)). Com isso, a direção de cada movimento pode ser codificada com um número $\{i|i = 0, 1, 2, 3\}$, denotando um ângulo de $i \times 90^\circ$ a partir da direção leste no sentido anti-horário. Com quatro direções possíveis, 2 bits são suficientes para representar cada direção. Um exemplo do uso dos códigos de Freeman com 4 e 8 direções para a representação de uma forma simples é mostrado na Figura 3.5.

Posteriormente, diversas modificações do código de cadeia de Freeman foram propostas. Em (LIU; ZALIK, 2005) foi proposto um método de código de cadeia que usa codificação de Huffman. Este trabalho explora a propriedade de que, geralmente, um elemento do código de Freeman é igual ao seu predecessor imediato ou ao antecessor deste. Ou seja, a probabilidade de mudanças bruscas de direção do contorno na distância de um

pixel é muito baixa. Se valendo disso, os autores analisaram os padrões de contorno de mais de 1000 curvas, e obtiveram uma distribuição de probabilidade das diferenças de ângulos entre segmentos contíguos de contorno, e utilizaram essas probabilidades como base para a geração do código de Huffman.

Aproximações poligonais são uma alternativa para a representação do contorno de um objeto. Segundo esta abordagem, cada componente conexo da máscara binária pode ser aproximado por um polígono. Com isso, basta armazenar as coordenadas dos vértices do polígono para representar a forma do objeto. Dependendo da complexidade da máscara do objeto, pode ser muito custoso representar a forma exata do objeto por aproximção poligonal, já que pode ser necessário incluir muitos vértices. Por esse motivo, na maioria das vezes a aproximação poligonal é utilizada para a codificação com perdas das formas.

Bandyopadhyay e Kondi (BANDYOPADHYAY; KONDI, 2005) abordaram a aproximação poligonal da forma de um objeto como um problema de otimização, que consiste da minimização de uma função que estima o custo da codificação dos vértices do polígono. Esta minimização está sujeita a um limite de tolerância para o erro máximo da aproximação. Os autores reduziram este problema ao de encontrar o menor caminho em um grafo acíclico dirigido (DAG - *Directed Acyclic Graph*).

Um método de representação e codificação da forma de um objeto dependente da imagem foi proposto por Luo (LUO, 2005). A idéia por trás deste método é que, como a forma de um objeto é utilizada para descrever objetos contidos na imagem, ela é altamente correlacionada com a imagem. Para explorar esta correlação, um método semi-automático de detecção de bordas foi criado. O contorno de um objeto é então codificado a partir de uma aproximação poligonal bastante simples do contorno do objeto, e de parâmetros utilizados pelo método de detecção de bordas para detectar a borda exata do objeto. Ou seja, o contorno do objeto não é codificado diretamente pelas coordenadas de seus pixels, mas sim pelos passos necessários para gerar ou detectar o contorno a partir da imagem. No entanto, este método se restringe a aplicações onde as formas dos objetos funcionam como meta-dados para descrição de imagens e/ou vídeos, já que muitas das funcionalidades desejáveis dos métodos de codificação baseados em objetos não são viáveis em representações dependentes da imagem, tais como transmissão individual de objetos, escalabilidade em nível de objeto, entre outras.

## 3.2 Representação e Codificação do Movimento dos Objetos

No contexto da codificação de vídeos, o movimento dos objetos é representado a partir dos parâmetros do modelo de movimento empregado. Os modelos de movimento mais comuns são o translacional e a transformação afim (HAN; WOODS, 1998). Veremos também métodos de representação de movimento baseados em malhas poligonais e métodos híbridos, que se valem de diferentes modelos de movimento para aumentar a eficiência da codificação.

Uma questão bastante delicada na codificação da informação de movimento dos objetos é a quantização dos parâmetros do modelo. Frequentemente os modelos de movimento dos objetos são estimados a partir de parâmetros representados por números de ponto flutuante. É possível se obter ganhos na compressão dos dados através da quantização destes parâmetros. Deve-se notar, no entanto, que a quantização destes parâmetros influencia no processo de predição das texturas (e, em alguns casos, dos contornos) dos objetos. Assim, o aumento no custo de codificação do erro de predição não pode ser maior do que o ganho obtido na quantização dos parâmetros do modelo.

Segundo o modelo translacional, o deslocamento de um ponto $(x, y)$ de um objeto $k$ é dado pelas componentes:

$$u_t^k(x, y) = d_x$$

$$v_t^k(x, y) = d_y.$$

Assim, o deslocamento de todos os pontos do objeto $k$ podem ser definidos apenas pelas componentes horizontal e vertical de movimento $d_x$ e $d_y$, respectivamente.

Em (SHAMIM; ROBINSON, 2002) é proposto um método de codificação de vídeos baseado em objetos visando altas taxas de compressão. Para isso, é utilizado um modelo simples de movimento translacional, onde os vetores de movimento são codificados com um número de bits variável, oferecendo escabilidade entre informação de movimento e informação de resíduo.

Segundo o modelo de transformação afim, o deslocamento de um ponto $(x, y)$ pertencente a um objeto $k$ é dado por:

$$u_a^k(x, y) = a_1^k + a_2^k x + a_3^k y$$

$$v_a^k(x, y) = a_4^k + a_5^k x + a_6^k y.$$

Neste caso, o movimento do objeto $k$ pode ser representado pelos parâmetros $\{a_1^k, \ldots, a_6^k\}$.

Quando o modelo afim é utilizado para representar o movimento dos objetos, frequentemente os parâmetros são codificados através de um código de tamanho fixo (HAN; WOODS, 1998).

Representar o movimento de uma região e/ou objeto pelo modelo afim implica na codificação dos seus 6 coeficientes. Quando o vídeo é composto de muitos objetos pequenos, isso gera um gasto considerável em termos de bits. Assim, métodos que utilizam diferentes modelos de movimento, adaptados a diferentes situações, podem economizar bits representando regiões simples com poucos parâmetros, e utilizando modelos mais complexos apenas quando estes são realmente necessários. Em regiões pequenas de um quadro, normalmente é possível aproximar o movimento apenas por uma translação, com pequenos erros. Mas a aproximação pelo modelo translacional tende a se degradar (e, consequentemente a eficiência da codificação) quando aplicada a grandes regiões.

Uma forma comum de se representar/compensar o movimento em um vídeo é através de malhas poligonais. Dado um quadro do vídeo, malhas podem ser utilizadas para dividir o domínio espacial em diversos polígonos simples (normalmente triângulos ou quadriláteros). O movimento é estimado nos vértices da malha pelo codificador, e os vetores de deslocamento dos vértices são armazenados e/ou enviados ao decodificador. Para efetuar a compensação de movimento, o deslocamento na localidade de cada pixel pode ser aproximado pela interpolação dos vetores de movimento dos vértices do polígono onde o pixel está contido.

As malhas podem ser regulares ou irregulares. Malhas irregulares geralmente são geradas com base no conteúdo da imagem, seja a partir de uma segmentação previamente computada, ou a partir de informação obtida diretamente dos pixels da imagem. A atualização das malhas varia muito de acordo com a abordagem. Em geral, no caso de malhas regulares, a cada quadro do vídeo é inicializada uma nova malha regular. Já no caso das malhas irregulares, os vértices costumam ser mantidos de um quadro para outro, apenas deslocados de acordo com o movimento no ponto. À medida que a malha vai se deformando/degenerando, modificações na estrutura podem ser feitas, como por exemplo, inserção e remoção de vértices, permutação de arestas, ou uma nova triangulação.

A triangulação de Delaunay com restrições foi utilizada em (GOKCETEKIN et al., 2000) para a compensação de movimento e a representação da forma dos objetos. A triangulação utiliza vértices que são criados de tal forma que as arestas dos triângulos se alinhem com as bordas de intensidade, e a densidade de vértices é proporcional à intensidade do movimento local. A topologia inicial da malha é dada por uma triangulação de Delaunay com restrições, onde os segmentos das bordas são utilizados como restrições. A partir daí, o movimento dos vértices é computado, e vértices são criados e removidos de triângulos de borda (triângulos cujos vértices apresentam movimentos distintos), e a triangulação é re-feita.

Os métodos híbridos se valem de diferentes técnicas de representação de movimento, e durante o processo de codificação avaliam qual técnica é a mais adequada para um determinado objeto do vídeo. Dada a segmentação do vídeo e, em alguns casos, uma descrição do conteúdo do vídeo, cada objeto pode ser classificado quanto às suas propriedades (como *background* estático, *background* rígido com movimento de câmera, objetos rígidos no *foreground*, objetos flexíveis no *foreground* com movimento coerente, etc.). Essa classificação pode ser utilizada para aplicar diferentes predições temporais em objetos com diferentes propriedades. Por exemplo, para objetos flexíveis, pode ser mais eficiente codificar e transmitir campos vetoriais densos de movimento; para objetos rígidos com movimento global, possivelmente é melhor transmitir parâmetros de movimento global; para um *background* com movimento de câmera, pode ser mais interessante estimar e transmitir parâmetros de câmera.

Métodos de codificação baseados em *sprites* são muito utilizados na codificação de vídeos, tendo sido inclusive incluído no padrão MPEG-4 (ISO/IEC, 2001). Técnicas baseadas em *sprites* (LU; GAO; WU, 2003; DASU; PANCHANATHAN, 2004; CHEUNG et al., 2008) (também referido como mosaico de vídeo (IRANI et al., 1996)) consistem na codificação de uma imagem estática que contém todos os pixels de um objeto que aparecem ao menos uma vez ao longo da sequência de quadros. Esta imagem pode ser utilizada para a codificação preditiva de um objeto em diferentes quadros. Técnicas baseadas em *sprites* são eficazes na codificação de um *background* rígido visto da mesma posição ao longo do vídeo. Se essa condição falha, a predição é ineficiente. Por isso muitas técnicas utilizam *sprites* para codificar o *background*, enquanto outras técnicas são utilizadas para objetos no *foreground*.

## 3.3 Representação e Codificação da Textura dos Objetos

A representação das texturas na codificação de objetos em vídeos normalmente ocupa a maior parte do *bitstream*. Daí a importância de uma codificação eficiente das texturas. Métodos utilizados na codificação de texturas de imagens e vídeos já constituem uma área de pesquisa relativamente madura. A compressão dos dados é obtida explorando propriedades como a alta correlação espacial e temporal, comuns em vídeos usuais. A correlação temporal geralmente é explorada ao se efetuar a compensação de movimento, substituindo a informação de cor original dos pixels por resíduo preditivo. Já a correlação espacial normalmente é explorada aplicando-se uma transformada aos dados, de tal forma que a representação no domínio da transformada concentre sua energia em alguns poucos coeficientes. A compressão então é alcançada através da quantização destes coeficientes. Dentre as transformadas que possuem a propriedade de alta compactação de energia, as mais utilizadas na codificação de imagens e vídeos são a DCT (*Discrete Cosine Transform*) e a DWT (*Discrete Wavelet Transform*). Por serem transformadas já

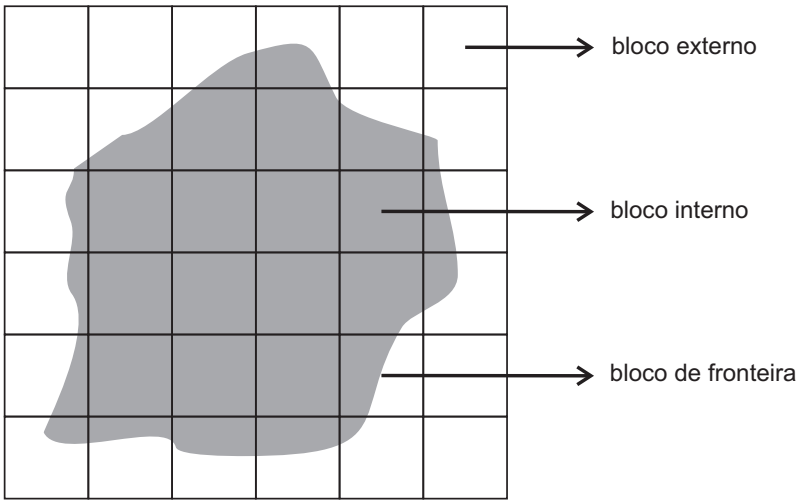



Figura 3.6: Exemplo de uma imagem dividida em blocos quadrados, contendo a máscara de um objeto. Podemos ter três tipos de blocos: blocos internos, blocos externos e blocos de fronteira.

muito estudadas e amplamente utilizadas em métodos de codificação de vídeos de primeira geração, normalmente as metodologias de codificação de texturas de objetos são adaptações destas transformadas para suportar regiões de formas arbitrárias.

A DCT é uma transformada amplamente utilizada nos métodos de codificação de vídeos baseados em blocos. Normalmente aplicada a blocos quadrados de tamanho fixo, a transformada do cosseno é computacionalmente eficiente e, quando aplicada a imagens usuais, tende a concentrar grande parte da energia em poucos coeficientes, fornecendo uma representação compacta.

O algoritmo SA-DCT (*Shape-Adaptive Discrete Cosine Transform*) (SIKORA; MAKAI, 1995) é um algoritmo de baixa complexidade que é uma simples extensão da DCT para a codificação de regiões de imagens com formas arbitrárias. Assim como nas abordagens de codificação baseadas na DCT, uma imagem qualquer (ou um quadro de um vídeo) contendo um objeto com forma arbitrária é dividido em blocos quadrados de mesmo tamanho. A Figura 3.6 mostra um exemplo de uma máscara binária de um objeto, e a divisão da imagem em blocos. Podemos ter três tipos de blocos:

**Blocos externos:** são blocos que não contém nenhum pixel da máscara do objeto. Para estes blocos não é necessário codificar nenhum dado;

**Blocos internos:** são blocos em que todos os pixels pertencem à máscara do objeto. Estes blocos são codificados através da DCT tradicional;

**Bloco de fronteira:** são blocos que contém pixels internos e externos à máscara do objeto. Para estes blocos, uma estratégia especial é adotada, conforme detalhado a seguir.

Para os blocos de fronteira, o algoritmo SA-DCT utiliza um conjunto de funções-base DCT pré-definidas e é executado separadamente ao longo das dimensões horizontal e vertical da imagem. A Figura 3.7 ilustra este método para um bloco de $8 \times 8$ pixels, contendo uma máscara de um objeto (Figura 3.7(a)). Sendo a máscara do objeto conhecida, primeiramente o comprimento $N, 0 < N < 9$ de cada coluna $j, 0 < j < 9$ é computado (ou seja, o número de pixels no interior do objeto em cada coluna do bloco). As colunas são

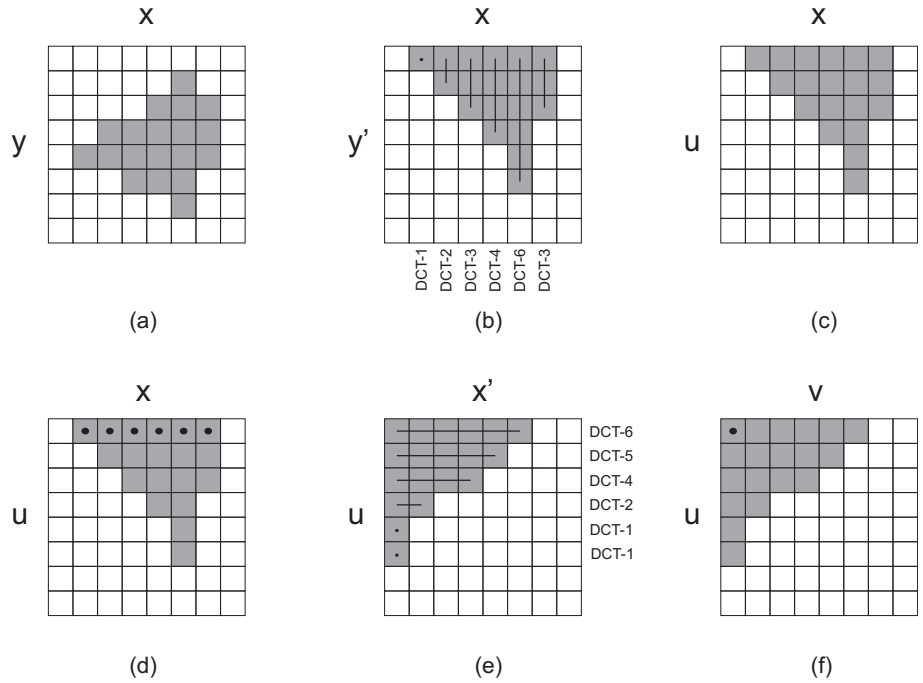



Figura 3.7: Exemplo da aplicação da SA-DCT em um bloco de $8 \times 8$ pixels.

então deslocadas e alinhadas à borda superior do bloco (Figura 3.7(b)). Dependendo do comprimento $N$ de cada coluna, uma matriz de transformação DCT-N:

$$\text{DCT-N}(p,k) = c_0 \cdot \cos\left(p\left(k+\frac{1}{2}\right)\cdot\left(\frac{\pi}{N}\right)\right), \quad k,p = 0 \rightarrow N-1, \tag{3.1}$$

contendo um conjunto de $N$ DCT-N vetores-base é selecionada. Sendo que $c_0 = \sqrt{1/2}$ se $p = 0$, caso contrário $c_0 = 1$, e $p$ representa o $p$-ésimo vetor-base DCT. Os $N$ coeficientes DCT verticais $c_j$ para cada coluna de valores $x_j$ são calculados utilizando a fórmula:

$$c_j = \left(\frac{2}{N}\right)\cdot \text{DCT-N} \cdot x_j. \tag{3.2}$$

Por exemplo, na Figura 3.7(b), a coluna mais à direita é transformada utilizando-se vetores-base DCT-3. Após a SA-DCT ser aplicada na direção vertical, os coeficientes DC para cada coluna se encontram na primeira linha do bloco (Figura 3.7(d)). Para aplicar a SA-DCT na direção horizontal, o comprimento de cada linha é computado, e as linhas são deslocadas e alinhadas à borda esquerda do bloco, e uma DCT horizontal adaptada ao tamanho de cada linha é calculada, novamente usando as Equações (3.1) e (3.2) (Figura 3.7(e)). Dessa forma, a transformada SA-DCT horizontal é aplicada a coeficientes da SA-DCT vertical de mesmo índice (ou seja, todos os coeficientes DC são agrupados e transformados na dimensão horizontal). A Figura 3.7(f) mostra a localização dos coeficientes DCT resultantes.

Note que o número de coeficientes DCT é idêntico do número de pixels dentro da máscara do objeto. Ainda, os coeficientes estão localizados em posições comparáveis às de um bloco DCT $8 \times 8$ convencional. O coeficiente DC está localizado no canto superior esquerdo do bloco e, dependendo da forma da máscara, o restante dos coeficientes se concentram em torno do coeficiente DC. Partindo do presuposto que a forma do objeto

é transmitida/decodificada primeiro, o decodificador pode executar a SA-DCT inversa a partir da operação:

$$x_j = \text{DCT-N}^T \cdot c_j^*,$$

nas direções horizontal e vertical. O coeficiente $c_j^*$ indica um valor possivelmente quantizado.

O problema que temos ao deslocar pixels dentro do bloco antes de aplicar a transformada, é que parte da correlação espacial existente entre pixels vizinhos é perdida, levando a uma menor eficiência da codificação. Por isso, muitos métodos de codificação utilizam técnicas de preenchimento para tratar blocos de fronteira, atribuindo valores para os pixels que estão fora da máscara do objeto, antes de aplicar a transformada. Esse preenchimento tem por objetivo possibilitar que o mesmo algoritmo da DCT utilizado em blocos internos seja utilizado em blocos de fronteira, sem deslocamento de pixels dentro do bloco, e de forma que os dados no domínio da transformada sejam eficientemente codificados (ou seja, apresentem alta concentração de energia em poucos coeficientes). Kaup (KAUP, 1999) propôs um método de extrapolação de baixa complexidade que resulta em coeficientes DCT com grande concentração de energia na região das baixas frequências. Em (MOON; KWEON; KIM, 1999), após o preenchimento de pixels externos ao objeto, um esquema de fusão de blocos de fronteira é utilizado para reduzir o número de blocos codificados. Isto é feito rotacionando os blocos e unindo pares de blocos por sobreposição, desde que não haja sobreposição dos pixels da máscara do objeto. Contudo, em maior ou menor grau, técnicas de preenchimento invariavelmente geram distorções e artefatos, já que introduzem informações que não existiam na imagem original.

Uma transformada *wavelet* é a decomposição de uma função por *wavelets*, que são funções cuja área total sobre a curva é zero, e sua energia é finita, representadas em diversas escalas diferentes. Ou seja, uma *wavelet* é uma função oscilatória cuja energia está localizada em uma determinada região. A transformada discreta *wavelet* (DWT) vem cada vez mais substituindo a DCT na codificação e imagens e vídeos. Isso se deve principalmente ao fato de que a DWT consegue representar características visuais importantes de forma compacta no domínio da transformada, sem apresentar artefatos de blocos mesmo com altas taxas de compressão, resultando em imagens decodificadas com melhor qualidade subjetiva.

A aplicação da DWT para a codificação de regiões com formas arbitrárias em duas dimensões, assim como no caso da DCT, exige que artifícios sejam feitos para que os coeficientes resultantes da transformada possam ser eficientemente codificados, ao mesmo tempo que os efeitos de borramento e artefatos na imagem reconstruída sejam minimizados.

A abordagem conhecida como *lifting scheme* (DAUBECHIES; SWELDENS, 1998) vem sendo bastante utilizada na codificação de imagens e vídeos, e sua principal característica é que todas as construções são derivadas no domínio espacial, contrastando com a abordagem tradicional onde isso é feito no domínio frequência. Entre as vantagens do *lifting scheme*, estão a possibilidade de se obter implementações mais rápidas, a possibilidade de se sobrescrever os dados originais pelos coeficientes da transformada na mesma localização da memória (abordagem chamada de *in-place*), a possibilidade de se obter a transformada inversa diretamente através do processo inverso da transformada direta, e transforma um sinal com tamanho arbitrário, não necessitando possuir tamanho da forma $2^n$. Esta última vantagem é especialmente importante na codificação de imagens e vídeos adaptativa à forma, já que apresenta melhores resultados na representação de regiões próximas às bordas do que a abordagem tradicional, sem a necessidade de qualquer tipo de

artifício.

Em (LI; LI, 2000) foi proposta uma SA-DWT utilizando *lifting scheme* que resulta em um número de coeficientes no domínio da transformada igual ao número de pixels da região original com forma arbitrária, mantendo propriedades como a correlação espacial, localidade e auto-similaridade através das sub-bandas. A decomposição *wavelet* também é obtida aplicando-se uma transformada *wavelet* 1-D a segmentos de pixels do objeto, porém, ao contrário do que é feito na SA-DCT, os pixels não são deslocados para as bordas da imagem. Dado um segmento de pixels, existem duas formas possíveis de se sub-amostrar esses pixels para a aplicação dos filtros passa-baixa e passa-alta: sub-amostragem par e sub-amostragem ímpar. Ao utilizar sub-amostragem par para o filtro passa-baixa e sub-amostragem ímpar para o filtro passa-alta, o primeiro coeficiente será passa-baixa, o segundo passa-alta, o terceiro passa-baixa, e assim por diante. Já com sub-amostragem ímpar para o filtro passa-baixa e par para o passa-alta, o primeiro coeficiente será passa-alta. Para a seleção do tipo de sub-amostragem (par ou ímpar) para um determinado segmento de pixels, dois referenciais são possíveis:

**Referencial local:** usa como referência a primeira coluna (ou linha) do segmento;

**Referencial global:** usa como referência a primeira coluna (ou linha) da imagem.

Ao utilizar referencial local, todos os segmentos de pixels terão o mesmo tipo de sub-amostragem (par ou ímpar), enquanto que ao utilizar referencial global, segmentos que começam em colunas (ou linhas) pares utilizam um tipo de amostragem diferente dos segmentos que começam em colunas (ou linhas) ímpares. Os autores argumentam que sub-amostragem com referencial local geralmente resulta em maior eficiência na codificação dos coeficientes, enquanto que sub-amostragem com referencial global normalmente resulta em maiores ganhos no processamento de sinais.

O algoritmo da SA-DWT (LI; LI, 2000) pode ser descrito da seguinte forma:

1. Identificar a primeira linha que contém pixels dentro da máscara do objeto;

2. Para cada linha, identificar o primeiro segmento de pixels consecutivos;

3. Aplicar a transformada *wavelet* 1-D adaptada ao tamanho do segmento, com uma estratégia de sub-amostragem adequada;

4. Os coeficientes passa-baixa são colocados na linha correspondente na banda passa-baixa, enquanto os coeficientes passa-alta são colocados na linha correspondente na banda passa-alta, conforme ilustrado nas Figuras 3.8(a) e 3.8(b) para um esquema de sub-amostragem global, onde linhas contínuas delimitam pixels internos à máscara do objeto e linhas pontilhadas delimitam pixels externos;

5. Execute as operações acima para o próximo segmento de pixels consecutivos;

6. Execute as operações acima para a próxima linha de pixels;

7. Execute as operações acima para cada coluna de coeficientes passa-alta e passa-baixa do objeto;

8. Execute as operações acima para os coeficientes passa-baixa e passa-alta do objeto até que o nível de decomposição *wavelet* desejado seja alcançado.


Figura 3.8: Exemplos de decomposição utilizando sub-amostragem com referencial global: sub-amostragem par (a) e sub-amostragem ímpar (b).

Martin *et al.* (MARTIN; LUKAC; PLATANIOTIS, 2006) propuseram um método semelhante ao proposto em (LI; LI, 2000) para a codificação de coeficientes de uma SA-DWT, porém, empregam uma estratégia de codificação simultânea da forma e a textura dos objetos. Os autores aplicam transformadas *wavelet* 1-D ao longo das duas dimensões espaciais, utilizando sub-amostragem com referencial global. Isto vai contra a recomendação de Li *et al.* para aplicação da SA-DWT para fins de codificação, porém, essa escolha se justifica por manter uma representação sem ambiguidades da máscara do objeto no domínio da transformada. Os coeficientes da transformada são então codificados segundo um algoritmo baseado em SPIHT (*Set Partitioning in Hierarchical Trees*) (SAID; PEARLMAN, 1996), modificado para transmitir simultaneamente informação sobre a máscara do objeto[1].

Um método que utiliza uma única transformada *wavelet* para um quadro com diversos objetos foi proposto por Kassim e Zhao (KASSIM; ZHAO, 2000). Os coeficientes da transformada são codificados através do método progressivo EZW. A transmissão de um único objeto é feita enviando apenas os coeficientes que pertencem à máscara deste objeto no domínio da transformada. Porém, como os coeficientes são obtidos a partir de um processo de filtragem, o conjunto de coeficientes dentro da máscara de um objeto no domínio da transformada em diferentes níveis de decomposição correspondem ao objeto em particular e também a partes de regiões circundantes. Por isso, um método de preenchimento deve ser utilizado para preencher coeficientes vizinhos de forma a reduzir distorções próximas à borda. Além do mais, o número de coeficientes transmitidos normalmente será maior que o número de pixels da máscara do objeto no domínio espacial.

Alguns métodos foram propostos para a codificação espaço-temporal utilizando *wavelets* adaptativas à forma (MINAMI et al., 1999; HAN; WOODS, 1997; LIU; WU; NGAN, 2007; LIU; NGAN; WU, 2008). Minami *et al.* (MINAMI et al., 1999) empregaram uma versão 3-D modificada do SPIHT para codificar objetos com formas arbitrárias. Han e Woods (HAN; WOODS, 1997) propuseram um método de codificação de objetos através de uma decomposição *wavelet* espaço-temporal. Após estimar e segmentar o movimento entre quadros consecutivos, pixels correspondentes de um mesmo objeto em quadros distintos são conectados, gerando trajetórias de pontos. Os pixels de uma mesma trajetória são então filtrados ao longo da dimensão temporal, gerando uma decomposição *wavelet* 1-D. Logo após todas as trajetórias terem sido filtradas, uma decomposição *wavelet* 2-D é efetuada no domínio espacial. Devido à oclusão, movimento de câmera e movimento

[1]Conforme será visto no Capítulo 5, uma extensão 3-D do algoritmo proposto por Martin *et al.* (MARTIN; LUKAC; PLATANIOTIS, 2006) foi empregado no presente trabalho para codificar simultaneamente texturas e formas dos objetos ao longo de uma representação espaço-temporal. A versão 3-D deste algoritmo está detalhada na Seção 5.1.

dos objetos, nem todos os pixels são temporalmente conectados em trajetórias. Pixels não conectados são ignorados e reconstruídos por interpolação no decodificador. Uma abordagem semelhante foi proposta em (LIU; WU; NGAN, 2007) com uma estratégia que diminui os efeitos de borda causados por pixels não conectados, e em (LIU; NGAN; WU, 2008) foi incluída uma transformada *wavelet* direcional, que identifica a direção ideal para aplicação dos filtros, de forma a ter uma maior concentração de energia em menos coeficientes e, consequentemente, tornando a codificação mais eficiente.

## 3.4 Conclusões

Neste Capítulo foi apresentado o problema da codificação de vídeos baseada em objetos. Este paradigma de codificação de vídeos foi criado com o objetivo de prover uma representação mais natural, do ponto de vista humano, do conteúdo de uma cena contida em um vídeo, possibilitando que novas funcionalidades, tais como a transmissão progressiva, a manipulação de objetos, e a busca baseada em conteúdo sejam disponibilizadas, com uma codificação eficiente. Para isso, objetos são representados através de três componentes: forma, movimento e textura. Estes três componentes estão fortemente relacionados, e todos eles são sensíveis à segmentação do vídeo.

Diversas abordagens para a codificação de forma foram propostas, apresentando vantagens e desvantagens de acordo com as características da forma a ser codificada, bem como as exigências da aplicação (transmissão progressiva, com ou sem perdas, etc).

A escolha do método de representação do movimento dos objetos normalmente estabelece um compromisso entre qualidade da aproximação e custo de codificação. Modelos mais complexos aproximam melhor o movimento de objetos reais; no entanto, necessitam que mais parâmetros sejam codificados. Em alguns casos, modelos mais simples podem produzir resultados aceitáveis com menos parâmetros a serem codificados. A escolha da melhor opção depende do tipo de movimento de objetos existentes no vídeo. Modelos mais complexos são mais generalistas, porém, nem sempre o ajuste ao modelo é uma tarefa trivial.

A representação das texturas geralmente representa a parte mais significativa do *bitstream* de um objeto codificado. A compressão dos dados pode ser obtida ao se explorar a redundância temporal e espacial. A redundância temporal normalmente é explorada através da compensação de movimento, substituindo a informação da cor original dos pixels por resíduo preditivo. Daí vem a importância de uma representação de movimento eficiente: apesar de não ocupar uma parte significativa do *bitstream*, um bom ajuste dos dados ao modelo de movimento garante uma redução na quantidade de informação necessária para se caracterizar as texturas. Métodos de codificação de texturas de objetos baseados na DWT têm se mostrado superiores aos métodos baseados na DCT. Além do mais, coeficientes de transformadas *wavelet* podem ser codificados de forma eficiente através de métodos progressivos, tais como o EZW ou o SPIHT. Em particular, estes métodos podem ser combinados com a codificação simultânea da forma (MARTIN; LUKAC; PLATANIOTIS, 2006), permitindo a transmissão progressiva dos objetos, ao mesmo tempo que exploram a redundância da representação das máscaras dos objetos.

# 4 UMA PROPOSTA PARA A SEGMENTAÇÃO DE VÍDEOS EM REGIÕES DE MOVIMENTO COERENTE

Neste Capítulo, é apresentada uma nova abordagem para a segmentação de vídeos baseada em objetos, onde objetos são definidos como regiões não-sobrepostas (em nível de pixel) no domínio espaço-temporal. Considera-se que essas regiões mantém suas características espaciais e fotométricas ao longo do tempo. A abordagem proposta combina vantagens dos métodos densos e dos métodos baseados em feições, como descrito a seguir.

Inicialmente, correspondências ao longo do tempo de pontos esparsos (chamados de partículas) são computadas, de tal forma que padrões temporais longos de movimento possam ser identificados. No entanto, ao invés de computar correspondências de pontos independentes (como é feito em diversos métodos baseados em feições), partículas vizinhas são tratadas como se elas estivesse ligadas, reduzindo a chance de ocorrência de *outliers* e evitando o problema da abertura. Ainda, a densidade de pontos amostrados (ou seja, de partículas) é adaptativa, e distribuições mais densas de partículas são usadas em regiões onde a precisão é mais importante (por exemplo, em bordas de movimento), economizando computação sem negligenciar regiões homogêneas. Para encontrar as correspondências de partículas em uma sequência de vídeo, é utilizada a abordagem proposta por Sand e Teller (SAND; TELLER, 2006), que baseia-se em partículas que são representadas com precisão sub-pixel. Após as correspondências das partículas serem encontradas em todos os quadros do vídeo, partículas com movimentos semelhantes são agrupadas em cada quadro da sequência. Os agrupamentos individuais de partículas em nível de quadro, são depois agrupadas em conjuntos maiores de partículas associadas a quadros diferentes, seguindo uma estratégia do tipo *ensemble clustering*. Finalmente, uma representação densa dos quadros do vídeo (ou seja, uma representação a nível de pixel) do agrupamento final de partículas é obtida.

O método se segmentação proposto é geral no sentido de que não se baseia em modelos de movimento, não impõe restrições de trajetória e segmenta múltiplos objetos de formas arbitrárias, sem precisar conhecer o número de objetos *a priori*. Ao invés de bordas de movimento, a segmentação é guiada pelo comportamento consistente do movimento de pontos amostrados nos quadros do vídeo. Essa estratégia permite extrair túneis mais longos no domínio espaço-temporal. Além disso, não é necessário nenhum tratamento especial no caso de mudanças na topologia de objetos, ou caso novos objetos surjam ao longo do vídeo. O método proposto tem a capacidade potencial de gerar volumes de oclusão e volumes de exposição, já que padrões de movimento são descobertos e associados a cada região em movimento, e os voxels dos volumes espaço-temporais podem ser classificados como pertencentes aos volumes de objetos, volumes de oclusão ou volumes de

exposição.

A abordagem proposta gera uma representação simples da cena, adequada para a codificação de vídeos baseada em objetos, e também entrega uma partição temporalmente persistente e mais redundante da cena do que métodos diretos de segmentação de vídeo e estratégias de predição de movimento comumente utilizadas por métodos de codificação.

Uma versão preliminar do método de segmentação de movimento proposto neste Capítulo está descrita em (SILVA; SCHARCANSKI, 2007), enquanto uma versão mais completa foi publicada em (SILVA; SCHARCANSKI, 2010).

O restante do Capítulo está organizado da seguinte forma: a Seção 4.1 apresenta uma visão geral do método de segmentação proposto, separando-o em etapas e explicando como se dá a interface entre cada etapa. A abordagem utilizada para a estimativa das trajetórias de partículas é mostrada na Seção 4.2. A Seção 4.3 apresenta o método de agrupamento de trajetórias de partículas, enquanto a Seção 4.4 trata da extração de uma segmentação densa para sequências de vídeos, utilizando os resultados obtidos nas etapas anteriores.

## 4.1 Visão Geral do Método

A estrutura da abordagem proposta para a segmentação de movimento pode ser dividida em três partes principais:

1. Estimativa de Trajetórias de Partículas (ver seção 4.2);

2. Segmentação de Trajetórias de Partículas (ver seção 4.3);

3. Extração da Segmentação Densa (ver seção 4.4).

A primeira parte diz respeito à seleção e o rastreamento de um conjunto de pontos da cena (chamados de partículas). Este estágio recebe como entrada os quadros originais do vídeo, e retorna como saída um conjunto de partículas e suas respectivas trajetórias. Durante a estimativa de trajetórias de partículas, as partículas cujos locais dos pontos correspondentes sofrem oclusão na cena são eliminadas, e novas partículas são criadas em regiões que se tornam visíveis ao longo da sequência.

A segunda parte trata da segmentação das trajetórias de partículas, de tal forma que partículas que se movem de forma coerente são agrupadas em um mesmo conjunto. Este estágio recebe como entrada as trajetórias das partículas computadas no primeiro estágio, e retorna rótulos para todas as partículas como saída, representando a segmentação de movimento de regiões dos quadros do vídeo de acordo com as trajetórias de partículas. A segmentação de trajetórias de partículas pode ser dividida em quatro passos:

1. **Agrupamento (*clustering*) de vetores de movimento de 2 quadros**: neste passo, agrupamentos das partículas são feitos a partir de vetores de deslocamento obtidos de pares de quadros. Apenas quadros da vizinhança são considerados (1, 2 e 3 unidades de tempo de distância), e os agrupamentos são obtidos de forma independente para cada par de quadros. Para cada par de quadros considerado, a entrada para este passo é a posição das partículas em cada quadro, e a saída é um conjunto de agrupamentos e seus respectivos rótulos, válidos para cada par de quadros considerado;

2. **Agrupamento de conjuntos (*ensemble clustering*) de partículas**: aqui, todos os agrupamentos computados no passo anterior são processados simultaneamente para

produzir uma única divisão do conjunto inteiro de partículas em sub-conjuntos de partículas com movimento coerente, chamados de meta-grupos (*meta-clusters*); diversos conjuntos de rótulos de agrupamentos são utilizados como entrada para este passo, e um único conjunto de rótulos de segmentação (diferentes partículas em movimento coerente compartilham o mesmo rótulo de segmentação) é retornado como saída;

3. **Validação de meta-grupos**: neste passo, partículas que foram segmentadas no passo anterior são comparadas a protótipos de meta-grupos em termos de movimento e posição espacial, para detectar partículas incorretamente rotuladas e, quando isto ocorre, partículas são re-rotuladas. Um conjunto de rótulos de segmentação é utilizado como entrada, e um conjunto de rótulos de segmentação corrigidos é retornado como saída;

4. **Filtragem espacial**: neste passo, *outliers* são eliminados, assim como grupos de partículas adjacentes que não são significativos. Os rótulos de partículas são analisados espacialmente, e ligações entre as partículas são criadas para definir a adjacência espacial em cada quadro. Pequenos grupos de rótulos de partículas adjacentes que não são significativos são então re-atribuídos. Este passo utiliza como entrada um conjunto de rótulos de partículas, e retorna como saída um conjunto filtrado de rótulos de partículas.

A terceira e última parte do método proposto de segmentação é a extração de uma segmentação densa. Este estágio utiliza como entrada os quadros originais do vídeo, os rótulos da segmentação retornados pelo segundo estágio, assim como as posições das partículas computadas no primeiro estágio, e retorna como saída os rótulos de segmentação correspondentes para cada pixel de cada quadro do vídeo. Isto é equivalente à segmentação de um volume espaço-temporal em diversos túneis. A extração de segmentação densa é executada criando-se funções implícitas para cada partícula, baseadas em seus respectivos movimentos e posição espacial. Esta representação de segmentação de movimento através de túneis pode ser empregada para obter predições de movimento eficientes em aplicações de codificação de vídeos.

Todos os estágios da abordagem proposta são processados sequencialmente. Cada estágio é executado para o vídeo inteiro antes que a execução passe para o próximo estágio. Por isso, o método de segmentação de movimento proposto não pode ser utilizado em aplicações de tempo real, sem um particionamento do vídeo.

## 4.2 Estimativa de Trajetórias de Regiões Baseada em Partículas

Nesta Seção, será mostrado como as partículas são selecionadas em cada quadro do vídeo, e como elas são localizadas ao longo do vídeo. A abordagem proposta por Sand e Teller (SAND; TELLER, 2006) é utilizada aqui, com pequenas modificações.

Algumas propriedades importantes da estimativa de trajetória utilizada aqui serão delineadas a seguir. Primeiro, a estimativa das trajetórias de partículas não requer qualquer restrição quanto à suavidade temporal do movimento. Isto significa que é robusto a movimentos abruptos de câmeras, e descontinuidades no movimento dos objetos. Segundo, a densidade da amostragem das partículas é adaptativa, no sentido de que regiões com mais detalhes são amostradas com mais partículas, enquanto regiões homogêneas são amostradas com menos partículas. Assim, uma precisão mais alta da segmentação de movimento

pode ser obtida em regiões com mais informação de movimento, onde uma maior densidade é necessária, enquanto se economiza computação nas regiões com menos informação de movimento. Terceiro, informação de movimento pode ser inferida das partículas vizinhas, reduzindo o efeito do problema da abertura em regiões homogêneas, onde não existe informação de movimento suficiente. Estas propriedades sugerem que esta abordagem de estimativa de trajetórias de partículas é potencialmente adequada para a estimativa de padrões de movimento coerente de longa duração, com o movimento representado em uma granularidade adaptada localmente. Estas propriedades são vantajosas na codificação de vídeo, já que a redundância dos dados pode ser explorada em conjuntos de quadros que não são vizinhos imediatos, ao contrário de diversas outras abordagens que utilizam apenas pares adjacentes de quadros (SMITH; BRADY, 1995; WANG et al., 2004; XU; KABUKA; YOUNIS, 2004).

O método para estimativa de trajetórias de partículas utilizado neste trabalho pode ser dividido em cinco passos:

1. Cálculo do fluxo ótico;
2. Adição de partículas;
3. Propagação de partículas;
4. Remoção de partículas;
5. Otimização da localização das partículas.

No passo do cálculo do fluxo ótico, são estimados vetores de movimento para cada pixel de cada quadro do vídeo, em relação ao quadro subsequente, para uso nos passos posteriores. No passo de adição de partículas, novas partículas são criadas e inseridas em cada quadro do vídeo, localizadas nos pixels onde um critério de proximidade é satisfeito, sendo este critério de proximidade adaptativo ao conteúdo da cena. No passo de propagação de partículas, as posições ocupadas pelas partículas no quadro no tempo $t$ são propagadas para o quadro no tempo $t + 1$ utilizando informação de movimento das partículas, a qual é obtida do cálculo de vetores de fluxo ótico. Neste trabalho, foi utilizado o método proposto por Sand e Teller (SAND; TELLER, 2006) para o cálculo do fluxo ótico. No passo de remoção de partículas, são eliminadas partículas que, após a execução do passo de propagação, se tornam oclusas ou saem do campo de visão. No passo de otimização da localização de partículas, as posições de partículas são reajustadas minimizando uma função de energia. Estes reajustes das posições das partículas são utilizados para corrigir erros de estimativa de movimento de partículas que podem ocorrer no passo de propagação de partículas, evitando a propagação de tais erros para quadros subsequentes (ver (SAND; TELLER, 2006)). O algoritmo inicialmente adiciona partículas ao primeiro quadro da sequência. Logo após, o segundo quadro da sequência é processado, e o algoritmo executa a propagação, remoção e otimização da localização das partículas para o segundo quadro. Após, algumas partículas são adicionadas ao segundo quadro para satisfazer o critério de proximidade de partículas, e o processo continua com a propagação, remoção e otimização da localização das partículas, considerando as posições das partículas no terceiro quadro. Estes passos são executados na ordem mencionada acima para os quadros subsequentes, até que o fim do vídeo seja alcançado. Neste trabalho, estes passos são executados em uma única passagem pelo vídeo. Conforme sugerido por Sand e Teller (SAND; TELLER, 2006), mais passagens podem ser executadas em diferentes direções (para frente e para trás), visando uma melhor distribuição e localização

de partículas, ao custo de um maior esforço computacional. No entanto, não foi possível notar uma vantagem considerável na segmentação de movimento final ao executar várias passagens a partir de experimentos.

Os passos de cálculo do fluxo ótico, adição, propagação e otimização de localização de partículas utilizados neste trabalho são idênticos aos propostos por Sand e Teller (SAND; TELLER, 2006), exceto pelos valores dos parâmetros, conforme será visto mais adiante. Já o método de remoção de partículas empregado no presente trabalho introduz uma modificação no método proposto por Sand e Teller, de forma que o número de falsas oclusões possa ser reduzido. Isso porque observou-se experimentalmente que mesmo que poucos erros de detecção de trajetória de partículas possam ocorrer, os melhores resultados da segmentação são obtidos quando partículas têm vidas mais longas. Erros de trajetórias de partículas são tratados em separado neste trabalho (ver Seção 4.3.2).

### 4.2.1 Cálculo do fluxo ótico

No presente trabalho, o fluxo ótico é calculado para cada par de quadros consecutivos do vídeo, provendo uma estimativa do movimento instantâneo nas localidades dos pixels, informação esta que será utilizada em diversas etapas posteriores do processo de segmentação de vídeos proposto. O método para o cálculo do fluxo ótico proposto por Sand e Teller (SAND; TELLER, 2006) e apresentado aqui utiliza uma abordagem variacional com uma estrutura hierárquica (*coarse-to-fine*), em que estimativas de movimento são feitas inicialmente em resoluções menores das imagens, e estas estimativas são então propagadas e refinadas em resoluções mais altas, até que se obtenha a convergência na resolução mais alta (resolução original da imagem). Em cada resolução, o algoritmo executa os seguintes passos:

- Otimização dos vetores de movimento, a partir de uma função objetivo com dois termos: erro de projeção e suavidade (Seção 4.2.1.1);
- Identificação das regiões de oclusão utilizando divergência dos vetores de movimento e erro de projeção dos pixels (Seção 4.2.1.2);
- Regularização dos vetores de movimento usando um filtro bi-lateral com suporte a oclusão (Seção 4.2.1.3).

A sequência de resoluções é obtida reduzindo-se recursivamente a resolução original por um fator escala $\eta = 0.9$, até alcançar o limite inferior de 0.05, resultando em 29 níveis de resolução. Após o redimensionamento da imagem, esta é filtrada por um *kernel* gaussiano com desvio padrão $\sigma = 1$ (SAND; TELLER, 2006).

Embora a abordagem empregada aqui funcione para imagens em tons de cinza e possa ser estendida para imagens de múltiplas bandas, sua aplicação é direcionada para imagens a cores. Assumimos como entrada para o algoritmo imagens no formato RGB. Para o cálculo do *optical flow*, a imagem RGB $I$ é substituída por uma imagem multi-canal $I^{[k]}$. Especificamente, são utilizados 5 canais ($k = 1, \ldots, 5$), sendo eles:

1. Brilho da imagem;
2. Derivada horizontal do brilho ($I_x$);
3. Derivada vertical do brilho ($I_y$);
4. Canal verde (G) menos o canal vermelho (R);

5. Canal verde (G) menos o canal azul (B).

Os canais de diferenças cromáticas (4 e 5) são escalados em 0.25 para reduzir o impacto de artefatos criados por amostragem de cor, comum em vídeos (SAND; TELLER, 2006).

#### *4.2.1.1 Otimização dos vetores de movimento*

Seja $u(x, y, t)$ e $v(x, y, t)$ as componentes horizontal e vertical, respectivamente, do fluxo ótico que mapeia um ponto $I(x, y, t)$ em outro ponto no quadro seguinte:

$$I(x + u(x, y, t), y + v(x, y, t), t + 1).$$

O processo de otimização dos vetores de movimento que compõem o fluxo ótico age minizando a função objetivo:

$$E_{flow} = E_{data}(u, v, t) + E_{smooth}(u, v, t), \tag{4.1}$$

onde $E_{data}$ é o termo que representa uma estimativa robusta do erro de projeção de um pixel no quadro seguinte, enquanto o termo $E_{smooth}$ mede a variação espacial dos vetores de movimento do fluxo ótico. $E_{data}$ é dado por:

$$E_{data}(u, v, t) = \sum_{x,y,k} r(x, y, t)\Psi([I^{[k]}(x + u(x, y, t), y + v(x, y, t), t + 1) - I^{[k]}(x, y, t)]^2),$$

sendo $\Psi(s^2) = \sqrt{s^2 + \epsilon^2}$ a norma robusta [1], com $\epsilon = 0.0001$. A visibilidade $r(x, y, t)$ é uma função rotuladora que tem como saída valores binários (0 ou 1) para cada pixel, de acordo com a sua probabilidade de oclusão. Esta função será descrita na Seção 4.2.1.2.

Já o termo $E_{smooth}$ é dado por:

$$E_{smooth}(u, v, t) = \\ \sum_{x,y}(\alpha_g + \alpha_l \cdot b(x, y, t)) \cdot \Psi(u_x(x, y, t)^2 + u_y(x, y, t)^2 + v_x(x, y, t)^2 + v_y(x, y, t)^2),$$

onde $\alpha_g$ é o fator de suavidade global de movimento e $\alpha_l$ é o fator de suavidade local, modulado pela suavidade local $b(x, y, t)$:

$$b(x, y, t) = e^{-\left(\frac{\sqrt{I_x(x,y,t)^2 + I_y(x,y,t)^2}}{2\sigma_b}\right)}.$$

Com base em experimentos, foram fixados os seguintes valores: $\alpha_g = 0.04$, $\alpha_l = 0.06$ e $\sigma_b = 0.008$. Estes valores estão diretamente relacionados à resolução e ao nível de ruído dos vídeos utilizados, e apresentaram um bom compromisso entre precisão e suavidade dos vetores de movimento do fluxo ótico.

Para a otimização da função objetivo (Eq. 4.1), é construído um sistema de equações sobre o domínio espacial da imagem:

$$\begin{cases} \dfrac{\partial E_{flow}}{\partial u(x, y, t)} = 0 \\ \\ \dfrac{\partial E_{flow}}{\partial v(x, y, t)} = 0 \end{cases}.$$

[1] Esta função é uma forma diferenciável da função de valor absoluto e não responde tão fortemente aos *outliers* como a norma $L^2$ (SAND; TELLER, 2006)

Uma abordagem de linearização semelhante à apresentada na Seção 2.1 é aplicada às equações acima, e a partir do resultado da linearização é construído um sistema de equações contendo duas equações para cada pixel (uma para cada componente do vetor do fluxo ótico no pixel correspondente). Com a finalidade de manter a simplicidade do texto, e também porque esta não é uma inovação do presente trabalho, não será mostrado aqui como são efetuadas a linearização, construção e solução do sistema de equações. Para maiores detalhes, consulte (SAND; TELLER, 2006).

#### *4.2.1.2 Detecção de oclusão*

Detecção de oclusão é a parte mais desafiadora dos métodos de estimativa de fluxo ótico, reconstrução estéreo, rastreamento de pontos e estimativa de movimento em geral. No presente trabalho, mais do que simplesmente resolver o problema da oclusão puramente com partículas, é utilizada a estimativa de fluxo ótico para prover informação sobre oclusões. Idealmente, a estimativa do fluxo ótico é capaz de incorporar detalhes sutis da superfície sendo ocluída, que não necessariamente são capturados pelas partículas.

Conforme foi visto na Seção 4.2.1.1, a máscara $r(x, y, t)$, que modula o termo $E_{data}$, rotula explicitamente pixels oclusos. Isto modela o fato de que alguns pontos da imagem somem de um quadro para outro. Para isso é utilizada uma combinação da divergência fluxo e o erro de projeção de pixel na identificação de pixels oclusos. Seja a divergência de fluxo definida como:

$$r_{div}(x, y, t) = \frac{\partial}{\partial x}\bar{u}(x, y, t) + \frac{\partial}{\partial y}\bar{v}(x, y, t)$$

onde $\bar{u}(x, y, t)$ e $\bar{v}(x, y, t)$ são as componentes horizontal e vertical dos vetores de fluxo ótico, respectivamente, nas posições $(x, y)$ do quadro no tempo $t$. Note que a divergência é positiva para bordas de "desoclusão", negativa para bordas de oclusão, e próximas de zero para bordas de cisalhamento. Em bordas de cisalhamento, os vetores de movimento dos dois lados da borda terão mesma direção, porém sentido contrários. Com isso as derivadas parciais acima deverão se anular. Em bordas de desoclusão e de oclusão, os vetores de movimento dos dois lados da borda terão direções diferentes. No caso de bordas de desoclusão, os vetores serão divergentes, enquanto em bordas de oclusão, serão convergentes. Se forem divergentes, a soma das derivdadas parciais acima será positiva, e será negativa se forem convergentes. Como estamos interessados apenas nas bordas de oclusão, a função $r_d(x, y, t)$ é definida como:

$$r_d(x, y, t) = \begin{cases} r_{div}(x, y, t) & , \quad r_{div}(x, y, t) < 0, \\ 0 & , \quad r_{div}(x, y, t) \geq 0. \end{cases}$$

O erro de projeção $r_e(x, y, t)$ também provê uma estimativa da probabilidade de oclusão:

$$r_e(x, y, t) = I(x, y, t) - I(x + \bar{u}(x, y, t), y + \bar{v}(x, y, t), t + 1).$$

Estas duas estimativas são combinadas para obter as máscara de oclusão, que é o conjunto de localizações onde $r(x, y, t) = 1$, sendo $r(x, y, t)$ dado por:

$$r(x, y, t) = \begin{cases} 0 & , \quad e^{-\left(\frac{|r_d(x,y,t)|}{2\sigma_d^2} + \frac{r_e(x,y,t)}{2\sigma_e^2}\right)} \geq \tau_r, \\ 1 & , \quad e^{-\left(\frac{|r_d(x,y,t)|}{2\sigma_d^2} + \frac{r_e(x,y,t)}{2\sigma_e^2}\right)} < \tau_r. \end{cases} \tag{4.2}$$

Os valores de $\sigma_d$, $\sigma_e$ e $\tau_r$ escolhidos foram 0.3, 20 e 0.5, respectivamente, baseados em experimentos. O desempenho da detecção de oclusão não é consideravelmente sensível a $\sigma_d$ e $\sigma_e$. No entanto, deve-se ter cuidado ao escolher o valor de $\tau_r$. Valores muito pequenos de $\tau_r$ geram muitos falsos negativos, enquanto valores altos pode gerar muitos falsos positivos.

#### *4.2.1.3 Filtro bi-lateral dos vetores de movimento*

A detecção de pixels oclusos faz parte do processo de modelagem de oclusão, mas ainda é necessário tratar o problema da mistura de propriedades de pixels próximos das bordas de movimento. Essa mistura ocorre para todos os tipos de bordas de movimento: desoclusão, oclusão e cisalhamento. Com o objetivo de aumentar a precisão dos vetores de movimento próximos às bordas de movimento, é utilizado um filtro bilateral baseado no trabalho de Xiao *et al.* (XIAO et al., 2006).

O filtro atualiza cada vetor do fluxo ótico com uma média ponderada de vetores vizinhos:

$$u'(x,y,t) = \frac{\sum_{x_1,y_1} u(x_1,y_1,t)w(x,y,x_1,y_1,t)}{\sum_{x_1,y_1} w(x,y,x_1,y_1,t)}.$$

A atualização para a componente $v$ é análoga. O algoritmo pesa os vizinhos de acordo com proximidade espacial, similaridade de cor, similaridade de movimento, e rótulo de oclusão:

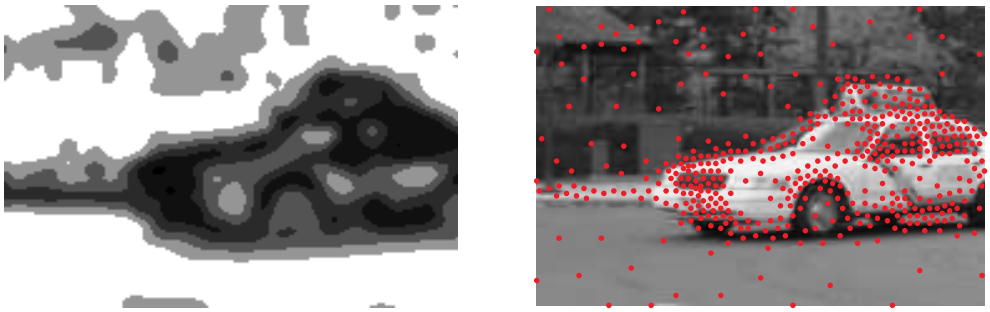
$$\begin{aligned} w(x,y,x_1,y_1,t) &= e^{-\left(\frac{\sqrt{(x-x_1)^2+(y-y_1)^2}}{2\sigma_x^2}\right)} \\ &\cdot e^{-\left(\frac{|I(x,y,t)-I(x_1,y_1,t)|}{2\sigma_i^2}\right)} \\ &\cdot e^{-\left(\frac{\sqrt{(u-u_1)^2+(v-v_1)^2}}{2\sigma_m^2}\right)} \\ &\cdot r(x_1,y_1,t), \end{aligned}$$

onde $u$ representa $u(x,y,t)$ e $u_1$ representa $u_1(x,y,t)$. Os valores de $\sigma_x$, $\sigma_i$ e $\sigma_m$ foram fixados em 4, 0.03 e 0.5, respectivamente, e $(x_1,y_1)$ foi limitado a uma distância máxima de 10 pixels de $(x,y)$ (SAND; TELLER, 2006).

### 4.2.2 Adição de Partículas

Seja $I(x,y,t)$ uma sequência de quadros, definida sobre as coordenadas espaciais $x$ e $y$ e a coordenada temporal $t$. Aqui, representaremos um quadro da sequência simplesmente por $I(x,y)$. Para determinar se uma nova partícula deve ser adicionada em um quadro $I(x,y)$, nós o filtramos com uma pilha de gaussianas com desvio padrão $\{\sigma(j) = 1.9^j | 0 \leq j \leq 5\}$, produzindo um conjunto de imagens $\{I_j(x,y)\}$, onde $j$ é o índice da escala correspondente à gaussiana de desvio padrão $\sigma(j)$.

Para cada pixel $(x,y)$ de $I(x,y)$, é determinada a sequência de escalas $\sigma(j)$ sobre a qual os correspondentes valores da imagem filtrada $I_j(x,y)$ são aproximadamente constantes. Assume-se que um pixel em $(x,y)$ é aproximadamente constante sobre $j$ escalas se $\|I_j(x,y) - I_1(x,y)\|_2 < \delta_{max}$ e $j$ é máximo (ou seja, $\|I_{j+1}(x,y) - I_1(x,y)\|_2 \geq \delta_{max}$). O valor de $\delta_{max}$ influenciará na densidade de partículas adicionadas. Quanto menor o valor de $\delta_{max}$, maior será a densidade de partículas adicionadas. Em nosso trabalho o valor de $\delta_{max}$ foi fixado em 10, com base em experimentos, pois este valor resultou em densidades que oferecem um bom compromisso entre precisão na segmentação e custo computacional, nos vídeos utilizados. Seja $z(x,y) = j$ o mapa de índices, e $s(x,y) = \sigma(j)$



(a) (b)

Figura 4.1: Exemplo do processo de adição de partículas: (a) mapa de escalas filtrado; (b) imagem original com partículas sobrepostas.

o mapa de escala, ambas informações irão nos auxiliar no processo de inclusão de novas partículas em um quadro $I(x, y)$.

Para obtermos uma distribuição espacial suave das partículas, o mapa de índices $z(x, y)$ é filtrado utilizando uma gaussiana com desvio padrão $\sigma_z$ e é arredondado para valores inteiros, resultando em um mapa de índices filtrado $\hat{z}(x.y)$. Com isso, o correspondente mapa de escala suavizado $\hat{s}(x, y) = \sigma(\hat{z}(x, y))$ é obtido. Em nossos experimentos, $\sigma_z$ foi fixado em 3. Este parâmetro afeta a intensidade da suavização da distrubuição espacial das partículas, e está diretamente relacionado à resolução das imagens: quanto maior for a resolução, maior deve ser $\sigma_z$.

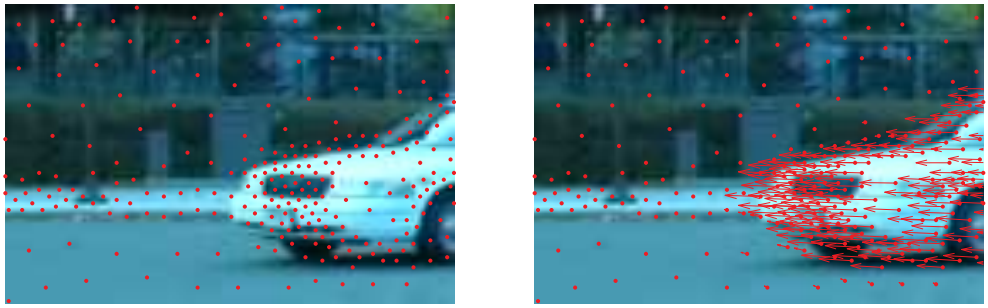
Adicionar partículas em um quadro é um processo iterativo. Uma partícula é inserida em um pixel $(x, y)$ se a sua distância euclidiana no domínio espacial em relação à partícula mais próxima é maior que $\hat{s}(x, y)$; novas partículas são adicionadas até que nenhum pixel satisfaça esta condição de distância mínima. Dessa forma, $\hat{s}(x, y)$ controla a densidade de partículas, que por sua vez vai ser proporcional ao contraste local. No primeiro quadro do vídeo, o processo começa sem qualquer partícula. Nos quadros subsequentes, o processo começa com o conjunto de partículas que "sobreviveram" (ou seja, que não foram removidas) do quadro anterior. A Figura 4.1(a) mostra um exemplo de um mapa de escala filtrado. Na Figura 4.1(b) é mostrado o resultado da adição de partículas (em vermelho) sobre a imagem original.

### 4.2.3 Propagação de Partículas

No estágio de propagação de partículas, o deslocamento de partículas do quadro atual para o quadro seguinte é estimado. Para propagar uma partícula $p$ do quadro $t$ para o quadro $t + 1$, utilizamos o fluxo ótico (*optical flow*) especificado pela componente horizontal de deslocamento $\bar{u}(x, y, t)$, e a componente vertical de deslocamento $\bar{v}(x, y, t)$:

$$x_p(t + 1) = x_p(t) + \bar{u}(x_p(t), y_p(t), t),$$
$$y_p(t + 1) = y_p(t) + \bar{v}(x_p(t), y_p(t), t).$$

Os valores de deslocamento para cada pixel da imagem são obtidos a partir do método de fluxo ótico proposto por Sand e Teller (SAND; TELLER, 2006). Os valores de deslocamento nas localidades das partículas são obtidos por interpolação bilinear a partir dos valores nos pixels. A Figura 4.2 mostra um exemplo da propagação de partículas com base no fluxo ótico. As posições originais das partículas são mostradas na Figura 4.2(a), enquanto na Figura 4.2(b) são mostrados os vetores de deslocamento correspondentes.



(a) (b)

Figura 4.2: Exemplo da propagação de partículas: (a) posição original das partículas; (b) vetores de movimento obtidos a partir do fluxo ótico.

### 4.2.4 Remoção de Partículas

No trabalho proposto por Sand e Teller (SAND; TELLER, 2006), a mesma função objetivo que é minimizada para otimizar (ou seja, para fazer um ajuste fino) a localização das partículas (ver Seção 4.2.5) é empregada como uma medida da confiabilidade da localização das partículas no estágio de remoção de partículas.

No trabalho de Sand e Teller (SAND; TELLER, 2006), uma partícula $p$ é removida em um quadro de tempo $t$ quando sua função de energia $E(p,t)$ (que será definida na Eq. (4.3), pág. 59) é maior que um limiar. No presente trabalho, partículas são removidas apenas se elas se tornam oclusas, ou saem do campo de visão.

Como as estimativas do fluxo ótico não são confiáveis em regiões próximas às bordas do quadro, consideramos no presente trabalho que o campo de visão é definido como o conjunto de pontos que estão a uma distância de pelo menos 5 pixels das bordas da imagem. Esta distância está relacionada à resolução das imagens, e quanto maior a resolução, maior deverá ser a área de incerteza. Então, quando uma partícula sai do campo de visão após o estágio de propagação em um dado quadro, ela é removida daquele quadro.

Para detectar quando uma partícula se torna oclusa, é utilizada a abordagem de detecção de oclusão empregada no método de fluxo ótico proposto por Sand e Teller (SAND; TELLER, 2006) e apresentado na Seção 4.2.1.2. Assim, partículas localizadas em pontos onde $r(x,y,t) = 1$ (ver Eq. 4.2) são eliminadas.

Conforme mencionado na Seção 4.2.1.2, erros na detecção de oclusão podem ocorrer, e os limiares foram escolhidos de forma a minimizar a ocorrência de falsos positivos. Falsos positivos, neste contexto, representam partículas que são erroneamente classificadas como oclusas. Por outro lado, isso faz com que a ocorrência de falsos negativos seja aumentada, permitindo que partículas migrem de um objeto para outro no decorrer do vídeo. Felizmente, segundo a abordagem de segmentação de partículas proposta, uma pequena porção de erros de rastreamento - partículas que são associadas a um objeto mas erroneamente mudam para outro objeto - podem ser recuperados em estágios de pós-processamento (ver Seção 4.3.2) sem que haja degeneração considerável no resultado final da segmentação.

No presente trabalho, se uma partícula se torna oclusa ou sai do campo de visão, esta partícula é eliminada e não mais é considerada no processo de rastreamento, mesmo que o local da cena atualmente associado a esta partícula se torne visível mais tarde.

### 4.2.5 Otimização da Localização de Partículas

O estágio de otimização da localização das partículas é necessário pois o estágio de propagação utiliza apenas informação de movimento do fluxo ótico disponível para as posições dos pixels da imagem. Como as partículas representam pontos interpolados a partir dos pixels, pode ocorrer o acúmulo de erro na estimativa do deslocamento da partícula a cada quadro e, por isso, a otimização da localização das partículas reduz este problema, tornando a estimativa mais confiável e resultando em tempos de vida mais longos.

No estágio de otimização da localização das partículas, a posição de cada partícula $p$ no quadro atual $t$ é otimizada minimizando-se uma função objetivo composta por três termos, chamados $E_p^{[c]}$, $E_f$ e $E_d$ conforme descrito abaixo.

O primeiro termo $E_p^{[c]}$ representa o erro de projeção da partícula $p$ no quadro do tempo $t$ com respeito à primeira aparição da partícula em um quadro da sequência:

$$E_p^{[c]}(p,t) = \Psi([I^{[c]}(x_p(t), y_p(t), t) - I^{[c]}(x_p(t_p^0), y_p(t_p^0), t_p^0)]^2),$$

sendo $I^{[c]}$ o canal de cor $c$ (R, G ou B) do vídeo; $x_p(t)$ e $y_p(t)$ representam as coordenadas espaciais da partícula $p$ no quadro do tempo $t$; e $t_p^0$ representa o tempo da primeira aparição da partícula $p$ na sequência. O termo $\Psi(a) = \sqrt{a+\epsilon}$ é a norma robusta[2], com $\epsilon = 0.0001$.

O segundo termo $E_f$ representa a diferença entre o deslocamento real da partícula $p$ no quadro do tempo $t$, e a estimativa correspondente baseada no deslocamento local do fluxo ótico:

$$\begin{aligned} E_f(p,t) = \Psi([\bar{u}(x_p(t-1), y_p(t-1), t-1) - (x_p(t) - x_p(t-1))]^2 + \\ [\bar{v}(x_p(t-1), y_p(t-1), t-1) - (y_p(t) - y_p(t-1))]^2), \end{aligned}$$

sendo $\bar{u}$ e $\bar{v}$ as componentes horizontal e vertical dos vetores de fluxo ótico, respectivamente.

O terceiro termo $E_d$ mede o movimento relativo entre partículas $p$ e $q$ ligadas no quadro de tempo $t$:

$$\begin{aligned} E_d(p,q,t) = l_{pq}(t)\Psi([(x_p(t) - x_p(t-1)) - (x_q(t) - x_q(t-1))]^2 \\ +[(y_p(t) - y_p(t-1)) - (y_q(t) - y_q(t-1))]^2), \end{aligned}$$

sendo $l_{pq}(t)$ o peso da ligação entre as partículas $p$ e $q$ no quadro de tempo $t$, e é calculado da seguinte forma: para cada quadro $t$ é computada a triangulação de Delaunay cujos vértices são as posições das partículas neste quadro. Conexões são criadas entre as partículas $p$ e $q$ caso haja uma aresta ligando estas partículas na triangulação de Delaunay correspondente. Para cada conexão $(p,q)$, computamos a diferença quadrática de movimento de acordo com os vetores do fluxo ótico:

$$\begin{aligned} D(p,q,t) = \\ (\bar{u}(x_p(t), y_p(t), t) - \bar{u}(x_q(t), y_q(t), t))^2 + \\ (\bar{v}(x_p(t), y_p(t), t) - \bar{v}(x_q(t), y_q(t), t))^2. \end{aligned}$$

O peso $l_{pq}(t)$ é então definido pela exponencial negativa:

$$l_{pq}(t) = e^{-\left(\frac{\sqrt{D(p,q,t)}}{2\sigma_l^2}\right)},$$

[2]Esta função é uma forma diferenciável da função de valor absoluto e não responde tão fortemente aos *outliers* como a norma $L^2$ (SAND; TELLER, 2006)

onde $\sigma_l$ foi fixado em 0.5, com base em experimentos.

Agrupando os três termos, temos a função objetivo completa:

$$E(p,t) = \sum_c E_p^{[c]}(p,t) + \alpha_f E_f(p,t) + \alpha_d \sum_{q \in L_p(t)} E_d(p,q,t), \tag{4.3}$$

sendo $L_p(t)$ o conjunto de partículas ligadas à partícula $p$ no quadro de tempo $t$; $\alpha_f$ e $\alpha_d$ são constantes que fornecem um compromisso entre os termos, e foram ajustadas para 5 e 10 no passo de otimização de localização, respectivamente ($\alpha_f$ e $\alpha_d$ foram ajustados com base em experimentos). O desempenho da estimativa de trajetórias não é muito sensível a mudanças nestas constantes. Então, os valores fixados aqui para $\alpha_f$ e $\alpha_d$ devem funcionar para uma grande variedade de vídeos. Só é necessário ter cuidado ao ajustar os valores de $\alpha_f$, já que o termo $E_f$ atua como uma restrição posicional; consequentemente, valores muito pequenos de $\alpha_f$ poderiam levar a erros significativos com respeito às estimativas do fluxo ótico.

O algoritmo de otimização minimiza a Eq. (4.3) utilizando um par de laços aninhados em torno de um sistema esparso de equações lineares. O laço externo computa iterativamente atualizações de $x_p(t)$ e $y_p(t)$ usando incrementos $dx_p(t)$ e $dy_p(t)$ computados pelo laço interno.

Na função objetivo $E$ (4.3), substituímos $x_p(t)$ por $x_p(t) + dx_p(t)$ e $y_p(t)$ por $y_p(t) + dy_p(t)$. Calculando-se as derivadas parciais, obtemos um sistema de equações, o qual o algoritmo resolve para $dx_p(t)$ e $dy_p(t)$:

$$\begin{cases} \dfrac{\partial E}{\partial dx_p(t)} = 0 \\ \\ \dfrac{\partial E}{\partial dy_p(t)} = 0 \end{cases}.$$

Para o termo $E_p$, é utilizado o método de linearização de Brox *et al*. (BROX et al., 2004):

$$\begin{aligned} I_z^{[c]}(x_p(t), y_p(t), t) = {} & I_x^{[c]}(x_p(t), y_p(t), t) dx_p(t) + \\ & I_y^{[c]}(x_p(t), y_p(t), t) dy_p(t) + \\ & I^{[c]}(x_p(t), y_p(t), t) - I^{[c]}(x_p(t_0), y_p(t_0), t_0), \end{aligned}$$

$$\begin{aligned} & \tfrac{\partial E_p^{[c](p,t)}}{\partial dx_p(t)} \approx \\ & \textstyle\sum_{\Delta_x} \sum_{\Delta_y} \Big( 2\Psi'([I_z^{[c]}(x_p(t) + \Delta_x, y_p(t) + \Delta_y, t)]^2) \\ & \qquad \cdot I_z^{[c]}(x_p(t) + \Delta_x, y_p(t) + \Delta_y, t) \cdot I_x^{[c]}(x_p(t) + \Delta_x, y_p(t) + \Delta_y, t) \Big), \end{aligned} \tag{4.4}$$

onde $\Psi'$ é a derivada de $\Psi$ em relação ao seu argumento $a$, enquanto $I_x$, $I_y$ e $I_z$ são as derivadas de $I(x,y,t)$ em relação às coordenadas espaciais $x$ e $y$, e em relação ao tempo $t$, respectivamente.

Para o termo $E_d$, a derivada parcial é direta (não necessita de linearização):

$$\begin{aligned}\frac{\partial E_d(p,q,t)}{\partial dx_p(t)} = \\ 2l_{pq}(t)\Psi'([(x_p(t) - x_p(t-1) + dx_p(t) - dx_p(t-1)) - \\ (x_q(t) - x_q(t-1) + dx_q(t) - dx_q(t-1))]^2 + \\ [(y_p(t) - y_p(t-1) + dy_p(t) - dy_p(t-1)) - \\ (y_q(t) - y_q(t-1) + dy_q(t) - dy_q(t-1))]^2) \\ \cdot[(x_p(t) - x_p(t-1) + dx_p(t) - dx_p(t-1)) - \\ (x_q(t) - x_q(t-1) + dx_q(t) - dx_q(t-1))].\end{aligned} \tag{4.5}$$

O termo $E_f$ utiliza a mesma linearização do termo $E_p$:

$$\begin{aligned}U &= \bar{u}_x dx_p(t-1) + \bar{u}_y dy_p(t-1) + \bar{u} - (x_p(t) - x_p(t-1) + dx_p(t) - dx_p(t-1)) \\ V &= \bar{v}_x dx_p(t-1) + \bar{v}_y dy_p(t-1) + \bar{v} - (v_p(t) - v_p(t-1) + dv_p(t) - dv_p(t-1)).\end{aligned}$$

$$\frac{\partial E_f(p,t)}{\partial dx_p(t)} \approx 2\Psi'(U^2 + V^2)U. \tag{4.6}$$

Aqui, os índices $(x_p(t-1), y_p(t-1), t-1)$ são omitidos de $\bar{u}$, $\bar{v}$ e suas respectivas derivadas espaciais $\bar{u}_x$, $\bar{u}_y$, $\bar{v}_x$ e $\bar{v}_y$.

Cada uma destas derivadas parciais é linear em relação a $dx_p(t)$ e $dy_p(t)$, exceto pelos fatores $\Psi'$. Para cada iteração do laço interno os termos $\Psi'$ são computados e os mesmos são mantidos constantes no sistema linear de equações:

```
Laço externo (4 iterações)
  Computar I_x^[c](p,t), I_y^[c](p,t)
  dx_p(t), dy_p(t) ← 0
    Laço interno (4 iterações)
      Computar termos Ψ'
      Resolver sistema de equações lineares para dx_p(t), dy_p(t)
    Fim Laço interno
    x_p(t) ← x_p(t) + dx_p(t)
    y_p(t) ← y_p(t) + dy_p(t)
Fim Laço Externo
```

O sistema esparso de equações lineares é resolvido a partir do método QMR (*Quasi-Minimum Residual*) (FREUND; NACHTIGAL, 1991), utilizando 200 iterações. Note que a linearização ocorre dentro do laço interno; o algoritmo ainda está otimizando a função objetivo original não-linearizada.

## 4.3 Segmentação de Regiões Segundo suas Trajetórias Baseada em Partículas

A representação de movimento em vídeos através de trajetórias de partículas é flexível, permitindo a identificação do movimento de objetos sem restrições globais de movimento (como modelos de câmera, modelos de cenas rígidas, etc), tratando oclusões e provendo granularidade local adaptativa. No entanto, extrair informação de alto-nível sobre estruturas em movimento com base em trajetórias de partículas não é uma tarefa trivial, pelas razões discutidas a seguir. Diferentes partículas que representam pontos da cena de uma mesma estrutura em movimento podem ter diferentes tempos de vida. Por exemplo, pode

ocorrer de um par de partículas que representam pontos de um mesmo objeto em movimento coerente não co-existir em um mesmo quadro do vídeo, por terem diferentes tempos de vida. Além disso, é difícil evitar a incidência de padrões de movimento errôneos, os quais podem ocorrer devido ao ruído, artefatos causados por iluminação e/ou quantização/compressão, falta de informação de movimento em partes da cena, e a ausência de um modelo de validação para o movimento das partículas.

Apenas partículas que aparecem simultaneamente em pelo menos dois quadros consecutivos podem ser comparadas diretamente em termos de seus movimentos. Contudo, duas partículas que não co-existem em qualquer quadro do vídeo podem ser comparadas indiretamente investigando-se como elas se movem em relação às outras partículas que tem interseção de tempos de vida com estas duas partículas. Por exemplo, sejam $F_1$ e $F_2$ os conjuntos de quadros onde duas partículas $p_1$ e $p_2$ existam, mas não exista interseção entre eles (ou seja, $F_1 \cap F_2 = \emptyset$). Outra partícula $p_3$ pode ter seu tempo de vida representado pelo conjunto de quadros $F_3$, tal que $F_1 \cap F_3 \neq \emptyset$ e $F_2 \cap F_3 \neq \emptyset$. Assim, a análise do movimento pode ser feita combinando-se informação de sub-conjuntos de partículas. Esta idéia de combinar informação de movimento (padrões de movimento) de sub-conjuntos de dados é similar à combinação de partições de dados em agrupamentos de conjuntos (*ensemble clustering*) (VISWANATH; JAYASURYA, 2006; STREHL; GHOSH, 2003; FRED; JAIN, 2005).

De acordo com a filosofia do agrupamento de conjuntos, dado um conjunto de dados com $n$ amostras, existem diferentes formas de gerar grupos a partir deste conjunto de dados (STREHL; GHOSH, 2003):

- Aplicando-se diferentes algoritmos de agrupamento ao conjunto inteiro de $n$ amostras;
- Aplicando-se o mesmo algoritmo de agrupamento com diferentes parâmetros;
- Aplicando-se algoritmos de agrupamento a diferentes representações dos dados (por exemplo, usando diferentes espaços de feições);
- Aplicando-se algoritmos de agrupamento a diferentes partições dos dados (ou seja, a sub-conjuntos das $n$ amostras).

O último item descreve a abordagem adotada no presente trabalho para agrupar partículas em movimento coerente. Um algoritmo de agrupamento é aplicado a cada conjunto de partículas que coexistem no vídeo (ou seja, que tenham interseção dos tempos de vida), e então estes agrupamentos de sub-conjuntos de partículas são combinados em grupos maiores de partículas.

A maioria dos métodos de agrupamento de conjuntos requerem o cálculo da similaridade par-a-par para todos os objetos na coleção. Em nosso caso, o número de partículas em um vídeo pode ser muito grande, e o custo de se calcular uma matriz de similaridade par-a-par pode ser impraticável. Uma opção interessante é considerar a integração de diferentes partições dos dados como um problema de correspondência de grupos (STREHL; GHOSH, 2003), onde grupos similares são identificados e combinados, formando meta-grupos. Um algoritmo que combina diversas partições de partículas em meta-grupos será apresentado na Seção 4.3.1, utilizando a abordagem de agrupamento de conjuntos proposta em (STREHL; GHOSH, 2003), modificado para selecionar automaticamente o número de meta-grupos.

Inconsistências no processo de agrupamento de partículas são esperadas, já que erros podem ocorrer no rastreamento de um grande número de partículas em movimento estocástico. Por essa razão, o estágio de agrupamento é seguido por um estágio de validação de grupos (ver Seção 4.3.2), onde o contexto da partícula, seu movimento e sua localização espacial são combinados para checar possíveis inconsistências. Finalmente, um estágio de filtragem (ver Seção 4.3.3) é executado para eliminar *outliers* e pequenos grupos de partículas que não são considerados significantes.

### 4.3.1 Agrupamento de Conjuntos de Partículas

De forma a agrupar partículas que estão em movimento coerente ao longo do vídeo, primeiramente são identificados sub-conjuntos de partículas que apresentam movimento similar em quadros vizinhos. Seja $P = \{p_1, p_2, \ldots, p_{n_p^v}\}$ o conjunto inteiro de $n_p^v$ partículas no vídeo, e $\vec{e_p}(t, t+l)$ o vetor de deslocamento da partícula $p$ entre os quadros de tempo $t$ e tempo $t+1$:

$$\vec{e_p}(t, t+l) = \begin{bmatrix} x_p(t+l) - x_p(t) \\ y_p(t+l) - y_p(t) \end{bmatrix}.$$

Para cada quadro de um tempo $t$, três agrupamentos são computados com as partículas presentes neste quadro (ou seja, $l = 1, 2, 3$), baseados nas seguintes feições:

1. vetores de deslocamento entre os quadros adjacentes ($\vec{e_p}(t, t+1)$);
2. vetores de deslocamento entre quadros distantes duas unidades de tempo ($\vec{e_p}(t, t+2)$);
3. vetores de deslocamento entre quadros distantes três unidades de tempo ($\vec{e_p}(t, t+3)$).

Estes três agrupamentos são utilizados em cada quadro para reforçar a tendência de partículas com padrões de movimento similares a se agruparem juntas, reduzindo a influência de *outliers* e inconsistências de agrupamentos individuais na partição final. A escolha de se utilizar vetores de deslocamento entre quadros com distâncias de 1, 2 e 3 unidades de tempo se deu com base na resolução temporal dos vídeos testados. A eficiência do algoritmo de agrupamento utilizando estes vetores de deslocamento depende, além da resolução temporal dos vídeos, também da magnitude dos movimentos esperados. Para vídeos com resoluções temporais muito altas e/ou movimentos muito lentos, recomenda-se utilizar vetores de deslocamento entre quadros mais distantes.

Assumindo $n_t$ quadros na sequência, teremos $n_h = 3n_t - 6$ agrupamentos do conjunto de partículas nestes $n_t$ quadros. Isto acontece porque alguns vetores de deslocamento $\vec{e_p}(t, t+l)$ não podem ser calculados:

- $\vec{e_p}(t, t+1)$, $\vec{e_p}(t, t+2)$ e $\vec{e_p}(t, t+3)$, quando $t = n_t$;
- $\vec{e_p}(t, t+2)$ e $\vec{e_p}(t, t+3)$ quando $t = n_t - 1$;
- $\vec{e_p}(t, t+3)$ quando $t = n_t - 2$.

Cada agrupamento $\{H^{(j)} | j \in \{1, \ldots, n_h\}\}$ divide o conjunto de $n_p^v$ partículas em $n^{(j)}$ grupos $\{h_i^{(j)} | i \in \{1, \ldots, n^{(j)}\}\}$, $H^{(j)} = \{h_i^{(j)}\}$, onde $h_i^{(j)}$ representa o $i$-ésimo grupo do $j$-ésimo agrupamento, e $n^{(j)}$ denota o número de grupos do $j$-ésimo agrupamento.

O método de *mean-shift* (COMANICIU; MEER, 2002) é empregado neste trabalho para obter os grupos $h_i^{(j)}$ em $H^{(j)}$, devido à capacidade deste método de identificar grupos com formas arbitrárias no espaço de feições. A banda do *kernel* do *mean-shift* foi ajustada em $3 \cdot l$, para todos os quadros $(t, t+l)$ e $l = 1, 2, 3$. A banda aumenta com $l$ de forma a reduzir a diversidade de agrupamentos e, consequentemente, produzir correspondências mais significativas entre grupos, pois a baixa diversidade entre os grupos é uma propriedade requerida pelo algoritmo de agrupamento de conjuntos empregado neste trabalho. Note que a banda do *kernel* do *mean-shift* está relacionada ao espectro de movimentos relativos esperado entre os objetos (ou seja, grupos de partículas). A banda pode ser ajustada de acordo com o tipo de movimento dos objetos (devagar, rápido) encontrado na sequência. Contudo, ela não é afetada pelo movimento global, tal como o movimento translacional induzido por uma câmera em movimento, por exemplo.

Neste estágio, nós temos três agrupamentos $\{H^{(j)}\}$ para cada quadro. Uma única partição dos dados é obtida para o conjunto inteiro de partículas no vídeo, combinando-se todos os agrupamentos (três para cada quadro) em um único agrupamento $H$. Para realizar esta tarefa, foi utilizado o algoritmo de *Meta Clustering* (STREHL; GHOSH, 2003), conforme detalhado a seguir.

Cada partícula $p_i (i = 1, 2, \ldots, n_p^v)$ é atribuída a um dos grupos $h_i^{(j)}$ em cada agrupamento $H^{(j)}$ baseado em seu padrão de movimento em diferentes intervalos de tempo $(l = 1, 2, 3)$. Portanto, cada grupo $h_i^{(j)}$ no agrupamento $H^{(j)}$ pode ser representado por um vetor de rótulos binários, onde a partícula $p_i$ é rotulada como '1' se ela pertence a $h_i^{(j)}$, ou '0' de outra forma (ver Tabela 4.1).

Inicialmente, de acordo com a abordagem de *Meta Clustering*, os $n_h$ grupos $h_i^{(j)}$, representados por vetores de rótulos binários, são transformados em um hiper-grafo. Um hiper-grafo é constituído por nodos e hiper-arestas, e é uma generalização de um grafo no sentido de que uma hiper-aresta pode conectar qualquer conjunto de nodos (enquanto que em um grafo usual, uma aresta conecta exatamente dois nodos). Cada agrupamento $H^{(j)}$ é representado por uma matriz binária, com $n_p^v$ linhas (uma para cada partícula do vídeo inteiro) e $n^{(j)}$ colunas (ou seja, a concatenação de $n^{(j)}$ vetores de rótulos binários, um para cada grupo). Nesta representação de hiper-grafo, nodos representam partículas e hiper-arestas representam grupos de partículas que ocorrem em 1, 2 e 3 quadros consecutivos. A Tabela 4.1 mostra um exemplo simples de hiper-grafo. Neste exemplo de uma sequência simples de 3 quadros, as linhas da Tabela representam as partículas ($p_{\{1,\ldots,6\}}$), os agrupamentos são representados por matrizes binárias $H^{(1,\ldots,3)}$, e os grupos individuais são representados pelas hiper-arestas $h_{1,2,3}^{(1)}$, $h_{1,2,3}^{(2)}$ e $h_{1,2}^{(3)}$. Note que, como temos 3 quadros neste exemplo, 3 agrupamentos são formados, utilizando:

1. vetores de movimento entre os quadros 1 e 2 ($H^{(1)}$);
2. vetores de movimento entre os quadros 1 e 3 ($H^{(2)}$);
3. vetores de movimento entre os quadros 2 e 3 ($H^{(3)}$).

Como uma partícula só pode pertencer a um único grupo $h_i^{(j)}$ em cada agrupamento $H^{(j)}$, as linhas na matriz binária correspondente somam '1' quando o grupo da partícula é conhecido (ou seja, a partícula é definida no intervalo de quadros $[t, t+l]$ utilizado para produzir os agrupamentos $H^{(j)}$). Linhas que correspondem a partículas que não foram atribuídas a nenhum grupo somam '0' (ou seja, a partícula correspondente não existe no intervalo de quadros $[t, t+l]$ sobre o qual o agrupamento $H^{(j)}$ é definido).

Tabela 4.1: Exemplo de um hiper-grafo com 8 hiper-arestas.

| | $H^{(1)}$ | | | $H^{(2)}$ | | | $H^{(3)}$ | |
|---|---|---|---|---|---|---|---|---|
| | $h_1^{(1)}$ | $h_2^{(1)}$ | $h_3^{(1)}$ | $h_1^{(2)}$ | $h_2^{(2)}$ | $h_3^{(2)}$ | $h_1^{(3)}$ | $h_2^{(3)}$ |
| $p_1$ | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| $p_2$ | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| $p_3$ | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| $p_4$ | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| $p_5$ | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| $p_6$ | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

A matriz concatenada $\mathbf{H} = H^{(1,\ldots,n_h)} = (H^{(1)} \ldots H^{(n_h)})$ é um hiper-grafo, com $n_p^v$ nodos (ou seja, partículas) e $n_{all} = \sum_{j=1}^{n_h} n^{(j)}$ hiper-arestas (ou seja, grupos de partículas). Cada vetor coluna $h_i, i = 1, \ldots, n_{all}$ de $H^{(j)}$ é uma hiper-aresta de um hiper-grafo $\mathbf{H}$, e representa um dos grupos de partículas do conjunto de agrupamentos $H^{(j)}, j = 1, \ldots, n_h$. A abordagem de *Meta Clustering* junta grupos de partículas (ou seja, hiper-arestas) em meta-grupos com base em uma medida de similaridade através da distância de Jaccard (ver Eq. (4.7)). A segmentação final das partículas é obtida atribuindo cada partícula do vídeo a um meta-grupo, e estes meta-grupos irão conduzir a segmentação final, conforme explicado na Seção 4.4.

Os meta-grupos são obtidos por agrupamento hierárquico das hiper-arestas. Para isso, uma matriz simétrica de similaridade $m$ é computada, onde $m(h_s, h_t) = m(h_t, h_s)$, contendo a similaridade calculada entre cada par de hiper-arestas $h_s$ e $h_t$ do hiper-grafo $\mathbf{H}$. A similaridade entre pares de grupos (ou seja, hiper-arestas) $h_s$ e $h_t$ é medida pela distância de Jaccard:

$$m(h_s, h_t) = \frac{h_s^\top h_t}{\|h_s\|_2^2 + \|h_t\|_2^2 - h_s^\top h_t}. \tag{4.7}$$

A matriz de similaridade $m$ computada é então utilizada como entrada para o algoritmo de agrupamento hierárquico, através do método de *single link* (JAIN; MURTY; FLYNN, 1999). O número ideal $K$ de meta-grupos é selecionado como sendo o número de grupos que é mais estável no dendrograma do agrupamento hierárquico. Observa-se como o número de grupos varia à medida que os valores de limiar aumentam de '0' para '1', quando o dendrograma é construído; $K$ é o número de grupos detectado no maior intervalo de incrementos de limiar consecutivos para o qual o número de grupos não se modifica, durante a construção do dendrograma. Isso está ilustrado na Figura 4.3, onde o intervalo $Z$ no dendrograma define o número ideal de meta-grupos $K = 3$ (já que esta é a maior sequência de incrementos do valor do limiar para o qual o número de meta-grupos é constante).

Após o número ideal $K$ de meta-grupos ser identificado, todas as hiper-arestas pertencentes ao mesmo meta-grupo são unidas em uma única meta-hiper-aresta. Suponha que os valores, originalmente lógicos, das partículas nas hiper-arestas sejam considerados pesos que indicam o grau de associação das partículas aos grupos $h_i$ correspondentes, e também aos agrupamentos $H^{(j)}$ (com '1' indicando uma forte associação e '0' indicando nenhuma associação). Então, partículas são atribuídas aos meta-grupos com base em seus pesos (ou grau de associação aos meta-grupos), e estes pesos são computados pela média aritmética dos pesos da partícula em todas as hiper-arestas $h_i$ pertencentes a cada meta-grupo. Suponha que tenhamos identificado 3 meta-grupos $M_{1,\ldots,3}$ em nosso hiper-grafo do exemplo mostrado na Tabela 4.1, cada um deles composto pelas seguintes hiper-arestas:

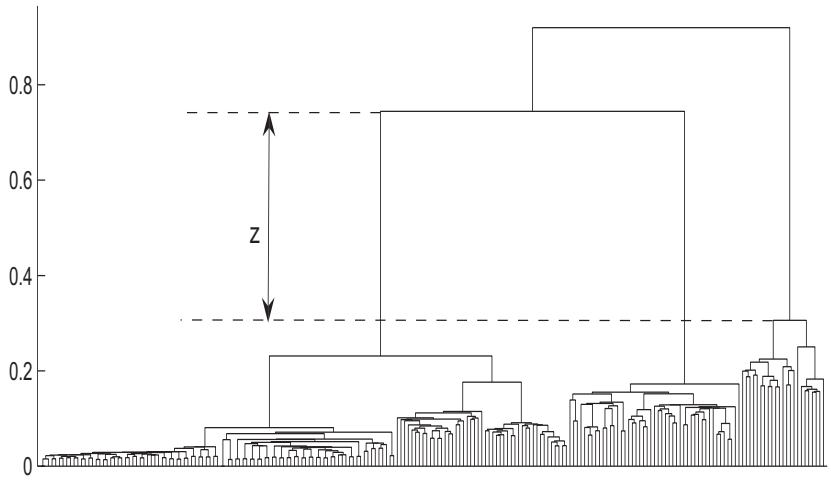



Figura 4.3: Exemplo de um dendrograma de meta-grupos. $Z$ representa o maior intervalo de valores consecutivos de limiar no qual o número de meta-grupos não se modifica ($K = 3$).

Tabela 4.2: Exemplo de 3 meta-grupos, representados por suas respectivas meta-hiper-arestas.

| | $M_1$ | $M_2$ | $M_3$ |
|---|---|---|---|
| $p_1$ | 0 | 1 | 0 |
| $p_2$ | 0 | 0.33 | 0.33 |
| $p_3$ | 0 | 0 | 1 |
| $p_4$ | 0 | 0 | 1 |
| $p_5$ | 1 | 0 | 0 |
| $p_6$ | 0.5 | 0 | 0 |

1. $M_1$: $h_3^{(1)}$ e $h_3^{(2)}$;
2. $M_2$: $h_1^{(1)}$, $h_1^{(2)}$ e $h_1^{(3)}$;
3. $M_3$: $h_2^{(1)}$, $h_2^{(2)}$ e $h_2^{(3)}$.

As meta-hiper-arestas correspondentes a estes 3 meta-grupos são mostradas na Tabela 4.2. Lembre que uma partícula pode estar associada a diferentes grupos $h_i$, e estes grupos são associados a diferentes meta-grupos (por exemplo, em alguns quadros uma partícula pode estar se movendo em relação a um *background* estático, mas a mesma partícula pode estar estática em outros quadros). Calculando a média aritmética do peso da partícula nas hiper-arestas resultará em uma meta-hiper-aresta que tem um registro para cada partícula no vídeo, descrevendo o grau de associação desta partícula com o meta-grupo correspondente. Quanto mais forte esta associação, mais próximo de '1' será esse registro; enquanto associações mais fracas terão valores próximos de '0'.

O último passo no agrupamento de conjuntos é a atribuição de cada partícula ao meta-grupo ao qual ela tem o maior grau de associação (ou seja, o vetor da meta-hiper-aresta que tem a maior média aritmética dos pesos calculados entre todas meta-hiper-arestas),

e no caso de um empate, o mesmo é quebrado aleatoriamente. No exemplo apresentado (Tabela 4.2), teríamos as seguintes atribuições:

1. $p_1 \rightarrow M_2$;
2. $p_2 \rightarrow$ atribuído tanto a $M_2$ ou $M_3$, aleatoriamente;
3. $p_3 \rightarrow M_3$;
4. $p_4 \rightarrow M_3$;
5. $p_5 \rightarrow M_1$;
6. $p_6 \rightarrow M_1$.

Ao final do processo de atribuição de partículas aos $K$ meta-grupos, o conjunto final de meta-grupos de partículas $\{M_b | b \leq K\}$ é obtido. O número final de meta-grupos pode ser menor que $K$ porque alguns meta-grupos podem não ter nenhuma partícula atribuída. Então, teremos $b$ rótulos ao final do processo de agrupamento de partículas (com $b$ sendo no máximo $K$).

Embora esta abordagem utilize agrupamentos obtidos de pares de quadros, o agrupamento de conjuntos tende a extrair padrões de longa-duração, sem sofrer do problema de acumulação de erro que está presente em abordagens puramente quadro-a-quadro (MEIER; NGAN, 1998; XU; KABUKA; YOUNIS, 2004). Além do mais, novos objetos que aparecem ao longo do vídeo não necessitam de tratamento especial.

### 4.3.2 Validação de Meta-Grupos de Partículas

Conforme mencionado anteriormente, erros no processo de rastreamento de partículas podem ocorrer, resultando em atribuições de partículas a meta-grupos errôneas. Para detectar estas inconsistências após o estágio de agrupamento de conjuntos (Seção 4.3.1), um passo de validação de meta-grupos é executado analisando-se as trajetórias no contexto das partículas juntamente agrupadas (no mesmo meta-grupo). Isto é particularmente importante quando partículas são atribuídas a um objeto mas migram para outro objeto durante o processo de rastreamento, devido a imperfeições na detecção de oclusão.

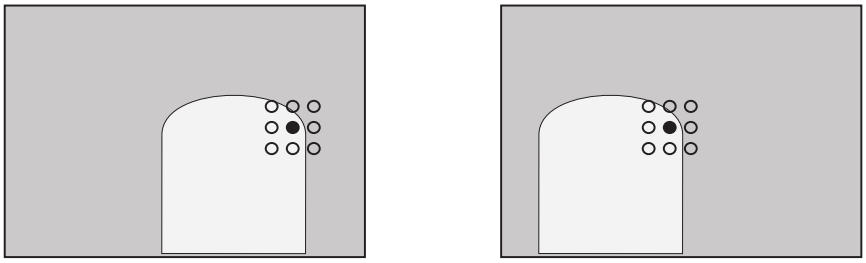
Seja $p = (x_p(t), y_p(t))$ a $p$-ésima partícula, representada por suas coordenadas espaciais no quadro de tempo $t$. O contexto desta partícula é definido por uma janela de tamanho $2W_V + 1$, dada por $(x_p(t) + \Delta_{V_x}, y_p(t) + \Delta_{V_y})$, onde:

$$-W_V \leq \Delta_{V_x}, \Delta_{V_y} \leq W_V \quad , \quad \Delta_{V_x}, \Delta_{V_y} \in \mathbb{Z}.$$

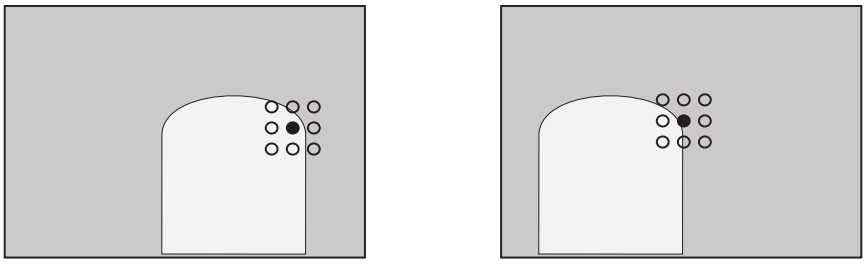
O tamanho ideal da janela depende da resolução espacial do vídeo, ou seja, quanto maior a resolução, maior deve ser $W_V$. Em todos os nossos experimentos, contextos de tamanho $W_V = 2$ foram utilizados.

O contexto de uma partícula movendo-se de forma coerente com suas partículas vizinhas geralmente não se modifica de um quadro para outro, desde que o conjunto esteja se movimentando de forma coerente e possamos assumir um movimento aproximadamente translacional entre quadros adjacentes. O erro de projeção de um ponto em uma janela $[-W_V, W_V]$ em um nível sub-pixel, que especifica o contexto da partícula $p$, dado o movimento da partícula entre o quadro de tempo $t$ e o quadro de tempo $t - \rho_V$, pode ser computado como segue:

$$D^{[\Delta_{V_x}, \Delta_{V_y}]}(p, t, \rho_V) = \Big\| I(x_p(t) + \Delta_{V_x}, y_p(t) + \Delta_{V_y}, t) - I(x_p(t - \rho_V) + \Delta_{V_x}, y_p(t - \rho_V) + \Delta_{V_y}, t - \rho_V) \Big\|_2 ,$$



(a) quadro no tempo $t - \rho_V$ (b) quadro no tempo $t$



(c) quadro no tempo $t - \rho_V$ (d) quadro no tempo $t$

Figura 4.4: Detecção de mudança de contexto: (a)-(b) $w$ melhores casamentos $< \tau_{best}$; (c)-(d) $w$ melhores casamentos $\geq \tau_{best}$.

De forma a detectar mudanças de contexto, é computada a média dos $w$-melhores "casamentos" (ou seja, a média dos $w$ menores erros de estimativa de movimento de $D^{[\Delta_{V_x}, \Delta_{V_y}]}(p, t, \rho_V)$ em $-W_V \leq \Delta_{V_x}, \Delta_{V_y} \leq W_V$), e se este valor médio é grande o suficiente (ou seja, maior que um limiar $\tau_{best}$), assumimos que a partícula $p$ mudou seu contexto no quadro de tempo $t$, e é marcada como inconsistente. O valor ideal para o limiar $\tau_{best}$ está relacionado à resolução temporal, no sentido de que quanto maior a resolução temporal, menor deve ser o valor de $\tau_{best}$. Para os vídeos testados, verificou-se que $\tau_{best} = 100$ resultou em um bom compromisso entre falsos positivos e falsos negativos. Também com base em experimentos, verificou-se que $w = 15$ e $\rho_V = 3$ oferecem um bom compromisso entre falsos positivos e falsos negativos. Note que a escolhoa para $\rho_V$ está fortemente relacionada à banda do *kernel* do *mean-shift*. Ambas constantes devem ser definidas em termos do espectro esperado de movimento relativo entre os objetos. Quando menor for movimento, maior deve ser $\rho_V$ (e menor a banda do *kernel*); da mesma forma, quanto maior for o movimento, menor deve ser $\rho_V$ (e maior deve ser a banda do *kernel*). A escolha de um valor para $w$ é discutida abaixo.

Escolheu-se utilizar apenas os $w$ melhores "casamentos" na janela do contexto para reduzir a influência de partículas próximas às bordas de movimento, já que seus contextos são afetados por movimentos diferentes em regiões adjacentes. No entanto, se $w$ é muito pequeno, pequenas transições entre regiões adjacentes não seriam detectadas como mudanças de contexto. Um exemplo de mudança de contexto em uma janela $3 \times 3$ é ilustrada na Figura 4.4. Note que, com um pequeno $w$, uma mudança de contexto de partícula poderia não ser detectada se a transição é lenta o suficiente para obter alguns poucos bons "casamentos".

Seja $\Omega_p = \{\omega_p^1, \omega_p^2, \ldots, \omega_p^{n_\omega}\}$ o conjunto ordenado de índices de quadros onde possíveis mudanças de contexto ocorrem para a partícula $p$. O tempo de vida da partícula $p$ é então dividido em intervalos:

$$\{[1, \omega_p^1), [\omega_p^1, \omega_p^2), ..., [\omega_p^{n_\omega - 1}, \omega_p^{n_\omega}), [\omega_p^{n_\omega}, n_t]\} \tag{4.8}$$

Lembre que uma partícula deve se mover consistentemente com o movimento coletivo das outras partículas em seu meta-grupo $M_b$, e sua vizinhança espacial não deve mudar muito. Para determinar se uma mudança de contexto realmente ocorreu (e a trajetória da partícula foi incorretamente calculada), seguimos o movimento e a localização espacial da partícula $p$, comparando-a a protótipos dos meta-grupos (conforme descrito abaixo), nos intervalos mostrados na Eq. (4.8). Assim, a similaridade entre os padrões de movimento da partícula $p$ e o protótipo de cada meta-grupo $M_b$ é dada por:

$$S(p,b,t) = e^{-(\frac{D_m(p,b,t)}{2\cdot\sigma_m^2}+\frac{D_s(p,b,t)}{2\cdot\sigma_s^2})}, \tag{4.9}$$

onde $D_m(p,b,t)$ e $D_s(p,b,t)$ denotam as diferenças de movimento e de localização espacial, respectivamente, entre a partícula $p$ e o protótipo do meta-grupo $M_b$ no quadro de tempo $t$, e são definidos abaixo. Na prática, os desvios padrões $\sigma_m$ e $\sigma_s$ representam a influência de cada uma das diferenças $D_m(p,b,t)$ e $D_s(p,b,t)$ no cálculo da similaridade $S(p,b,t)$. Os desvios padrões $\sigma_m$ e $\sigma_s$ foram fixados em '1', com base em experimentos. A diferença de movimento $D_m(p,b,t)$ é definida como:

$$D_m(p,b,t) = \sqrt{(u_p(t) - u^{[b]}(t))^2 + (v_p(t) - v^{[b]}(t))^2},$$

onde $u_p(t) = x_p(t) - x_p(t-1)$, $v_p(t) = y_p(t) - y_p(t-1)$, e $u^{[b]}(t)$ e $v^{[b]}(t)$ são, respectivamente, as componentes horizontal e vertical do vetor de movimento representativo do meta-grupo $M_b$ entre os quadros de tempo $t$ e de tempo $t-1$. Este vetor de movimento representativo é computado da seguinte forma:

1. Computa-se o conjunto de vetores de movimento $\{[u_p(t), v_p(t)] | \forall p \subset M_b\}$;
2. Computa-se a ordem reduzida dos respectivos vetores de movimento, em relação ao vetor de referência dado pelos valores mínimos de deslocamento horizontal e vertical das partículas, dentre todas as partículas pertencentes ao meta-grupo $M_b$: $[\texttt{min}(u_p(t) | \forall p \subset M_b), \texttt{min}(v_q(t) | \forall q \subset M_b)]$;
3. O vetor correspondente à mediana das distâncias ordenadas $\|\cdot\|_2$ em relação ao vetor de referência é atribuído a $[u^{[b]}(t), v^{[b]}(t)]$.

A diferença de localização espacial para a partícula $p$, $D_s(p,b,t)$, é definida como:

$$D_s(p,b,t) = \tfrac{1}{k} \cdot \textstyle\sum_{q=1}^{k} \sqrt{(x_p(t) - x_{p,q}^{[b]}(t))^2 + (y_p(t) - y_{p,q}^{[b]}(t))^2},$$

onde $\{(x_{p,q}^{[b]}(t), y_{p,q}^{[b]}(t)) | q = 1, ..., k\}$ são as coordenadas espaciais das $k$ partículas mais próximas que pertencem ao meta-grupo $M_b$ em relação à partícula $p$ no quadro de tempo $t$.

Para determinar a qual meta-grupo a partícula $p$ deveria ser atribuída no intervalo $[\omega_p^1, \omega_p^2)$, verifica-se qual meta-grupo $M_b$ maximiza a seguinte medida de similaridade (ver Eq. (4.9)) neste intervalo:

$$b_{max} = \max_b \sum_{t=\omega_p^1}^{\omega_p^2 - 1} S(p,b,t). \tag{4.10}$$

Note que se uma mudança de contexto é detectada, isso não necessariamente implica em uma mudança de meta-grupo após a validação. Isso apenas fornece um marcador que nos dá uma evidência de que a estimativa da trajetória da partícula pode estar incorreta. A análise da coerência espacial e de movimento expressa na Eq. (4.10) é que indica se uma mudança de meta-grupo ocorreu.

### 4.3.3 Filtragem Espacial

O estágio final no processo de classificação é a filtragem espacial das partículas. O objetivo da filtragem espacial é eliminar *outliers* e grupos de partículas adjacentes que não são significativos (ou seja, atribuídos a pequenos fragmentos de regiões isoladas).

Para representar a adjacência espacial de partículas, a triangulação de Delaunay $DT(t)$ é computada com base nas posições $(x_p(t), y_p(t))$ das partículas em cada quadro de tempo $t$. Duas partículas são consideradas adjacentes se elas compartilham uma aresta na triangulação $DT(t)$. A associação entre partículas adjacentes é representada atribuindo-se pesos binários às arestas de $DT(t)$; nesse caso, uma aresta recebe '1' se ela conecta duas partículas pertencentes ao mesmo meta-grupo, ou ela recebe '0' se conecta partículas pertencentes a meta-grupos diferentes. Todos os componentes conexos de $DT(t)$ são examinados[3] e aqueles muito pequenos são atribuídos a outros meta-grupos. Em nossos experimentos, consideramos que um componente conexo é muito pequeno caso tenha menos de 20 partículas. Este limiar depende da resolução do vídeo, e deve ser maior para vídeos com maior resolução. Os componentes considerados pequenos são atribuídos aos meta-grupos que compartilham mais arestas com peso '0', ou seja, aos meta-grupos com os quais compartilham a maior borda espacial.

## 4.4 Extração da Segmentação Densa

Em diversas tarefas de visão computacional e processamento de imagens, e em diversos problemas de codificação de vídeo, é necessário extrair uma representação densa da segmentação de movimento. Isso significa que devemos determinar quais pixels são atribuídos a cada objeto em movimento.

Neste estágio conhecemos as posições das partículas em cada quadro, assim como os rótulos de meta-grupos para cada partícula. Então, necessitamos apenas atribuir cada pixel ao meta-grupo de partículas mais similar. Para executar esta tarefa, os pixels são comparados com conjuntos de partículas em termos de movimento e proximidade espacial através de funções implícitas, conforme explicado a seguir. Seja $P = p_1, p_2, \ldots, p_{n_p^v}$ o conjunto inteiro das $n_p^v$ partículas do vídeo, e $\lambda_p \in \{1, \ldots, K\}$ o conjunto de rótulos que indicam para cada partícula, a qual meta-grupo $b = 1, \ldots, K$ esta partícula está associada. Então, para cada partícula $p$ em um quadro de tempo $t$, representada por suas coordenadas espaciais $(x_p^t, y_p^t)$, é atribuído um *kernel* gaussiano:

$$G_p^t(x, y, u, v, t) = \frac{1}{(2\pi)^2 |\Sigma|^{1/2}} \exp\left( -\frac{1}{2} \begin{bmatrix} x - x_p^t \\ y - y_p^t \\ \bar{u}(x, y, t) - \hat{u}_p^t \\ \bar{v}(x, y, t) - \hat{v}_p^t \end{bmatrix}^{\mathrm{T}} \Sigma^{-1} \begin{bmatrix} x - x_p^t \\ y - y_p^t \\ \bar{u}(x, y, t) - \hat{u}_p^t \\ \bar{v}(x, y, t) - \hat{v}_p^t \end{bmatrix} \right), \tag{4.11}$$

onde $\hat{u}_p^t = x_p^t - x_p^{t-1}$ e $\hat{v}_p^t = y_p^t - y_p^{t-1}$ representam o deslocamento da partícula $p$ no quadro de tempo $t$, e $\Sigma$ é a matrix de covariância, dada por:

$$\Sigma = \begin{bmatrix} \sigma_S & 0 & 0 & 0 \\ 0 & \sigma_S & 0 & 0 \\ 0 & 0 & \sigma_M & 0 \\ 0 & 0 & 0 & \sigma_M \end{bmatrix},$$

[3] Duas partículas estão no mesmo componente conexo se e apenas se existe um caminho entre elas composto apenas de arestas de peso '1'.

sendo $\sigma_S$ e $\sigma_M$ os desvios padrões espacial e de movimento. Na prática, $\sigma_S$ e $\sigma_M$ representam a influência da localização espacial e do movimento, respectivamente, na resposta do *kernel* gaussiano $G_p^t(x, y, u, v, t)$. Estes coeficientes são escolhidos com base em um compromisso entre precisão e suavidade das bordas dos objetos, e pode ser modificado de acordo com a aplicação. Quanto menor for o valor de $\sigma_S$, mais precisa a será a borda. Quando maior for o valor de $\sigma_S$, mais suave a será a borda. Por outro lado, $\sigma_M$ controla o peso da informação de movimento utilizada na classificação dos pixels. Valores grandes de $\sigma_M$ devem ser evitados pois isso causa fragmentação das regiões de um objeto. No presente trabalho, $\sigma_S$ e $\sigma_M$ foram fixados em 50 e 1, com base em experimentos. Uma matriz $\Sigma$ diagonal foi utilizada pois considera-se que as duas coordenadas espaciais $x$ e $y$ e as componentes $u$ e $v$ do vetor de deslocamento são independentes, *a priori*. Isso significa que a cada partícula será associada uma função implícita, que define a verossimilhança da atribuição de um pixel $(x, y, t)$ ao padrão de movimento representado pela partícula $p$ em um espaço de 4 dimensões (duas dimensões espaciais e duas dimensões de movimento). Com isso, pixels localizados próximos à partícula $p$, com movimento similar à partícula $p$, vão produzir grandes valores de $G_p^t(x, y, \bar{u}, \bar{v}, t)$, enquanto pixels distantes da partícula $p$, ou com vetores de movimento muito distintos, irão produzir pequenos valores de $G_p^t(x, y, \bar{u}, \bar{v}, t)$. Para atribuir um pixel a um meta-grupo, este pixel é comparado com todas as partículas de cada meta-grupo utilizando a Eq. (4.11), e atribui-se o pixel ao meta-grupo de partículas que fornece a maior soma de funções implícitas de partículas[4]. Um pixel de uma imagem $I(x, y, t)$ é então atribuído ao meta-grupo $M_N$ de partículas que maximiza a soma de gaussianas na posição correspondente do pixel:

$$\max_N \sum_{\{p|\lambda_p=N\}} G_p^t(x, y, \bar{u}(x, y, t), \bar{v}(x, y, t), t). \tag{4.12}$$

Note que as componentes do fluxo ótico ($\bar{u}(x, y, t)$ and $\bar{v}(x, y, t)$) são utilizadas aqui como informação de movimento a nível de pixel, e comparadas com o movimento das partículas $(\hat{u}_p^t, \hat{v}_p^t)$ através de funções implícitas dadas pela soma de gaussianas. O pixel é então atribuído ao meta-grupo contendo partículas próximas ao pixel que estão se movendo mais coerentemente com relação ao fluxo ótico local, ou seja, o máximo da Eq. (4.12). Quanto mais precisa é a estimativa do fluxo ótico, mais precisos são os contornos dos objetos segmentados. A utilização de somas de gaussianas resulta em contornos espacialmente suaves, mesmo quando poucas partículas estão disponíveis junto às bordas de movimento dos objetos.

O resultado final da atribuição dos pixels aos meta-grupos de partículas em todos os quadros do vídeo pode ser visto como um conjunto de volumes (representando objetos) no domínio espaço-temporal, onde pixels são representados por voxels.

## 4.5 Conclusões

Neste Capítulo foi proposto um método de segmentação de vídeos em objetos constituintes que apresentam movimento coerente. O método proposto combina características de métodos diretos e de métodos baseados em feições. O movimento é inicialmente computado a partir de conjuntos esparsos de pontos (partículas), permitindo a identificação

[4] Alternativamente, podemos comparar um pixel apenas com as $k$ partículas mais próximas. Nos experimentos feitos neste trabalho, cada pixel foi comparado às 30 partículas mais próximas, obtendo-se resultados virtualmente idênticos aos obtidos comparando pixels com todas as partículas, com a vantagem de economizar o tempo de computação da extração da segmentação densa em mais de 90%.

de padrões de movimento temporalmente longos. Contudo, ao invés de computar trajetórias de partículas de forma independente, estas são conectadas, reduzindo a influência de *outliers*. Uma abordagem de agrupamento de conjuntos é utilizada para segmentar as trajetórias de partículas. Esta abordagem resolve dois problemas críticos da segmentação de partículas simultaneamente: 1) lidar com um grande conjunto de dados (um vídeo com cerca de 100 quadros pode conter mais de 10000 partículas); e 2) comparar trajetórias de partículas com tempos de vidas distintos (e muitas vezes sem interseção). Uma segmentação densa é obtida comparando-se o movimento e a localização espacial dos pixels com conjuntos de partículas já segmentadas.

O método de segmentação proposto aqui não impõe restrições quanto ao modelo de câmera, tipo de objeto ou suavidade de trajetórias, identifica múltiplos objetos com movimento distintos, e não necessita de tratamento especial para objetos que sofrem oclusão ou que surjam na cena. Ainda, apesar da correspondência entre partículas ser feita apenas para conjuntos esparsos de pontos, entre pares de quadros vizinhos, a análise/segmentação do movimento é efetuada ao longo de vários quadros simultaneamente, permitindo que padrões de movimento longos sejam distinguidos, mesmo que em parte do vídeo as bordas de movimento sejam inexistentes.

Contudo, o método pode ser sensível a alguns parâmetros, em especial à banda do *kernel* do algoritmo *mean-shift* (COMANICIU; MEER, 2002), que restringe a amplitude de movimentos relativos detectáveis entre objetos.

# 5 UMA PROPOSTA PARA A CODIFICAÇÃO DE VÍDEOS BASEADA EM SEGMENTAÇÃO

Na abordagem de codificação descrita neste Capítulo, é utilizada como ponto de partida a segmentação de movimento de um vídeo obtida pelo método não-supervisionado descrito no Capítulo 4 e em (SILVA; SCHARCANSKI, 2010). Além dos rótulos da segmentação para cada pixel (ver Seção 4.4), são utilizados também os rótulos das partículas computadas pelo método de segmentação (ver Seção 4.3), bem como a posição de cada partícula em cada quadro (ver Seção 4.2). Como partículas são representadas com precisão sub-pixel e têm longos tempos de vida, são indicadores esparsos do movimento dos objetos, e são utilizadas neste trabalho para estimar as transformações afins empregadas nas predições de movimento, conforme será visto na Seção 5.2. Os dois tipos de quadros utilizados, I (*intra-coded*) e B (*bi-directional coded*), são codificados de formas diferentes. As partículas associadas aos objetos são utilizadas para estimar as predições de movimento para cada objeto nos quadros B, através de transformações afins (ou seja, cada quadro B é reconstruído através da compensação de movimento obtida a partir dos dois quadros I adjacentes e dos erros de predição). Os quadros I são empilhados, formando grupos de quadros I que são codificados diretamente através de uma versão 3-D do algoritmo ST-SPIHT (ver Seção 5.1). Cada objeto do vídeo é predito nos quadros B com base nos quadros I adjacentes, de instantes de tempo $t_B - \Delta t_{PI}$ e $t_B + \Delta t_{FI}$ ($t_B - \Delta t_{PI} < t_B < t_B + \Delta t_{FI}$), utilizando transformações afins. Os erros de predição nos quadros B entre $t_B - \Delta t_{PI}$ e $t_B + \Delta t_{FI}$ são representados por *bistreams* separados.

Definimos aqui um GOP (*Group of Pictures*) como sendo um bloco auto-contido de quadros I e B representando parte do vídeo. A Figura 5.1 ilustra um GOP de nove quadros com uma distância de três quadros entre quadros I consecutivos. O tamanho do GOP e a distância entre quadros I consecutivos é fixa, e o primeiro e o último quadros do GOP são sempre codificados como quadros I. No exemplo da Figura 5.1, os quadros dos instantes de tempo $t$, $t+4$ e $t+8$ são empilhados e codificados pelo ST-SPIHT 3-D, enquanto os erros de predição para cada objeto nos quadros B de instantes de tempo $t+1$, $t+2$, $t+3$, $t+5$, $t+6$ e $t+7$ são empilhados e codificados separadamente pelo ST-SPIHT 3-D (um *bitstream* para cada objeto).

Seja ainda GOI (*Group of I frames*) um grupo de quadros I codificados juntos, e GOB (*Group of B frames*) um conjunto de quadros B que são codificados juntos (ou seja, quadros I e B que pertencem a um mesmo GOP). Uma visão geral do processo de codificação proposto é mostrada na Figura 5.2. Durante a codificação de um GOP, os frames do GOI são primeiramente codificados e então decodificados, de forma a obter uma versão do GOI com os erros de decodificação. O GOI decodificado é utilizado como referência para as predições de movimento nos quadros do GOB. Os movimentos dos objetos em um GOB

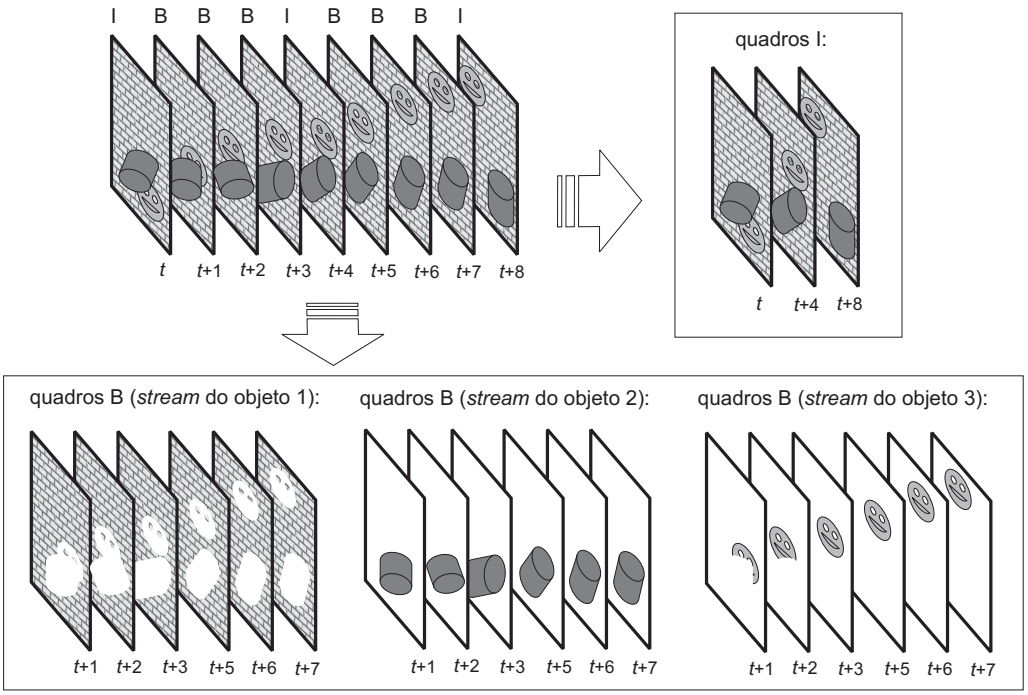



Figura 5.1: Ilustração de uma sequência de 9 quadros, com 3 quadros B intercalados entre quadros I consecutivos. A sequência é codificada com 4 *bitstreams* ST-SPIHT diferentes: um para os quadros I, e um para cada objeto nos quadros B.

são descritos por transformações afins baseadas no movimento de partículas associadas a cada objeto. Os erros de predição são codificados em um *bitstream* ST-SPIHT 3-D, junto à informação lateral (parâmetros das transformações afins para cada objeto em cada quadro do GOB). O processo de decodificação é executado de forma direta, decodificando o GOI, computando as predições de movimento através dos coeficientes das transformações afins, e então decodificando o erro de predição juntamente com as máscaras dos objetos, para cada objeto no GOB, conforme mostrado na Figura 5.3.

Os parâmetros necessários para a decodificação dos dados de entrada são armazenados no início do *bitstream*. Inicialmente são armazenados:

1. Número de linhas - 2 *bytes*;
2. Número de colunas - 2 *bytes*;
3. Número de quadros - 2 *bytes*;
4. Número de objetos - 1 *byte*;
5. Máximo nível de decomposição *wavelet* (ver seção 5.1) - 1 *byte*;
6. Nível de codificação de forma[1] - 1 *byte*;
7. Tamanho do GOP - 1 *byte*;
8. Tamanho do GOB - 1 *byte*.

[1]O nível de codificação de forma no algoritmo ST-SPIHT define o nível de quantização no qual o algoritmo força a codificação da máscara que define a forma do objeto para aqueles pixels que não foram codificados nos níveis de quantização anteriores. Mais detalhes na seção 5.1

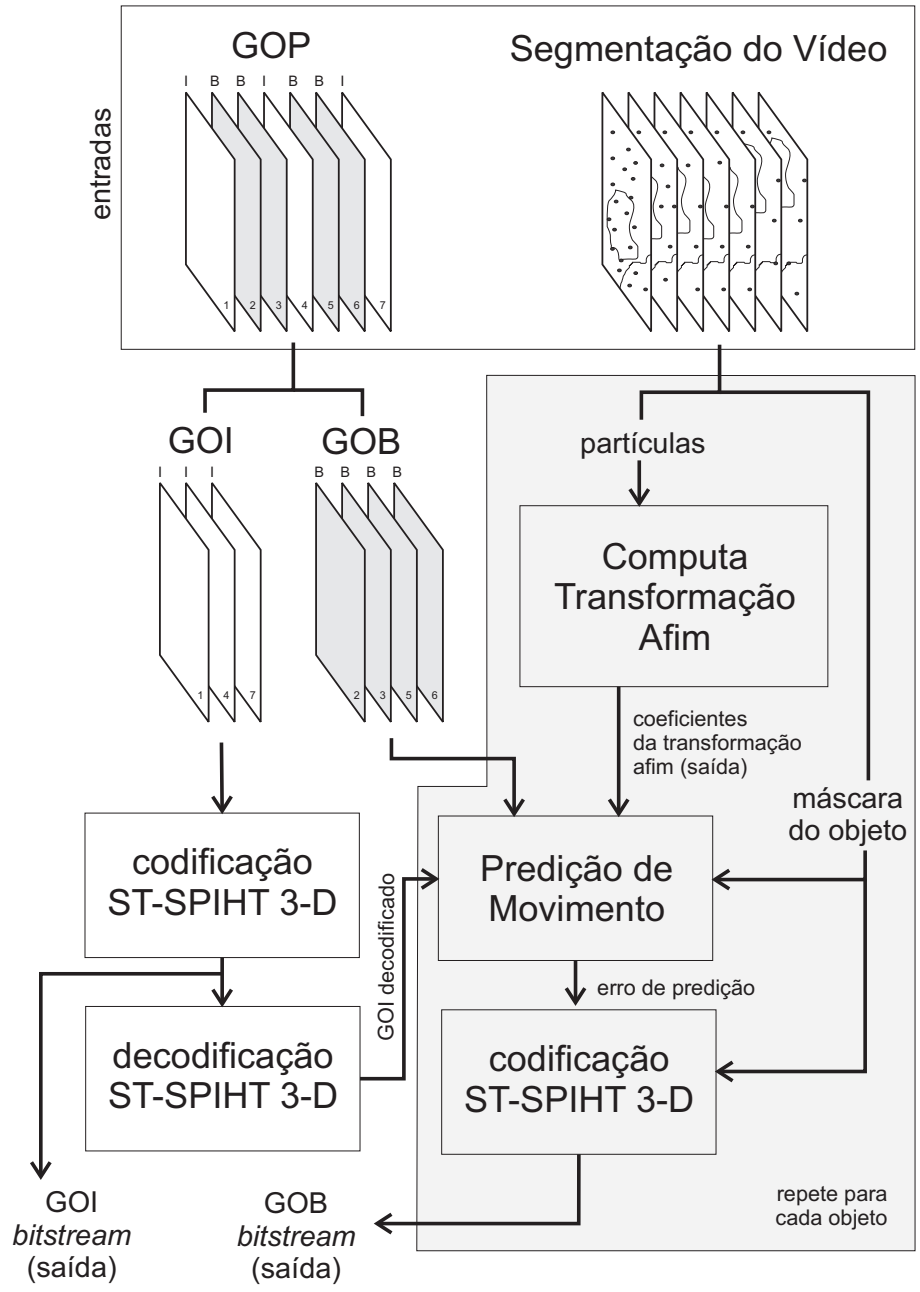



Figura 5.2: Visão geral do processo de codificação do método proposto.

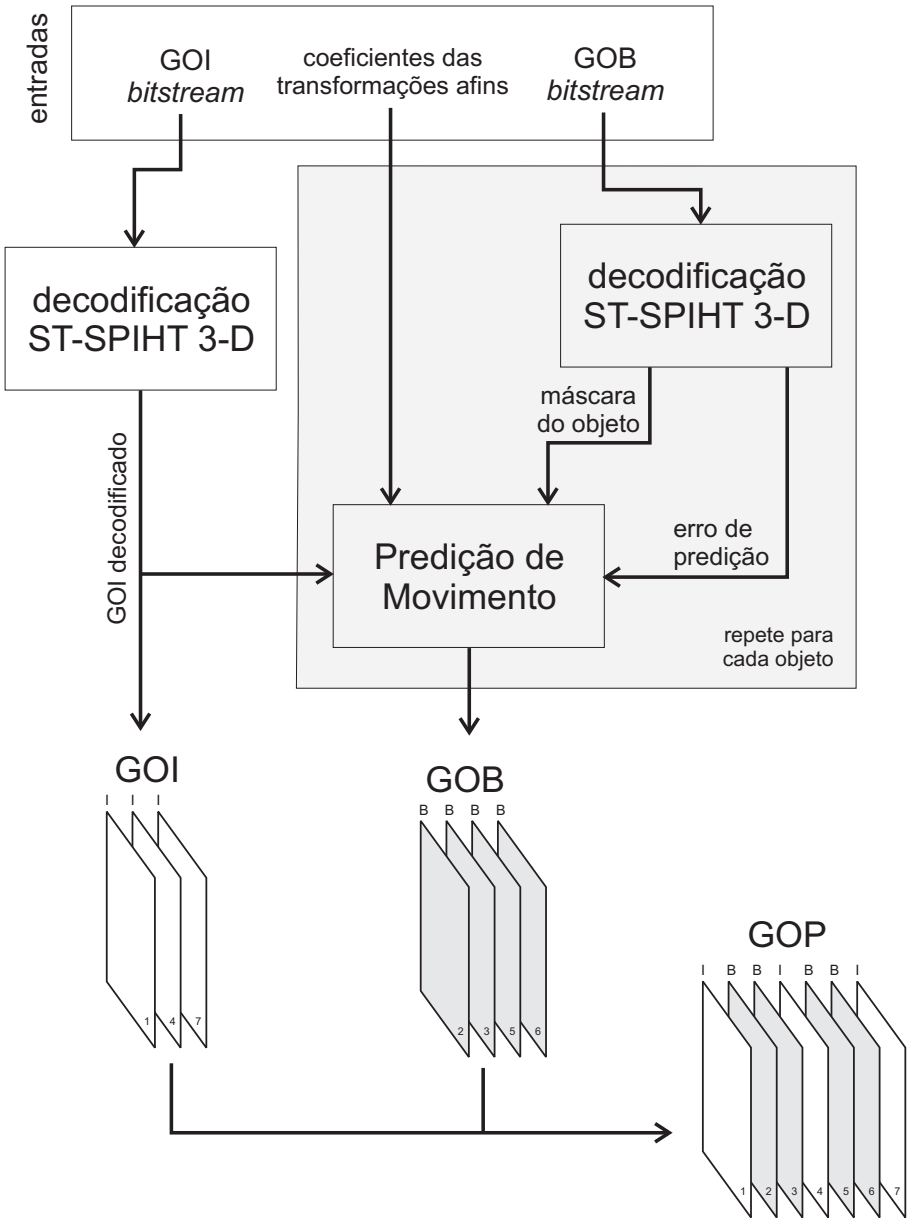



Figura 5.3: Visão geral do processo de decodificação do método proposto.

A seguir, os coeficientes das transformações afins para a predição de movimento nos quadros do GOB são armazenados. Para cada objeto de cada quadro no GOB, os coeficientes de duas transformações afins são armazenados, uma relativa ao quadro I anterior, e outra relativa ao próximo quadro I na sequência. Uma transformação afim é representada com 6 números binários de ponto flutuante com 32 bits cada (IEEE, 2008), dos quais 4 coeficientes formam a matriz de transformação linear e 2 coeficientes constituem o vetor de translação. Finalmente, para cada GOI e para cada objeto de cada GOB, *bitstreams* independentes são criados, conforme detalhado na Seção 5.3.

Nas próximas Seções, são detalhados o método ST-SPIHT 3-D utilizado (Seção 5.1), a estratégia de predição de movimento (Seção 5.2), e como os quadros I e B são codificados (Seção 5.3).

## 5.1 ST-SPIHT 3-D

Na abordagem proposta neste trabalho, foi utilizada uma extensão 3-D do método 2-D proposto por Martin *et al.* (MARTIN; LUKAC; PLATANIOTIS, 2006), que por sua vez extendeu o método SPIHT tradicional (SAID; PEARLMAN, 1996) codificando formas de objetos e coeficientes de uma transformada *wavelet* discreta (DWT) simultaneamente. De forma análoga ao método proposto por Martin *et al.* (MARTIN; LUKAC; PLATANIOTIS, 2006), em que as formas dos objetos são representados por máscaras binárias planares, onde um pixel é igual a '1' se ele está dentro do objeto e '0' se ele está fora do objeto, aqui as formas espaço-temporais dos objetos são representadas por máscaras binárias volumétricas, onde um voxel é '1' se ele está dentro do objeto e '0' se ele está fora do objeto.

O processo de codificação completo de um objeto consiste de três etapas:

1. **Pré-processamento**: a partir dos quadros originais do vídeo (ou dos erros de predição de movimento) e a segmentação densa do vídeo, são criados dois componentes: a máscara binária e a textura do objeto. A máscara volumétrica é representada por $M : \mathbb{Z}^3 \rightarrow \{0, 1\}$, onde $M(x, y, t) = 1$ indica que o pixel de coordenadas espaciais $x$ e $y$ no quadro de tempo $t$ está 'dentro' do objeto, enquanto $M(x, y, t) = 0$ indica que ele está 'fora' do objeto; a textura do objeto é representada por $I : \mathbb{Z}^3 \rightarrow \mathbb{R}^3$, onde $I(x, y, t) = [I(x, y, t)_Y, I(x, y, t)_{C_r}, I(x, y, t)_{C_b}]$ é um vetor contendo as três componentes de cor (no espaço $\mathrm{YC}_r\mathrm{C}_b$) do pixel de coordenadas espaciais $x$ e $y$ no quadro de tempo $t$. Ainda, $I(x, y, t) = [0, 0, 0] \forall (x, y, t)$ quando $M(x, y, t) = 0$; ou seja, a textura nas posições externas ao objeto é fixada em zero.

2. **SA-DWT**: Cada canal de cor de $I(x, y, t)$ é transformado por uma SA-DWT com sub-amostragem de referencial global, e os coeficientes são armazenados nas mesmas posições dos pixels do objeto, ou seja, onde $M(x, y, t)$ é igual a 1. Com isso é criado um campo vetorial $I_T : \mathbb{Z}^3 \rightarrow \mathbb{R}^3$ de coeficientes $I_T(x, y, t) = [I_T(x, y, t)_Y$, $I_T(x, y, t)_{C_r}$, $I_T(x, y, t)_{C_b}]$ da transformada. Esta etapa está descrita na Seção 5.1.1.

3. **Codificador ST-SPIHT 3-D**: uma extensão do algoritmo SPIHT é aplicada codificando simultaneamente os coeficientes da textura transformada $I_T$ e a máscara binária $M$, gerando um único *bitstream*. Esta etapa está detalhada na Seção 5.1.2.

A decodificação segue o processo inverso, executando o decodificador ST-SPIHT 3-D, aplicando-se a SA-DWT inversa e efetuando o pós-processamento dos dados (que pode

ser simplesmente uma conversão para o espaço de cores RGB, ou adicionar os resíduos decodificados à predição de movimento).

### 5.1.1 SA-DWT

O primeiro nível da SA-DWT é obtido aplicando-se uma DWT 1-D sucessivamente três vezes, uma em cada dimensão - horizontal, vertical e temporal - apenas nos segmentos de pixels (ou voxels) que estão dentro do objeto que está sendo codificado. Uma estatégia de sub-amostragem com referencial global é utilizada, de forma que o filtro passa-baixa é aplicado nas posições pares de pixels (ou voxels), e o filtro passa-alta é aplicado nas posições ímpares de pixels (ou voxels). Para cada segmento em cada dimensão, um segmento com coeficientes passa-baixa e passa-alta intercalados é computado e colocado na posição do segmento original (essa estratégia é chamada de *in-place lifting DWT* (MARTIN; LUKAC; PLATANIOTIS, 2006)). Assim, a máscara $M$ com a forma do objeto no domínio espacial permanece a mesma no domínio da transformada.

Um exemplo da aplicação de um nível da SA-DWT para um objeto simples é mostrado na Figura 5.4. Na primeira linha aparece a máscara do objeto, representado em um volume de tamanho $8 \times 8 \times 4$ pixels (ou voxels), sendo que os pixels (ou voxels) que estão dentro do objeto aparecem em preto e numerados, enquanto os que estão fora do objeto aparecem em branco. Na segunda linha aparecem as representações dos coeficientes (L para coeficientes passa-baixa e H para coeficientes passa-alta) obtidos após a aplicação da SA-DWT na dimensão horizontal. Note que a SA-DWT é aplicada isoladamente em cada segmento horizontal contíguo de pixels de cada quadro, dentro do objeto. Para o primeiro quadro, mais especificamente, temos 5 segmentos contíguos:

1. Pixels 1, 2, 3 e 4;
2. Pixel 5;
3. Pixel 6;
4. Pixels 7, 8, 9 e 10;
5. Pixel 11.

A disposição dos coeficientes após a aplicação da SA-DWT na dimensão vertical é mostrada na terceira linha. Para o primeiro quadro, por exemplo, temos 6 segmentos verticais contíguos:

1. Pixels 1, 5 e 7;
2. Pixel 2;
3. Pixel 8;
4. Pixel 3;
5. Pixel 9;
6. Pixels 4, 6, 10 e 11.

Os coeficientes correspondentes à aplicação da SA-DWT vertical (L para passa-baixa e H para passa-alta) aparecem na segunda letra em cada pixel. Por fim, a aplicação da SA-DWT na dimensão temporal resulta na disposição de coeficientes mostrada na quarta linha. Os coeficientes correspondentes à aplicação da SA-DWT temporal (L para passa-baixa e H para passa-alta) aparecem na terceira letra em cada pixel. Para a segunda linha, por exemplo, temos 2 segmentos temporais contíguos:

1. Pixels 12, 34 e 63;

2. Pixels 35 e 64.

Após aplicar a SA-DWT nas dimensões horizontal, vertical e temporal, temos oito tipos de coeficientes: LLL, LHL, HLL, HHL, LLH, LHH, HLH e HHH. Deve ser observado que nesta estratégia de SA-DWT, o filtro passa-baixa é aplicado apenas aos pixels (ou voxels) em posições pares, e o filtro passa-alta é aplicado apenas aos pixels (ou voxels) em posições ímpares, com base em uma referência global, independente de onde um segmento contíguo de pixels inicia. Esta estratégia mantém uma correspondência de um-para-um entre a máscara do objeto no domínio espaço-temporal e no domínio da trasformada.

O segundo nível da SA-DWT é computado aplicando-se a mesma estratégia descrita acima para segmentos de coeficientes LLL consecutivos, e substituindo-os pelos correspondentes coeficientes do segundo nível. Uma ilustração do segundo nível da SA-DWT para o mesmo objeto da Figura 5.4 é mostrado na Figura 5.5. A primeira linha mostra os coeficientes obtidos após o primeiro nível de decomposição da SA-DWT, ou seja, após a aplicação da SA-DWT nas dimensões horizontal, vertical e temporal, em todos os pixels contidos no objeto; a segunda linha mostra os coeficientes obtidos após o segundo nível de decomposição da SA-DWT, ou seja, após a aplicação da SA-DWT nas dimensões horizontal, vertical e temporal, em todos os coeficientes LLL do primeiro nível. As localidades que terminam com '1' correspondem aos coeficientes do primeiro nível de decomposição, e localidades que terminam com '2' correspondem aos coeficientes do segundo nível. Se desejarmos computar o terceiro nível de decomposição da SA-DWT, basta utilizar a mesma estratégia a segmentos de coeficientes LLL2 consecutivos, e assim por diante.

### 5.1.2 Codificador ST-SPIHT 3-D

Para codificar a informação de textura dos objetos, os valores de pixels (ou erros de predição de movimento) são representados no espaço de cores $YC_bC_r$, sem sub-amostragem cromática (4:4:4), e são codificados utilizando uma árvore de orientação espacial com uma relação de parentesco que é uma extensão 3-D da estrutura proposta em (KASSIM; LEE, 2003). A Figura 5.6 ilustra esta estrutura e mostra o arranjo espacial dos coeficientes de uma SA-DWT com dois níveis de decomposição. Dada uma decomposição SA-DWT de altura $n_{max}$, os coeficientes LLL$n_{max}$ do canal Y constituem o nível mais alto da árvore (as raízes), e:

- Coeficientes LLL$n_{max}$ têm como filhos os coeficientes HLL$n_{max}$, LHL$n_{max}$, HHL$n_{max}$, LLH$n_{max}$, HLH$n_{max}$, LHH$n_{max}$ e HHH$n_{max}$ do mesmo canal;

- Coeficientes LLL$n_{max}$ do canal Y que não têm filhos no mesmo canal terão como descendentes os coeficientes LLL$n_{max}$ dos canais de crominância;

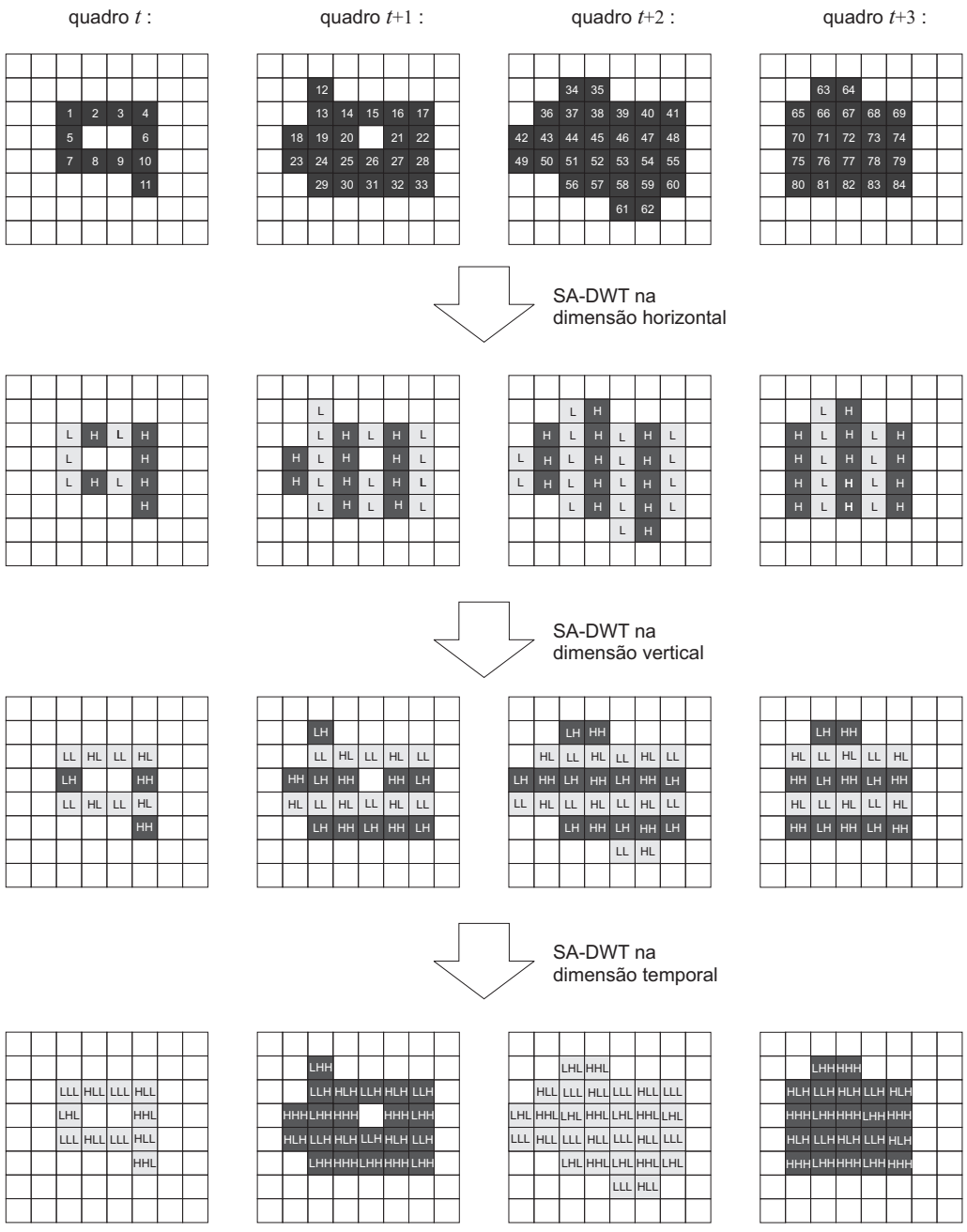



Figura 5.4: Exemplo da aplicação de um nível da SA-DWT em um objeto simples representado em um volume de tamanho $8 \times 8 \times 4$.

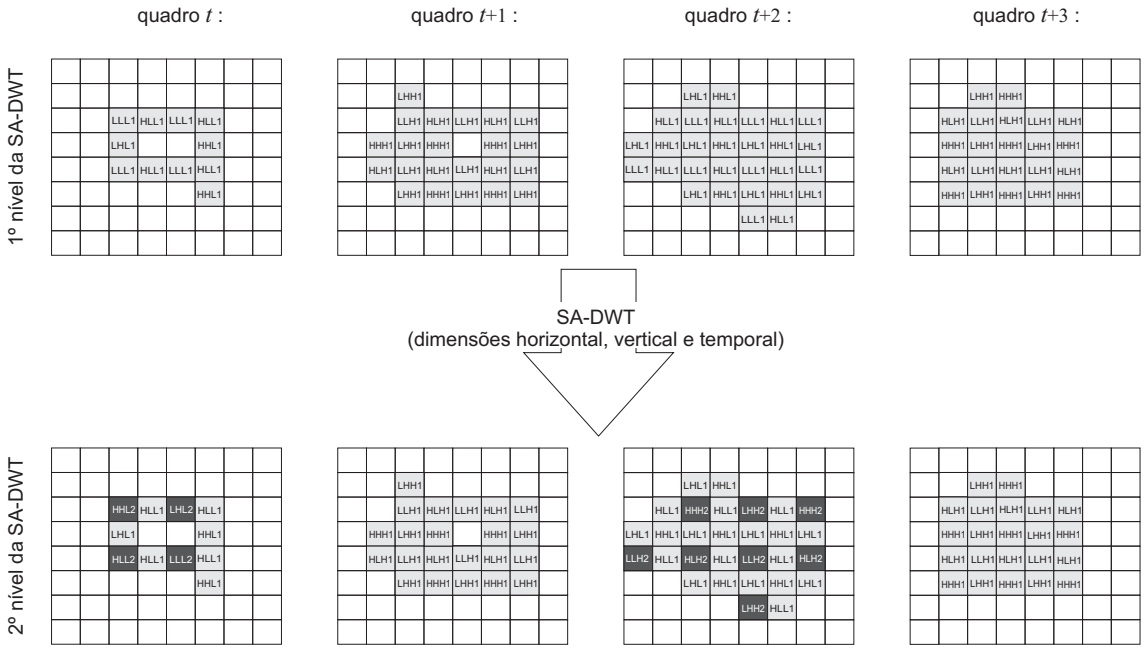



Figura 5.5: Exemplo da aplicação do segundo nível da SA-DWT em um objeto simples representado em um volume de tamanho $8 \times 8 \times 4$.

- Coeficientes passa-alta de nível $n$ (HLL$n$, LHL$n$, HHL$n$, LLH$n$, HLH$n$, LHH$n$ e HHH$n$) têm como filhos os coeficientes de nível $n-1$ do mesmo tipo no mesmo canal.

Com esta estratégia, todos os coeficientes da SA-DWT são alcançáveis em sub-árvores a partir dos coeficientes LLL$n_{max}$ do canal Y, e sucessivos refinamentos são obtidos descendo através da estrutura de árvore. Esta estrutura de parentesco impõe uma sub-amostragem cromática implícita, já que os coeficientes dos canais $C_b$ e $C_r$ estão em um nível abaixo dos coeficientes correspondentes do canal Y. Em outras palavras, dado um certo nível da árvore de orientação espacial, os canais cromáticos estão representados com metade da resolução do canal Y. No entanto, após alcançar a máxima resolução no canal Y, os canais cromáticos ainda podem ser refinados até que alcancem a resolução máxima.

Seja $\mathbf{G} = \{(x,y,t)|M(x,y,t) = 1\}$ o conjunto de todas as coordenadas dos pixels que estão dentro do objeto, e $\overline{\mathbf{G}} = \{(x,y,t)|M(x,y,t) = 0\}$ o complemento de $\mathbf{G}$, contendo todas as coordenadas dos pixels que estão fora do objeto. As definições do algoritmo SPIHT (SAID; PEARLMAN, 1996) são utilizadas aqui, tais como a lista de pixels significativos (LSP), a lista de pixels não-significativos (LIP) e a lista de conjuntos não-significativos (LIS). Ainda, seja $\mathbf{T}$ um conjunto de nodos da árvore de orientação espacial. Um nodo em particular pode ser referenciado por $(x,y,t,c)$, onde $x$ e $y$ são coordenadas espaciais, $t$ é o tempo de um determinado quadro, e $c$ é um dos canais de cor (Y, $C_r$ ou $C_b$). A significância $\mathbf{S}_b(\mathbf{T})$ de um nodo $\mathbf{T}$ no plano de bits $b$ é definida por:

$$\mathbf{S}_b(\mathbf{T}) = \begin{cases} 1, & \max_{(x,y,t,c)\in\mathbf{T}}\{|I_T(x,y,t)_c|\} \geq 2^b, \\ 0, & \text{de outra forma} \end{cases} \tag{5.1}$$

As definições dos seguintes conjuntos de nodos também são utilizadas:

- $\mathbf{O}(x,y,t,c)$: conjunto de todos os nodos filhos do nodo $(x,y,t,c)$;

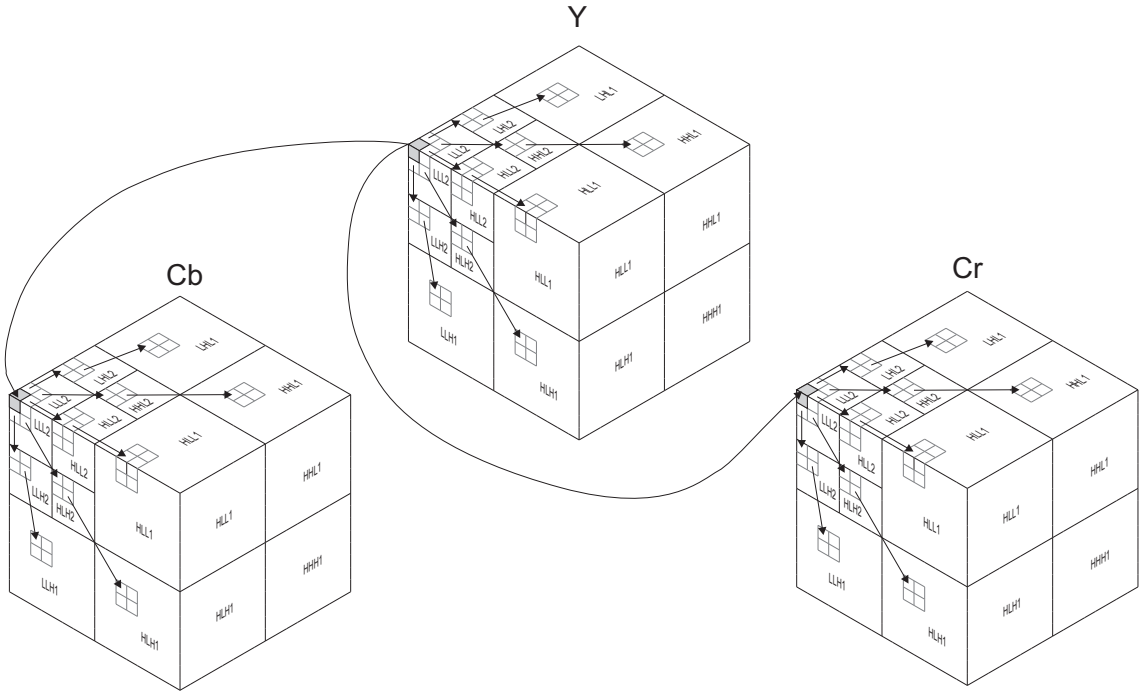



Figura 5.6: Relação de parentesco da árvore de orientação espacial.

- $\mathbf{D}(x, y, t, c)$: conjunto de todos os nodos da sub-árvore de descendentes do nodo $(x, y, t, c)$;
- $\mathbf{H}$: conjunto de todos os nodos-raíz (ou seja, todos os coeficientes LLL$n_{max}$ do canal Y);
- $\mathbf{L}(x, y, t, c) = \mathbf{D}(x, y, t, c) - \mathbf{O}(x, y, t, c)$.

Uma entrada do tipo A na lista de conjuntos não-significativos (LIS) se refere a $\mathbf{D}(x, y, t, c)$, enquanto uma entrada do tipo B na mesma lista se refere a $\mathbf{L}(x, y, t, c)$.

O algoritmo ST-SPIHT utiliza três funções "$\alpha$-tests": $\alpha_p$ ("$\alpha$ *pixel test*") indentifica se um pixel está dentro ou fora do objeto sendo codificado:

$$\alpha_p(x, y, t) = \begin{cases} 1, & (x, y, t) \in \mathbf{G} \\ 0, & \text{de outra forma;} \end{cases} \tag{5.2}$$

$\alpha_{SD}$ ("$\alpha$ *set-discard test*") identifica se conjuntos $\mathbf{T}$ de nodos (pixels) estão totalmente fora do objeto sendo codificado:

$$\alpha_{SD}(\mathbf{T}) = \begin{cases} 0, & \mathbf{T} \subseteq \overline{\mathbf{G}} \\ 1, & \text{de outra forma;} \end{cases} \tag{5.3}$$

$\alpha_{SR}$ ("$\alpha$ *set-retain test*") identifica se conjuntos $\mathbf{T}$ de nodos (pixels) estão totalmente dentro do objeto sendo codificado:

$$\alpha_{SR}(\mathbf{T}) = \begin{cases} 1, & \mathbf{T} \subseteq \mathbf{G} \\ 0, & \text{de outra forma.} \end{cases} \tag{5.4}$$

O pseudo-código do codificador ST-SPIHT 3-D é mostrado na Figura 5.7, e sua sub-rotina para codificação de forma de um conjunto de nodos (SCS - *Shape Code Set*) é

mostrada na Figura 5.8. Seguindo a mesma filosofia do método SPIHT original (SAID; PEARLMAN, 1996), o algoritmo utiliza um método de ordenação dos coeficientes, e esta ordenação não é transmitida explicitamente. Ao invés disso, seu funcionamento se baseia no fato de que o caminho de execução do codificador pode ser replicado pelo decodificador ao se transmitir os resultados das comparações em seus pontos de ramificação, desde que o codificador e o decodificador utilizem o mesmo algoritmo de ordenação. Na verdade, não é necessário ordenar todos os coeficientes, mas apenas selecionar a cada passo aqueles que estão dentro de um determinado intervalo de magnitude.

O codificador recebe como entrada a máscara $M$ do objeto, os coeficientes da SA-DWT $I_T$, o nível de codificação da forma $\lambda$, e a máscara binária $N : \mathbb{Z}^3 \rightarrow \{0, 1\}$, cuja função será discutida mais adiante. O nível de codificação da forma $\lambda$ determina o nível de quantização (ou plano de bits) no qual a rotina força a codificação de pixels que ainda não estão nem no conjunto $\mathbf{G}$ nem no conjunto $\overline{\mathbf{G}}$, ou seja, pixels de coordenada $(x, y, t)$ cujo valor da máscara $M(x, y, t)$ ainda não poderia ser determinado pelo decodificador. A sub-rotina SCS (Figura 5.8) tem justamente esta função.

No algoritmo ST-SPIHT 3-D (ver Figura 5.7), assim como no algoritmo ST-SPIHT original (MARTIN; LUKAC; PLATANIOTIS, 2006), além das comparações envolvendo a magnitude dos coeficientes (através do teste de significância da Eq. (5.1)), também são transmitidos (escritos) os resultados dos "$\alpha$-tests" (Eq. (5.2), (5.3) e (5.4)), possibilitando que o decodificador reconstrua as formas dos objetos à medida que os nodos da árvore de orientação espacial são analisados. Os resultados dos "$\alpha$-tests" são chamados de "bits de forma". Note que, pelo algoritmo da Figura 5.7, toda a vez que um nodo ou uma sub-árvore da árvore de orientação espacial é visitado pelo codificador ST-SPIHT 3-D, a máscara volumétrica é analisada nas localidades correspondentes, e os bits de forma são enviados para o decodificador sinalizando se estas localidades estão totalmente dentro, parcialmente dentro, ou totalmente fora do objeto que está sendo codificado. À medida que os bits de forma são enviados, o decodificador gradualmente reconstrói a máscara do objeto. À medida que a máscara do objeto é reconstruída, o codificador não necessita enviar todos os bits de forma, já que a máscara do objeto nas localidades de alguns nodos já é conhecida.

Porém, o algoritmo ST-SPIHT original (MARTIN; LUKAC; PLATANIOTIS, 2006) trata apenas do problema da codificação de um único objeto individual. Ao se utilizar este algoritmo para codificar diversos objetos não sobrepostos de um vídeo, haverá redundância na representação das formas dos objetos. No presente trabalho, uma modificação é proposta, de modo que esta redundância seja explorada, reduzindo a quantidade de bits de forma que necessitam ser enviados ao decodificador. Para isso, assumimos que os túneis dos objetos são codificados em uma ordem pré-definida. Isso significa que, após uma máscara de objeto ser decodificada, o decodificador é capaz de saber que os pixels desta máscara não pertencem a nenhum outro objeto do vídeo. Com isso, os bits de forma relacionados a esses pixels não precisam ser codificados para os próximos objetos. Assim, apenas para o primeiro objeto a ser codificado os conjuntos $\mathbf{G}$ e $\overline{\mathbf{G}}$ são inicializados vazios. Ao codificar o segundo objeto de um sequência, o conjunto $\overline{\mathbf{G}}$ já é inicializado com as coordenadas dos pixels pertencentes ao primeiro objeto. Ao codificar o terceiro, $\overline{\mathbf{G}}$ é inicializado com as coordenadas dos pixels do primeiro e do segundo objetos, e assim por diante. O último objeto, por sua vez, já terá sua forma conhecida ($\mathbf{G}$ e $\overline{\mathbf{G}}$ completos) e por isso nenhum bit de forma precisa ser escrito no *bitstream*. A máscara $N$, utilizada como entrada do algoritmo da Figura 5.7, é utilizada para manter um registro dos pixels cujo objeto correspondente já é conhecido. Então, para cada pixel de coordenadas $(x, y, t)$ que

**Entrada:** $I_T$, $M$, $N$ e $\lambda$

1. **Inicialização:**

   - Calcule nível inicial de quantização $b = b_{max} = \left\lfloor \log_2 \left( \max_{(x,y,t,c)} \{|I_T(x,y,t,c)|\} \right) \right\rfloor$
   - Inicialize LSP $= \emptyset$; LIP $= \mathbf{H}$; LIS $= \{(x,y,t,c)$ tipo-A$|(x,y,t,c) \in \mathbf{H}, \mathbf{D}(x,y,t,c) \neq \emptyset\}$
   - Inicialize $\mathbf{G}$ e $\overline{\mathbf{G}}$ de acordo com $M$ e $N$.

2. **Passo de ordenação:**

   (a) Para cada entrada $(x,y,t,c) \in$ LIP:

      i. Se $\alpha_p(x,y,t,c)$ ainda não é conhecido, então escreva $\alpha_p(x,y,t,c)$, e atualize $\mathbf{G}$ e/ou $\overline{\mathbf{G}}$

      ii. Se $\alpha_p(x,y,t,c) = 1$, então:

         A. Escreva $S_b(x,y,t,c)$

         B. Se $S_b(x,y,t,c) = 1$, então mova $(x,y,t,c)$ para a LSP e escreva o sinal de $I_T(x,y,t,c)$

      iii. Se $\alpha_p(x,y,t,c) = 0$ então remova $(x,y,t,c)$ da LIP

   (b) Para cada entrada $(x,y,t,c) \in$ LIS, faça: se for uma entrada do tipo A, $\mathbf{T} = \mathbf{D}(x,y,t,c)$; se for do tipo B, $\mathbf{T} = \mathbf{L}(x,y,t,c)$

      i. Se $b \geq \lambda$ e a forma não é totalmente conhecida, então:

         A. Se $\alpha_{SD}(\mathbf{T})$ não é conhecido, então escreva $\alpha_{SD}(\mathbf{T})$, e atualize $\mathbf{G}$ e/ou $\overline{\mathbf{G}}$

         B. Se $\alpha_{SD}(\mathbf{T}) = 0$, então remova $(x,y,t,c)$ da LIS e vá para a próxima entrada da LIS (vá para o passo 2.b)

         C. Se $\alpha_{SD}(\mathbf{T}) = 1$, então:

            - Se $\alpha_{SR}(\mathbf{T})$ ainda não é conhecido, então escreva $\alpha_{SR}(\mathbf{T})$ e atualize $\mathbf{G}$ e/ou $\overline{\mathbf{G}}$
            - Se $\alpha_{SR}(\mathbf{T}) = 0$ e $b = \lambda$, então execute SCS($\mathbf{T}$)

      ii. Se a forma é totalmente conhecida e $\alpha_{SD}(\mathbf{T}) = 0$, então remova $(x,y,t,c)$ da LIS e vá para a próxima entrada da LIS (vá para o passo 2.b)

      iii. Se for uma entrada do tipo A e $\alpha_{SD}(\mathbf{T}) = 1$, então:

         A. Escreva $S_b(\mathbf{D}(x,y,t,c))$

         B. Se $S_b(\mathbf{D}(x,y,t,c)) = 1$, então:

            - Para cada $(x',y',t',c') \in \mathbf{O}(x,y,t,c)$:
               - Escreva $S_b(x',y',t',c')$
               - Se $S_b(x',y',t',c') = 1$, então adicione $(x',y',t',c')$ na LSP e escreva o sinal de $I_T(x',y',t',c')$
               - Se $S_b(x',y',t',c') = 0$, então adicione $(x',y',t',c')$ na LIP
            - Se $\mathbf{L}(x,y,t,c) \neq \emptyset$, então mova $(x,y,t,c)$ para o fim da LIS como uma entrada do tipo B; senão, remova $(x,y,t,c)$ da LIS

      iv. Se for uma entrada do tipo B e $\alpha_{SD}(\mathbf{T}) = 1$, então:

         A. Escreva $S_b(\mathbf{L}(x,y,t,c))$

         B. Se $S_b(\mathbf{L}(x,y,t,c)) = 1$, então:

            - Adicione cada $(x',y',t',c') \in \mathbf{O}(x,y,t,c)$ no fim da LIS como uma entrada do tipo A
            - Remova $(x,y,t,c)$ da LIS

3. **Passo de refinamento:**

   - Para cada $(x,y,t,c) \in$ LSP, exceto aqueles considerados significantes no passo de ordenação atual, escreva o $b$-ésimo bit mais significativo de $I_T(x,y,t,c)$

4. **Atualização do passo de quantização:**

   - Decremente $b$ de 1 e vá para o passo 2.

Figura 5.7: Codificador ST-SPIHT 3-D.

**Entrada:** sub-árvore $\mathbf{T}$ com raiz $(x, y, t, c)$

1. Se $(x, y, t, c)$ é uma entrada do tipo A, então:

   (a) Para cada $(x', y', t', c') \in \mathbf{O}(x, y, t, c)$:

      i. Se $\alpha_p(x', y', t', c')$ ainda não é conhecido, então escreva $\alpha_p(x', y', t', c')$ e atualize $\mathbf{G}$ e/ou $\overline{\mathbf{G}}$

      ii. Se $\mathbf{D}(x', y', t', c') \neq \emptyset$, então:

         A. Se $\alpha_{SD}(\mathbf{D}(x', y', t', c'))$ ainda não é conhecido, então escreva $\alpha_{SD}(\mathbf{D}(x', y', t', c'))$ e atualize $\mathbf{G}$ e/ou $\overline{\mathbf{G}}$

         B. Se $\alpha_{SD}(\mathbf{D}(x', y', t', c')) = 0$, então termine o processamento de $\mathbf{D}(x', y', t', c')$

         C. Se $\alpha_{SD}(\mathbf{D}(x', y', t', c')) = 1$, então:

            - Se $\alpha_{SR}(\mathbf{D}(x', y', t', c'))$ ainda não é conhecido, então escreva $\alpha_{SR}(\mathbf{D}(x', y', t', c'))$ e atualize $\mathbf{G}$ e/ou $\overline{\mathbf{G}}$
            - Se $\alpha_{SR}(\mathbf{D}(x', y', t', c')) = 0$, então vá para o passo 1 tratando $\mathbf{D}(x', y', t', c')$ como uma nova entrada do tipo A

2. Se $(x, y, t, c)$ é uma entrada do tipo B, então:

   (a) Para cada $(x', y', t', c') \in \mathbf{O}(x, y, t, c)$, vá para o passo 1 tratando $\mathbf{D}(x', y', t', c')$ como uma nova entrada do tipo A

Figura 5.8: Rotina para a codificação da forma de um conjunto de nodos (SCS).

já foi codificado em um objeto anteriormente, $N(x, y, t) = 1$, enquanto que para todos os pixels $(x', y', t')$ ainda desconhecidos, $N(x', y', t') = 0$ Esta estratégia requer que as formas dos objetos sejam decodificadas na mesma ordem em que elas foram codificadas. E mais, a forma de um objeto deve estar completamente decodificada antes que o próximo objeto comece a ser decodificado. Embora isto implique em alguma limitação da escalabilidade, isto não é um problema para diversas aplicações. Na verdade, isto seria uma limitação apenas em transmissões com baixíssimas taxas de bits, ou quando um objeto único precisa ser transmitido e ele não é o primeiro codificado. Em nossos experimentos, os objetos são codificados em ordem decrescente do número de pixels, ou seja, objetos maiores são codificados e decodificados antes. Isto se deve ao fato de que o custo relativo da codificação da forma de um objeto é maior para objetos menores, ou seja, o número de bits por pixel gastos na codificação da máscara é relativamente maior. Assim, esta estratégia de codificar objetos maiores antes ajuda a equilibrar o custo em termos de bits por pixel para todos os objetos.

Durante a execução do codificador, dois contadores acompanham o número de posições em que o decodificador potencialmente sabe que estão dentro do objeto (ou seja, $|\mathbf{G}|$) ou fora do objeto (ou seja, $|\overline{\mathbf{G}}|$). Quando um destes contadores alcança o número total de pixels que está dentro do objeto ($M(x, y, t) = 1$) ou fora do objeto ($M(x, y, t) = 0$), a forma do objeto já pode ser totalmente inferida pelo decodificador, e a partir daí nenhum bit de forma é mais codificado.

O decodificador segue exatamente o mesmo fluxo de execução do codificador. No entanto, para ser inicializado, o decodificador deve antes receber os seguintes parâmetros, que são armazenados no início do *bitstream* do ST-SPIHT 3-D, e são individuais para cada objeto:

1. Número de pixels no interior do objeto ($|\mathbf{G}|$) - 4 *bytes*;

2. Nível de quantização inicial ($b_{max}$) - 1 *byte*.

## 5.2 Predição de Movimento

Seja $I(t)$ o quadro original do vídeo (com três componentes de cor) no instante de tempo $t$, e sejam $t_B$, $t_{PI}$ e $t_{FI}$, respectivamente, o instante de tempo do quadro B que desejamos predizer, o instante de tempo do quadro I imediatamente anterior, e o instante de tempo do quadro I imediatamente posterior. Por exemplo, se $t_B = 6$ no exemplo da Figura 5.1, então $t_{PI} = 5$ e $t_{FI} = 9$. Seja $F_k^{(t_{PI})}(t_B)$ o quadro $I(t_{PI})$ deformado pela transformação afim computada com base no deslocamento das partículas associadas ao $k$-ésimo objeto do vídeo, do quadro no instante $t_{PI}$ para o quadro no instante $t_B$. De forma similar, seja $F_k^{(t_{FI})}(t_B)$ o quadro $I(t_{FI})$ deformado pela transformação afim computada com base no deslocamento das partículas associadas ao $k$-ésimo objeto do vídeo, do quadro no instante $t_{FI}$ para o quadro $t_B$. A predição $F_k(t_B)$ do $k$-ésimo objeto no quadro do instante $t_B$ é obtida interpolando $F_k^{(t_{PI})}(t_B)$ e $F_k^{(t_{FI})}(t_B)$:

$$F_k(t_B) = \frac{F_k^{(t_{PI})}(t_B) \times (t_{FI} - t_B) + F_k^{(t_{FI})}(t_B) \times (t_B - t_{PI})}{t_{FI} - t_{PI}}. \tag{5.5}$$

O $k$-ésimo objeto é interpolado no quadro B com base nas duas imagens deformadas $F_k^{(t_{PI})}(t_B)$ e $F_k^{(t_{FI})}(t_B)$ para se obter predições mais precisas.

As imagens deformadas $F_k^{(t_{PI})}(t_B)$ e $F_k^{(t_{FI})}(t_B)$ são obtidas por transformações afins estimadas a partir do movimento do $k$-ésimo objeto. Seja $\tau_k(t, t_B)$ a transformação afim que representa o movimento do $k$-ésimo objeto do instante $t \in \{t_{PI}, t_{FI}\}$ ao instante $t_B$. Os coeficientes da transformação afim são calculados com base na estimativa de movimento do conjunto de partículas atribuídas ao $k$-ésimo objeto. Para estimar o movimento do $k$-ésimo objeto entre os quadros dos instantes $t_{PI}$ e $t_B$ nós utilizamos as partículas que coexistem no intervalo $[t_{PI}, t_B]$; e para estimar o movimento do $k$-ésimo objeto do quadro do instante $t_B$ para o quadro do instante $t_{FI}$ utilizamos as partículas que coexistem no intervalo $[t_B, t_{FI}]$. As partículas que coexistem nas interseções $P_k(t_{PI}) \cap P_k(t_B)$ e $P_k(t_{FI}) \cap P_k(t_B)$ são utilizadas para computar as transformações afins $\tau_k(t_{PI}, t_B)$ e $\tau_k(t_{FI}, t_B)$, respectivamente, onde $P_k(t)$ é o conjunto de partículas associadas ao $k$-ésimo objeto no quadro de instante $t$. A seguir, as transformações afins $\tau_k(t_{PI}, t_B)$ e $\tau_k(t_{FI}, t_B)$ são utilizadas para computar as imagens deformadas $F_k^{(t_{PI})}(t_B)$ e $F_k^{(t_{FI})}(t_B)$, respectivamente. Seja $C$ a matriz $3 \times 2$ da transformação afim $\tau_k(t_{PI}, t_B)$ com 6 coeficientes:

$$C = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ b_1 & b_2 \end{bmatrix},$$

onde $a_{11}$, $a_{12}$, $a_{21}$ e $a_{22}$ são os coeficientes da matriz de transformação linear, e $b_1$ e $b_2$ são os dois componentes do vetor de translação. Os coeficientes da matriz $C$ são obtidos pela seguinte minimização:

$$C = \arg\min ||[\ X_{PI} \quad Y_{PI}\ ] - [\ X_B \quad Y_B \quad 1\ ]C||_2,$$

onde $X_{PI}$ e $Y_{PI}$ são vetores contendo as coordenadas horizontais e verticais, respectivamente, do conjunto de partículas $P_k(t_{PI}) \cap P_k(t_B)$ no quadro do instante $t_{PI}$, e $X_B$ e $Y_B$ são vetores contendo as coordenadas horizontais e verticais, respectivamente, do conjunto de partículas $P_k(t_{PI}) \cap P_k(t_B)$ no quadro do instante $t_B$. Para computar a minimização acima, precisamos conhecer o deslocamento de pelo menos três partículas. Se o conjunto $P_k(t_{PI}) \cap P_k(t_B)$ tiver cardinalidade menor que 3, a transformação afim correspondente

não é calculada, e a predição se torna simplesmente $F_k(t_B) = F_k^{(t_{FI})}(t_B)$, ou seja, apenas o quadro I posterior é utilizado como referência na predição. A mesma idéia se aplica ao conjunto $P_k(t_{FI}) \cap P_k(t_B)$, se ele tem cardinalidade menor que 3 (ou seja, se temos menos de 3 partículas que coexistem nos quadros de instantes $t_{FI}$ e $t_B$, utilizamos apenas o quadro I anterior como referência para a predição).

## 5.3 Codificação dos quadros I e B

Após calcular as predições de movimento para todos os objetos nos quadros B de instantes $t_B$ de um GOB, ou seja, $\{F_k(t_B)|k = 1, \ldots, K\}$, onde $K$ é o número de objetos, codificamos os erros de predição $E_k(t_B) = I_k(t_B) - F_k(t_B)$ junto com a máscara do objeto utilizando o ST-SPIHT 3-D, onde $I_k(t_B)$ é o $k$-ésimo objeto no quadro original do instante $t_B$. Assim, codificar um GOB consiste em computar as predições de movimento para cada objeto no conjunto de quadros B, e então codificar os erros de predição junto com a forma do objeto utilizando a abordagem ST-SPIHT 3-D.

Conforme mencionado anteriormente, cada GOI é codificado em um *stream* único e independente (ver Figura 5.1), como se contivesse um único objeto. Como a forma deste objeto é conhecida (ou seja, corresponde a todos os pixels de todos os quadros do GOI), os bits de forma não precisam ser codificados.

## 5.4 Conclusões

Neste Capítulo foi proposto um método de codificação de vídeos baseado em objetos, com o objetivo de validar o método de segmentação de vídeos descrito no Capítulo 4. O método utilizado para a codificação de vídeos visa obter eficiência na codificação, sem abandonar algumas funcionalidades da codificação baseada em objetos, tais como a transmissão progressiva e o controle de taxa de transmissão individual para cada objeto.

A abordagem ST-SPIHT 3-D utilizada aqui permite que a forma dos objetos no domínio espaço-temporal seja transmitida simultaneamente com as texturas, explorando as redundâncias espacial e temporal. Além do mais, ao codificar diversos quadros em um único *bitstream* ST-SPIHT, cada bit decodificado aprimora a qualidade de vários quadros simultaneamente, resultando em um controle de taxa de transmissão mais eficiente do que os métodos que codificam e decodificam cada quadro de forma independente. Ainda, uma compensação de movimento eficiente é obtida ao utilizar informação sobre o movimento das partículas, devidamente associadas aos diferentes objetos do vídeo, obtida durante o processo de segmentação do vídeo (ver Capítulo 4).

Porém, o método proposto possui algumas limitações, conforme discutido a seguir. Em primeiro lugar, a informação lateral não é codificada de forma eficiente; parâmetros das transformações afins, utilizadas para a compensação de movimento, são representados por números de ponto flutuante e inseridos diretamente no *bitstream*, sem qualquer processamento visando a compactação destes dados. Isto, somado ao fato de que a forma de um objeto deve ser totalmente conhecida antes que o próximo objeto possa começar a ser decodificado, limita o desempenho do método proposto na codificação em baixas taxas de bits. Em segundo lugar, como os quadros I são utilizados como referência para a compensação de movimento nos quadros B, a taxa de bits utilizada nos quadros I deve ser determinada durante a codificação, o que limita parcialmente o potencial de escalabilidade do método.

# 6 RESULTADOS EXPERIMENTAIS

Neste Capítulo são apresentados resultados experimentais obtidos a partir dos métodos propostos nos Capítulos 4 e 5. Comparações com outros métodos são feitas e discutidas.

## 6.1 Resultados experimentais do método de segmentação de vídeos proposto

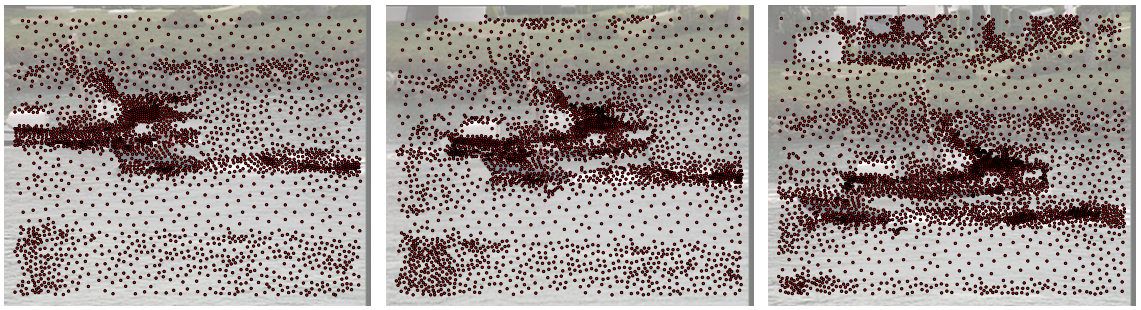
### 6.1.1 Vídeos reais

O método de segmentação de vídeos proposto foi testado em onze vídeos reais. Destes, dez são vídeos clássicos (*bus*, *city*, *coastguard*, *flower*, *hall*, *husky*, *mobile*, *news*, *soccer* and *stefan*), e estão disponíveis no endereço `http://media.xiph.org/video/derf/`. Todos estes vídeos têm resolução CIF (288x352 pixels), e foram utilizadas sequências de 60 quadros de cada vídeo. O outro vídeo (*VCars*, com 51 quadros) tem resolução de 480x720 pixels e foi utilizado no trabalho de Sand e Teller (SAND; TELLER, 2006) para demonstrar o potencial do método de rastreamento de partículas, e está disponível no endereço `http://rvsn.csail.mit.edu/pv/data/input-video/`.

As Figuras de 6.1 a 6.6 mostram o resultado de vários passos intermediários do processo de segmentação, para os quadros 10, 30 e 50 do vídeo *coastguard*. Os quadros originais são mostrados na Figura 6.1. As partículas rastreadas são mostradas na Figura 6.2. Ao todo, 6548 partículas foram criadas ao longo da sequência de 60 quadros, com um tempo médio de vida de aproximadamente 27 quadros. Na Figura 6.3 são mostrados os agrupamentos de partículas obtidos a partir o método *mean-shift*, em pares de quadros adjacentes. Partículas de um mesmo grupo são mostradas com a mesma cor. Note que não há, nesta etapa, correspondência entre grupos em quadros distintos. Nos quadros 10 e 30 foram identificados 2 grupos de partículas, enquanto no quadro 50 foram identificados 3 grupos. Os meta-grupos de partículas, obtidos após o agrupamento de conjuntos, são mostrados na Figura 6.4. Neste momento, já existe uma correspondência entre os grupos ao longo do vídeo, ou seja, partículas representadas por uma mesma cor pertencem ao mesmo grupo, mesmo que em quadros distintos. O resultado final da segmentação de partículas, após as etapas de validação dos meta-grupos e filtragem espacial, é mostrado na Figura 6.5. Note que, no quadro 10, algumas partículas que estão sobre a água e após o agrupamento de conjuntos haviam sido erroneamente atribuídas ao meta-grupo correspondente ao barco maior - representadas por triângulos azuis - na Figura 6.4, após as etapas de pós-processamento foram re-atribuídas ao meta-grupo correspondente ao *background* da imagem - representado por cruzes verdes - na Figura 6.5. O resultado da segmentação densa (após a atribuição dos pixels da imagem a cada um dos meta-grupos de partículas)

(a) (b) (c)

Figura 6.1: Quadros (a) 10, (b) 30 e (c) 50 do vídeo *coastguard* original.



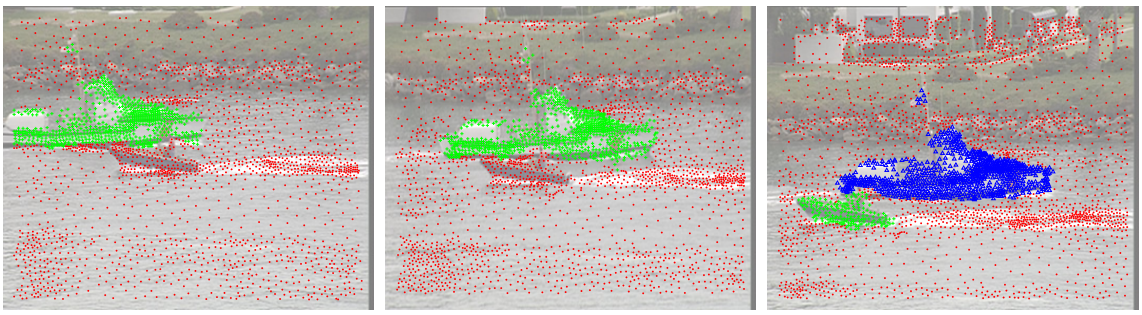
(a) (b) (c)

Figura 6.2: Rastreamento de partículas para os quadros (a) 10, (b) 30 e (c) 50 do vídeo *coastguard*.

é mostrado na Figura 6.6. A representação da segmentação dos objetos através de túneis - volumes espaço-temporais - é mostrada na Figura 6.7. Os túneis correspondentes aos dois barcos da sequência são mostrados através de dois pontos de vista diferentes. Note que os túneis são preservados ao longo do tempo e não sofrem fragmentação. Isto se deve à robustez do método de rastreamento utilizado, combinado à capacidade do método de segmentação de corrigir inconsistências, que nos levam a uma segmentação satisfatória, apesar do alto nível de ruído da sequência original e as flutuações da superfície da água.

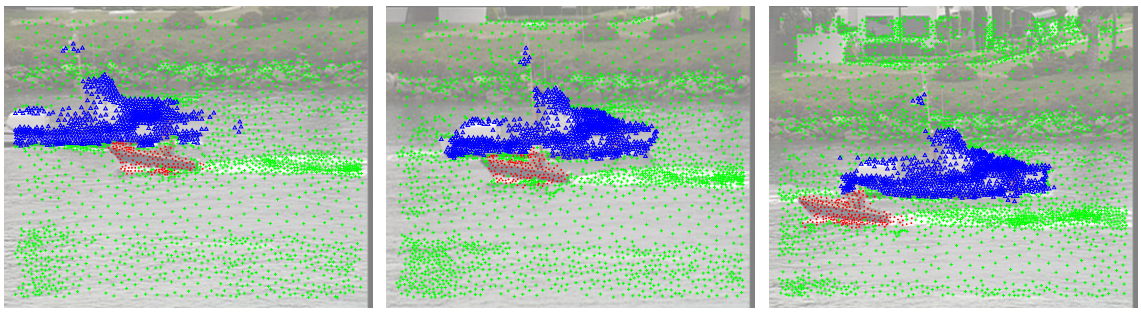
Os quadros 5, 15 e 25 originais do vídeo *VCars*, bem como os grupos individuais de partículas para os mesmos quadros, a segmentação final das partículas e a segmentação densa, são mostrados na Figura 6.8. Note que o efeito da projeção perspectiva neste vídeo é bastante nítido. Por essa razão, e também devido ao fato de que o *mean-shift* utiliza uma banda fixa, restringindo o espectro de movimentos de objetos detectados, não foi possível dinstinguir os dois carros que estão em primeiro plano nos quadros 5 e 15, utilizando apenas agrupamentos de pares de quadros individuais, conforme mostrado na segunda linha da Figura 6.8. No entanto, o agrupamento de conjuntos os identificou como objetos distintos, conforme mostrado na terceira linha da Figura 6.8. Isto ocorre porque o agrupamento de conjuntos reconhece grupos de partículas que tendem a ser agrupadas juntas ao longo de todo o vídeo, retornando resultados significativos mesmo quando bordas de movimento não são claras ou agrupamentos incorretos são encontrados. Esta propriedade também é importante na codificação de vídeos, já que a redundância dos objetos individuais é melhor explorada.

O vídeo *VCars* é um vídeo temporalmente simétrico. Ou seja, os quadros de 1 até 26



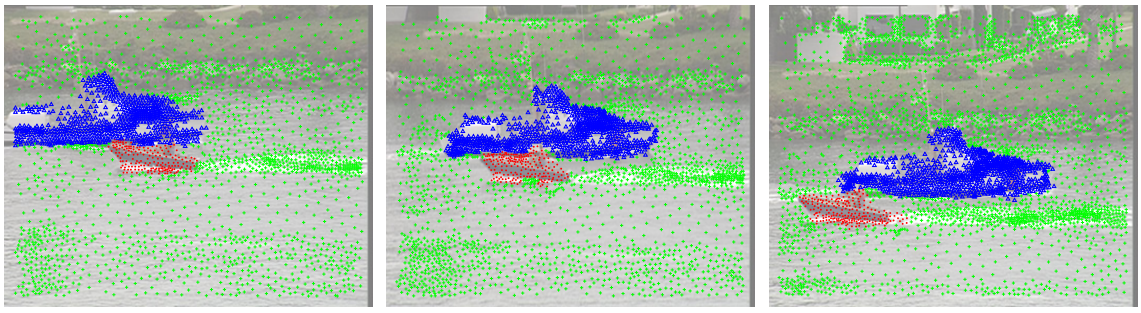
(a) (b) (c)

Figura 6.3: Agrupamentos de partículas entre pares de quadros vizinhos, para os quadros (a) 10, (b) 30 e (c) 50 do vídeo *coastguard*.



(a) (b) (c)

Figura 6.4: Meta-grupos de partículas, para os quadros (a) 10, (b) 30 e (c) 50 do vídeo *coastguard*.

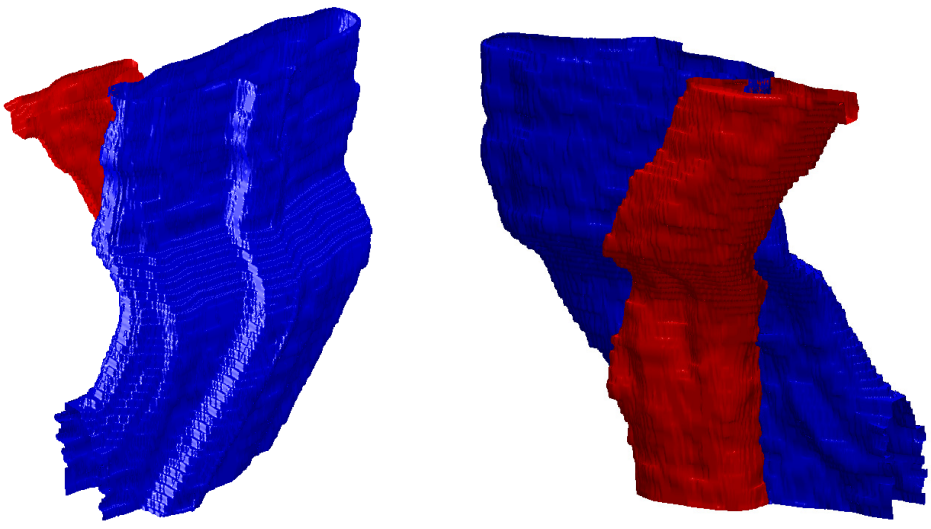


(a) (b) (c)

Figura 6.5: Agrupamento final de partículas após validação de meta-grupos e filtragem espacial, para os quadros (a) 10, (b) 30 e (c) 50 do vídeo *coastguard*.

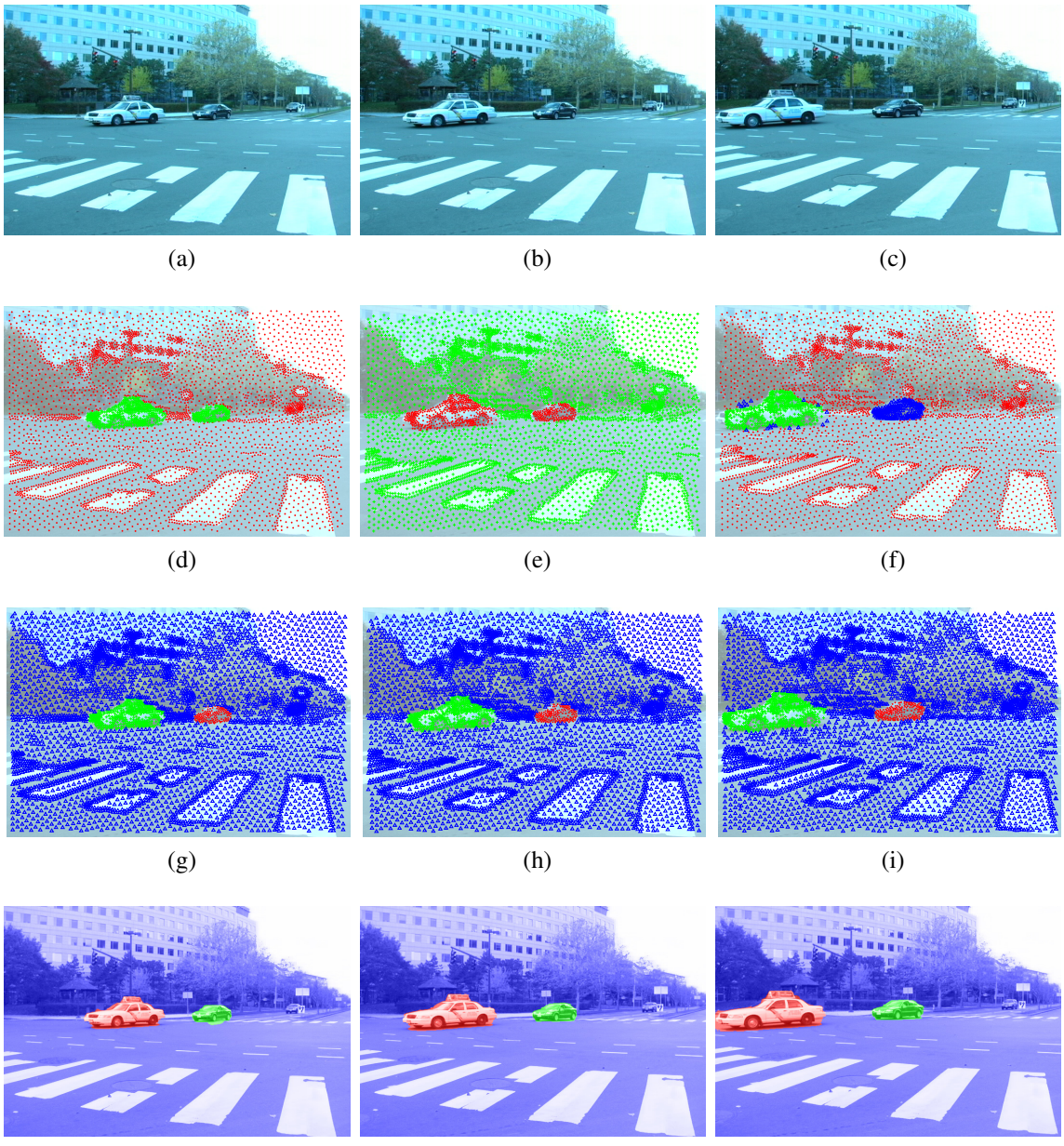
(a) (b) (c)

Figura 6.6: Segmentação densa para os quadros (a) 10, (b) 30 e (c) 50 do vídeo *coastguard*.



(a) (b)

Figura 6.7: Representação dos túneis para os barcos do vídeo *coastguard*.



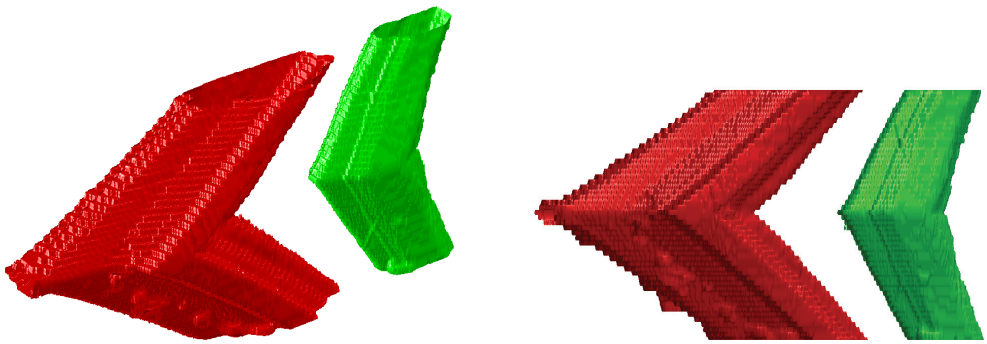
(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figura 6.8: Quadros 5 (primeira coluna), 15 (segunda coluna) e 25 (terceira coluna) do vídeo *VCars*: quadros originais (a),(b),(c); agrupamentos obtidos pelo método *mean-shift* (d),(e),(f); agrupamento final das partículas (g),(h),(i); e segmentação densa (j),(k),(l).



(a) (b)

Figura 6.9: Túneis gerados a partir da segmentação do vídeo *VCars* segundo dois pontos de vista diferentes.

são originais, enquanto de 27 a 51 contém os mesmos primeiros 25 quadros, porém na ordem inversa. Isto é útil para testar a consistência da segmentação, que deve ser também simétrica. A Figura 6.9 mostra os túneis gerados a partir desta sequência, que apresentam uma boa simetria. Se compararmos os rótulos dos pixels da primeira parte do vídeo (quadros de 1 a 25) com os correspondentes na segunda metade do vídeo (quadros de 51 a 27), temos uma taxa de correspondência entre rótulos de 99,84% (ou seja, apenas 0,16% dos pixels correspondentes têm rótulos diferentes). Já se compararmos apenas o primeiro com o último quadro, temos uma taxa de correspondência entre rótulos de 99,61% (ou seja, apenas 0,39% dos pixels correspondentes têm rótulos diferentes).

A Figura 6.10 mostra os resultados da segmentação para os quadros 10, 30 e 50 de outros quatro vídeos testados. Os dois primeiros vídeos (*city* e *flower*) apresentam uma segmentação satisfatória, enquanto os dois últimos (*news* e *soccer*) apresentam problemas, conforme discutido a seguir. A sub-segmentação obtida para o vídeo *soccer* se deve a uma combinação de dois problemas distintos: em primeiro lugar, o método de rastreamento de partículas utilizado falha ao tentar rastrear pontos que se movem muito rápido em regiões muito pequenas. Isso porque o fluxo ótico não consegue identificar movimentos rápidos muito localizados, e a função objetivo utilizada para otimizar a localização das partículas (Eq. (4.3)) não corrige o problema, pois um de seus termos força as partículas a se moverem junto com suas vizinhas que apresentam fluxo ótico semelhante. Em segundo lugar, quando corpos articulados têm movimentos muito complexos, a tendência é que o algoritmo do *mean-shift* gere uma sobre-segmentação. Isso se deve ao fato de que *mean-shift* utiliza uma banda fixa, e dessa forma um mesmo objeto articulado pode gerar uma variedade de movimentos com magnitudes e direções diversas. Dependendo do nível de complexidade e articulação dos movimentos, tal como ocorre com o movimento dos corpos no vídeo *soccer*, a sobre-segmentação pode ser tão grande que não é possível inferir uma correlação entre os grupos de partículas gerados pelo *mean-shift* em diferentes quadros, fazendo com que o desempenho do agrupamento de conjuntos seja bastante prejudicado. O problema poderia ser amenizado modificando o critério de distância mínima entre partículas (ver Seção 4.2.2), de forma a aumentar o número total de partículas, sob o custo de um maior esforço computacional. No vídeo *news*, a sub-segmentação se deve à limitação da banda fixa do *mean-shift*. Como os movimentos presentes neste vídeo são lentos, o *mean-shift* não é capaz de distinguir entre grupos de partículas com movimentos distintos sem que um ajuste da banda seja feito. Além do mais, por conter objetos que são corpos articulados de movimentos complexos, os grupos de partículas obtidos pelo *mean-*

*shift* não apresentam uma boa correlação entre si, dificultando o trabalho do agrupamento de conjuntos.

O método de segmentação proposto foi testado em um computador com processador Quad Core de 2.4GHz e 4GB de memória RAM, com Windows Vista. O cálculo do fluxo ótico foi implementado na liguagem C++, enquanto a estimativa de trajetórias de partículas, a segmentação de trajetórias de partículas e a extração da segmentação densa foram implementadas em Matlab. Para os vídeos de resolução CIF com 60 quadros o processo completo de segmentação leva cerca de 7 horas. Apesar de não constituir a maior parte do tempo de execução, acreditamos que o gargalo no desempenho do método proposto está no cálculo do fluxo ótico (cerca de 45 minutos), visto que o restante do processo não está implementado de forma eficiente e aparesenta um grande potencial de otimização. Todavia, ainda há um outro impedimento para a implementação em tempo real do método proposto: para que a segmentação das trajetórias das partículas possa ser iniciada, as trajetórias das partículas ao longo de todos os quadros da sequência já deve ser conhecida.

### 6.1.2 Vídeos sintéticos

Para analisar quantitativamente o método de segmentação proposto, foram utilizados dois vídeos sintéticos: um deles foi utilizado no trabalho de Ristivojevic e Konrad (RISTIVOJEVIC; KONRAD, 2006) para a extração de túneis espaço-temporais (*syntseq*) e consiste de 30 quadros, com uma resolução de 343x419 pixels em tons de cinza (8 bits por pixel); já o outro vídeo foi criado para este trabalho (*syntobjects*), e consiste de 30 quadros, com uma resolução 288x384 pixels representados no espaço RGB (24 bits por pixel).

A Figura 6.11 apresenta resultados experimentais para o vídeo *syntseq*. Este vídeo consiste de um objeto sintético em forma de grão de feijão que se move em relação a um *background* estático. Este objeto tem seu movimento descrito por uma transformação afim (com rotação, translação e escala), e sofre oclusão, induzindo uma mudança espacial topológica. Ristivojevic e Konrad propuseram um método para a extração de túneis de objetos, volumes de oclusão e volumes de exposição (RISTIVOJEVIC; KONRAD, 2006), e sugeriram que estes conceitos poderiam ser utilizados em uma próxima geração de métodos de compressão de vídeos. Então, comparamos os resultados obtidos por Ristivojevic e Konrad com a abordagem proposta neste trabalho. Os quadros 10 e 30 do vídeo *syntseq* são mostrados na primeira e na segunda colunas, respectivamente, da Figura 6.11. As segmentações de partículas correspondentes são mostradas nas Figuras 6.11(a) e 6.11(d). Os resultados da segmentação densa são mostrados nas Figuras 6.11(b) e 6.11(e), assim como os volumes de oclusão e volumes de exposição correspondentes, conforme proposto por Ristivojevic e Konrad (RISTIVOJEVIC; KONRAD, 2006). Os rótulos, do preto ao branco, representam: 1) *background*; 2) volume de oclusão do *background*; 3) volume de exposição do *background*; 4) objeto; e 5) volume de oclusão do objeto. Os resultados correspondentes obtidos por Ristivojevi e Konrad são mostrados nas Figuras 6.11(c) e 6.11(f). Os volumes de oclusão e volumes de exposição são obtidos no presente trabalho estimando-se transformações afins para cada região em cada quadro, e propagando estas regiões através da sequência utilizando estas transformações. Estas transformações afins são estimadas com base no movimento das partículas pertencentes à mesma região. Para comparar a exatidão da segmentação obtida por ambos os métodos, foi criada uma verdade de campo (*ground truth*) manualmente para este vídeo. Assim, foi obtida uma taxa de 99.75% de pixels de objetos corretamente classificados para o método proposto,

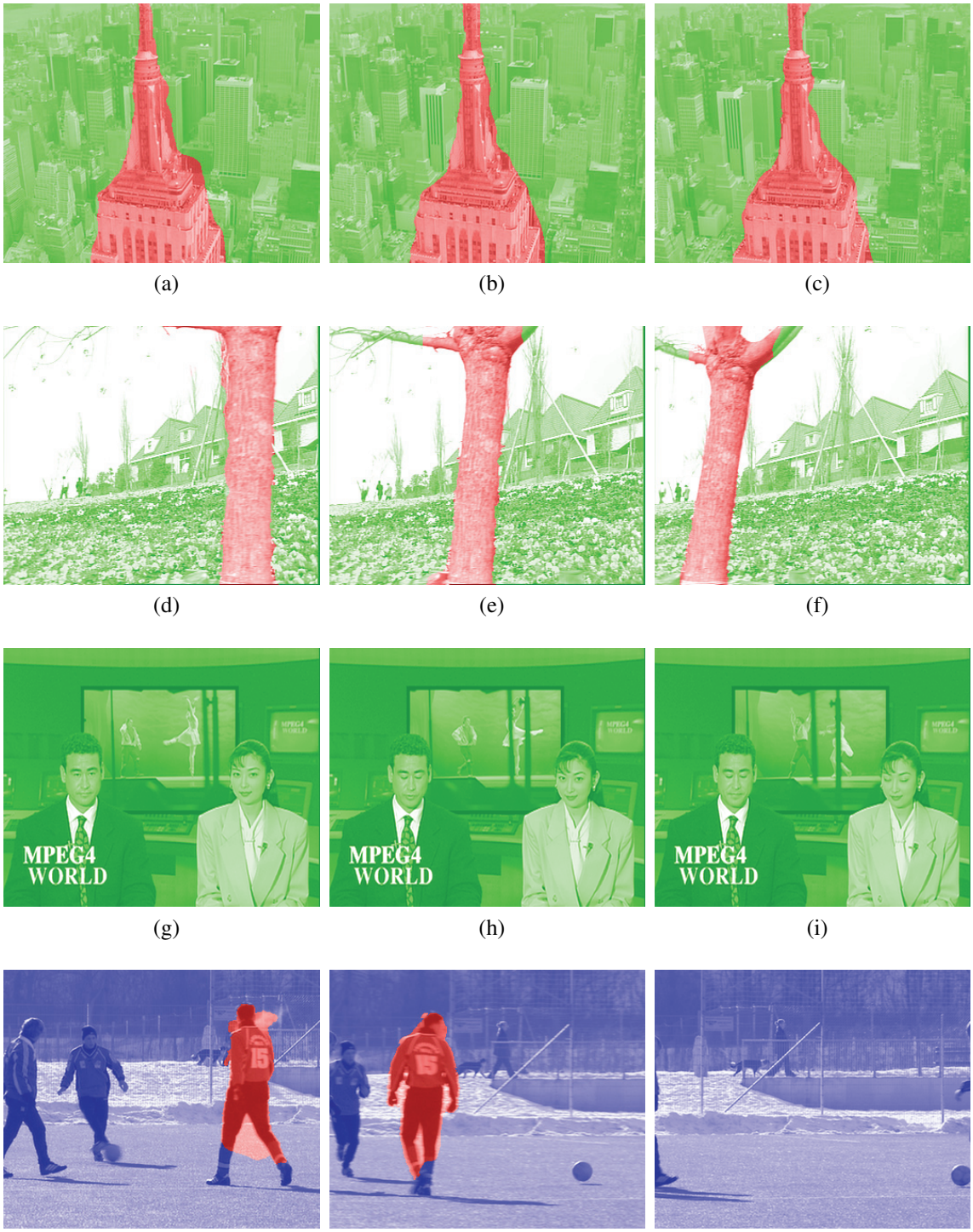
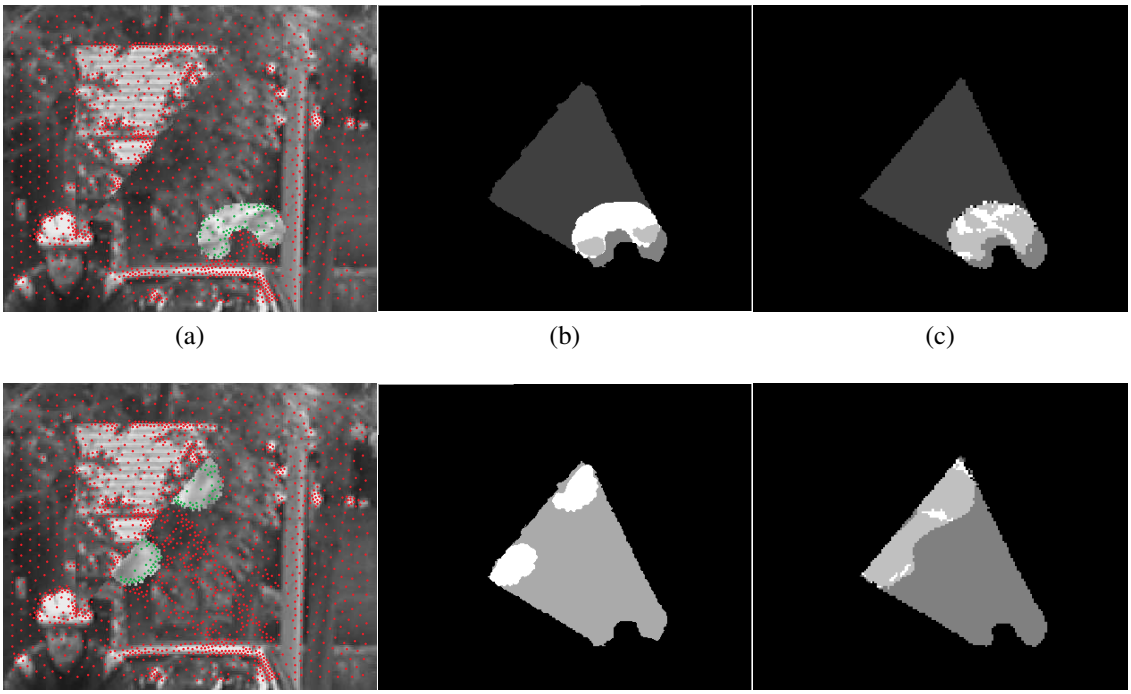



Figura 6.10: Resultados da segmentação para os quadros 10 (primeira coluna), 30 (segunda coluna) e 50 (terceira coluna) dos vídeos: *city* (a),(b), (c); *flower* (d),(e), (f); *news* (g),(h), (i) e *soccer* (j),(k), (l).



(a) (b) (c)

(d) (e) (f)

Figura 6.11: Segmentação para os quadros 5 (primeira linha) e 30 (segunda linha) de um vídeo sintético: (a),(d) resultados da segmentação de partículas; (b),(e) túnel de objeto, volumes de oclusão volumes de exposição obtidos com o método proposto; e (c),(f) túnel de objeto, volumes de oclusão e volumes de exposição pelo método proposto por Ristivojevic e Konrad (RISTIVOJEVIC; KONRAD, 2006).

contra 99.35% obtidos pelo método de Ristivojevic e Konrad. Estas medidas de exatidão se referem apenas à classificação dos pixels aos objetos (objeto vs. *background*, neste caso). Nota-se que a abordagem proposta trata a oclusão e mudanças topológicas de uma maneira mais simples, e obtém volumes de oclusão e volumes de exposição confiáveis, sem as limitações da abordagem de Ristivojevic e Konrad discutidas na Seção 2.1. Além do mais, os resultados demonstram que a abordagem proposta trata de forma mais simples e funcional as mudanças topológicas causadas pelas oclusões. Note que utilizando a abordagem de segmentação de vídeos proposta, temos a possibilidade de utilizar, além da segmentação densa, a informação do movimento das partículas correspondentes, o que pode ser utilizado como uma informação extra de movimento em diversas aplicações. Por exemplo, a oclusão e o movimento no interior dos objetos pode ser inferido através da informação das partículas e da segmentação densa combinadas. No experimento mostrado acima, foi empregada uma transformação afim para executar esta tarefa. No entanto, deve-se notar que o algoritmo de segmentação é geral e não implica nenhuma restrição de movimento.

Na Figura 6.12 são apresentados os resultados da segmentação pelo método proposto para o vídeo *syntobjects*. Este vídeo apresenta um *background* com movimento, e três objetos em movimento na frente. Um objeto rotaciona e translada, enquanto os outros apresentam apenas movimento de translação. Os quadros 5 e 20 da sequência original são mostrados nas Figuras 6.12(a) e 6.12(d). Os rótulos dos objetos segmentados são mostrados nas Figuras 6.12(b) e 6.12(e), enquanto os túneis de objetos (preto), volumes de oclusão (cinza) e volumes de exposição (branco) são mostrados nas Figuras 6.12(c) e

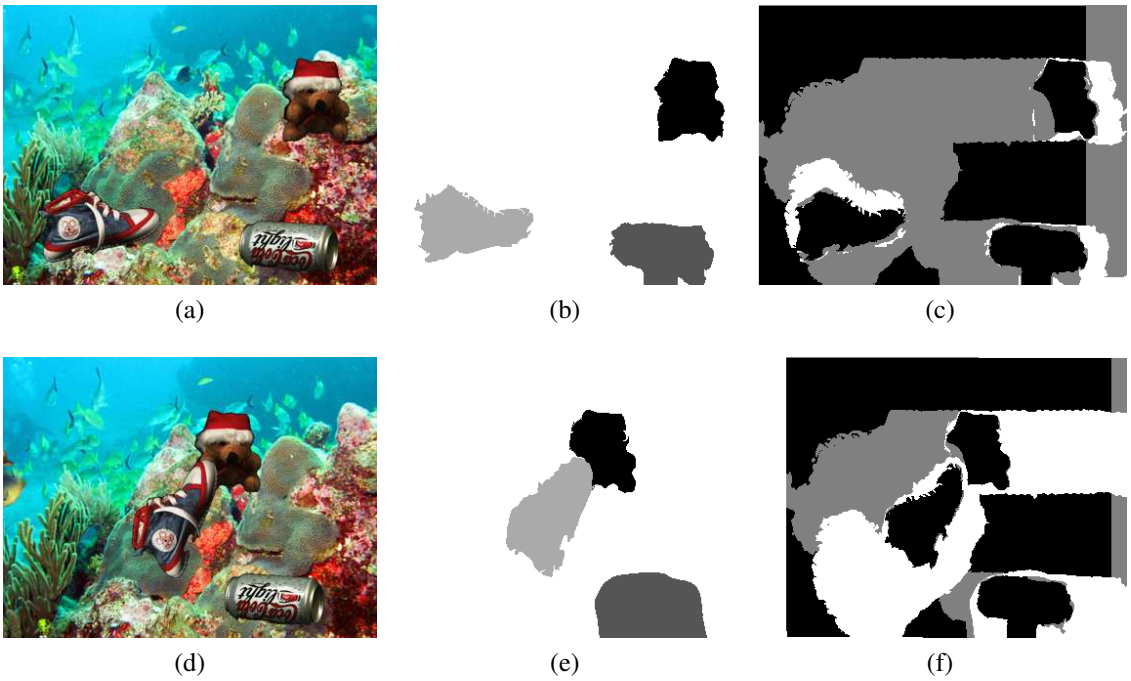


Figura 6.12: Resultados da segmentação para os quadros 5 (primeira linha) e 20 (segunda linha) de um vídeo sintético com três objetos em movimento: (a),(d) quadros originais; (b),(e) rótulos de segmentação de objetos; (c),(f) túneis de objetos, volumes de oclusão e volumes de exposição.

6.12(f). O método proposto produz uma classificação de objetos correta em 96.12% dos pixels. Uma interessante propriedade deste vídeo é que um objeto (lata de refrigerante), move-se junto com o *background* da cena por 10 quadros, enquanto exibe um movimento diferente no restante dos quadros do vídeo. Mesmo sem bordas de movimento ao longo de um terço da sequência, o método proposto segmentou este objeto do *background* da cena em todos os quadros, devido à sua capacidade de extrair padrões de movimento prolongados.

## 6.2 Resultados experimentais do método de codificação de vídeos proposto

O método de codificação proposto foi testado em 10 vídeos reais para verificar sua eficácia, nomeadamente: *bus*, *city*, *coastguard*, *flower*, *hall*, *husky*, *mobile*, *news*, *soccer* e *stefan* (disponíveis em `http://media.xiph.org/video/derf/`). Todos os vídeos têm resolução CIF ($288 \times 352$ pixels no canal de luminância e $144 \times 176$ pixels nos canais cromáticos), e a resolução dos canais cromáticos foi duplicada antes da codificação, já que o algoritmo do método proposto não utiliza sub-amostragem cromática. Para cada vídeo, foram codificados apenas os primeiros 60 quadros como teste, e considerou-se que estes 60 quadros formam um GOP.

O desempenho do método proposto de codificação é comparado ao padrão estado-da-arte H.264 (ISO/IEC, 2003). Foi utilizado o *software* de referência JM-H.264 (versão 16.0) para comparação. Para comparar resultados de codificação em diferentes taxas de bits, utilizamos o algoritmo de controle de taxa no JM-H.264. Mais especificamente, foram utilizadas as seguintes opções no JM-H.264:

- `FramesToBeEncoded=60`
- `SourceHeight=288`
- `SourceWidth=352`
- `ProfileIDC=244`
- `YUVFormat=3`
- `RateControlEnable=1`
- `HierarchicalCoding=2`
- `RCUpdateMode=3`
- `Bitrate=` variou-se de `200000` ($2 \times 10^5$) até `2000000` ($2 \times 10^6$)

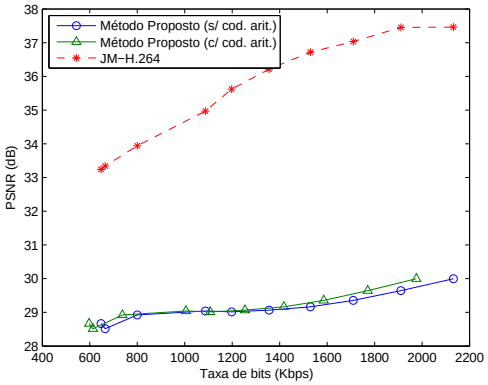
Todos os outros parâmetros tiveram seus valores padrão mantidos.

Para cada taxa de bits utilizada no JM-H.264, verificou-se o tamanho $S$ do arquivo codificado, e calculou-se o número médio de bits por pixel ($N_{bpp}$). A seguir, o mesmo vídeo foi codificado pelo método proposto com $N_{bpp}$ por pixel, da seguinte forma: (a) o GOI é codificado utilizando-se $1.5 \times N_{bpp}$ bits por pixel como alvo; (b) então o GOI é decodificado, são computadas as transformações afins, calculadas as predições de movimento e o erro de predição para cada objeto nos quadros B; (c) os erros de predição para cada objeto nos quadros do GOB são então codificados utilizando-se os bits restantes, ou seja, $S$ menos os bits utilizados com o cabeçalho, os coeficientes das transformações afins e a codificação do GOI, com um número médio de bits por pixel igual para cada objeto. As segmentações dos objetos são fornecidas pelo método de segmentação proposto no Capítulo 4. Como o método ST-SPIHT 3-D pode ter sua taxa de bits totalmente controlada, o número de bits utilizado pelo método proposto é igual ao número de bits utilizado pelo JM-H.264, mas a qualidade do vídeo decodificado pode diferir significativamente, e com isso os dois métodos podem ser comparados. Os parâmetros utilizados no método de codificação proposto foram:

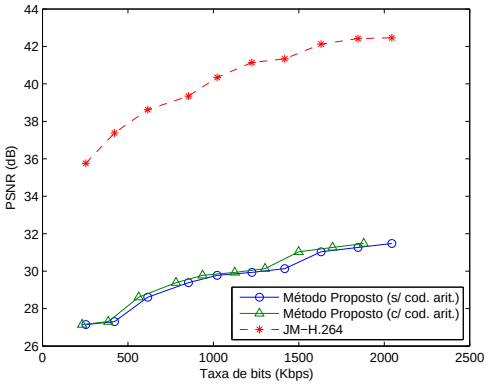
- Tamanho do GOP = 60
- Distância entre quadros I consecutivos = 10
- Nível de decomposição *wavelet* = 4
- Nível de codificação da forma = 20

Os valores médios de PSNR obtidos pelo método proposto e pelo JM-H.264 para os vídeos decodificados foram comparados segundo o canal de luminância (espaço $YC_bC_r$), e são mostrados nas Figuras 6.13 e 6.14. São incluídas ainda nestas Figuras os resultados obtidos pela codificação aritmética dos *bitstreams* ST-SPIHT 3-D do método proposto. Os valores médios de PSNR dos canais de luminância obtidos para quatro taxas de bits diferentes são mostrados na Tabela 6.1 para o método proposto, e na Tabela 6.2 para o JM-H.264.

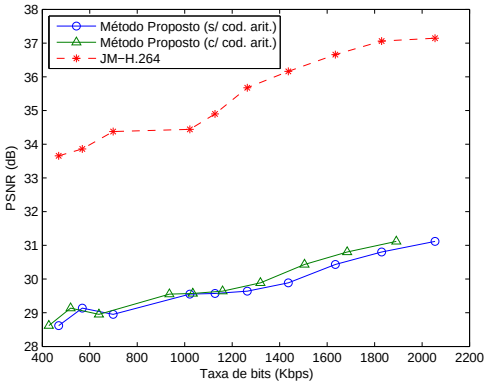
Os quadragésimos quadros das sequências *coastguard*, *mobile* e *stefan* originais são mostrados na Figura 6.15. A Figura 6.16 mostra os mesmos quadros decodificados pelo



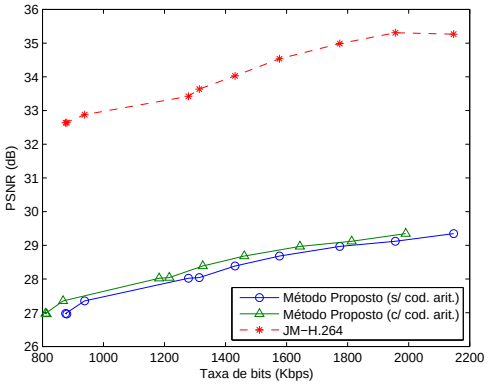

(a) bus



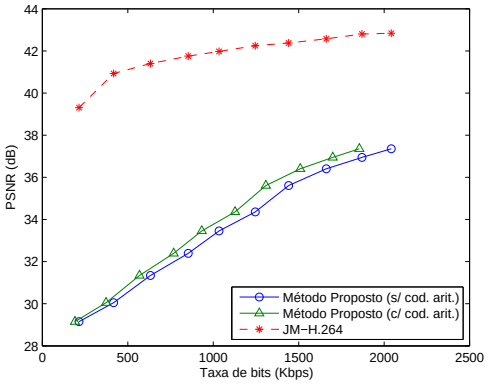

(b) city



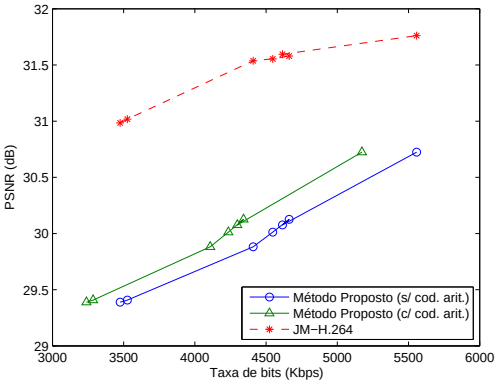

(c) coastguard




(d) flower

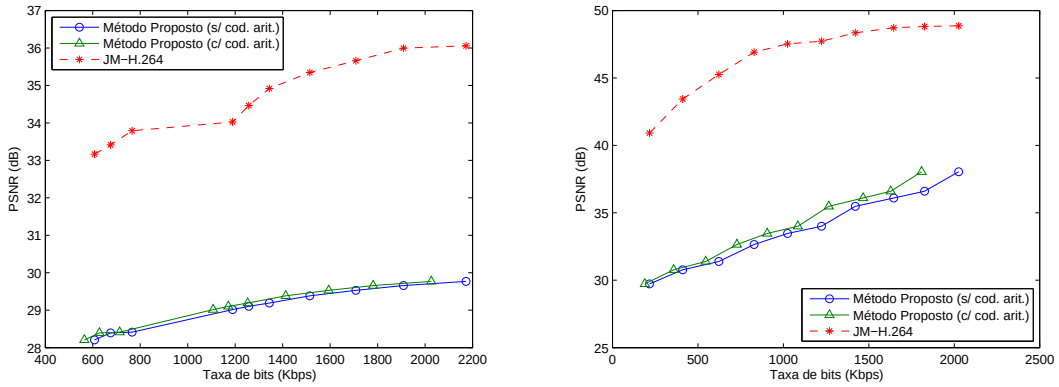



(e) hall




(f) husky

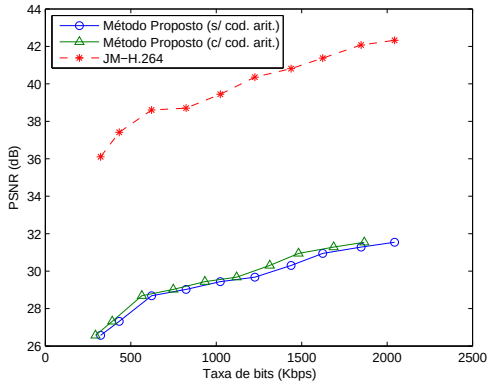
Figura 6.13: PSNR médio do canal de luminância dado em termos das taxas de bits, obtidos pelo método proposto e pelo JM-H.264 para os vídeos: (a) *bus*, (b) *city*, (c) *coastguard*, (d) *flower*, (e) *hall* e (f) *husky*.



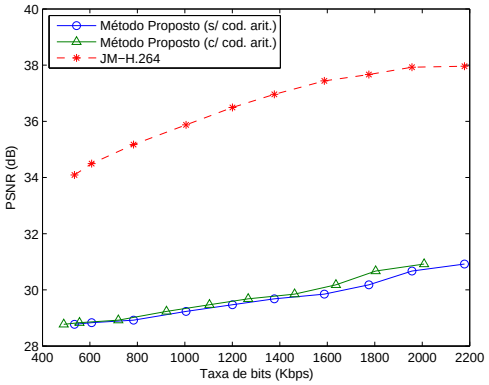

(a) mobile




(b) news




(c) soccer




(d) stefan

Figura 6.14: PSNR médio do canal de luminância dado em termos das taxas de bits, obtidos pelo método proposto e para o JM-H.264 para os vídeos: (a) *mobile*, (b) *news*, (c) *soccer* e (d) *stefan*.

Tabela 6.1: PSNR médio do canal de luminância, dado em dB, das sequências decodificadas pelo método proposto.

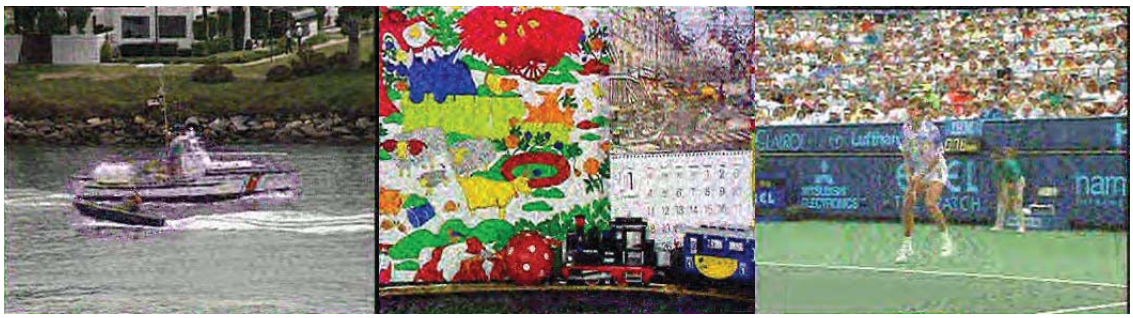
| | 600kbps | 1000kbps | 1400kbps | 1800kbps |
|---|---|---|---|---|
| *bus* | 28.9 | 29.0 | 29.2 | 29.6 |
| *city* | 28.6 | 29.8 | 30.1 | 31.3 |
| *coastguard* | 29.0 | 29.6 | 29.9 | 30.8 |
| *flower* | 27.4 | 28.0 | 28.7 | 29.1 |
| *hall* | 31.1 | 33.5 | 35.6 | 36.9 |
| *husky* | 29.4 | 30.0 | 30.1 | 30.2 |
| *mobile* | 28.4 | 29.1 | 29.4 | 29.7 |
| *news* | 31.4 | 33.5 | 35.5 | 36.6 |
| *soccer* | 28.7 | 29.4 | 30.3 | 31.3 |
| *stefan* | 28.9 | 29.5 | 29.8 | 30.7 |

Tabela 6.2: PSNR médio do canal de luminância, dado em dB, das sequências decodificadas pelo JM-H.264.

| | 600kbps | 1000mbps | 1400mbps | 1800kbps |
|---|---|---|---|---|
| *bus* | 33.9 | 35.6 | 36.7 | 37.5 |
| *city* | 38.6 | 40.3 | 41.3 | 42.4 |
| *coastguard* | 34.4 | 34.9 | 36.2 | 37.1 |
| *flower* | 32.9 | 33.6 | 34.5 | 35.3 |
| *hall* | 41.4 | 42.0 | 42.4 | 42.8 |
| *husky* | 31.0 | 31.5 | 31.6 | 31.6 |
| *mobile* | 33.8 | 34.5 | 35.3 | 36.0 |
| *news* | 45.3 | 47.5 | 48.4 | 48.8 |
| *soccer* | 38.6 | 39.4 | 40.8 | 42.0 |
| *stefan* | 35.2 | 36.5 | 37.4 | 37.9 |

(a) bus (b) coastguard (c) husky

Figura 6.15: Quadragésimo quadro original do vídeo: (a) *coastguard*, (b) *mobile* e (c) *stefan*.



(a) (b) (c)

(d) (e) (f)

Figura 6.16: Quadragésimos quadros dos vídeos *coastguard*, *mobile* e *stefan* decodificados a 600Kbps pelo método proposto (a)-(c) e pelo JM-H.264 (d)-(f).

(a) (b) (c)

(d) (e) (f)

Figura 6.17: Quadragésimos quadros dos vídeos *coastguard*, *mobile* e *stefan* decodificados a 1.2Mbps pelo método proposto (a)-(c) e pelo JM-H.264 (d)-(f).

método proposto e pelo JM-H.264 a 600kbps. Já os quadros decodificados a 1200kbps são mostrados na Figura 6.17.

Os resultados acima servem para indicar a que distância o método proposto está do estado-do-arte em termos de eficiência de codificação. No entanto, deve-se notar que o método de codificação proposto trata de um paradigma diferente, com características distintas e que apresenta vantagens em aplicações distintas. Além do mais, o método proposto não é otimizado para aplicações do tipo *low-end*. A informação lateral, composta pela forma das regiões e coeficientes das transformações afins, não depende da taxa de bits, ou seja, o número de bits alocados para estas informações são os mesmos tanto para taxas altas como para taxas baixas de bits. Ainda, coeficientes das transformações afins são incluídos no *bitstream* sem quantização ou codificação eficiente.

Ao observar as curvas de Taxas de bits *vs.* PSNR nas Figuras 6.13 e 6.14, percebe-se que as curvas de desempenho do método proposto são mais "bem comportadas" que as curvas de desempenho do JM-H.264, que por sua vez se mostram mais irregulares. Isto pode ser explicado em parte pela estratégia de controle da taxa de bits empregada. O JM-H.264 codifica um quadro de cada vez, e a cada quadro processado, o algoritmo de controle de taxa re-calcula os parâmetros de quantização para o quadro seguinte, de acordo com a quantidade de bits utilizada nos quadros anteriores. Como esta estimativa dos parâmetros de quantização não é muito precisa, a cada quadro o algoritmo compensa o erro em relação à taxa alvo nos quadros anteriores. Isto faz com que o comportamento do desempenho do JM-H.264 seja errático, à medida que a taxa de bits varia. O método proposto, por outro lado, codifica todos os quadros de um GOI ou de um GOB inteiros de uma só vez, através de uma codificação progressiva. Com isso, a cada bit que vai sendo codificado, a qualidade de todo um conjunto de quadros vai sendo aumentada regularmente.

O método de codificação proposto foi testado em um computador com processador Quad Core de 2.4GHz e 4GB de memória RAM, com Windows Vista. O algoritmo de codificação ST-SPIHT 3-D foi implementado em C++, enquanto o restante do processo de codificação foi implementado em Matlab. Para os vídeos testados, o processo de codificação completo de um vídeo levou em média 4 horas e 30 minutos. A decodificação variou de 45 minutos para a menor taxa de bits (200 kbps) até 3 horas para a maior taxa de bits (2000 kbps).

# 7 CONCLUSÕES

Um método para a identificação e a segmentação não-supervisionada de movimento coerente em vídeos adaptativamente amostrados foi proposto nesta tese. Esta técnica provê uma nova forma de ligação entre informação de baixo nível em vídeos a conceitos de alto nível que podem ser empregados diretamente na codificação de vídeos. Esta abordagem por ser útil em várias outras tarefas de processamento de imagens e visão computacional, incluindo rastreamento de objetos, recuperação de informações e análise de vídeos.

Utilizando como entrada a técnica de segmentação de movimento desenvolvida, foi proposto também um método de codificação de vídeos baseado em objetos. Este método utiliza transformações afins, inferidas do movimento das partículas, para predizer o movimento dos objetos. Quadros I (*intra-coded*), assim como erros de predição de cada objeto para os quadros B (*bi-directional predicted*), são codificados empilhados utilizando uma extensão do método ST-SPIHT (MARTIN; LUKAC; PLATANIOTIS, 2006), que codifica simultaneamente a cor e a forma dos objetos. O método ST-SPIHT, que originalmente trata apenas da codificação de objetos 2-D individuais, foi modificado para codificar diversos objetos representados por volumes espaço-temporais (chamados de túneis). Objetos diferentes podem ser transmitidos com diferentes taxas de bits, e os objetos podem ser decodificados progressivamente. Porém, ao contrário dos métodos tradicionais que codificam um quadro de cada vez, aqui cada bit decodificado melhora a qualidade geral da sequência, e não de um quadro específico. Estas características são vantajosas em aplicações tais como transmissão de vídeo pela *Web*.

O método de segmentação de vídeos proposto nos permite identificar padrões temporalmente descontínuos de movimento, e é robusto ao movimento abrupto de câmera. Além disso, nenhuma restrição global de movimento é imposta à cena e/ou aos objetos. O problema da abertura é reduzido ao se inferir movimento em regiões homogêneas pela propagação de informação de movimento na vizinhança das amostras espaciais (ou seja, partículas). No entanto, a solução utilizada para abordar o problema da abertura tem uma desvantagem: quando regiões homogêneas se tornam oclusas, a propagação de movimento pode permitir estimativas errôneas de movimento, que devem ser corrigidas através de passos adicionais de consistência, ao custo de maior esforço computacional.

O método de segmentação de partículas proposto utiliza agrupamento de conjuntos para combinar grupos de partículas obtidos para quadros adjacentes, permitindo a identificação de padrões de movimento de longa duração, os quais são representados através de volumes espaço-temporais chamados de túneis. A identificação de padrões de movimento de longa duração é crucial para se tirar vantagem da redundância em codificação de vídeo baseada em segmentação. O algoritmo *mean-shift* (COMANICIU; MEER, 2002) é empregado para obter os grupos de partículas associados a quadros adjacentes, e tem a

propriedade importante de identificar grupos com formas arbitrárias no espaço de feições. No entanto, o *mean-shift* apresenta uma importante desvantagem: ele requer o uso de uma banda fixa, reduzindo a magnitude e variedade dos padrões de movimento detectados. Assim, ao fixar um determinado valor para a banda no *mean-shift*, restringimos o desempenho do método a um certo espectro de magnitudes de movimentos entre os objetos. Por exemplo, no vídeo *soccer* (ver Seção 6.1.1), a magnitude de movimento de cada um dos jogadores e mesmo da bola é bastante diferente. Mesmo se considerarmos um único jogador, diferentes partes do seu corpo podem ser mover em velocidades distintas. Dessa forma, quando temos movimentos de magnitudes muito distintas, a banda fixa do *mean-shift* tende a gerar sub-segmentação ou sobre-segmentação. O valor de $\rho_V$ (ver Seção 4.3.2) também necessita ser ajustado de acordo com o espectro de movimentos entre os objetos. Isto pode ser uma desvantagem em aplicações onde uma ampla gama de magnitudes de movimento é esperada. Outra limitação do método de segmentação proposto diz respeito aos tipos de movimentos que são melhor suportados com esta abordagem. Como são gerados grupos de partículas com base na similaridade dos vetores de movimento projetados em 2-D, sequências com um efeito de projeção perspectiva mais intenso e/ou com movimento 3-D complexo tende a serem sobre-segmentadas ou sub-segmentadas. De forma geral, podemos dizer que quando o tamanho dos objetos da cena são comparáveis à distância da câmera em relação ao objeto, o efeito da projeção perspectiva deverá ser considerável. Ainda, como o método de codificação proposto utiliza transformações afins para representar o movimento entre quadros, se os objetos segmentados não apresentam movimento aproximável por uma transformação afim, a predição de movimento tende a ser menos eficiente, reduzindo o desempenho do algoritmo de codificação.

Conforme discutido na Seção 5.1, é utilizada uma estratégia que codifica formas de objetos juntamente com os erros de predição. Para evitar a codificação de dados redundantes, quando um objeto é codificado, as formas dos objetos previamente codificados são levadas em consideração. Isto torna a decodificação do conjunto inteiro de objetos mais eficiente, enquanto mantém importantes características da codificação de vídeos baseada em objetos, como escalabilidade a nível de objeto. No entanto, esta estrátégia limita a escalabilidade de objeto, já que um objeto só pode começar a ser decodificado quando a forma dos objetos anteriores já são conhecidas. Apesar disso, se a aplicação não requer a transmissão de objetos individuais, ou a decodificação em baixíssimas taxas de bits, esta limitação não é relevante, visto que as formas dos objetos tendem a ser totalmente decodificadas no início do *bitstream*.

Ainda, embora o método de codificação proposto permita a decodificação escalável de objetos, o erro de predição de movimento nos quadros B deve ser calculada com base nos quadros I decodificados. Assim, devemos definir a taxa de bits do GOI durante a codificação. Isto torna o mecanismo de controle de taxa amarrado à taxa de bits do GOI, e a qualidade geral da sequência decodificada não é ótima, no sentido de que quadros decodificados do GOI e do GOB podem apresentar qualidades diferentes.

Algumas melhorias ao método proposto podem ser abordadas em trabalhos futuros:

1. A distância entre quadros I consecutivos pode ser adaptativa, ao invés de fixa. Uma seleção cuidadosa de quadros I, baseada no movimento dos objetos da sequência, pode melhorar a predição de movimento, permitindo que os quadros B sejam codificados de forma mais eficiente;

2. Um método mais eficiente para a codificação dos coeficientes das transformações afins pode ser empregado;

3. Como os resíduos da compensação de movimento usualmente concentram energia em pequenas regiões, os coeficientes *wavelet* podem ser quebrados em blocos e codificados separadamente, com potenciais ganhos na eficiência da codificação (LIN; GRAY, 2001);

4. O método de segmentação, assim como a predição de movimento baseada em partículas, pode ser extendida de planar para 3-D.

Resultados experimentais obtidos com o método de segmentação proposto foram apresentados, e avaliações numéricas com vídeos sintéticos foram computadas. O método de codificação proposto foi aplicado a vídeos reais segmentados, e os resultados foram comparados aos obtidos pelo codificador JM-H.264. Comparações em termos de PSNR foram apresentadas para diferentes taxas de bits, mostrando onde o método proposto se posiciona em relação ao estado da arte na eficiência da codificação.

# REFERÊNCIAS

BANDYOPADHYAY, S.; KONDI, L. Optimal bit allocation for joint contour-based shape coding and shape adaptive texture coding. In: IMAGE PROCESSING, 2005. ICIP 2005. IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2005. v.1, p.I – 589–92.

BEUCHER, S.; MEYER, F. **The Morphological Approach to Segmentation** : The Watershed Transformation. [S.l.: s.n.], 1993. p.433–481.

BOULT, T.; BROWN, L. G. Factorization-based segmentation of motions. In: VISUAL MOTION, 1991., PROCEEDINGS OF THE IEEE WORKSHOP ON. **Anais...** [S.l.: s.n.], 1991. p.179–186.

BRADY, N.; BOSSEN, F.; MURPHY, N. Context-based arithmetic encoding of 2D shape sequences. In: IMAGE PROCESSING, 1997. PROCEEDINGS., INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 1997. v.1, p.29 –32 vol.1.

BRIASSOULI, A.; KOMPATSIARIS, I.; MEZARIS, V. Joint Motion and Color Statistical Video Processing for Motion Segmentation. In: MULTIMEDIA AND EXPO, 2007 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2007. p.2014–2017.

BROX et al. High Accuracy Optical Flow Estimation Based on a Theory for Warping. In: ECCV 2004. **Proceedings...** [S.l.: s.n.], 2004.

CANNY, J. A computational approach to edge detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Washington, DC, USA, v.8, n.6, p.679–698, Nov. 1986.

CHEUNG, H. K. et al. New Block-Based Motion Estimation for Sequences with Brightness Variation and Its Application to Static Sprite Generation for Video Compression. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.18, n.4, p.522 –527, april 2008.

COMANICIU, D.; MEER, P. Mean shift: a robust approach toward feature space analysis. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.24, p.603–619, 2002.

COSTEIRA, J.; KANADE, T. A Multibody Factorization Method for Independently Moving Objects. **International Journal of Computer Vision**, [S.l.], v.29, p.159–179, 1998.

CREMERS, D. Dynamical statistical shape priors for level set based tracking. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.28, n.8, p.1262–1273, August 2006.

CREMERS, D.; KOHLBERGER, T.; SCHNÖRR, C. Nonlinear shape statistics in Mumford–Shah based segmentation. In: EUROPEAN CONFERENCE ON COMPUTER VISION (ECCV), Copenhagen. **Anais...** Springer, 2002. p.93–108. (LNCS, v.2351).

CREMERS, D.; OSHER, S. J.; SOATTO, S. Kernel density estimation and intrinsic alignment for shape priors in level set segmentation. **International Journal of Computer Vision**, [S.l.], v.69, n.3, p.335–351, September 2006.

CREMERS, D.; SOATTO, S. Variational space-time motion segmentation. In: COMPUTER VISION, 2003. PROCEEDINGS. NINTH IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2003. p.886–893 vol.2.

DASU, A.; PANCHANATHAN, S. A wavelet-based sprite codec. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.14, n.2, p.244 – 255, february 2004.

DAUBECHIES, I.; SWELDENS, W. Factoring Wavelet Transforms into Lifting Steps. **Journal of Fourier Analysis and Applications**, [S.l.], v.4, n.3, p.245–267, 1998.

DEMONCEAUX, C.; KACHI-AKKOUCHE, D. Fast motion estimation and motion segmentation using multi-scale approach. In: IMAGE PROCESSING, 2004. ICIP '04. 2004 INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2004. v.1, p.377–380 Vol. 1.

DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum likelihood from incomplete data via the EM algorithm. **Journal of the Royal Statistical Society**, [S.l.], v.39, n.1, p.1–38, 1977.

FEGHALI, R.; MITICHE, A. Spatiotemporal motion boundary detection and motion boundary velocity estimation for tracking moving objects with a moving camera: a level sets pdes approach with concurrent camera motion compensation. **IEEE Transactions on Image Processing**, [S.l.], v.13, n.11, p.1473–1490, Nov. 2004.

FRED, A.; JAIN, A. Combining multiple clusterings using evidence accumulation. **Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.27, p.835–850, 2005.

FREEMAN, H. On the Encoding of Arbitrary Geometric Configurations. **IRE Transactions on Electronic Computers**, [S.l.], v.EC-10, n.2, p.260 –268, june 1961.

FREUND, R.; NACHTIGAL, N. QMR: a quasi-minimal residual method for non-hermitian linear systems. **Numerische Mathematik**, [S.l.], v.60, p.315–339, Dec. 1991.

GAO, J.; JUNG, S.; THAKOOR, N. A motion field reconstruction scheme for smooth boundary video object segmentation. In: IMAGE PROCESSING, 2004. ICIP '04. 2004 INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2004. v.1, p.381–384 Vol. 1.

GELGON, M.; BOUTHEMY, P.; LE CADRE, J. Recovery of the trajectories of multiple moving objects in an image sequence with a PMHT approach. **Image and Vision Computing Journal**, [S.l.], v.23, n.1, p.19–31, 2005.

GOKCETEKIN, M. et al. Mesh based segmentation and update for object based video. In: IMAGE PROCESSING, 2000. PROCEEDINGS. 2000 INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2000. v.1, p.343 –346 vol.1.

GRUBER, A.; WEISS, Y. Multibody factorization with uncertainty and missing data using the EM algorithm. In: COMPUTER VISION AND PATTERN RECOGNITION, 2004. CVPR 2004. PROCEEDINGS OF THE 2004 IEEE COMPUTER SOCIETY CONFERENCE ON. **Anais...** [S.l.: s.n.], 2004. v.1, p.I–707–I–714 Vol.1.

HAN, S.-C.; WOODS, J. Spatiotemporal subband/wavelet coding of video with object-based motion information. In: IMAGE PROCESSING, 1997. PROCEEDINGS., INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 1997. v.2, p.629 –632 vol.2.

HAN, S. C.; WOODS, J. W. Adaptive coding of moving objects for very low bit rates. **IEEE Journal on Selected Areas in Communications**, [S.l.], v.16, n.1, p.56–70, 1998.

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In: FOURTH ALVEY VISION CONFERENCE. **Anais...** [S.l.: s.n.], 1988. p.147–151.

HARTLEY, R.; VIDAL, R. The multibody trifocal tensor: motion segmentation from 3 perspective views. In: COMPUTER VISION AND PATTERN RECOGNITION, 2004. CVPR 2004. PROCEEDINGS OF THE 2004 IEEE COMPUTER SOCIETY CONFERENCE ON. **Anais...** [S.l.: s.n.], 2004. v.1, p.I–769–I–775 Vol.1.

ICHIMURA, N. A robust and efficient motion segmentation based on orthogonal projection matrix of shape space. In: COMPUTER VISION AND PATTERN RECOGNITION, 2000. PROCEEDINGS. IEEE CONFERENCE ON. **Anais...** [S.l.: s.n.], 2000. v.2, p.446–452 vol.2.

IEEE. **IEEE Std 754-2008**: standard for floating-point arithmetic. [S.l.], 2008.

IRANI, M.; ANANDAN, P. About Direct Methods. In: INTERNATIONAL WORKSHOP ON VISION ALGORITHMS: THEORY AND PRACTICE. **Proceedings...** Springer-Verlag, 2000. p.267–277.

IRANI, M. et al. Efficient representations of video sequences and their applications. **Signal Processing: Image Communication**, [S.l.], v.8, n.4, p.327 – 351, 1996.

ISO/IEC. **ISO 13818-2**: information technology - generic coding of moving pictures and associated audio information: video. [S.l.], 2000.

ISO/IEC. **Overview of the MPEG-4 Standard**. [S.l.], 2001.

ISO/IEC. **ISO 14496-10**: coding of audiovisual objects–part 10: advanced video coding. [S.l.], 2003.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. **ACM Computing Surveys**, [S.l.], v.31, p.264–323, 1999.

JODOIN, P. M.; ROSENBERGER, C.; MIGNOTTE, M. Segmentation Framework Based on Label Field Fusion. **IEEE Transactions on Image Processing**, [S.l.], v.16, n.10, p.2535–2550, 2007.

KANATANI, K. Motion segmentation by subspace separation and model selection. In: COMPUTER VISION, 2001. ICCV 2001. PROCEEDINGS. EIGHTH IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2001. v.2, p.586–591 vol.2.

KASSIM, A.; LEE, W. S. Embedded color image coding using SPIHT with partially linked spatial orientation trees. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.13, n.2, p.203 – 206, feb 2003.

KASSIM, A.; ZHAO, L. Rate-scalable object-based wavelet codec with implicit shape coding. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.10, n.7, p.1068 –1079, oct 2000.

KAUFF, P. et al. Functional coding of video using a shape-adaptive DCT algorithm and an object-based motion prediction toolbox. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.7, n.1, p.181 –196, Feb. 1997.

KAUP, A. Object-based texture coding of moving video in MPEG-4. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.9, n.1, p.5 –15, feb 1999.

KUNT, M.; IKONOMOPOULOS, A.; KOCHER, M. Second-generation image-coding techniques. **Proceedings of the IEEE**, [S.l.], v.73, n.4, p.549–574, 1985.

LI, S.; LI, W. Shape-adaptive discrete wavelet transforms for arbitrarily shaped visual object coding. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.10, n.5, p.725 –743, aug 2000.

LI, T. et al. Projective Factorization of Multiple Rigid-Body Motions. In: COMPUTER VISION AND PATTERN RECOGNITION, 2007. CVPR '07. IEEE CONFERENCE ON. **Anais...** [S.l.: s.n.], 2007. p.1–6.

LI, Y.; HATZINAKOS, D.; VENETSANOPOULOS, A. A multi-frame, region-feature based technique for motion segmentation. In: IMAGE PROCESSING, 1999. ICIP 99. PROCEEDINGS. 1999 INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 1999. v.1, p.11–15 vol.1.

LIN, K.; GRAY, R. Video residual coding using SPIHT and dependent optimization. In: DATA COMPRESSION CONFERENCE, 2001. PROCEEDINGS. DCC 2001. **Anais...** [S.l.: s.n.], 2001. p.113 –122.

LIU, Y. K.; ZALIK, B. An efficient chain code with Huffman coding. **Pattern Recognition**, [S.l.], v.38, n.4, p.553 – 557, 2005.

LIU, Y.; NGAN, K. N.; WU, F. 3-D Shape-Adaptive Directional Wavelet Transform for Object-Based Scalable Video Coding. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.18, n.7, p.888 –899, july 2008.

LIU, Y.; WU, F.; NGAN, K. N. 3-D Object-Based Scalable Wavelet Video Coding With Boundary Effect Suppression. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.17, n.5, p.639 –644, may 2007.

LLOYD, S. P. Least squares quantization in pcm. **IEEE Transactions on Information Theory**, [S.l.], v.28, p.129–137, 1982.

LOWE, D. G. Distinctive Image Features from Scale-Invariant Keypoints. **International Journal of Computer Vision**, [S.l.], p.91–110, Nov. 2004.

LU, Y.; GAO, W.; WU, F. Efficient background video coding with static sprite generation and arbitrary-shape spatial prediction techniques. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.13, n.5, p.394 – 405, May 2003.

LUBLINERMAN, R.; CAMPS, O.; SZNAIER, M. Dynamics Based Robust Motion Segmentation. In: COMPUTER VISION AND PATTERN RECOGNITION, 2006 IEEE COMPUTER SOCIETY CONFERENCE ON. **Anais...** [S.l.: s.n.], 2006. v.1, p.1176–1184.

LUO, H. Image-dependent shape coding and representation. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.15, n.3, p.345 – 354, march 2005.

MA, Y. et al. **An Invitation to 3-D Vision**: from images to geometric models. [S.l.]: Springer, 2003. (Interdisciplinary Applied Mathematics Series).

MANSOURI, A. R.; KONRAD, J. Multiple motion segmentation with level sets. **IEEE Transactions on Image Processing**, [S.l.], v.12, n.2, p.201–220, 2003.

MANSOURI, A. R.; KONRAD, J.; CHOMAUD, T. A comparative evaluation of algorithms for fast computation of level set PDEs with applications to motion segmentation. In: IMAGE PROCESSING, 2001. PROCEEDINGS. 2001 INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2001. v.3, p.636–639 vol.3.

MARTIN, K.; LUKAC, R.; PLATANIOTIS, K. SPIHT-Based Coding of the Shape and Texture of Arbitrarily Shaped Visual Objects. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.16, n.10, p.1196 –1208, october 2006.

MEIER, T.; NGAN, K. Automatic segmentation of moving objects for video object plane generation. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.8, n.5, p.525–538, 1998.

MINAMI, G. et al. 3-D wavelet coding of video with arbitrary regions of support. In: SIGNALS, SYSTEMS, AND COMPUTERS, 1999. CONFERENCE RECORD OF THE THIRTY-THIRD ASILOMAR CONFERENCE ON. **Anais...** [S.l.: s.n.], 1999. v.2, p.1422 –1425 vol.2.

MITICHE, A.; EL-FEGHALI, R.; MANSOUI, A. R. Motion tracking as spatio-temporal motion boundary detection. **Robotics and Autonomous Systems**, [S.l.], v.43, n.1, p.39–50, 2003.

MITICHE, A.; SEKKATI, H. Optical Flow 3D Segmentation and Interpretation: a variational method with active curve evolution and level sets. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.28, n.11, p.1818–1829, 2006.

MOON, J. H.; KWEON, J. H.; KIM, H. K. Boundary block-merging (BBM) technique for efficient texture coding of arbitrarily shaped object. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.9, n.1, p.35 –43, feb 1999.

MUSMANN, H.; HOTTER, M.; OSTERMANN, J. Object-oriented analysis-synthesis coding of moving images. **Signal Processing: Image Communication**, [S.l.], v.1, n.2, p.117–138, October 1990.

PEI, S. C.; LIOU, L. G. Motion-based grouping of optical flow fields: the extrapolation and subtraction technique. **IEEE Transactions on Image Processing**, [S.l.], v.6, n.10, p.1358–1363, 1997.

RISTIVOJEVIC, M.; KONRAD, J. Space-time image sequence analysis: object tunnels and occlusion volumes. **IEEE Transactions on Image Processing**, [S.l.], v.15, p.364–376, 2006.

SAID, A.; PEARLMAN, W. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.6, n.3, p.243 –250, June 1996.

SAND, P.; TELLER, S. Particle Video: long-range motion estimation using point trajectories. In: COMPUTER VISION AND PATTERN RECOGNITION, 2006 IEEE COMPUTER SOCIETY CONFERENCE ON. **Anais...** [S.l.: s.n.], 2006. v.2, p.2195–2202.

SCHINDLER, K.; U, J.; WANG, H. Perspective n-View Multibody Structure-and-Motion Through Model Selection. In: EUROPEAN CONFERENCE ON COMPUTER VISION, 9. **Anais...** [S.l.: s.n.], 2006. p.606–619.

SEKKATI, H.; MITICHE, A. Joint dense 3D interpretation and multiple motion segmentation of temporal image sequences: a variational framework with active curve evolution and level sets. In: IMAGE PROCESSING, 2004. ICIP '04. 2004 INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2004. v.1, p.553–556 Vol. 1.

SEKKATI, H.; MITICHE, A. Concurrent 3-D motion segmentation and 3-D interpretation of temporal sequences of monocular images. **IEEE Transactions on Image Processing**, [S.l.], v.15, n.3, p.641–653, 2006.

SEKKATI, H.; MITICHE, A. Joint optical flow estimation, segmentation, and 3D interpretation with level sets. **Computer Vision and Image Understanding**, [S.l.], p.89–100, Aug. 2006.

SHAMIM, A.; ROBINSON, J. Object-based video coding by global-to-local motion segmentation. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.12, n.12, p.1106 – 1116, Dec. 2002.

SHEN, Z.; FRATER, M.; ARNOLD, J. Quad-Tree Block-Based Binary Shape Coding. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.18, n.6, p.845 –850, june 2008.

SHI, J. et al. Good features to track. In: COMPUTER VISION AND PATTERN RECOGNITION, 1994. PROCEEDINGS CVPR '94., 1994 IEEE COMPUTER SOCIETY CONFERENCE ON. **Anais...** [S.l.: s.n.], 1994. p.593–600.

SIKORA, T.; MAKAI, B. Shape-adaptive DCT for generic coding of video. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.5, n.1, p.59 –62, feb 1995.

SILVA, L. S.; SCHARCANSKI, J. Unsupervised Identification of Coherent Motion in Video. In: COMPUTER GRAPHICS AND IMAGE PROCESSING, 2007. SIBGRAPI 2007. XX BRAZILIAN SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2007. p.231–238.

SILVA, L.; SCHARCANSKI, J. Video Segmentation Based on Motion Coherence of Particles in a Video Sequence. **IEEE Transactions on Image Processing**, [S.l.], v.19, n.4, p.1036 –1049, april 2010.

SMITH, P.; CIPOLLA, R.; DRUMMOND, T. Layered motion segmentation and depth ordering by tracking edges. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.26, n.4, p.479–494, 2004.

SMITH, S.; BRADY, J. ASSET-2: real-time motion segmentation and shape tracking. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.17, n.8, p.814–820, 1995.

STREHL, A.; GHOSH, J. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. **Journal of Machine Learning Research**, [S.l.], v.3, p.583–617, 2003.

SUGAYA, Y.; KANATANI, K. ichi. Geometric Structure of Degeneracy for Multi-body Motion Segmentation. In: ECCV WORKSHOP SMVP. **Anais...** [S.l.: s.n.], 2004. p.13–25.

SUN, Y.; AHMAD, I. A robust and adaptive rate control algorithm for object-based video coding. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.14, n.10, p.1167 – 1182, october 2004.

TALLURI, R. et al. A robust, scalable, object-based video compression technique for very low bit-rate coding. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.7, n.1, p.221 –233, Feb. 1997.

TORR, P. Geometric motion segmentation and model selection. **Phil. Transactions Royal Society of London**, [S.l.], v.356, p.1321–1340, 1998.

TORR, P. H. S.; ZISSERMAN, A. Feature Based Methods for Structure and Motion Estimation. In: INTERNATIONAL WORKSHOP ON VISION ALGORITHMS: THEORY AND PRACTICE. **Proceedings...** Springer-Verlag, 2000. p.278–294.

TORRES, L.; KUNT, M.; PEREIRA, F. Second Generation Video Coding Schemes And Their Role In Mpeg-4. In: MPEG-4. EUROPEAN CONFERENCE ON MULTIMEDIA APPLICATIONS, SERVICES AND TECHNIQUES. **Anais...** [S.l.: s.n.], 1996. p.799–824.

TRON, R.; VIDAL, R. A Benchmark for the Comparison of 3-D Motion Segmentation Algorithms. In: COMPUTER VISION AND PATTERN RECOGNITION, 2007. CVPR '07. IEEE CONFERENCE ON. **Anais...** [S.l.: s.n.], 2007. p.1–8.

VAZQUEZ, C.; LAGANIERE, R.; MITICHE, A. Joint multiregion segmentation and parametric estimation of image motion by basis function representation and level set evolution. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v.28, p.782–793, 2006.

VIDAL, R. et al. Two-View Multibody Structure from Motion. **International Journal of Computer Vision**, [S.l.], v.68, p.7–25, 2006.

VIDAL, R.; HARTLEY, R. Motion segmentation with missing data using PowerFactorization and GPCA. In: COMPUTER VISION AND PATTERN RECOGNITION, 2004. CVPR 2004. PROCEEDINGS OF THE 2004 IEEE COMPUTER SOCIETY CONFERENCE ON. **Anais...** [S.l.: s.n.], 2004. v.2, p.II–310–II–316 Vol.2.

VIDAL, R.; SASTRY, S.; MA, Y. Generalized principal component analysis (GPCA). **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], p.1945–1959, 2005.

VIDAL, R.; SINGARAJU, D. A closed form solution to direct motion segmentation. In: COMPUTER VISION AND PATTERN RECOGNITION, 2005. CVPR 2005. IEEE COMPUTER SOCIETY CONFERENCE ON. **Anais...** [S.l.: s.n.], 2005. v.2, p.510–515 vol. 2.

VISWANATH, P.; JAYASURYA, K. A Fast and Efficient Ensemble Clustering Method. In: PATTERN RECOGNITION, 2006. ICPR 2006. 18TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2006. v.2, p.720–723.

WANG, H.; LIN, H. A spectral clustering approach to motion segmentation based on motion trajectory. In: MULTIMEDIA AND EXPO, 2003. ICME '03. PROCEEDINGS. 2003 INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2003. v.2, p.II–793–6 vol.2.

WANG, J. et al. Moving object segmentation using graph cuts. In: SIGNAL PROCESSING, 2004. PROCEEDINGS. ICSP '04. 2004 7TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2004. v.1, p.777–780 vol.1.

XIAO, J. et al. Bilateral Filtering-Based Optical Flow Estimation with Occlusion Detection. In: ECCV (1). **Anais...** [S.l.: s.n.], 2006. p.211–224.

XU, H.; KABUKA, M.; YOUNIS, A. Automatic moving object extraction for content-based applications. **IEEE Transactions on Circuits and Systems for Video Technology**, [S.l.], v.14, n.6, p.796–812, 2004.

XU, Y.; YU, H. Contour-Based Motion Segmentation Using Few Priors. In: SIGNAL PROCESSING, THE 8TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2006. v.2.

YAN, J.; POLLEFEYS, M. A General Framework for Motion Segmentation: independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In: ECCV 2006. **Proceedings...** [S.l.: s.n.], 2006.