Técnicas para Segmentação Temporal de Vídeo Digital

Thiago Teixeira Santos [1]
thsant@ime.usp.br
Carlos Hitoshi Morimoto (orientador)

25 de junho de 2002

[1]Suportado pelo Laboratório de Imagens Visão e Estruturas Espaciais (LIVES) das Faculdades SENAC.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Material audiovisual (AV) em formato digital já soma enormes acervos em bancos de dados pelo mundo todo. A Internet e as grandes larguras de banda tornaram possível distribuir este material e a potência dos computadores pessoais atuais processá-la. Surge, então, a necessidade de estruturar esta informação, indexá-la e recuperá-la com base em seu conteúdo (*content-based information retrieval*, CBIR).

Tal material, para que possa ser pesquisado e navegado, deve ser estruturado em unidades menores, que permitam sua representação, de onde possam ser extraídas características para indexação, pesquisa e rápido acesso, pelo usuário, aos pontos chave de seu interesse.

Daí surge a idéia de *shot* e a *segmentação temporal*. Um *shot* é um conjunto de quadros sucessivos em uma seqüência de vídeo digital, que pode ser considerado como uma unidade de decomposição. De um *shot*, podem ser extraídas características diversas, tal como objetos, formas dominantes, cor, movimento espacial e movimento temporal dentre centenas de outras que alimentam os classificadores e permitem CBIR automática. A segmentação temporal é o processo pelo qual a seqüência de vídeo é decomposta nestas unidades.

O trabalho em Tópicos em Ciência da Computação (MAC 5701) foi ambientar-nos com o problema. Consistiu na leitura de um conjunto de artigos, sete dos quais foram inicialmente citados como proposta no plano de estudos que antecedeu esta monografia, implementações e testes de dois dos algoritmos mais utilizados na área, a implementação e teste de um novo método proposto por nós e um artigo submetido ao Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens (SIBGRAPI).

O restante desta monografia segue com uma apresentação em maior profundidade do problema da segmentação temporal, na Seção 1. Nas Seções de 2 a 7, resumimos os artigos lidos. Na Seção 8, descrevemos os resultados obtidos com as implementações, como apresentados no artigo escrito para o SIBGRAPI. Nossas conclusões e trabalho futuro seguem, finalmente, na Seção 9.

2 O problema da segmentação temporal

Um *shot* é uma seqüência de quadros sucessivos obtidos por uma operação da câmera, isto é, o que é obtido entre o "ligar e desligar" da câmera. A Figura 1 ilustra a transição entre dois *shots* no filme *Challenge at Glen Canyon*, obtido a partir do Open Video Project [15].

Figura 1: Uma transição abrupta entre dois *shots* de *Challenge at Glen Canyon*.

As transições entre *shots* sucessivos passa freqüentemente por efeitos de edição comuns na televisão e no cinema. A Figura 2, referente ao filme *Airline Safety and Economy* [15], ilustra o efeito obtido pela aplicação de *fades*. A transição entre os *shots* é feita através de um *fade-out*, um escurecimento gradual dos quadros do primeiro *shot*, até obter-se um quadro negro, seguido logo após por um *fade-in*, um clareamento do quadro negro até o surgimento dos quadros do segundo *shot*. Já a Figura 3 mostra um outro efeito bastante comum: a *dissolução* é obtida com um escurecimento do *shot anterior* concorrentemente ao clareamento do *posterior*, obtendo assim uma mescla dos dois segmentos durante um certo intervalo de tempo.

Figura 2: Uma transição gradual entre dois *shots* via *fades* em *Airline Safety and Economy*.

Seja $V = \langle q_0, q_1, q_2, ..., q_T \rangle$ uma seqüência de vídeo digital, onde q_i

Figura 3: Uma transição gradual entre dois *shots*, via dissolução, em *Airline Safety and Economy.*

(quadro) é uma imagem digital de dimensão $M \times N$. O problema da delimitação de *shots*, ou *shot boundary detection* (SBD), como iremos nos referir daqui em diante, é encontrar uma seqüência de quadros $\langle q_{i_0}, q_{i_1}, ..., q_{i_S} \rangle$ tal que $\langle q_{i_k}, ..., q_{i_{k+1}} \rangle$ é um *shot* para todo $0 \leq k \leq S-1$. Em outras palavras, queremos delimitar todos os *shots* existentes em V. Este problema também é conhecido por *segmentação temporal de vídeo.*

Posto o problema, como avaliar a qualidade de uma solução para SBD? Baseando-se em anotação humana da seqüência de vídeo, duas medidas são importantes: *cobertura* e *precisão*, como observado por Ferman e Tekalp [4]. Introduzidas para a avaliação de performance de sistemas de recuperação de informação, elas são definidas por

$$\text{cobertura} = \frac{\text{corretos}}{\text{corretos} + \text{falsos negativos}}, \tag{1}$$

$$\text{precisão} = \frac{\text{corretos}}{\text{corretos} + \text{falsos positivos}}. \tag{2}$$

Na próxima seção, apresentaremos o resumo de um artigo que dá um panorama da área, abordando as soluções clássicas para o problema de SBD, tanto no domínio de vídeo não-comprimido como no domínio MPEG.

3 Abordagens em Segmentação Temporal de Vídeo

Koprinska e Carrato apresentaram um *survey* da área em um artigo publicado em 2001 pela *Image Communication* [7]. Eles dividem os métodos em dois grupos: os que atuam na seqüência de vídeo sem compressão (uma seqüência simples de imagem ordenadas) e os que trabalham em vídeo codificado no padrão MPEG, utilizando características do próprio sistema de codificação para auxliar na resolução do problema.

3.1 Segmentação em Vídeo Não-comprimido

De maneira geral, para transições abruptas, os métodos baseiam-se em definir uma medida de similaridade entre os quadros do vídeo, comparar quadros sucessivos e, se a similaridade for baixa o suficiente, segundo algum critério, definir aquele ponto como uma fronteira entre *shots*.

Conforme o método utilizado para avaliar a similaridade entre quadros, Koprinska e Carrato dividem os algoritmos em três grupos: comparação de pixels, de histogramas e por blocos.

3.1.1 Comparação de Pixels

Considere uma função

$$d_p(q_i, q_j) = \sum_{x=1}^{M} \sum_{y=1}^{N} |q_i(x,y) - q_j(x,y)| \tag{3}$$

onde $q_i(x, y)$ é a intensidade do pixel (x, y) em uma quadro em escala cinza q_i, ou, para quadros coloridos no domínio RGB (*red*, *green*, *blue*),

$$d_p(q_i, q_j) = \sum_{c \in \{R,G,B\}} \sum_{x=1}^{M} \sum_{y=1}^{N} |q_i^c(x,y) - q_j^c(x,y)|. \tag{4}$$

Podemos dizer que um quadro q_i delimita um *shot* se

$$d_p(q_i, q_{i+1}) > k_p, \tag{5}$$

onde k_p é um limiar definido experimentalmente.

Este método, também conhecido como *template matching*, é de fácil implementação (apresentaremos alguns resultados na Seção 8). Porém, ele possui uma grande desvantagem: é muito sensível a grandes alterações em pequenas áreas dos quadros. Trata-se de um sério problema pois tal situação ocorre quando objetos estão se movimentando no vídeo ou quando ocorre uma operação de câmera, como uma panorâmica ou um *zoom*. Por isto, o método gera uma grande quantidade de falsos positivos, perdendo precisão. Filtros de borramento podem reduzir este problema, já que forçam o *template matching* a levar em consideração alguma informação sobre a vizinhança de cada pixel. Outra forma de compensar esta deficiência é a abordagem em blocos.

3.1.2 Comparação de Histogramas

Uma abordagem mais robusta às movimentações diversas (câmera e objetos) é a comparação por histogramas. Um histograma h_i de uma imagem em níveis de cinza q_i é um vetor L-dimensional, onde L é o número de níveis e $h_i[l]$ é igual ao número de pixels na imagem com nível de cinza igual a l. Considere então a função

$$d_h(q_i, q_j) = \sum_{l=1}^{L} |h_i[l] - h_j[l]|. \tag{6}$$

Daí, podemos dizer que um quadro q_i delimita um *shot* se

$$d_h(q_i, q_{i+1}) > k_h, \tag{7}$$

onde k_h é um limiar definido.

Koprinska e Carrato citam o método de Zhang et al. [13], onde os níveis de cinza são substituídos por 64 valores ($0 \leq l \leq 63 = L$), obtidos pela combinação dos dois bits mais significativos de cada um dos três componentes de cor em quadros coloridos (R, G e B). Este método é computacionalmente mais eficiente pois gera um número bem menor de valores possíveis no histograma, se comparado aos 2^{24} valores utilizados ao se trabalhar com os 8 bits das 3 cores.

Alguns pesquisadores, em detrimento da equação 6, utilizaram o teste χ^2 entre os histogramas, na esperança de acenturar as diferenças entre os *shots*. Porém, tal método acentuou também as diferenças devido ao movimento, de modo a não melhorar a performance obtida por d_h, além de ser computacionalmente mais pesado. Os autores também relatam o trabalho de Gargi et al. [5] sobre a performace da abordagem de histogramas em domínios de cor diferentes, sendo os sistemas YIQ, L*a*b* e Munsell considerados como os de melhor acurácia.

A desvantagem da abordagem por histogramas se deve ao fato que quadros completamente diferentes podem ter histogramas muito similares, levando a falsos negativos. Porém, a probabilidade de situações como essa ocorrem é baixa de modo que não compromete a eficácia do método. Tal desvantagem se origina do fato dos histogramas não levarem em consideração qualquer informação espacial dos quadros (exatamente o oposto do que ocorre com *template matching*). Para compensar o excesso de informação espacial na comparação de pixels e sua total ausência na comparação por histogramas, surgiram as abordagens por blocos.

3.1.3 Comparação de Blocos

Aplicar as medidas de similaridade a blocos da imagem ao invés da imagem inteira tem se mostrado uma forma interessante de incrementar a performance das mesmas. Considere o conjunto de blocos b sem sobreposição obtidos pela aplicação de uma grade ao quadro (Figura 4). Desta forma, decidimos se um quadro q_i é um delimitador se

$$\sum_{b}^{B} c_b d(q_i^b, q_{i+1}^b) > k_b \tag{8}$$

onde B é o número total de blocos, q_i^b é o bloco b no quadro i e k_b é um limiar definido. A função d é uma variante para blocos das medidas de *template matching* ou diferença de histogramas e c_b é um peso para o bloco.

Técnicas mais complexas poderiam buscar por casamento de blocos, ao invés de simplesmente comparar blocos correspondentes na grade, o que tornaria a abordagem ainda mais robusta às diversas movimentações (tal idéia é revisitada nas técnicas com *macroblocks* e compensação de movimento no domínio MPEG).

Koprinska e Carrato citam também o trabalho de Xiong et al. [11] em blocos, que nós abordaremos na Seção 7.

3.1.4 Abordagem por Modelos

As abordagens vistas até o momento são *bottom-up*, pois se concentram na análise dos dados. Há um outro grupo de métodos, *top-down*, que visam estabelecer modelos matemáticos para as transições entre *shots*. Tais métodos

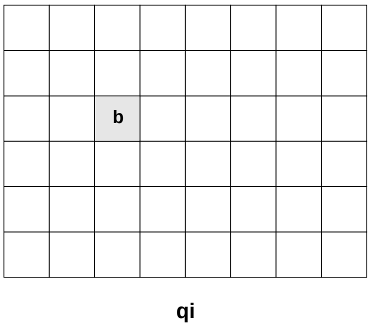

Figura 4: Divisão do quadro q_i em um conjunto de blocos sem sobreposição.

visam modelar os diversos tipos de transição através de funções lineares, distribuições probabilísticas de diferenças inter-quadros ou por modelos markovianos. Na Seção 5, apresentaremos o trabalho de Yu et al., também citado por Koprinska e Carrato, que ilustra uma abordagem *top-down*.

3.2 Segmentação de Vídeo no Domínio MPEG

A codificação de uma seqüência de vídeo em formato MPEG gera um série de estrutura que podem ser utilizadas como base em métodos de segmentação temporal. Antes de apresentar estas abordagens, apresentaremos um breve resumo sobre codificação MPEG.

3.2.1 A Codificação MPEG

O padrão JPEG de compressão reduz redundância espacial no vídeo, comprimindo os quadros como imagens simples. Ele representa a imagem pelos coeficientes da sua transformada dos cossenos (DCTs) que são posteriormente quantificados, eliminando-se aqueles coeficientes com menor quantidade de informação. O resultado é então representado por códigos de Huffman. A Figura 5 dá uma visão geral do processo.

O padrão MPEG de compressão utiliza o JPEG para reduzir a redundância espacial e usa a técnica de compensação de movimento para reduzir a redundância temporal. Há três tipos de quadros em um vídeo MPEG: I (*intra*) quadros, P (*predicted*) quadros e B (*bi-directional*) quadros. Os I quadros servem como os quadros base de referência, de onde os outros serão reconstruídos, e é codificado via JPEG. Os quadros P são reconstruídos a partir de um quadro I (ou de outro quadro P) que o sucede na seqüência, através do procedimento ilustrado na Figura 6.

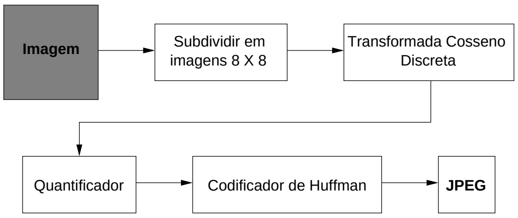

Figura 5: Codificação JPEG.

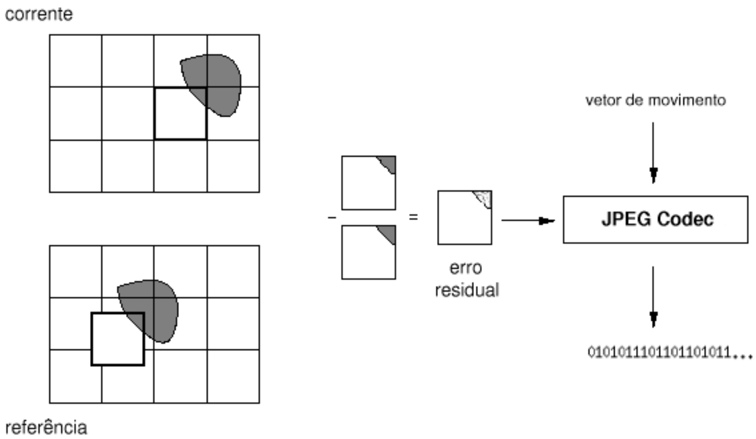

Figura 6: Codificação MPEG. O vetor de movimento e o erro residual são armazenados para posterior reconstrução.

Os quadros são divididos em blocos (*macroblocks*). Para cada bloco do quadro P sendo reconstruído, é procurado um bloco referência no quadro I (ou P) mais próximo que o sucede. Os quadros são comparados e o erro residual, junto com o vetor de movimento (*motion vector*), é codificado via JPEG. A reconstrução dos blocos dos quadros B ocorre de maneira similar, mas através da interpolação entre dois blocos de referência, um pertencente a um quadro anterior e outro a um posterior. Entretanto, quando os blocos são tão dissimilares que sua reconstrução via compensação de movimento resulta em um código *maior* que sua respectiva codificação JPEG, esta é preferida àquela.

3.2.2 Técnicas com Coeficientes DCT

Koprinska e Carrato citam os trabalhos de Arman et al. [1] e Zhang et al. [14] com coeficientes DCT obtidos dos quadros I de seqüências em MPEG. Arman et al. atilizam um subconjunto dos coeficientes DCT c_j para criar um vetor $v_i = \langle c_1, c_2, ..., c_n \rangle$ para cada I quadro q_i. A função de diferença é dada pelo produto interno dos vetores dos quadros comparados:

$$d_{DCT}(q_i, q_j) = \frac{v_i \cdot v_j}{|v_i||v_j|}. \tag{9}$$

Dois limiares k_1 e k_2, $0 < k_1 < k_2 < 1$ são utilizados. Se $d_{DCT}(q_i, q_j) > k2$, a transição é marcada. Porém, se $k_1 \leq d_{DCT}(q_i, q_j) \leq k2$, os quadros são descompactados e checados via diferença de histogramas. Note que já não se pode falar em diferença entre q_i e q_{i+1} pois não há garantias que os I quadros observados sejam sucessivos (é improvável que o sejam).

Já Zhang et al. calculam as diferenças entre os coeficientes correspondentes nas duas imagens, obtendo uma medida de forma similar a diferença de pixels. A decisão é novamente baseada na superação de um limiar.

Estes dois algoritmos possuem grandes desvantagens. Sua amostragem temporal, devida ao uso exclusivo de I quadros, pode levar a muitos falsos positivos. A abordagem também é sensível às movimentações de objetos e operações de câmera.

3.2.3 Técnicas com *Macroblocks*

O modo como os blocos de P e B quadros são reconstruídos dá boas dicas sobre as fronteiras dos *shots*. Se a maioria dos blocos de um quadro P são codificados via JPEG, isto é, não puderam ser reconstruídos de um I (ou P) quadro sucessor, significa que é um bom candidato a delimitador. Já um B quadro é um bom candidato quando a reconstrução da maioria dos seus blocos se dá principalmente via seu antecessor.

3.2.4 Vetores de Movimento

Se operações com a câmeras são uma fonte de falsos positivos, principalmente em métodos como o *template matching*, então identificá-las é uma forma de incrementar a robustez dos métodos. Os vetores de compensação de movimento, utilizados no padrão de MPEG, são um bom indicador para tais operações, como pode se ver na Figura 7, pois tomam padrões facilmente identificáveis para várias das possíves movimentações da câmera.

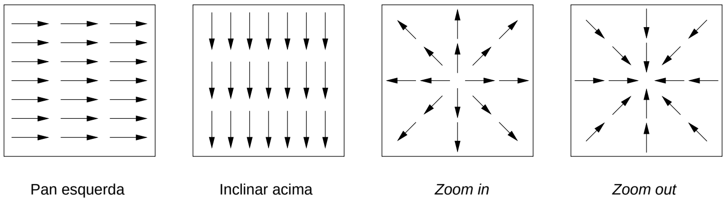

Figura 7: Padrões observados no vetores de movimento para diferentes operações de câmera.

Há métodos no domínio MPEG combinando diversas das técnicas vistas acima: *macroblocks* com vetores de movimento, *macroblocks* e coeficientes DCT com vetores, etc. Há, ainda, outras técnicas, baseadas em imagens DC, que apresentaremoas na Seção 6 quando falarmos do trabalho de Yeo e Liu [9]. A vantagem de trabalhar no domínio MPEG é primeiramente não criar *overhead* devido a decodificação do vídeo e, segundamente, operar em uma quantidade menor de dados, beneficiando-se da compressão.

3.3 *Twin-comparison*

A maioria dos métodos vistos até o aqui, como foram descritos, procuram por transições abruptas. Diferenças acumuladas podem ser utilizadas quando se trata de transições graduais, como fazem Zhang et al. [13] em seu *twin comparison.*

Neste método, uma função diferença (no caso, a de histogramas) é utilizada contra dois limiares k_h e k_1, $k_1 < k_h$. Se a diferença entre os quadros exceder k_h, uma transição abrupta é declarada. Se, porém, a diferença não passar do limiar maior mas exceder k_1, o quadro é apontado como um possível início de uma transição gradual, q_s. O sistema, então, deveria observar o progresso das diferenças acumuladas (Figura 8). Se a diferença acumulada com o tempo exceder k_h, o quadro corrente é declarado término, q_e, e, junto a q_s, demarcará uma transição gradual. Entretanto, se antes disso a diferença consecutiva ficar novamente abaixo de k_1, q_s é descartado e o processo se inicia novamente a partir dali.

É importante observar que, em transições graduais como *fades*, a diferença consecutiva pode voltar a ficar abaixo de k_1. Valores de tolerância podem ser usados, então, para evitar que o candidato q_s seja descartado prematuramente. *Twin-comparison* é um método simples com bons resultados e que pode ser usado com outras diferenças além da de histogramas, como

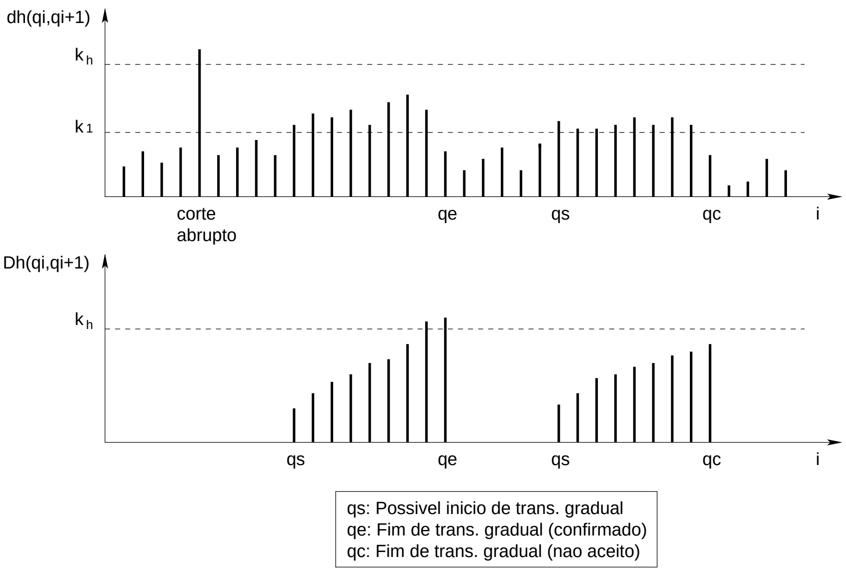

Figura 8: Diferença de histogramas em *twin-comparison*: consecutiva (acima) e acumulada (abaixo).

template matching ou diferença entre coeficientes DCT.

3.4 Conclusões

Koprinska e Carrato terminam seu artigo com algumas considerações importantes. A maioria dos métodos não é capaz de classificar as transições graduais quando as encontram, além de ainda terem muita dificuldade de discerní-las da movimentação de objetos. O problema do ajuste dos vários limiares é outra questão importante (*clustering* e uma possível solução, como veremos na Seção 4). Os autores finalizam com o apelo pela uniformização de testes e pela implementação dos vários algoritmos de forma acessível por várias arquiteturas, via interface *Web* ou programação Java.

4 Melhorias via Filtragem e *Clustering*

Em dezembro de 1998, o *Journal of Visual Commuinication and Image Representation* publicou um artigo de Ferman e Tekalp [4] sobre melhorias das medidas tradicionais (Seção 3) com o uso de filtragem e *clustering*. Medidas como diferenças de pixels, histogramas, coeficientes DCT, etc. não só podem ser melhorados separadamente, via filtros, como também podem ser combinados em *clustering* para compensarem suas deficiências.

4.1 Filtragem dos Sinais da Diferenças

A abordagem de Ferman e Tekalp se baseia no sinal obtido pelas diferenças quadro a quadro ao longo do tempo, ou seja, $i \times d(q_i, q_{i+1})$. Dentro de um *shot*, as diferenças deveriam resultar em um sinal com lenta variação. Quando uma grande variação ocorre, entretanto, um pico deveria ser observado. Detectar transições seria, então, procurar por tais picos, via limiarização. Porém, novamente as movimentações de objetos e da câmera dificultam o problema, gerando picos espúrios. A Figura 9 exibe o gráfico da diferença entre histogramas, d_h, em um segmento do filme *Eu, tu, eles*.

Para aumentar a robustez com relação às movimentações, técnicas de filtragem, comuns às áreas de processamento de sinais e de imagens, podem ser utilizados para destacar os picos mais interessantes e reduzir os espúrios. Considere os sinais $d[i] = d_h(q_i, q_{i+1})$ e $\tilde{d}[i]$, este último obtido pela passagem de um filto passa-baixas em $d[i]$. Para submeter à limiarização, utilizamos o sinal $g[i]$ dado por

$$d'[i] = d[i] - \tilde{f}[i]; \tag{10}$$

$$g[i] = \begin{cases} d'[i] & \text{se } f'[i] > 0 \\ 0 & \text{caso contrário} \end{cases} \tag{11}$$

Os filtros passa-baixas utilizados por Ferman e Tekalp foram os filtros *média* e *mediana*. O próprio sinal $g[i]$ pode ser filtrado também, numa repetição do processo. Os autores mostram que os melhores resultados são alcançados quando o filtro mediana é utilizado para processar $g[i]$, já previamente obtido de $d[i]$ pelo filtro da média, obtendo-se assim um novo sinal $g_2[i]$. O filtro da média, utilizado primeiramente, atenua flutuações no sinal *intra* e *intershots*. O mediana, em seguida, atenua ainda mais as flutuações *intrashots* sem prejudicar os picos *intershots*.

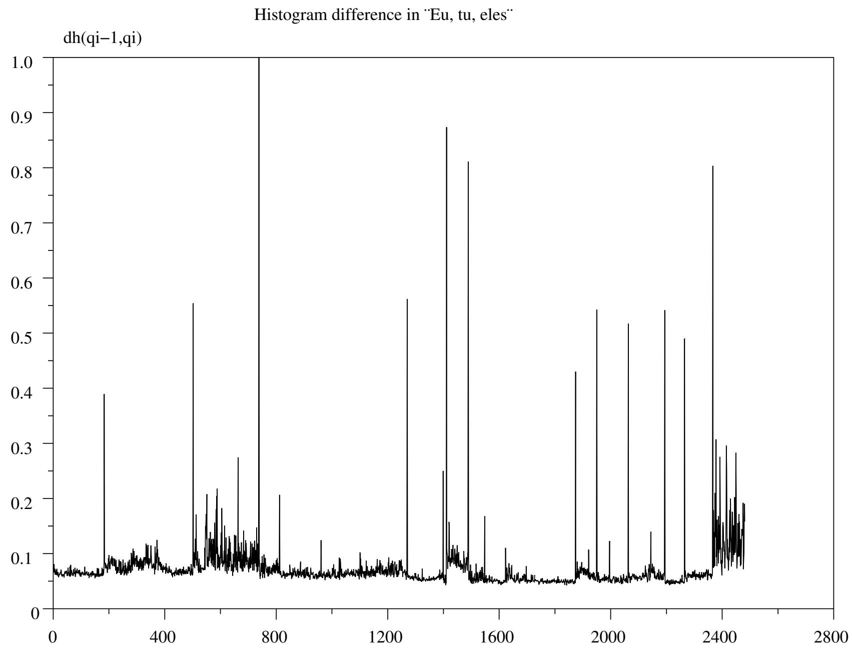

Figura 9: $d_h(q_i, q_{i+1})$ em um segmento do filme *Eu, tu, eles*.

Uma questão importante ao se lidar com os filtros baseados em média e mediana é a do tamanho da janela a ser utilizada. Janelas longas degeneram tanto a cobertura quanto a precisão. Além disso, como as características do sinal mudam ao longo do tempo (cenas de amor e cenas de ação em um filme, com certeza, não produzem sinais com flutuações semelhantes!) um tamanho fixo para a janela não leva aos melhores resultados. Uma escolha adaptativa para o tamanho é o mais adeqüado: Ferman e Tekalp testam amostras do sinal com cinco janelas diferentes, de tamanhos 7, 11, 15, 21 e 31. Calcula-se a variância da amostra para as janelas e, daí, configura-se o filtro da média para o tamanho que produziu a maior variância, com o objetivo de reduzir os picos espúrios por movimentação, enquanto o filtro mediana é utilizado com o tamanho que produziu a menor variância, afim de privilegiar as características locais do sinal.

4.2 Clustering

Combinar duas medidas capazes de compensar as deficiências uma da outra é uma alternativa que produz bons resultados. Ferman e Tekalp combinam as diferenças entre pixels e entre histogramas (d_p e d_h, vistas na Seção 3), em um *clustering k-means* ($k = 2$), o qual classifica cada quadro como sendo *intrashot* ou fronteira, como pode ser visto no *scatter plot* da Figura 10, referente ao filme *Eu, tu, eles*.

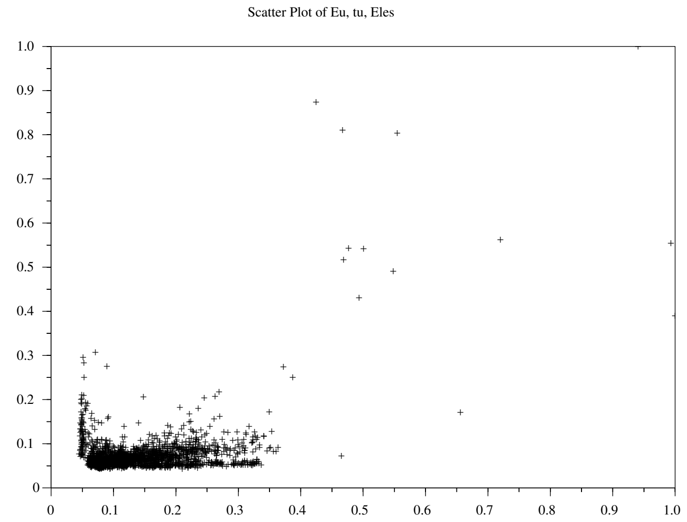

Figura 10: Diferença entre pixels × diferença entre histogramas em um fragmento de *Eu, tu, eles*. Os quadros *intrashot* encontram-se mais próximos da origem e os fronteiriços mais afastados.

O uso de sinais filtrados incrementa a performance dos resultados obtidos, pois afastam ainda mais os centros dos dois *clusters*. Os melhores resultados são obtidos ao se utilizar os sinais $g_{p2}[i]$ e $g_{h2}[i]$, referentes às diferenças de pixels e histogramas, respectivamente.

O algoritmo proposto pelos autores segue abaixo. Como processar toda a seqüência de vídeo é muito longo em tempo, sobretudo para um sistema interativo, o algoritmo trabalha sobre segmentos do vídeo, gerando resultados continuamente ao durante o processo:

```
FermanTekalp-SBD
1   Selecione um limiar de rejeição k_r
2   Selecione um intervalo [i_1, i_2]
3   Proceda com o clustering 2-means no intervalo
4   Calcule as médias dos clusters, μ_1 μ_2
5   Se |μ_1 − μ_2| < k_r
6       Descarte os quadros candidatos a fronteira em [i_1, i_2]
7       Defina um novo intervalo [i_1, i_3], i_3 > i_2
8       Retorne para o passo 3
9   Caso contrário
10      Rotule os candidatos em [i_1, i_2] como fronteiras
11      Defina um novo intervalo [i_2, i_3]
12      Retorne para o passo 3
```

4.2.1 Resultados

A Tabela 1 apresenta os resultados obtidos pelos autores para uma seqüência de vídeo em MPEG-1 de 25 minutos. Tal seqüência possuia cenas de ação, dialógos, rápidas panorâmicas de câmera e os dois tipo de transição, abruptas e graduais. A tabela nos mostra resultados obtidos com e sem filtragem e, nas duas últimas linhas, o resultado da combinação das medidas em *clustering 2-means*. Os histogramas utilizados em d_h e g_{h2} foram obtidos no espaço de cor YCbCr, via concatenação dos valores nos três canais.

	Corretos	Perdidos	Falsos +	Cobertura	Precisão
d_h	478	44	155	0,91570881	0,75513428
g_{h2}	473	49	42	0,90613027	0,9184466
d_p	516	6	4171	0,98850575	0,11009174
g_{p2}	508	14	33	0,97318008	0,93900185
d_p $d_{[}h2]$	517	5	3936	0,99042146	0,1161015
g_{p2} g_{h2}	511	11	26	0,9789272	0,95158287

Tabela 1: Resultados obtidos com e sem a aplicação de filtros em 25 minutos de vídeo (adaptado de Ferman e Tekalp [4]).

Os resultados ilustram bem a eficácia do método de filtragem e *clustering*, sobretudo quando aplicados ao *template matching*. Os autores sugerem que as melhorias obtidas via filtragem são maiores em medidas fortemente suscetíveis à variações *intrashot*, como a diferença entre pixels.

5 Modelos para Transformações

Como visto na Seção 3, alguns métodos utilizam modelos para as transformações. Neste artigo de Yu et al. [10], um algoritmo de dois passos varre a seqüência de vídeo uma primeira vez para encontrar transições abruptas, via diferença de histogramas, em quadros de menor resolução (maior velocidade para o sistema). Posteriormente, os *shots* encontrados são submetidos a uma análise por modelos, que busca por *fades* e dissoluções.

5.1 Caracterização de *Fades*

Em *fades*, o primeiro *shot* gradualmente desaparece antes que o segundo gradualmente apareça. Já em dissoluções, os dois processos ocorrem simultaneamente. A partir desta observação, os autores modelam uma transição gradual entre os instantes t_1 e t_2 como:

- *Fades*
  - *Fade out*

$$E(x,y,t) = I(x,y,t_1)\alpha(t) + O(x,y,t_2)[1-\alpha(t)] + B(x,y) \quad (12)$$

  - *Fade in*

$$E(x,y,t) = O(x,y,t_1)\alpha(t) + I(x,y,t_2)[1-\alpha(t)] + B(x,y) \quad (13)$$

- Dissolução

$$E(x,y,t) = I(x,y,t_1)\alpha(t) + I(x,y,t_2)[1-\alpha(t)] + B(x,y) \quad (14)$$

onde $\alpha(t)$ é uma função linear decrescente (Figura 11), $E(x,y,t)$ é a intensidade total do pixel (x,y) no quadro q_t, $I(x,y,t)$ é a intensidade sem *background*, $B(x,y)$ é o *background* e $O(x,y,t) = 0, \forall x,y$. O *background* pode ser originário, por exemplo, de textos e legendas presentes continuamente durante a transição.

Os autores observam, então, que $\alpha(t)$ é uma *blurring function*. Isto significa que, ao final de um *fade-out* e no início de um *fade-in*, as intensidades são dadas por $B(x,y)$ e o número de bordas da imagem é baixo, isto é, o número de pixels de borda (*edge counting*, EC) é próximo de zero. Daí, o algoritmo abaixo é proposto. Seja EC' o gradiente de EC e k_{min}, k_1, k_2, k_3 e k_4 limiares dados. Temos então:

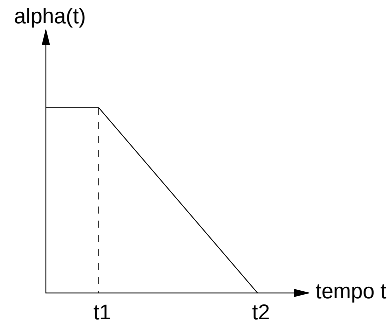

Figura 11: A função $\alpha(t)$ utilizada para modelar *fades* e dissoluções.

YBH-FADEDETECTION
1 Calcule $EC[i]$ para cada quadro q_i no segmento $[s,t]$
2 Procure por um mínimo local em $EC[s,t]$ menor que k_{min}
3 Para cada mínimo i_{min}, encontre os máximos esquerdo e direito mais próximos i_{esq}, i_{dir}
4 Se $\sum_{i \in]i_{esq}, i_{min}]} \frac{EC'[i]}{(i_{min} - i_{esq})} \in [k_1, k_2]$
então o segmento $]i_{esq}, i_{min}]$ é um *fade-out*
5 Se $\sum_{i \in [i_{min}, i_{dir}]} \frac{EC'[i]}{(i_{dir} - i_{min})} \in [k_2, k_3]$
então o segmento $[i_{min}, i_{dir}[$ é um *fade-in*

5.2 Caracterização de Dissoluções

Quanto às dissoluções, podemos dizer que sempre existe um certo i_k tal que

$$I(x,y,i_k) = \frac{I(x,y,i_{esq}) + I(x,y,i_{dir})}{2}, \forall(x,y) \tag{15}$$

onde i_{esq} e i_{dir} representam, respectivamente, o início e o término da dissolução. Baseados nisso, os autores definem a *double chromatic difference* (DCT) para um quadro q_i como

$$DCD[i] = \sum_{x,y} f\left(\left|\frac{I(x,y,i_{esq}) + I(x,y,i_{dir})}{2} - I(x,y,i)\right|\right) \tag{16}$$

onde f é uma função de limiarização. A idéia por trás da DCD é que o quadro que marca o meio da dissolução será um mínimo local. Com isso, temos algoritmo

YBH-DissolveDetection

```
Calcule EC[i] para cada quadro q_i no segmento [s,t]
Procure por um mínimo local em EC[s,t]
Para cada mínimo i_min, encontre os máximos esquerdo e direito mais
    próximos i_esq, i_dir
Para cada i no intervalo [i_esq, i_dir], calcule DCD_i
Se min_DCD ∈ (i_min − n_1, i_min + n_2)
    então (i_esq, i_dir) demarcam uma dissolução
    caso contrário, (i_esq, i_dir) pode ser, ainda
    uma operação de câmera (zoom, pan, etc.)
```

onde n_1 e n_2 são pequenos inteiros utilizados para demarcar uma pequena região em torno do mínimo local $DCD[i_{min}]$.

Yu et al. terminam seu artigo afirmando que seu método é robusto e computacionalmente pouco intensivo. Infelizmente, os autores citam apenas um resultado e, mesmo assim, não há referências nem à cobertura nem à precisão obtida. Nada é dito sobre a seqüência de vídeo utilizada no teste.

6 Análise em Baixa Resolução

Para obter ganhos em velocidade sem perdas expressivas na qualidade da segmentação, podemos operar em versões dos quadros em menor resolução. Yeo e Liu [9] apresentam uma forma de segmentar o vídeo comprimido em MPEG utilizando, para cada quadro, uma imagem DC, que reduz a resolução através da média dos pixels em blocos da imagem.

6.1 Imagens DC

Felizmente, a compressão JPEG utilizada em MPEG permite a construção quase instantânea, para quadros I, de imagens com resolução 8 vezes menor que original.

Como visto na Seção 3 (ver também Gonzalez e Woods [6], capítulos 3 e 6), a compressão JPEG baseia-se na transformada DCT bi-dimensional de cada bloco $N \times N$, dada por

$$DCT(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) cos \left[\frac{(2x+1)u\pi}{2N} \right] \left[\frac{(2y+1)v\pi}{2N} \right] \quad (17)$$

onde

$$\alpha(n) = \begin{cases} \sqrt{\frac{1}{N}} & \text{para } u = 0 \\ \sqrt{\frac{2}{N}} & \text{para } u = 1, 2, ..., N-1 \end{cases} \quad (18)$$

Considere um bloco b de um quadro $q(x,y)$. Como os blocos tem dimensão 8×8, temos que

$$\begin{aligned} DCT^{q^b}(0,0) &= \sqrt{\frac{1}{8}} \cdot \sqrt{\frac{1}{8}} \cdot \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} q^b(x,y) \cdot \cos 0 \cdot \cos 0 && (19) \\ &= \frac{1}{8} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} q^b(x,y) && (20) \end{aligned}$$

$DCT^{q^b}(0,0)$ encontra-se armazenado diretamente na *stream* JPEG.

Um termo DC é a média da intensidade dos pixels em um bloco $N \times N$ de uma imagem. Uma forma de reduzir a resolução de uma imagem é dividí-la em blocos $N \times N$ sem sobreposição e gerar uma nova imagem subtituindo os blocos por seus correspondentes termos DC. Logo, temos uma forma muito eficiente de obter *imagens DC* de um original JPEG pois

$$DC_8^{q^b} = \frac{1}{64}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} q^b(x,y) \quad (21)$$

$$= \frac{1}{8}DCT_8^{q^b}(0,0) \quad (22)$$

Assim, se queremos obter uma seqüência de imagens DC de uma seqüência de vídeo codificada em MPEG, temos, ao menos, uma forma de construir rapidamente essas imagens para os quadros I (como visto anteriormente, estes estão codificados em JPEG). Para obter então uma seqüência completa, precisamos ainda de uma forma de construir as imagens DC para os quadros P e B. Yeo e Liu o fazem através dos vetores de compensação de movimento e dos coeficientes DCT, a partir dos quadros I. Infelizmente, tal processo é caro computacionalmente: oito multiplicações de matrizes 8×8 e quatro somas de matrizes para cada termo DC.

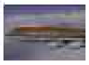

Figura 12: Uma imagem (320×240) e sua correspondente DC_8 (40×30).

Trabalhar com baixa resolução traz outras vantagens, além rapidez em trabalhar em imagens menores. O borramento decorrente da diminuição de resolução torna métodos como *template matching* mais robustos às pequenas movimentações de objetos e câmera. Quanto à diferença de histogramas, observa-se que a diminuição de resolução utilizada pouco altera o sinal das diferenças ao longo do tempo, o que demonstra que o ganho de velocidade quase não afeta a performance da medida.

6.2 Detecção de Cortes Abruptos

Yeo e Liu evitam o uso de limiares globais em benefício de uma abordagem que beneficia a análise local. Tomando o sinal das diferenças dado por $d[i] = d(q_i, q_{i+1})$ uma janela (*slide window*) de tamanho m e um fator k, os autores declaram uma transição abrupta entre q_i e q_{i+1} se

1. $d[i]$ é o máximo no intervalo dado por uma janela simétrica de tamanho 2m+1, ou seja, $d[i-m+1], ..., d[i-1], d[i], d[i+1], ..., d[i+m-1]$.
2. $d[i]$ é k vezes maior que o segundo maior valor na janela.

O parâmetro m deve ser menor que o tamanho do menor *shot* possível pois não deve haver duas transições ocorrendo dentro da janela. A restrição 2 tem como objetivo diminuir o número de falsos positivos devidos a panorâmicas e *zoom*, operações que se manifestam geralmente como uma seqüência de valores altos e próximos, um do outro, no sinal $d[i]$.

6.3 Detecção de Transições Graduais

Analisar a diferença entre quadros sucessivos já não é o bastante quando se trata de transições graduais. Assim, os autores se utilizam de um sinal das diferenças dado por

$$d_i^k = d(q_i, q_{i+k}). \tag{23}$$

A idéia de Yeo e Liu é que, se k for apropriadamente grande, os sinal d_i^k tomará uma forma muito característica durante as transições graduais: se k for maior que o intervalo de transição, *platôs* surgirão no gráfico do sinal.

Para entender o método, é necessário pensar novamente em um modelo linear para as transições graduais, de forma semelhante à Seção 5. Considere, para uma transição gradual entre c_1 e c_2 no intervalo $[\alpha_1, \alpha_2]$:

$$g_n = \begin{cases} c_1, & n < \alpha_1, \\ \frac{c_2 - c_1}{\alpha_2 - \alpha_1}(n - \alpha_2) + c_2, & \alpha_1 \leq n < \alpha_2, \\ c_2, & n \geq \alpha_2. \end{cases} \tag{24}$$

Analisando g_n, observamos que

$$d_i^k(g_n) = \begin{cases} 0, & n < \alpha_1 - k, \\ \frac{|c_2 - c_1|}{\alpha_2 - \alpha_1}[n - (\alpha_1 - k)], & \alpha_1 - k \leq n < \alpha_2 - k, \\ |c_2 - c_1|, & \alpha_2 - k \leq n < \alpha_1 \\ -\frac{|c_2 - c_1|}{\alpha_2 - \alpha_1}(n - \alpha_2), & \alpha_1 \leq n < \alpha_2, \\ 0, & n \geq \alpha_2 \end{cases} \tag{25}$$

Concluimos, então, que surge um "platô" constante ($|c_2 - c_1|$) imediatamente antes da transição, como pode ser observado na Figura 13.

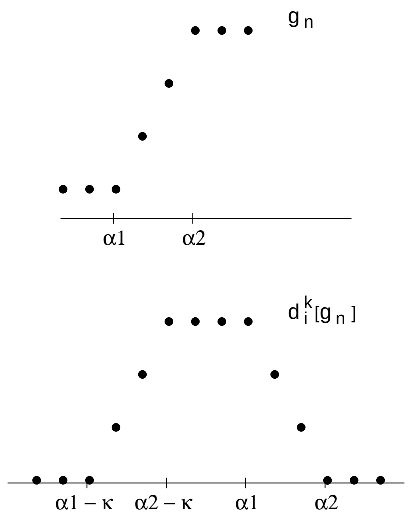

Figura 13: Os sinais g_n e $d_i^k[g_n]$.

Assim, para um certo k fixo, q_i e q_{i+k} delimitam uma transição gradual se

1. d_i^k é próximo a d_j^k, $j = i - s, ..., i - 1, i + 1, ..., i + s$.
2. $d_i^k \geq l \cdot d_{i-\lfloor k/2 \rfloor - 1}^k$ e $d_i^k \geq l \cdot d_{i+\lfloor k/2 \rfloor + 1}^k$, para algum l grande.

As condição 1 e 2 combinadas asseguram a existência do platô. É importante notar que o método funciona mesmo no caso das transições não serem lineares, bastando apenas a existência do "platô", independendo da forma como ele "ascende" ou "descende".

6.4 Resultados

Yeo e Liu em seu artigo exibem vários resultados, variando seus parâmetros m, k e l. A cobertura obtida é de $0,95$. Porém, por citarem apenas o número de falsos positivos e negativos e não apresentarem o número total de transições em seu conjunto de testes, não foi possível estimar a precisão do método.

7 Amostragem Espacial e Temporal

Ganhos em velocidade com pouca perda em eficiência podem ser obtidos amostrando-se os quadros tanto espacialmente como temporalmente. Os trabalhos de Xiong e Lee [11] [12] exemplificam tal processo.

7.1 Amostragem Espacial: *Net Comparison*

Xiong, Lee e Ip [11] estudam como uma amostragem baseada em blocos poderia impactar a eficiência da segmentação. Como primeiro passo, eles definem uma divisão para o quadro, baseada em uma grade de blocos sem sobreposição, de forma semelhante ao visto previamente na Figura 4. O tamanho do bloco tem estreita relação com a robustez do método a movimentações: quanto menor o bloco, mais ele se aproxima da comparação entre pixels e torna-se mais sensível a tais eventos. Os autores propõe, então, que os blocos tenham dimensão $2\delta \times 2\delta$, onde δ é velocidade máxima sem quebra [2], que traduz o maior movimento possível entre dois quadros. Na prática, $\delta = 10$ funciona bem, segundo os autores. Isto garante que, não havendo uma transição abrupta, ao menos um quarto dos pixels de blocos correspondentes são iguais.

Os autores definem uma transição abrupta como uma mudança em, no mínimo, 50% dos blocos correspondentes entre dois quadros sucessivos. Há uma mudança entre dois blocos q_i^b e q_j^b se

$$\left| \sum_{(x,y)} q_i^b(x,y) - \sum_{(x,y)} q_j^b(x,y) \right| > k_b \tag{26}$$

onde k_b é um limiar.

A amostragem espacial consiste aqui em examinar apenas alguns dos blocos. Seja B o número de blocos em cada quadro, B_c o número de quadros modificados (*changed*), S o número de blocos amostrados (*sampled*) e S_c o número blocos modificados dentre os amostrados. O erro obtido pela amostragem é dado por $Er = Er_1 + Er_2$ onde Er_1 e Er_2 são

$$\begin{aligned} Er_1 &= Pr\left(\frac{S_c}{S} > 0.5 | \frac{B_c}{B} < 0.5\right) & (27) \\ &= \frac{1}{0.5B} \sum_{I=0}^{0.5N-1} \frac{\sum_{J=0.5S}^{S} C_I^J C_{B-I}^{S-J}}{C_S^B} & (28) \end{aligned}$$

$$Er_2 = Pr\left(\frac{S_c}{S} < 0.5 | \frac{B_c}{B} > 0.5\right) \quad (29)$$

$$= \frac{1}{B - 0.5B + 1} \sum_{I=0.5B}^{B} \frac{\sum_{J=0}^{0.5S-1} C_I^J C_{B-I}^{S-J}}{C_S^B}. \quad (30)$$

A Figure 14 mostra o erro em função de S para uma imagem 318×238 ($B = 192$). Observe que, com cerca de metade dos blocos, já é possível obter taxas de erro bem pequenas.

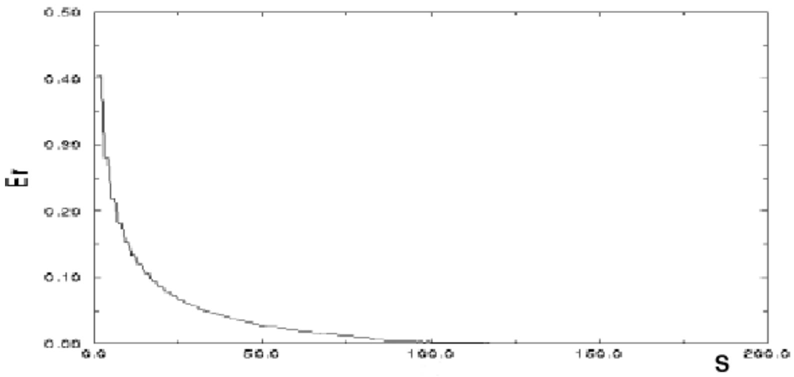

Figura 14: O erro obtido em função da amostragem para uma imagem 318×238 (de Xiong, Lee e Ip [11]).

7.2 Amostragem Temporal: *myStep*

Todos os métodos vistos até aqui implicavam em uma varredura de todos os quadros da seqüência de vídeo. Xiong e Lee [12] mostram que, se pudermos fazer suposições com relação ao tamanho das transições graduais e dos *shots* em $V = \langle q_0, q_1, q_2, ..., q_T \rangle$, é possível detectar todos as transições em menos que $O(T)$.

Tome uma constante $myStep$ tal que

$$2 \cdot gt_{max} < myStep < shot_{min} \quad (31)$$

onde gt_{max} denota o tamanho, em quadros, da maior transição gradual existente em V e $shot_{min}$ o tamanho do menor dos *shots* em V.

Com base nisso, os autores propõe o algoritmo

```
XIONGLEE-SBD
1   l ← 0
2   enquanto l < T
3      r ← l + myStep
4      se d(q_l, q_r) > k_1
5         l_0 ← l
6         r_0 ← r
7         enquanto r − l ≠ 1
8            m ← (l + r)/2
9            se d(q_l, q_m) > d(q_m, q_r)
10              então l ← m
11              caso contrário, r ← m
12        se d(q_l, q_r) > k_1
13           então declare transição abrupta em l
14           caso contrário, se d_e(q_{l_0}, q_{r_0}) > k_2
15              declare transição gradual em l
16     l ← l + myStep/2
```

onde k_1 e k_2 são limiares e $d_e(q_i, q_j)$ é uma função que mede a diferença entre as bordas dos quadros q_i e q_j.

As restrições sobre o valor de $myStep$ garantem que, para toda transição, abrupta ou gradual, um teste será aplicado a um par de quadros q_l e q_r, respectivamente antes e depois da transição (Figura 15 (c)). A iteração da linha 2 de XIONGLEE-SBD, quando encontra o sistema no estado retratado em (a), leva-o aos estados (b) ou (c). O estado (b) é reduzido pelo laço ao estado (c). A partir de (c), as linhas 3-15 garantem a examinação da transição. As linhas 7-11 encapsulam uma busca binária que estreita o intervalo entre l e r até isolar a transição, gradual ou abrupta, representadas, respectivamente, por uma área hachurada e um X na Figura 15. É fácil ver que o algoritmo leva $O(\frac{T}{myStep} + S\lfloor \log myStep \rfloor + 1) < O(T)$ para processar o vídeo V, onde S é o total de transições na seqüência, sendo $S << T$.

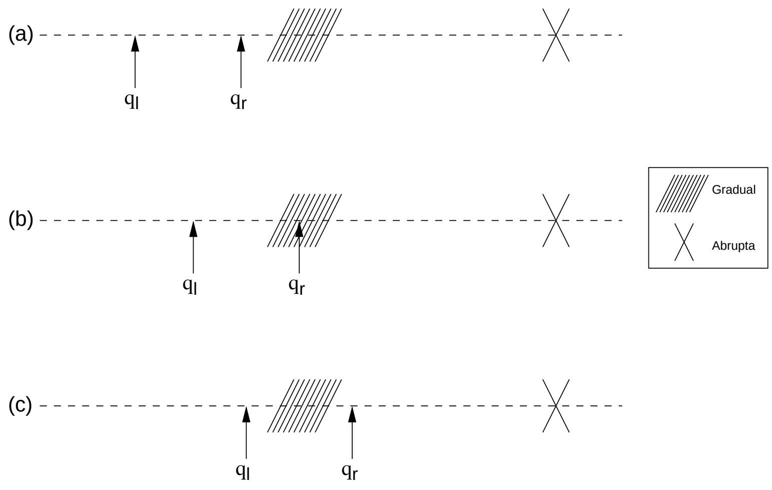

Figura 15: O uso de *myStep* para amostragem temporal de seqüência de vídeo, sem perda de transições.

8 Implementações e Testes

Em maio de 2002, submetemos um artigo [8] ao Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens (SIBGRAPI), infelizmente não aceito pelo comitê. O artigo fazia uma rápida apresentação do problema, apresentava as medidas de *template matching* e diferença de histogramas e introduzia a diferença entre a contagem dos pixels de borda (*edge counting*) como medida para análise de todo o segmento, ao contrário dos métodos de Yu et al. [10] e Xiong e Lee [12] que utilizavam variantes desta abordagem apenas no contexto de transições graduais.

Foram utilizadas seqüências com cerca de 5 minutos de duração de três filmes diferentes: *Eu, tu, eles*, com movimentação contínua e diálogos, *Villa-Lobos*, um segmento de diálogo entre dois personagens, e a animação por computação gráfica *Shrek*, repleta de cenas de ação e intenso uso de *motion blur* nos segmentos de maior movimento. Os quadros coloridos foram extraídos a uma taxa de 15 quadros/segundo e os dados foram analisados sem compressão.

O método utilizado consistia em obter o sinal temporal $d_*[i]$ das diferenças entre quadros sucessivos, para cada tipo de medida (d_p, d_h e d_e) e aplicar vários limiares para identificação de transições (abruptas somente). Baseado no observado no trabalho de Ferman e Tekalp [4], aplicamos duas filtragens diferentes aos sinais: a subtração pela mediana (janela 5) do sinal, $\Im_m^5 d_*[i]$, e uma derivação discreta, obtida por

$$d'_*[i] = d_*[i] - d_*[i-1]. \tag{32}$$

Os resultados podem vistos nas Tabelas 2, 3 e 4, que retratam o obtido, em cada caso, pelo limiar que alcançou melhor desempenho. C denota cobertura e P precisão. Note como *template matching*, cuja precisão é sempre comprometida por picos espúrios no sinal, causados por movimentação, tem o maior incremento relativo de eficiência pela aplicação dos filtros.

	$d_p[i]$		$d_p[i] - \Im_m^5 d_p[i]$		$d'_p[i]$	
Filme	C	P	C	P	C	P
Eu, tu, eles	1	0.857	1	0.923	1	0.857
Villa-Lobos	1	1	1	1	1	1
Shrek	0.951	0.609	0.951	0.928	0.951	0.928

Tabela 2: Diferença de pixels aplicada a segmentos de três filmes.

	$d_h[i]$		$d_h[i] - \Im_m^5 d_h[i]$		$d'_h[i]$	
Filme	C	P	C	P	C	P
Eu, tu, eles	1	1	1	1	1	1
Villa-Lobos	1	1	1	1	1	1
Shrek	0.951	0.722	0.951	0.796	0.951	0.722

Tabela 3: Diferença de histogramas aplicada a segmentos de três filmes.

	$d_e[i]$		$d_e[i] - \Im_m^5 d_e[i]$		$d'_e[i]$	
Filme	C	P	C	P	C	P
Eu, tu, eles	0.917	0.846	0.917	1	0.917	0.917
Villa-Lobos	1	1	1	1	1	1
Shrek	0.829	0.065	0.756	0.188	0.756	0.146

Tabela 4: Diferença de contagem de bordas aplicada a segmentos de três filmes.

9 Conclusões

Nossa solução via *edge counting* mostrou-se cara computacionalmente e pouco efetiva. A diferença de histogramas tem sido a medida preferida e mais adotada pelos vários pesquisadores da área, dada sua robustez. Contudo, diferença de pixels com a aplicação de filtros também é capaz de obter bons resultados.

SBD é um problema difícil sem uma solução definitiva ainda. Além da melhoria dos níveis de cobertura e precisão obtidos atualmente, são necessários ainda métodos para eficientemente *caracterizar* as transições graduais encontradas como *fades*, dissoluções ou outras formas de edição.

Além disso, no nível semântico, embora um *shot* seja uma unidade interessante de estruturação, ele não representa propriamente uma *cena*, como é observado por Chang e Sundaram [3] . Uma cena de um filme pode consistir de vários *shots*. Na construção automática de ambientes 3D a partir de vídeo, por exemplo, deveríamos conhecer todos os *shots* relacionados a um ambiente afim de ter toda a informação disponível para sua reconstrução.

Como visto no trabalho de Yeo e Liu, desempenho e eficiência podem ser obtidos em imagens de baixíssima resolução. Um estudo interessante poderia ser realizado com relação à aplicação de espaços escala, uma forma bem conhecida para análise de aspectos macroscópicos da informação visual.

A representação de *shots* para indexação é outro ponto interessante. Para tornar um vídeo "navegável" de forma não-seqüencial, é necessária uma representação do *shot* que passe ao usuário uma idéia da informação que está contida lá. A escolha de um ou mais *quadros chaves* tem sido outro tópico abordado quando se analisa o problema da segmentação temporal.

Assim, há um grande leque de possibilidades a serem exploradas. Esperamos focar em alguns destes tópicos para a elaboração da dissertação de mestrado a ser defendida no programa de pós-graduação desse departamento.

Referências

[1] F. Arman, A. Hsu, M.-Y. Chiu *Image Processing on Compressed Data for Large Video Databases*, Proceedings of First ACM International Conferenceon Multimedia (1993).

[2] D. H. Ballard e C. M. Brown, *Computer Vision*, Prentice Hall (1982).

[3] S.-F. Chang e H. Sundaram, *Structural and Semantic Analysis od Video*, IEEE International Conference on Multimedia (2000).

[4] A. M. Ferman and A. M. Tekalp, *Efficient Filtering and Clustering Methods for Temporal Video Segmentation and Visual Summarization*, Journal of Visual Communication and Image Representation (1998).

[5] U. Gargi, R. Kasturi, S. Antani, *Performance Characterization and Comparison of Video Indexing Algorithms*, Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR) (1995).

[6] R. C. Gonzalez e R. E. Woods, *Digital Image Processing*, Addison-Wesley (1992).

[7] I. Koprinska and S. Carrato, *Temporal Video Segmentation: A Survey*, Signal Processing: Image Comunication, Elsevier Science (2001).

[8] T. T. Santos e C. H. Morimoto, *Comparison of Temporal Video Segmentation Techniques*, LATIN/IME-USP e LIVES/CEI-SENAC (2002).

[9] B.-L. Yeo and B. Liu, *Rapid Scene Analysis on Compressed Video*, IEEE Transactions on Circuits and Systems for Video Technology (1995).

[10] H. Yu, G. Bozdagi e S. Harrington, *Feature-based Hierarchical Video Segmentation*, Proceedings of International Conference on Image Processing (1997).

[11] W. Xiong, J. C.-M. Lee, M.-C. Ip, *Net Comparison: A Fast and Effective Method for Classifying Image Sequences*, Proceedings of SPIE Conference on Storage and Retrieval for Imagem and Video Databases III (1995).

[12] W. Xiong and J. C.-M. Lee, *Efficient Scen Change Detection and Camera Motion Annotation for Video Classification*, Computer Vision and Image Understanding, ACM (1998).

[13] H. J. Zhang, A. Kankanhalli, S. W. Smoliar, *Automatic Partitioning of Full-motion Video*, Multimedia Systems (1993).

[14] H. J. Zhang, C. Y. Low, Y. H. Gong, S. W. Smoliar, *Video Parsing Using Compressed Data*, Proceedings of SPIE Conference on Image and Video Processing II (1994).

[15] The Open Video Project, `http://www.open-video.org/`.