Resumo

A popularização de aplicativos e dispositivos capazes de produzir, exibir e editar conteúdos multimídia fez surgir a necessidade de se adaptar, modificar e customizar diferentes tipos de mídia à diferentes necessidades do usuário. Nesse contexto, a área de Personalização e Adaptação de Conteúdo busca desenvolver soluções que atendam a tais necessidades.

Sistemas de personalização, em geral, necessitam conhecer os dados presentes na mídia, surgindo, assim, a necessidade de uma indexação do conteúdo presente na mídia. No caso de vídeo digital, os esforços para a indexação automática utilizam como passo inicial a segmentação de vídeos em unidades de informação menores, como tomadas e cenas. A segmentação em cenas, em especial, é um desafio para pesquisadores graças à enorme variedade entre os vídeos e a própria ausência de um consenso na definição de cena.

Diversas técnicas diferentes para a segmentação em tomadas e em cenas estão presentes na literatura. Uma forma particular de realizar a segmentação do vídeo em cenas é através da análise de coerência entre tomadas, onde busca-se unir tomadas semelhantes de maneira a formarem "cenas". Vantagens do uso da coerência de tomadas frente à outras técnicas incluí o baixo custo computacional e a independência de domínio.

Assim, este trabalho têm por objetivo apresentar uma técnica de segmentação de vídeo em tomadas e em cenas através da coerência entre tomadas. Para melhorar os resultados obtidos, utiliza-se uma análise de movimento baseada em fluxo óptico, capaz de significativamente reduzir o número de falsos positivos alcançado pela técnica. Descreve-se, ainda, detalhes de uma implementação multi-thread da técnica de segmentação de vídeo.

Sumário

Capítulo

1

Introdução

Com o advento e a popularização de uma grande gama de diferentes dispositivos capazes de processar e acessar dados multimídia, tais como computadores pessoais, celulares e *tablets*, houve uma mudança no paradigma de produção de conteúdos. Até então, conteúdos multimídia eram produzidos majoritariamente por especialistas. Atualmente, tais dispositivos contam com o auxílio de aplicativos e facilitam a produção e disponibilização de conteúdo por usuários leigos (como no YouTube[1] , por exemplo) resultando em um aumento expressivo no volume de informação disponível. Como consequência, ocorre o problema da "sobrecarga de informação", termo cunhado por Toffler (1984), caracterizado pela dificuldade de se localizar, de modo eficiente, o conteúdo que seja de interesse para o usuário. Uma área recente, no campo da Ciência da Computação, que tenta contribuir com soluções para esse problema é a Personalização e Adaptação de Conteúdo (P&A)(Lu et al., 2011; Manzato, 2011).

A adaptação de conteúdo tem como objetivo disponibilizar material multimídia adequado para cada tipo, condição, estado e conectividade dos mais diversos aparelhos. A personalização, caso particular da adaptação, procura encontrar métodos de customizar e/ou filtrar os dados segundo as preferências, necessidades e interesses de um usuário específico (Magalhães e Pereira, 2004). O enfoque dos pesquisadores, nos últimos anos, centra-se na personalização de conteúdo, desenvolvendo serviços categorizados em seleção de conteúdo, sistemas de recomendação e sistemas de sumarização (Adomavicius e Tuzhi-

[1] http://www.youtube.com

lin, 2005). Na seleção de conteúdo, o usuário define, interativamente, critérios de busca de itens multimídia. Na recomendação, itens de possível interesse do usuário são oferecidos automaticamente com base em seu histórico de uso ou em um perfil de preferências. Por fim, a sumarização tem como meta produzir índices daquilo que pode ser relevante ao usuário, oferecendo versões reduzidas do conteúdo, como um trecho significativo de um texto, por exemplo, para que possam ser selecionadas pelos usuários.

Os sistemas de personalização, em geral, apresentam uma necessidade em comum: o conhecimento dos dados contidos no conteúdo, chamados metadados. Tal necessidade surge na área de P&A para que o sistema possa se adequar às exigências dos usuários. Os metadados podem ser classificados, basicamente, como de baixo ou de alto nível semântico. Metadados de baixo nível semântico descrevem características inerentes à mídia em si, tais como histogramas, sistema de cor, tipo de compressão, entre outros. Metadados de alto nível, por sua vez, descrevem características conceituais do conteúdo, tais como identificação e/ou presença de pessoas, localidade, assunto, entre outros (Snoek et al., 2005).

A extração de metadados, chamada de Indexação Multimídia, pode ser realizada de modo automático ou manual (Brunelli et al., 1999), e aplicada a diversos tipos de mídia. No caso particular de vídeo digital, os vários esforços para realizar a indexação automática primeiro segmentam o vídeo em unidades menores de informação, mais gerenciáveis, como quadros e tomadas, para então aplicar a extração de metadados (Chaisorn et al., 2003; Chen e Li, 2010; Liu et al., 2009; Ogawa et al., 2008; Wang et al., 2008; Yu et al., 2007) e obter segmentações de maior nível semântico, como cenas.

A segmentação de vídeo em quadros é uma área já estabelecida, com a existência de técnicas maduras tanto para vídeos comprimidos como para vídeos sem compressão.

A área de segmentação de vídeo em tomadas ainda é uma área de pesquisa ativa, procurando superar desafios como a dependência de domínio e limitações quanto a efeitos de transição complexos. Segundo Hanjalic (2002), a segmentação do vídeo em tomadas é especialmente importante, pois segmentações com maior grau de abstração dependem dela.

A segmentação do vídeo em cenas é importante devido ao fato de cena ser um conceito subjetivo bem difundido entre as pessoas. Tal subjetividade impõe dificuldades, como a falta de uma definição única e formal, resultando em uma área pouco investigada e com grande variação entre resultados obtidos. Segmentações desse tipo potencializam a utilização de aplicações voltadas ao público em geral, seja em anotações, seja na criação de resumos de um vídeo ou na localização de cenas específicas em uma base de vídeos de larga escala, reduzindo a sobrecarga de informação.

Assim, este trabalho tem como objetivo apresentar uma técnica de segmentação de vídeo em tomadas e cenas que seja totalmente automático, não dependa da intervenção humana em nenhum momento, apto a segmentar uma ampla gama de vídeos, seja simples de ser implementado e, também, apresente um bom desempenho.

O restante deste trabalho é assim dividido: o segundo capítulo apresenta detalhes da técnica de segmentação de vídeo. Já o terceiro capítulo apresenta detalhes de uma implementação, em Java, da técnica proposta.

Por fim, no quarto e último capítulo, é apresentada uma série de considerações quanto a técnica descrita neste trabalho.

Capítulo

2

Descrição da Técnica

Neste capítulo, são apresentadas as principais características da técnica desenvolvida. Na **Seção 2.1**, descreve-se em alto nível as fases que compõe a técnica. Na **Seção 2.2** é apresentado a primeira fase da técnica, a segmentação em tomadas. Já na **Seção 2.3** descreve-se a segunda fase do algoritmo, a segmentação em cenas.

2.1 Descrição inicial da técnica

A técnica de segmentação de vídeo é dividida de maneira semelhante ao modelo proposto por Hu *et al.* (2011), que divide a segmentação em três fases:

- **Segmentação em tomadas:** Geralmente, a fase inicial da maioria das técnicas propostas para a segmentação em cenas. Incluí a obtenção de informações de diferentes níveis com o objetivo de unir quadros em tomadas.

- **Extração de quadros-chave e cálculo de características:** A fase começa logo após a segmentação em tomadas, onde são calculadas características diversas das tomadas e a extração de quadros-chave para simplificar a posterior segmentação em cenas.

- **Segmentação em cenas:** Geralmente a última fase na segmentação de vídeo. Incluí comparações das características calculadas anteriormente e o agrupamento

das tomadas em cenas. Resulta em um conjunto de tomadas agrupadas de maneira a formar uma cena.

O algoritmo proposto, entretanto, pode ser dividido em duas fases distintas e mais amplas que as propostas por Hu *et al.* (2011).

- Fase de segmentação em tomadas e obtenção de dados para a segmentação em cenas (Fase um).
- Fase de segmentação em cenas e procedimentos adicionais para refinar a segmentação (Fase dois).

Tal divisão foi adotada graças à saída esperada pelo usuário. Caso o usuário deseje a segmentação em tomadas, apenas a primeira fase do algoritmo poderia ser executada. A **Figura 2.1** apresenta uma breve descrição da técnica de segmentação de vídeo em tomadas e cenas.

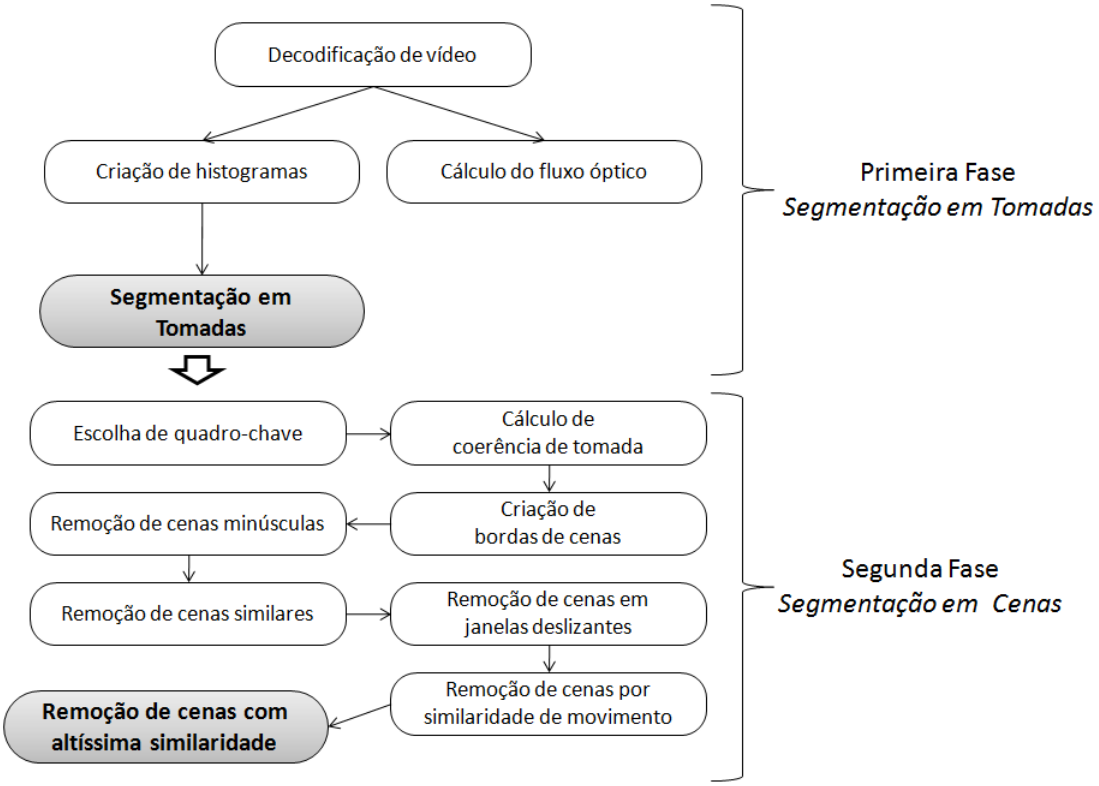


Figura 2.1: Diagrama descrevendo as duas fases da técnica desenvolvida, com ênfase no último procedimento de cada fase.

Os procedimentos adotados em cada uma das fases são descritos nas seções posteriores. A **Seção 2.2** detalha a primeira fase do algoritmo, a segmentação em tomadas. Já a **Seção 2.3** apresenta a segunda fase do algoritmo, a segmentação em cenas.

2.2 Fase um: Segmentação em tomadas

A primeira fase do algoritmo tem como objetivo transformar o fluxo de vídeo de entrada em um conjunto de tomadas, além de calcular variáveis necessárias (como o fluxo óptico) para a segunda fase do algoritmo.

A entrada de dados constitui o passo fundamental da primeira fase do algoritmo. Esse mecanismo de entrada de dados deve, idealmente, prover compatibilidade à diversos formatos de entrada e, preferencialmente, um desempenho satisfatório, sob pena de comprometer o desempenho de toda a técnica em si.

Com o quadro obtido, seja através da decodificação do fluxo de vídeo, seja através de quadros individuais já providos, entra em cena dois procedimentos: a extração dos histogramas de cada quadro e o cálculo do fluxo óptico. A extração dos histogramas é descrito na **Subseção 2.2.1** e o cálculo do fluxo óptico é descrito na **Subseção 2.2.2**. Após o final de tais procedimentos, a técnica realiza a segmentação em tomadas em si, detalhada na **Subseção 2.2.3**

2.2.1 Criação de histogramas do vídeo

Segundo Marques (2011), o histograma de uma imagem é uma representação gráfica da frequência de cada nível de cor em uma imagem. A importância dos histogramas reside no fato de ser uma representação de tamanho reduzido de uma imagem, tornando o processamento mais rápido.

Em imagens coloridas, o número de possíveis níveis de cores é muito alto, atingindo cerca de 17 milhões de combinações possíveis entre os 256 níveis de cor no sistema RGB. Assim, para reduzir tal volume de dados, o vídeo colorido sofre o processo de quantização (Marques, 2011): cores próximas são agrupadas em um conjunto e, dentro desse conjunto, todas as cores são igualadas. Ao número de grupos da quantização costuma-se dar o nome de *bins* (caixas).

Assim, um vídeo no espaço de cor RGB com 12 *bins* na proporção 4:4:4 (4 para vermelho, 4 para verde e 4 para azul) seria representado em apenas 64 valores contra quase 17 milhões de valores sem o processo de quantização. Esse valor é obtido através da combinação de todos os quatro valores de vermelho, quatro para verde e quatro para azul, formando então 64 combinações entre elas. Esse processo apresenta vantagens tanto na redução do tamanho do histograma como em velocidade de processamento de operações sobre o histograma resultante. Como desvantagem, cita-se que a divisão de um histograma em *bins* resulta na perda irreversível de informações tais como pequenos detalhes da imagem e degradês.

De mesma forma, na técnica descrita neste trabalho, converte-se o quadro do espaço de cor de entrada (por exemplo, RGB) para o espaço de cor HSV. Utilizou-se o padrão HSV com 180 níveis de cor para o *Hue*, 256 níveis para o *Saturation* e 256 níveis de *Value*. O histograma gerado utiliza-se da proporção de 8 *bins* para o *Hue*, 4 *bins* para o *Saturation* e 4 *bins* para o *Value* para redução do espaço requerido do histograma. Após calculado, o histograma é ainda normalizado ao valor um, ou seja, a soma de todos os níveis do histograma é um. Tal normalização é necessária para a adotação de limiares capazes de representar histogramas de quadros de diferentes resoluções.

2.2.2 Cálculo do fluxo óptico do vídeo

O fluxo óptico é uma reta, com ponto inicial e final, que descreve a movimentação de um determinado *pixel* entre quadros do vídeo. Assim, para detectar a movimentação de um *pixel*, compara-se quadros adjacentes em busca de tais deslocamentos.

O fluxo óptico é calculado utilizando-se dois quadros consecutivos. No primeiro quadro (ou quadro anterior), é realizado um procedimento de escolha de um determinado número de pontos de interesse, como por exemplo bordas de imagens, texturas, entre outros. Esses pontos de interesses são então "casados" com os mesmos pontos no segundo quadro (ou quadro posterior). Esse casamento ocorre através de uma série de variáveis como a janela de procura e o índice de similaridade mínimo. Ao final desse procedimento, um conjunto de pontos inicial e final do movimento detectado é retornado. O valor utilizado para algumas das mais importantes variáveis são descritos abaixo:

- **Máximo de pontos de interesse:** 1000 pontos.
- **Número máximo de iterações:** 20 iterações.
- **Tamanho da janela de busca:** janela de busca de tamanho 15x15.

Com isso, a técnica extraí os histogramas de todos os quadros do vídeo de entrada e também o seu fluxo óptico, armazenados adequadamente para a análise posterior. Após esses procedimentos, entra em prática a segmentação em tomadas, baseada nos histogramas de cada quadro, descrita na **Subseção 2.2.3**.

2.2.3 Segmentação do vídeo em tomadas

A segmentação do vídeo em tomadas é um procedimento que utiliza, basicamente, duas operações de comparação entre histogramas: a intersecção de histogramas e a diferença absoluta de histogramas. O procedimento de segmentação em tomadas pode ser descrito como um método de janelas deslizantes de tamanho variável, duplas (cada quadro faz

parte de duas janelas), independentes (cada janela deslizante independe da outra) e com limiar independente e adaptável (ou seja, há um limiar único para cada janela, calculado sobre os dados que a compõe). A **Figura 3.3** apresenta um exemplo da divisão de um determinado número de quadros em janelas deslizantes com valores de limiares independentes calculados.

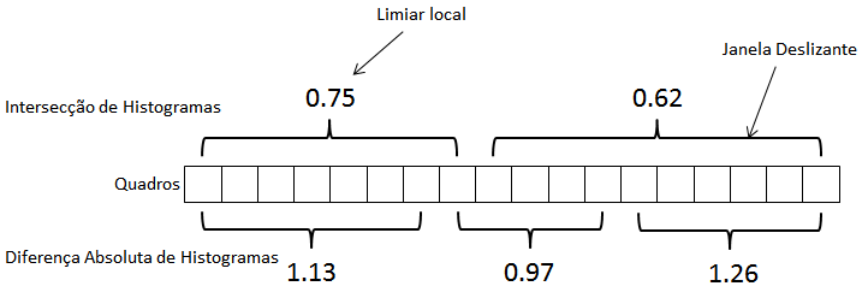


Figura 2.2: Exemplo da divisão dos quadros em duas janelas deslizantes independentes com limiares independentes para cada janela deslizante.

A intersecção de histogramas é representado por um valor no intervalo $[0, 1]$, onde zero significa nenhuma similaridade e 1 significa similaridade total (ou seja, histogramas iguais).

A diferença absoluta consiste em comparar cada nível de cor correspondente e encontrar a diferença, em módulo, de cada nível, somando as diferenças de cada nível para formar um valor único. Assim, a diferença absoluta de histogramas normalizados resulta em um valor do intervalo $[0, 2]$, onde zero significa "histogramas iguais" e 2.0 significa histogramas "completamente diferentes".

Com os valores de intersecção e diferença absoluta de histogramas de quadros dois-a-dois cria-se, então, duas janelas deslizantes, uma para a intersecção e outra para a diferença absoluta. A janela da intersecção é criada calculando-se o menor valor de intersecção entre todos os valores calculados. Após encontrado tal valor, uma borda da janela deslizante é detectada:

- Quando o valor de intersecção de histogramas entre dois quadros adjacentes for menor que 0.25.
- Quando o valor de intersecção for menor que o menor valor encontrado acrescido de 50% e também for menor que 0.4.

Para a diferença absoluta de histogramas calcula-se o maior valor de diferença encontrado nos histogramas de quadros adjacentes. As bordas da janela deslizante da diferença de histogramas são detectadas:

- Quando o valor de diferença absoluta de histogramas for maior que 1.5.

- Quando o valor da diferença absoluta de histogramas for maior que 85% do valor máximo encontrado e maior ou igual a 0.9.

As bordas de janelas são projetadas para encontrar transições "fáceis", ou seja, que possuam valores de histogramas bastante diferentes. Deve-se notar que:

1. As janelas deslizantes são independentes entre si, embora possam apresentar diversas bordas compartilhadas.

2. Todas as bordas de janelas são consideradas, desde já, como transições de tomada.

Com isso, a detecção de tomadas simples e evidentes é resolvida de início, de forma a não influenciar na detecção das outras transições de tomadas, mais complexas que as demais. Para essas transições, vasculha-se cada uma das janelas deslizantes e realiza-se uma série de análises. O limiar para cada janela deslizante é calculada de forma diferenciada.

O limiar de cada janela deslizante da intersecção de histogramas corresponde a 50% da média dos valores de intersecção entre todos os quadros da janela. Já o limiar para a diferença absoluta de histogramas é formado por 9 vezes a média dos valores encontrados na diferença absoluta de histogramas.

Com os valores calculados, realiza-se a detecção das bordas da transição em si. Para cada quadro, vasculha-se a posição e as janelas correspondentes àquele quadro. Caso o valor de diferença absoluta extrapole o máximo permitido do valor calculado para aquela janela de diferença ou seja inferior ao limiar calculado para a janela de intersecção de histogramas, uma transição de tomada é encontrada.

Para corrigir problemas de detecção em transições graduais, adota-se a seguinte heurística: caso exista uma outra transição em uma janela de N quadros anteriores, os quadros que compõe a última transição até a transição "atual" são consideradas como uma mesma transição gradual. Por padrão, adotou-se o valor utilizado de N foi 3.

Finalmente, a primeira fase do algoritmo se encerra, obtendo-se uma série de transições de tomadas abruptas e graduais.

2.3 Fase dois: Segmentação em cenas

A segunda fase do algoritmo exige a presença de um conjunto de tomadas e de um conjunto de vetores de fluxo óptico para operar. O objetivo desta fase do algoritmo é a de agrupar, progressivamente, as tomadas em segmentos maiores e com maior grau semântico, as cenas.

Primeiro, encontra-se um conjunto de quadro-chaves para cada tomada (processo descrito na **Subseção 2.3.1**). Após, calcula-se a coerência de tomadas baseado nos

quadros-chaves encontrados (processo descrito na **Subseção 2.3.2**), após ter a coerência calculada, entra em cena a análise do valor de coerência com o objetivo de se obter uma segmentação inicial do vídeo em cenas (procedimento descrito na **Subseção 2.3.3**).

Depois de a segmentação inicial de cenas ter sido obtida, uma série de procedimentos para reduzir o número de transições incorretas são realizados. Inicialmente, utiliza-se os histogramas para uma análise de similaridade em procedimentos descritos nas **Subseções 2.3.4**, **2.3.5** e **2.3.6**. Após, utiliza-se o fluxo óptico calculado para a remoção de algumas cenas com dinâmica de movimento similar, procedimento apresentado na **Subseção 2.3.7** e, por fim, uma última análise de similaridade de histogramas apresentado na **Subseção 2.3.8**.

2.3.1 Seleção de quadros-chaves

O primeiro procedimento realizado na segunda fase do algoritmo é a seleção de um conjunto de quadros-chaves para cada tomada do vídeo. O uso de quadros-chaves apresenta como vantagem a redução expressiva no processamento em fases posteriores, já que um grande número de quadros (por exemplo, uma tomada inteira) pode ser representado adequadamente por apenas um quadro.

Diversas técnicas são propostas na literatura para a seleção do quadro-chave. Duas das abordagens mais comuns são a seleção do quadro mediano ou do primeiro quadro de cada tomada. Tais métodos, porém, podem apresentar problemas como a falta de representatividade de tomadas complexas, ou ainda, em casos de erro na segmentação de tomadas, onde múltiplas tomadas são detectadas como apenas uma tomada.

Assim, desenvolveu-se uma técnica de escolha de quadros-chaves que tenta representar tanto tomadas simples, complexas e casos onde a separação entre a transição da tomada não foi corretamente detectada. Para isso, ao invés de selecionar apenas um quadro-chave, procura-se um conjunto de quadros-chaves para representar a tomada.

O procedimento de seleção adotado para a seleção de quadros-chaves é o seguinte:

1. Calcula-se o grau de similaridade entre todos os quadros da tomada.
2. Seleciona-se o quadro que apresenta maior similaridade com os demais quadros.
3. Adiciona-se o quadro selecionado ao conjunto de quadros-chaves, se nenhum quadro já presente no conjunto de quadros-chaves for similar ao quadro-chave candidato.
4. Repete-se os passos 2 ao 4, até que o quadro candidato tenha baixa representatividade ou não exista quadro candidato.

O grau de similaridade é obtido através da intersecção de histogramas de todos os histogramas de cada tomada. Dois quadros são considerados “similares” caso a intersecção

entre seus dois histogramas for igual ou maior que 95%. De mesma forma, dois quadros são ditos dissimilares caso a similaridade for inferior à 95%.

Por padrão, adota-se 20% como representatividade mínima para que um quadro possa ser considerado como quadro candidato. Ou seja, um determinado quadro deve ser 95% "similar" a pelo menos 20% do número de quadros presente na tomada. Assim, essa técnica permite a seleção de diversos quadros-chaves (ou apenas um quadro-chave) que são suficientemente dissimilares entre si e que, ao mesmo tempo, são representativos aos demais quadros da tomada.

2.3.2 Cálculo da coerência de tomadas

A coerência de tomadas é uma medida numérica que procura medir o quão similar uma determinada tomada é frente a determinado conjunto de tomadas anteriores. Neste caso, adotou-se uma versão modificada da técnica Backward Shot Coherence (BSC) (Rasheed e Shah, 2003), cuja principal característica adicional é a introdução de uma medida de incremento de valor conforme a tomada comparada se aproxime da tomada em análise.

Na técnica BSC, a medida de coerência entre duas tomadas quaisquer é dada pela **Equação 2.1**.

$$SC_j^i = \max_{f^x \in K_i, f^y \in K_j} \left(D\left(f^x, f^y\right) \right) \tag{2.1}$$

Onde f^x é um histograma do quadro-chave do conjunto de quadros-chaves K_i da tomada i, f^y é um histograma do quadro-chave do conjunto de quadros-chaves K_j da tomada j e $D\left(f^x, f^y\right)$ é a medida de comparação entre os histogramas f^x e f^y, no caso, a intersecção de histogramas. Assim, em outras palavras, o valor de coerência entre duas tomadas quaisquer é o maior valor da intersecção de histogramas entre os quadros-chaves das duas tomadas.

O valor BSC de cada tomada do vídeo é obtido, assim, através da análise da coerência de tomadas entre uma tomada "base" e N tomadas anteriores. A **Equação 2.2** apresenta o cálculo do BSC originalmente proposto por Rasheed e Shah (2003).

$$BSC_i = \max_{1 \leq k \leq N} \left(SC_i^{i-k} \right) \tag{2.2}$$

A técnica desenvolvida, porém, adota uma pequena modificação ao cálculo original do valor BSC. Como especificado originalmente, o valor BSC não distingue a maior importância de tomadas mais "próximas" daquelas mais "distantes" da tomada em análise, podendo inclusive determinar o valor BSC baseado em uma única tomada no limite do valor N determinado, causando falhas em casos de mudança progressiva de contexto ou de cenários em si.

Para isso, na técnica desenvolvida, calcula-se o valor BSCW (BSC *Weighted*), com a introdução de uma medida chamada de Temporal Memory (TM). A **Equação 2.3** descreve o cálculo do valor BSCW.

$$BSCW_i = \max_{1 \leq k \leq N} \left(SC_i^{i-k} \cdot TM_{N-k} \right) \tag{2.3}$$

O valor TM é calculado conforme a **Equação 2.4**.

$$TM_k = 1.0 + (0.05 \cdot k) \tag{2.4}$$

Com o valor TM, o valor SC aumenta progressivamente 5% conforme a tomada em análise aproxime-se cronologicamente da tomada "base". Para evitar problemas nos procedimentos posteriores, os valores de BSCW maiores que 1.0 são convertidos para o valor 1.0 (ou seja, 100% de coerência).

2.3.3 Criação de bordas de cenas

Assim que o valor BSCW é calculado para todas as tomadas, é executado o procedimento em que analisa-se os valores BSCW para encontrar as bordas da cena. Rasheed e Shah (2003), em seu trabalho, analisam o valor de BSC em busca de vales: caso um seja detectado, uma nova borda de cena é detectada. Na técnica BSC e também na técnica desenvolvida, essa borda de cena é chamada de Potential Scene Boundarie (PSB).

Na técnica aqui descrita, porém, PSBs são detectados quando o valor de BSCW apresenta alguma das seguintes condições:

- O valor BSCW apresenta uma redução de 15% ou mais em comparação com o BSCW da tomada anterior.

- O valor BSCW apresenta uma redução de 5% ou mais o próximo valor também apresenta uma reduação de 5% ou mais.

Ao fim deste procedimento, tem-se uma primeira versão da segmentação do vídeo em cenas, cujo resultado apresenta um bom índice de abrangência (*recall*), mas também com um elevado índice de falsos positivos (causando baixa precisão). Assim, para reduzir o número de falsos positivos da segmentação em cenas, utiliza-se uma série de procedimentos e heurísticas atuando em série para que as cenas detectadas até então sejam melhor avaliadas, de forma a serem unidas, para que o número de falsos positivos diminua.

2.3.4 Remoção de cenas minúsculas

O primeiro procedimento para reduzir a ocorrência de falsos positivos na segmentação em cenas é uma tentativa de eliminar cenas minúsculas. Cenas desse tipo podem ocorrer no procedimento descrito na **Subseção 2.3.3** quando uma determinada tomada apresenta baixíssima coerência com todas as N tomadas anteriores, mas que a tomada seguinte não apresente tal comportamento. Nesse caso, essa redução brusca no valor BSCW ocasionará uma cena contendo apenas uma tomada. A **Figura 2.3** apresenta um exemplo de uma cena minúscula.

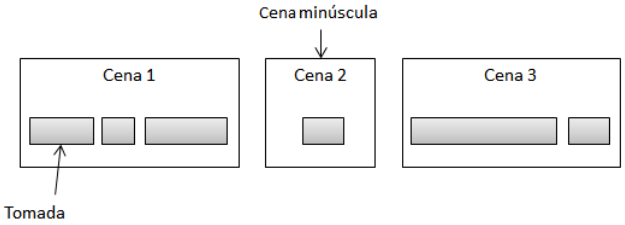


Figura 2.3: Exemplo de uma cena minúscula contendo apenas uma tomada.

As cenas minúsculas, nesse caso, são eliminadas determinando-se qual a cena adjacente (a anterior ou a posterior) que mais se assemelha à cena minúscula em análise. Após determinada a semelhança entre as cenas, a cena minúscula será fundida com a cena com a qual apresentar a maior semelhança.

A análise de semelhança entre cenas utiliza-se de três medidas: $maxC$, $medC$ e $minC$. A $maxC$ é o maior valor da intersecção de histogramas entre os quadros-chaves de duas cenas quaisquer. O valor $medC$ é a média do valor da intersecção de histogramas entre os quadros-chaves de duas cenas quaisquer e, finalmente, $minC$ é o menor valor da intersecção de histogramas entre os quadros-chaves de duas cenas quaisquer. O conjunto de quadros-chaves de cada cena é formado pela união de todos os quadros-chaves de todas as tomadas que formam aquela cena específica.

A premissa básica do algoritmo adotado diz respeito à quando uma cena minúscula não deve ser fundida. Caso o maior valor encontrado de $medC$ entre a cena minúscula e as cenas anterior e posterior for menor que 0.2 (ou 20% de similaridade média) ou se a cena minúscula apresentar 150 quadros ou mais de duração (ou seja, a única tomada da cena minúscula for de grande duração), então a cena minúscula não deve ser fundida a nenhuma outra cena.

As fusões com a cena minúscula podem ocorrer com a cena anterior ou posterior. Considere como C_a a cena anterior a cena minúscula e C_p a cena posterior a cena minúscula e C a cena minúscula. A **Tabela 2.1** apresenta os casos em que a cena minúscula será fundida e com qual cena adjacente ocorrerá tal fusão.

Tabela 2.1: Casos onde uma cena minúscula será fundida com a cena anterior ou com a cena posterior.

Fusão com a anterior	Fusão com a posterior
Se $maxC(C_a, C) >= maxC(C, C_p)$ e $medC(C_a, C) >= medC(C, C_p)$	Se $maxC(C, C_p) >= maxC(C_a, C)$ e $medC(C, C_p) >= medC(C_a, C)$
Se $maxC(C, C_p) >= maxC(C_a, C) \cdot 1.2$	Se $maxC(C_a, C) >= maxC(C, C_p) \cdot 1.2$
Se $minC(C_a, C) >= minC(C, C_p) \cdot 1.1$	Se $minC(C, C_p) >= minC(C_a, C) \cdot 1.1$

Para os casos limites (a cena minúscula é a primeira ou a última cena do vídeo), adota-se a seguinte estratégia:

- Funde-se a cena inicial com a posterior caso $minC(C, C_p) >= 0.2$ e a tomada da cena minúscula tiver menos que 150 quadros de duração.
- Funde-se a cena final com a anterior caso $minC(C_a, C) >= 0.2$ e a tomada da cena minúscula tiver menos que 150 quadros de duração.

Com isso, diversos falsos positivos são eliminados do procedimento anterior, com a introdução de poucos (ou, no melhor dos casos, nenhum) falsos negativos. Destaca-se que nem todas as cenas com apenas uma tomada são eliminadas. Algumas cenas com a tomada longa ou cenas minúsculas com alta dissimilaridade com as cenas adjacentes são mantidas intactas nesse procedimento.

2.3.5 Remoção de cenas similares adjacentes

O segundo procedimento adotado para a redução de falsos positivos trata-se de uma análise entre os histogramas dos quadros-chaves de cenas adjacentes. Ao contrário do procedimento descrito na **Subseção 2.3.4**, não se considera o tamanho da cena envolvida na análise.

Neste procedimento, compara-se os histogramas de cenas duas-a-duas adjacentes de maneira a se obter o valor *maxC* entre os quadros-chaves das duas cenas. Caso algum par de quadros-chaves apresentar o valor de intersecção maior ou igual a um determinado limiar, as duas cenas são então consideradas aptas a serem fundidas.

Para evitar a fusão de cenas muito dissimilares mas que possuam pelo menos um par de quadros-chaves similares, analisa-se também o valor *minC* entre as duas cenas. Caso o valor de *minC* seja maior que o limiar pré-definido, a cena é fundida. Assim, exige-se que 1) as cena possuam pelo menos um par de valores com similaridade de 60% (0.6) e valor *minC* de 0.2 (20% de similaridade mínima) ou 2) o valor *maxC* entre as duas cenas seja maior que 0.8 (80% de similaridade máxima).

A **Figura 2.4** exemplifica um caso de fusão entre duas cenas, com o auxílio dos valores *maxC* e *minC*.

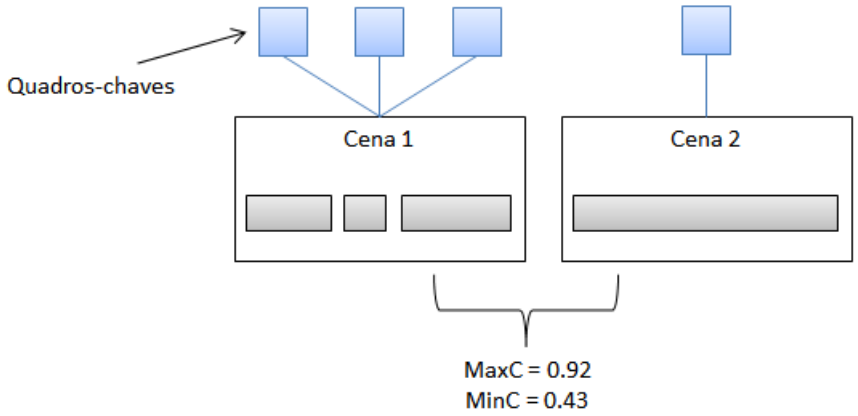


Figura 2.4: Exemplo de duas cenas adjacentes que serão fundidas devido a análise dos valores $maxC$ e $minC$.

2.3.6 Remoção de cenas em janelas deslizantes

O terceiro procedimento para redução do número de falsos positivos trata-se da análise em janelas deslizantes de similaridades para a fusão de cenas similares. Ao contrário dos dois primeiros procedimentos com o mesmo propósito, a tentativa de remover cenas similares em janelas procura, se possível, unir mais que duas janelas ao mesmo tempo, reduzindo drasticamente o número de falsos positivos do conjunto inicial de cenas. Na técnica, utilizou-se janelas deslizantes de tamanho três. A **Figura 2.5** exemplifica a união de três cenas em uma só através do procedimento de remoção de cenas em janelas deslizantes.

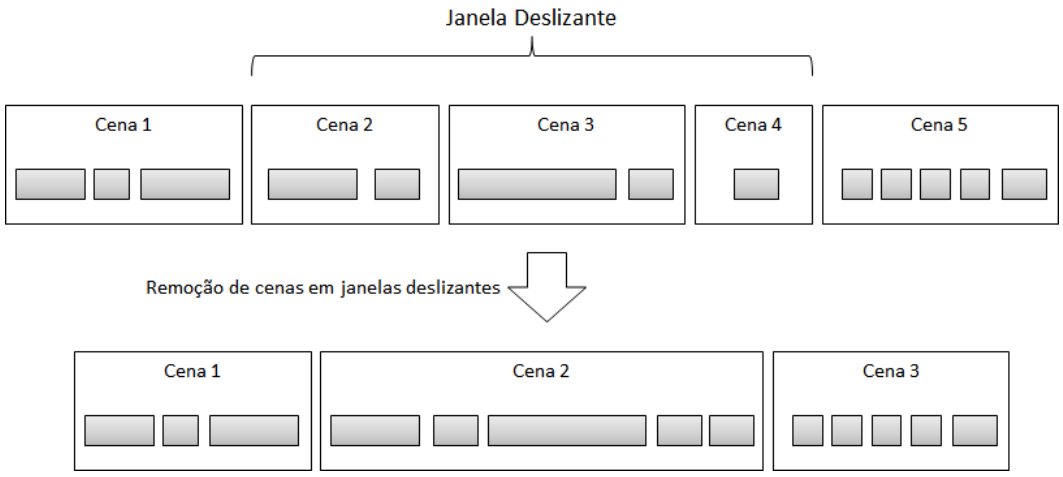


Figura 2.5: Exemplo de fusão entre três cenas adjacentes através do procedimento de remoção de cenas em janelas deslizantes.

Considere três cenas C_1, C_2 e C_3. Calcula-se os valores $maxC(C_1, C_2)$, $maxC(C_2, C_3)$, $maxC(C_1, C_3)$, $medC(C_1, C_2)$, $medC(C_2, C_3)$, $medC(C_1, C_3)$, $minC(C_1, C_2)$, $minC(C_2, C_3)$

e $minC(C_1, C_3)$ (ou seja, todas as combinações de $maxC$, $medC$ e $minC$ entre as três cenas) para as cenas C_1, C_2 e C_3. Com os valores calculados, unem-se as cenas:

- C_1**,** C_2 **e** C_3 - Caso apresentem $minC(C_1, C_3) >= 0.5$ e $maxC(C_1, C_3) >= 0.8$. Caso $maxC$ não seja maior que 0.8, se $medC(C_1, C_3) >= 0.3$, então a cena é fundida.
- C_1 **e** C_2 - Caso apresentem $minC(C_1, C_2) >= 0.25$ e $maxC(C_1, C_2) >= 0.8$. Caso $maxC$ não seja maior que 0.8, se $medC(C_1, C_2) >= 0.3$, então a cena é fundida.
- C_2 **e** C_3 - Caso apresentem $minC(C_2, C_3) >= 0.25$ e $maxC(C_2, C_3) >= 0.8$. Caso $maxC$ não seja maior que 0.8, se $medC(C_2, C_3) >= 0.3$, então a cena é fundida.

Nota-se que uma cena que já é resultado de uma fusão não pode ser fundida novamente. Tal condição garante que não haverá múltiplas fusões, que poderia causar um aumento demasiado no número de falsos negativos.

O procedimento primeiro procura fundir toda a janela em apenas uma única cena. Nesse caso, o valor mínimo requerido para o $minC$ é 0.5 (ou seja, para quaisquer dois quadros-chaves das cenas consideradas, a intersecção de histogramas será maior ou igual à 0.5) entre a C_1 e a cenas C_3. O valor 0.5 é o dobro do valor mínimo requerido nos demais casos, já que a união de três cenas em uma só requer um grau de confiança mais elevado que uma união entre apenas duas cenas adjacentes. Na prática, ocorrem poucas fusões de três cenas, haja visto o grande requerimento para a similaridade mínima.

2.3.7 Remoção de cenas por similaridade de movimento

O quarto método de remoção de falsos positivos é consideravelmente diferente dos outros métodos apresentados pois sua principal medida não é obtida através da análise de histogramas.

Em cenas muito movimentadas (como cenas de ação), o pequeno tamanho da cena e sua alta complexidade a tornam difíceis de serem corretamente segmentadas baseando-se apenas em histogramas (características de cor). Nesses casos, são reconhecidos um grande conjunto de cenas (normalmente englobando poucas tomadas) disjuntas, mas que representam a mesma "ação".

Para solucionar tal problema, utilizou-se uma análise de movimento baseada na quantidade de pontos significativos do fluxo óptico entre quadros-adjacentes.

Nesse procedimento, cada quadro possuí um determinado valor de fluxo óptico. O valor indica quantos *pixels* moveram-se do quadro anterior ao quadro atual, limitado tanto pela janela de busca como pelo tamanho do bloco pesquisado. Na fase de obtenção do fluxo óptico, vetores de movimento com distância entre os pontos menores ou iguais a um foram descartados, para que apenas pontos significativos sejam considerados.

Assim, para cada tomada, calcula-se a quantidade de pontos obtidos na tomada como a soma dos pontos encontrados em cada um dos quadros que a compõe (descrito na **Equação 2.5**). No caso de cena, o valor de fluxo óptico utilizado é a média dos valores encontrados nas tomadas que compõe a cena, descrito na **Equação 2.6**

$$FluxoTomada_i = \sum_{a=0}^{N} FluxoQuadro_a \tag{2.5}$$

$$FluxoCena_i = \frac{\sum_{a=0}^{N} FluxoTomada_a}{N} \tag{2.6}$$

Onde N é o número de quadros que compõe a tomada na **Equação 2.5** e é o número de tomadas que compõe a cena na **Equação 2.6**.

Com o valor $FluxoCena$ de cenas adjacentes, realiza-se uma série de comparações em janelas deslizantes de tamanho três. Assim, dadas três cenas C_1, C_2 e C_3:

- Unem-se as cenas C_1, C_2 e C_3 caso a diferença entre o valor de $FluxoCena$ entre C_1 e C_3 seja menor ou igual a 25%. Além disso, o valor de $FluxoCena$ entre as cenas C_1 e C_2 ou o valor entre as cenas C_2 e C_3 deve ser inferior ou igual à 25%.

- Unem-se as cenas C_2 e C_3 caso a diferença entre o valor de $FluxoCena$ entre C_2 e C_3 seja menor ou igual a 25%.

- Unem-se as cenas C_1, C_2 caso a diferença entre o valor de $FluxoCena$ entre C_1 e C_2 seja menor ou igual a 25%.

Assim como no terceiro procedimento (apresentado na **Subseção 2.3.6**), cenas que se formaram através da fusão por movimento não são fundidas novamente.

Com isso, o procedimento reconhece cenas adjacentes com mesma dinâmica de movimento como apenas uma cena, mas sem necessariamente eliminar cenas com alta movimentação. Para isso, utiliza-se outro passo de fusão que, agora, procura encontrar cenas adjacentes que possuam valores de fluxo óptico acima de determinado limiar.

Na técnica, dada as informações já apresentadas na **Seção 2.2.2**, utiliza-se o valor de limiar determinado pela **Equação 2.7**.

$$FluxoOptico_{limiar} = \frac{Video_{largura} \cdot Video_{altura}}{TamanhoJanelaFluxoOptico^3} \tag{2.7}$$

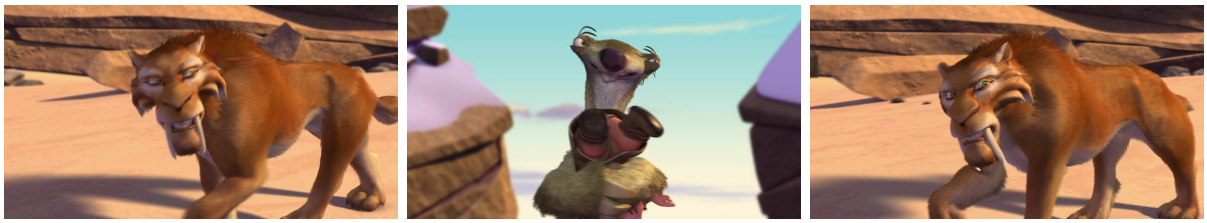
Onde $Video_{largura}$ e $Video_{altura}$ é a largura e altura (em *pixels*), respectivamente, do fluxo de vídeo de entrada e $TamanhoJanelaFluxoOptico$ é o tamanho da janela de procura especificado na criação dos vetores do fluxo óptico.

Com isso, as cenas resultantes possuem uma dinâmica de movimento consideravelmente dissimilares, sendo que cenas consecutivas com dinâmica de movimento similar ou alta movimentação foram unidas em apenas uma cena.

2.3.8 Remoção de cenas adjacentes com altíssima similaridade

O último procedimento introduzido com o objetivo de reduzir o número de falsos positivos é baseada na constatação de que, depois dos procedimentos descritos acima, a segmentação de cenas tende a separar cenas com quadros-chaves dissimilares, porém, que possuam uma altíssima similaridade com a cena posterior.

Um exemplo de tal erro de segmentação pode ser vista na **Figura 2.6**. Nela, nota-se que uma única tomada (Figura 2.6(b)) muito dissimilar às tomadas adjacentes (Figuras 2.6(a) e 2.6(c)) resulta em uma separação em três cenas, quando todas as mesmas pertencem à mesma cena.



(a) Quadro-chave da Cena 1 (b) Quadro-chave da Cena 2 (c) Quadro-chave da Cena 3

Figura 2.6: Exemplo de detecção de erro de detecção na transição de cena.

Assim, é necessária a introdução de mais um procedimento para tentar evitar a ocorrência de tais casos. Para isso, adota-se uma abordagem similar às adotadas previamente: analisa-se os histogramas dos quadros-chaves em busca de alta similaridade. Ao contrário dos outros métodos, neste procedimento procura-se cenas pequenas, com até cinco tomadas, e a comparação não é realizada com a cena mediana, apenas com as cenas no extremo de uma janela deslizante de três cenas.

O processo, embora se assemelhe ao processo de remoção em janelas deslizantes (descrito na **Subseção 2.3.6**), é mais rigoroso em relação à intersecção média dos histogramas dos quadros-chave.

Sejam três cenas C_1, C_2 e C_3, as três cenas serão transformadas em apenas uma única cena caso:

- C_1 e C_3 apresentem valor $maxC(C_1, C_3) >= 0.95$, ou seja, houver pelo menos um par de quadros-chaves com 95% de similaridade.

- C_1 e C_3 apresentem valor $medC(C_1, C_3) >= 0.5$, ou seja, a média da intersecção dos histogramas dos quadros-chaves tiver similaridade maior ou igual a 50%.

- C_1 e C_3 apresentem valor $minC(C_1, C_3) >= 0.25$, ou seja, o para quaisquer dois pares de histogramas dos quadros-chaves, a intersecção entre as mesmas deve ser maior ou igual a 25% de similaridade.

Com isso, cenas maiores intercaladas por cenas pequenas com histogramas altamente dissimilares são unidas, formando uma única cena com maior semântica.

Na prática, poucas cenas são fundidas nesse procedimento, haja visto que poucas cenas apresentam tal tipo de problema que, em geral, a análise de movimentação consegue identificar corretamente. Ainda assim, mesmo quando o método é executado, devido ao alto grau de confiança, o procedimento gera resultados com reduzido valor de falsos positivos, sem gerar falsos negativos por conta disso. Ao final deste precedimento, a técnica retorna um conjunto de transições de cenas, finalizando seu processamento.

Capítulo

3

Descrição da Implementação

Neste capítulo, são apresentados os principais detalhes quanto à implementação da técnica descrita no segundo capítulo. A **Seção 3.1** apresenta a interface de entrada e saída da implementação. A **Seção 3.2** apresenta, por sua vez, os detalhes específicos da implementação da técnica, tais com a divisão em duas fases e os métodos utilizados para cada procedimento.

3.1 Interface

A primeira experiência do usuário ao se utilizar de uma nova ferramenta é a facilidade e eficiência de utilização da interface da ferramenta. Uma interface exageradamente complexa pode, por exemplo, afastar o usuário leigo, haja visto a alta curva de aprendizagem envolvida na utilização da nova ferramenta.

A interface gráfica desenvolvida para a técnica tem por objetivo torná-la utilizável por usuários leigos sem, contudo, afetar a eficácia da técnica em si. Tal característica é obtida através da remoção das opções avançadas da interface básica do usuário. Dessa forma, um usuário leigo é capaz de interagir com a técnica utilizando-se apenas as opções mais básicas e fundamentais. Um usuário especialista, por sua vez, pode acessar as opções avançadas da técnica, contendo diversas opções extras.

Ao iniciar a aplicação, o usuário é apresentado à janela inicial da aplicação, onde deve selecionar o vídeo que deseja que seja segmentado, seguido das saídas que deseja: a

segmentação em cenas e/ou em tomadas. Por padrão, ambas as saídas estão selecionadas, significando que tanto a segmentação em tomadas como a segmentação em cenas produzirá algum resultado. A **Figura 3.1** apresenta a janela inicial.

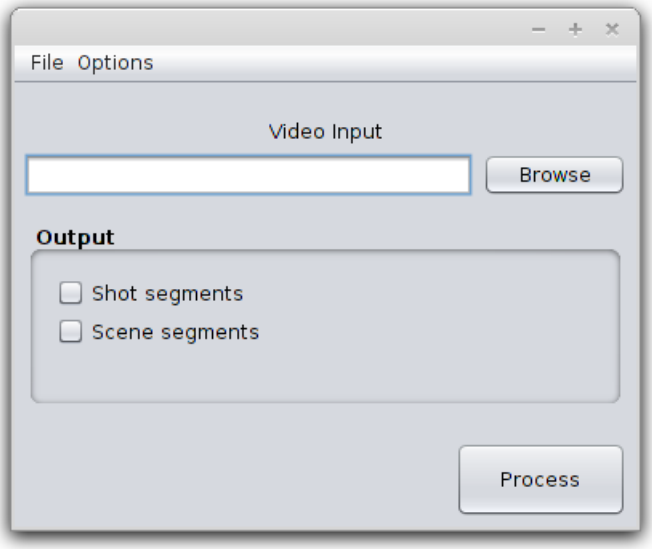


Figura 3.1: Tela inicial da ferramenta de segmentação de vídeo.

Com isso, o usuário leigo é capaz de utilizar a aplicação de forma rápida e sem necessitar de conhecimentos sobre a segmentação. Para o usuário mais avançado, a tela de opções apresenta uma série de parâmetros específicos que podem ser modificados de forma a satisfazer determinadas necessidades. Por exemplo, caso o usuário queira que a aplicação divida os núcleos de processamento em uma determinada proporção ou caso deseje modificar parâmetros utilizados pela técnica para realizar a segmentação. A **Figura 3.2** apresenta a janela de opções avançadas.

Quanto à saída, o usuário recebe como resultado da segmentação de tomadas e de cenas um arquivo de registro no formato XML contendo as transições de tomadas/cenas do vídeo de entrada. Tal arquivo, baseia-se em modelo proposto pela TRECVid[1], apresentando o número do quadro inicial e final da transição, além do tipo de transição.

Originalmente proposto para a segmentação de tomadas, o modelo utilizado trata-se de uma versão simplificada para abranger as transições de tomadas graduais ("GRA") ou abruptas ("CUT"). Embora não tenha sido especificado para a segmentação em cenas, tal modelo é assim utilizado modificando-se o tipo de transição para "SCE" (*Scene*). A **Figura 3.3** apresenta um exemplo de segmentação em tomadas do vídeo "video.mp4".

[1]http://goo.gl/cGc51

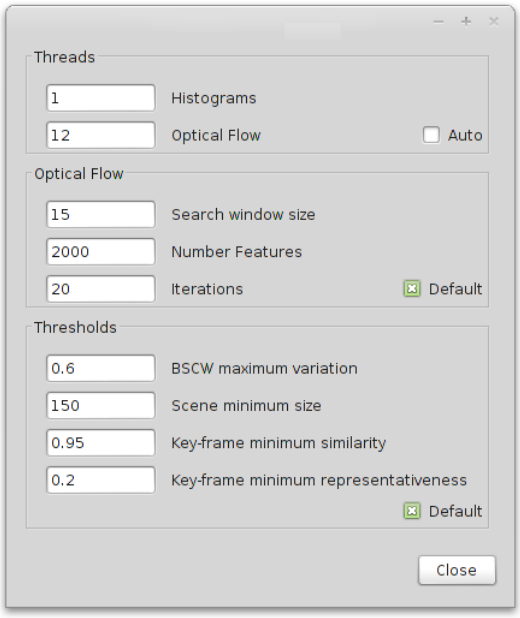


Figura 3.2: Tela de opções avançadas da ferramenta de segmentação de vídeo.

```
<?xml version="1.0" encoding="UTF-8"?>
<refSeg src="video.mp4" creationMethod="Manual" totalFNum="6543">
        <trans type="CUT" preFNum="814" postFNum="815" />
        <trans type="CUT" preFNum="2497" postFNum="2498" />
        <trans type="CUT" preFNum="4251" postFNum="4252" />
        <trans type="GRA" preFNum="4293" postFNum="4299" />
        <trans type="CUT" preFNum="4354" postFNum="4355" />
        <trans type="GRA" preFNum="5996" postFNum="6010" />
        <trans type="GRA" preFNum="6046" postFNum="6054" />
        <trans type="CUT" preFNum="6137" postFNum="6138" />
</refSeg>
```

Figura 3.3: Exemplo de arquivo XML obtido para a segmentação em tomadas.

Com o arquivo de registro de transição obtido, pode-se utilizar a segmentação do vídeo em cenas e/ou tomadas para diversas aplicações de Personalização e Adaptação, como o *framework* de anotações multimídia desenvolvido por Manzato (2011).

3.2 Implementação da Segmentação de Vídeo

A ferramenta foi implementada na linguagem Java[2] versão 1.7 utilizando-se da biblioteca JavaCV[3]. A biblioteca JavaCV é um *wrapper* para diversas bibliotecas utilizadas para a visão computacional, tais como a OpenCV[4] e a FFmpeg[5]. Com tal biblioteca, diver-

[2]http://www.java.com/
[3]http://code.google.com/p/javacv/
[4]http://opencv.org/
[5]http://www.ffmpeg.org/

sos aspectos consolidados como a decodificação de vídeo e extração de histogramas são automatizados e otimizados através do reuso de código.

Na implementação da técnica utilizada, criou-se três classes básicas: `Processamento`, `Tomada` e `Cena`. A classe `Processamento` é responsável por coordenar a maioria dos procedimentos realizados pela técnica de segmentação tais como decodificação, agrupamento de tomadas e/ou cenas, entre outras. A classe `Tomada`, por sua vez, representa uma tomada formada por um conjunto não-nulo de histogramas, além de outras funcionalidades tais como métodos de seleção de quadro-chave, cálculo de coerência entre tomadas ou ainda o cálculo do fluxo óptico. A classe `Cena`, por outro lado, contém um conjunto não-nulo de objetos da classe `Tomada`, apresentando métodos tais como a união entre duas cenas, cálculo da média do fluxo óptico, entre outros.

Diversas outras classes foram criadas para realizar tarefas específicas, tais como a decodificação do vídeo. As classes auxiliares desenvolvidas e suas principais funcionalidades são descritas abaixo:

- **Decodificador:** Classe responsável por tratar diretamente o fluxo de vídeo de entrada. Tem como objetivo abrir o arquivo de vídeo, decodificar um número específico de quadros e armazená-los temporariamente para que um objeto da classe `Processamento` analise os quadros individualmente.

- **Utils:** Classe formada apenas por métodos estáticos utilizados por diferentes classes. Seu principal objetivo é o de manter funcionalidades triviais, como calcular a intersecção entre dois histogramas, em apenas uma classe globalmente acessível.

- **XMLHandler:** Classe formada apenas por métodos estáticos utilizado pela classe `Processamento`, apenas. Têm por objetivo tratar a entrada e a saída de arquivos de registro no formato XML especificado na **Seção 3.1**.

- **Movimento:** Classe formada por uma lista de objetos da classe **Reta**. Responsável por calcular a "quantidade de movimento" de um quadro, dado os vetores do fluxo óptico calculado pelo método *calculateOpticalFlow* da classe `Utils`.

- **Reta:** Classe representando uma reta representada por dois objetos da classe `Ponto`.

- **Ponto:** Classe que representa um ponto no espaço da imagem analisada. Formada por duas variáveis inteiras representando a posição do ponto no eixo das abscissas e no eixo das ordenadas.

A descrição detalhada das classes e métodos envolvidos em cada procedimento da técnica são descritos nas subseções posteriores.

3.2.1 Primeira Fase

A primeira fase do algoritmo têm por objetivo extrair os histogramas dos quadros do vídeo, calcular os vetores do fluxo óptico quadro-a-quadro e segmentar o vídeo em tomadas.

A biblioteca OpenCV provê, por padrão, métodos para leitura tanto de vídeos de entrada como quadros em arquivos de formatos diversos. Optou-se por utilizar a entrada de dados através de fluxos de vídeo, já que o segundo método necessitaria de um processo de decodificação e gravação de todos os quadros do vídeo de entrada, causando um aumento considerável no requisito de espaço em disco e, também, reduzindo a velocidade geral do algoritmo devido ao excessivo número de acessos ao sistema de armazenamento.

A decodificação de quadros pode ser realizada de duas maneiras simples na biblioteca JavaCV: através da classe `OpenCVFrameGrabber` ou da classe `FFmpegFrameGrabber`. A primeira opção utiliza-se do método padrão adotado no OpenCV. A segunda opção adota o FFmpeg, uma biblioteca de codificação e decodificação disponível em diversos sistemas operacionais. Adotou-se a decodificação baseada na biblioteca FFmpeg, já que a mesma provê decodificação *multi-thread* em processadores com suporte a tal recurso (escalável conforme o número de núcleos de processamento existente), que resulta em uma alta velocidade de decodificação.

A decodificação é disparada, inicialmente, pela criação de um objeto da classe `Decodificador` pelo objeto da classe `Processamento`. A decodificação não é realizada integralmente, ou seja, alguns quadros do vídeo são decodificados e a decodificação é pausada temporariamente.

No momento da pausa na decodificação, o controle do programa volta à classe `Processamento` com o intuito de extrair os histogramas e o fluxo óptico do conjunto de quadros decodificados. A extração dos histogramas, que resulta em um vetor com um histograma para cada quadro do vídeo, é realizado pelo método *createHSV_Histogram* da classe `Utils`, que retorna o histograma HSV calculado pela biblioteca OpenCV.

Já o fluxo óptico é calculado armazenando-se, para cada quadro do vídeo, um objeto da classe `Movimento` em uma lista apropriada. O cálculo do fluxo óptico é realizado pelo método *calculateOpticalFlow* da classe `Utils`.

Ao final do cálculo do fluxo óptico e da extração dos histogramas, os quadros do vídeo armazenados no objeto da classe `Decodificador` são eliminados (com exceção do último quadro) e um novo conjunto de quadros é decodificado. O processo se repete até que todos os quadros do vídeo tenham sido decodificados e processados.

Nota-se que o procedimento realizado pela implementação é diferente do fluxo de trabalho "decodificar - processar - desalocar" para cada quadro, tradicionalmente adotado em técnicas de segmentação de vídeo. Na implementação descrita, ao contrário da abordagem tradicional, o custo de memória é um fator importante a se considerar: como os

quadros são armazenados temporariamente para processamento posterior, o tamanho de cada quadro individual e sua quantidade impactam diretamente o custo de memória da técnica. Por exemplo, armazenar um quadro RGB de resolução 1920x1080 necessita de, pelo menos, 5.932 MB de memória. Nesse cenário, mil quadros (cerca de 34 segundos de vídeo, com a taxa de 30 quadros por segundo) requerem cerca de 6 GB de memória principal.

Tal abordagem foi adotada para melhor aproveitar a divisão em múltiplas *threads* proporcionando maior rapidez no processamento dos procedimentos de extração dos histogramas e cálculo do fluxo óptico, realizados concorrentemente. O processamento concorrente é realizado no método *decHistOF* da classe `Processamento`, onde cria-se vetores de objetos das classes internas `HistWorker` e `OpticalFlowWorker`, dividindo-se os quadros decodificados entre cada objeto das classes criadas. No método *decHistOF*, cada posição na lista de objetos de `HistWorker` e `OpticalFlowWorker` são `threads` separadas responsáveis pelo processamento de um número pré-definido de quadros do vídeo de entrada, obtido na classe `Decodificador`.

Assim, por exemplo, considerando um processador com 4 núcleos lógicos de processamento, com uma divisão de uma *thread* para extração de histogramas e três *threads* para o cálculo do fluxo óptico, com 100 quadros decodificados, resultaria nos índices iniciais e finais exemplificados na **Tabela 3.1**.

Tabela 3.1: Exemplo de divisão de 4 *threads*, 1 para o extração de histogramas e 3 para o cálculo do fluxo óptico, com 100 quadros decodificados.

Thread	Índice inicial	Índice final	Tipo
$Thread_1$	0	99	Extração de histogramas
$Thread_2$	0	32	Cálculo de fluxo óptico
$Thread_3$	33	66	Cálculo de fluxo óptico
$Thread_4$	67	99	Cálculo de fluxo óptico

Conforme exemplifica a **Tabela 3.1**, a última *thread* de cada tipo (extração de histogramas ou cálculo do fluxo óptico) usualmente apresenta um número diferenciado de quadros a processar, devido à possíveis erros de arredondamento e/ou truncamento na divisão do número de quadros decodificados pelo número de *threads* disponíveis.

A vantagem deste modelo de separação de *threads* é a facilidade de uso e o encapsulamento *intra-threads*. Ou seja, após definidos os índices de trabalho para cada *thread*, não é mais necessário nenhum tipo de comunicação entre as *threads* até que todas elas tenham concluído seu processamento.

Somente após o final de todas as *threads* das classes `HistWorker` e `OpticalFlowWorker` é que os quadros armazenados temporariamente são eliminados.

Após a extração das *features* ter sido realizada, entra em cena a segmentação em tomadas. Para isso, é necessário criar-se dois vetores de similaridade (baseada na intersecção de histogramas) e dissimilaridade (baseada na diferença absoluta de histogramas). O grau de dissimilaridade entre dois histogramas é calculado pelo método *compareHistDiff* e o grau de similaridade é calculo pelo método *compareHistInter*, ambos presentes na classe `Utils`. Após os vetores terem sido calculados, entra em ação o procedimento de segmentação em tomadas, realizado de maneira tradicional (sem *multi-thread*), através do método *segmentarTomadas* da classe `Processamento`.

O procedimento de criação das tomadas resulta em uma série de objetos da classe `Tomada` contendo, cada um, os histogramas referentes à tomada que o objeto representa. Esse procedimento marca, também, o fim da primeira fase do algoritmo. O usuário, neste ponto, pode acionar o método *geraXMLTomadas*, que gera o arquivo de registro especificado na **Seção 3.1**, para a segmentação em tomadas.

3.2.2 Segunda Fase

A implementação da segunda fase da técnica é mais simples que a primeira fase, haja visto que a maioria dos procedimentos são realizados serialmente, sem a utilização de recursos *multi-thread*. Nota-se que, embora haja a divisão lógica em duas fases na técnica (segmentação em tomadas e em cenas), o mesmo não ocorre na implementação. Os procedimentos realizados e alguns detalhes de suas implementações são descritos abaixo:

- **Seleção dos quadros-chaves:** A seleção dos quadros chaves é realizada através método *calcKeyframes* da classe `Processamento`. Nesse método, cria-se um número pré-definido de objetos da classe privada `KeyframesWorker`, cada um com um número específico de tomadas para selecionar os quadros-chaves das tomadas, sendo executados em paralelo. A seleção de quadros propriamente dita é realizada pelo método público *criarKeyframes* da classe `Tomada`.

- **Cálculo da coerência de tomadas:** O cálculo do valor BSCW para cada tomada é realizado pelo método *calculateBSCW* da classe `Processamento`. Nele, um conjunto de objetos da classe privada `BSCWorker` são criados, cada qual responsável por calcular o valor BSCW concorrentemente. O valor BSCW é atribuído a cada tomada diretamente através do método *calcBSCW*, disponível publicamente na classe `Tomada`.

- **Criação de bordas de cenas:** A criação dos PSBs é realizado através do método *criarPSB* da classe `Processamento`. Com ela, determina-se se uma tomada pertence ou não à uma transição e, através da análise linear de tal afirmativa, cria-se o conjunto de cenas, cada uma representada por um objeto da classe `Cena`.

- **Remoção de cenas minúsculas:** O primeiro procedimento realizado após as cenas terem sido criadas é o método *removeCenasPequenas* da classe `Processamento`. Destaca-se que o método é realizado serialmente (sem o uso de *multi-threads*) e que a fusão de cena é realizada através do método *fundirCena* da classe `Cena`.

- **Remoção de cenas similares adjacentes:** O procedimento é realizado pelo método *removeCenasAdjacentes* da classe `Processamento`.

- **Remoção de cenas em janelas deslizantes:** O procedimento é realizado pelo método *removeCenasSimilaresJanelas* da classe `Processamento`.

- **Remoção de cenas por similaridade de movimento:** O procedimento é realizado pelo método *removeJanelaDeslizanteMovimento* da classe `Processamento`. Neste caso, o cálculo do limiar $FluxoOptico_{limiar}$ é realizado com o apoio da classe `Decodificador`, que ainda retém informações do vídeo, tal como a resolução.

- **Remoção de cenas adjacentes com altíssima similaridade:** O último procedimento para a fusão de objetos da classe `Cena` é realizada pelo método *removeIntercaladasSimilares*, presente na classe `Processamento`.

Após o método *removeIntercaladasSimilares* ter sido executado, o algoritmo está preparado para retornar ao usuário o arquivo de registro contendo as transições de cenas, realizado pelo método *geraXMLCenas* da classe `Processamento`.

Capítulo

4

Conclusões

Este relatório técnico apresentou uma técnica de segmentação de vídeo em tomadas e cenas. Apresentou-se, ainda, detalhes de uma implementação da técnica na linguagem Java com o auxílio de bibliotecas *open-source* tais como a OpenCV e a JavaCV.

A técnica desenvolvida apresenta uma importante vantagem sobre técnicas similares (Chen e Li, 2010; Rasheed e Shah, 2003) que também utilizam o agrupamento de tomadas através da comparação de histogramas: a independência de domínio. Rasheed e Shah (2003), por exemplo, após a segmentação baseada no valor BSC, realiza uma comparação da "quantidade de movimento" entre cenas em potencial utilizando, para isso, dados obtidos dos vetores de movimento do fluxo MPEG-1 de entrada. Semelhantemente, na técnica apresentada por Chen e Li (2010), mede-se a quantidade de movimento utilizando-se de informações dos vetores de movimento do fluxo MPEG-4 de entrada.

Embora a utilização dos vetores de movimento seja computacionalmente eficiente, o uso de tais informações restringe as técnicas desenvolvidas. Os vetores de movimento, embora largamente utilizados em formatos de codificação de vídeo que utilizam-se da estimação de movimento (tais como o MPEG-1, MPEG-2 e MPEG-4), não estão presentes em diversos formatos de codificação de vídeo. O formato MJPEG e todos os formatos de vídeo descomprimido são exemplos de formatos que não disponibilizam vetores de movimento no fluxo de dados. Os recursos obtidos pelas técnicas de Rasheed e Shah (2003) e Chen e Li (2010) utilizando os vetores de movimento são obtidos, na técnica desenvolvida, através do fluxo óptico, que pode ser calculado sobre qualquer fluxo de

vídeo de entrada que possa ser descomprimido e/ou decodificado, tornando a técnica independente de formato de vídeo de entrada.

A implementação, graças ao uso da biblioteca FFmpeg, é capaz de tratar dezenas formatos de vídeo diferentes, incluindo formatos populares tais como MPEG-1, MPEG-2, MPEG-4, RealVideo, FlashVideo e Windows Media Video, além de diversos formatos de vídeo descomprimido. Graças ao uso de métodos concorrentes (como a extração dos histogramas e o cálculo do fluxo óptico), a implementação é capaz de reduzir o impacto negativo de se calcular o fluxo óptico. Além disso, por não estar restrito a um número específico de *threads*, o uso em computadores com grande número elevado de núcleos de processamento ou até mesmo em *clusters* resultaria em um aumento significativo no desempenho da implementação.

Referências

Adomavicius, G.; Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, v. 17, p. 734–749, 2005.
Disponível em `http://dx.doi.org/10.1109/TKDE.2005.99` (Acessado em 21-02-2012)

Brunelli, R.; Mich, O.; Modena, C. A survey on the automatic indexing of video data,. *Journal of Visual Communication and Image Representation*, v. 10, n. 2, p. 78–112, 1999.
Disponível em `http://www.sciencedirect.com/science/article/pii/S1047320397904041` (Acessado em 21-02-2012)

Chaisorn, L.; Chua, T.-S.; Lee, C.-H. A multi-modal approach to story segmentation for news video. *World Wide Web*, v. 6, p. 187–208, 10.1023/A:1023622605600, 2003.
Disponível em `http://dx.doi.org/10.1023/A:1023622605600` (Acessado em 21-02-2012)

Chen, H.; Li, C. A practical method for video scene segmentation. In: *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, 2010, p. 153–156.

Hanjalic, A. Shot-boundary detection: unraveled and resolved? *Circuits and Systems for Video Technology, IEEE Transactions on*, v. 12, n. 2, p. 90–105, 2002.

Hu, W.; Xie, N.; Li, L.; Zeng, X.; Maybank, S. A survey on visual content-based video indexing and retrieval. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, v. 41, n. 6, p. 797–819, 2011.

Liu, J.; Li, M.; Liu, Q.; Lu, H.; Ma, S. Image annotation via graph learning. *Pattern Recogn.*, v. 42, p. 218–228, 2009.

Disponível em http://dl.acm.org/citation.cfm?id=1453255.1453378 (Acessado em 21-02-2012)

Lu, Y.; Sebe, N.; Hytnen, R.; Tian, Q. Personalization in multimedia retrieval: A survey. *Multimedia Tools and Applications*, v. 51, p. 247–277, 10.1007/s11042-010-0621-0, 2011. Disponível em http://dx.doi.org/10.1007/s11042-010-0621-0 (Acessado em 21-02-2012)

Magalhães, J.; Pereira, F. Using MPEG standards for multimedia customization. *Signal Processing: Image Communication*, v. 19, n. 5, p. 437–456, 2004.
Disponível em http://dx.doi.org/10.1016/j.image.2004.02.004 (Acessado em 21-02-2012)

Manzato, M. G. *Uma arquitetura de personalização de conteúdo baseada em anotações do usuário*. Tese (doutorado em ciência da computação e matemática computacional), Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2011.

Marques, O. *Practical Image and Video Processing Using MATLAB*. Wiley, IEEE Press, 690 p., 2011.

Ogawa, A.; Takahashi, T.; Ide, I.; Murase, H. Cross-lingual retrieval of identical news events by near-duplicate video segment detection. In: *Proceedings of the 14th international conference on Advances in multimedia modeling*, Berlin, Heidelberg: Springer-Verlag, 2008, p. 287–296 (*MMM'08*, v.1).
Disponível em http://dl.acm.org/citation.cfm?id=1785794.1785826 (Acessado em 21-02-2012)

Rasheed, Z.; Shah, M. Scene detection in hollywood movies and tv shows. In: *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2003, p. II – 343–8 vol.2.

Snoek, C. G. M.; Worring, M.; van Gemert, J.; Geusebroek, J.-M.; Koelma, D.; Nguyen, G. P.; de Rooij, O.; Seinstra, F. Mediamill: exploring news video archives based on learned semantics. In: *Proceedings of the 13th annual ACM international conference on Multimedia*, New York, NY, USA: ACM, 2005, p. 225–226 (*MULTIMEDIA '05*, v.1).
Disponível em http://doi.acm.org/10.1145/1101149.1101188 (Acessado em 21-02-2012)

Toffler, A. *Future Shock*. 1 ed. Bantam, 576 p., 1984.

Wang, C.; Jing, F.; Zhang, L.; Zhang, H.-J. Scalable search-based image annotation. *Multimedia Systems*, v. 14, n. 4, p. 205–220, 2008.
Disponível em `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.150.3175` (Acessado em 21-02-2012)

Yu, H.; Su, B.; Lu, H.; Xue, X. News video retrieval by learning multimodal semantic information. In: *Proceedings of the 9th international conference on Advances in visual information systems*, Berlin, Heidelberg: Springer-Verlag, 2007, p. 403–414 (*VISUAL'07*, v.1).
Disponível em `http://dl.acm.org/citation.cfm?id=1783294.1783340` (Acessado em 21-02-2012)