Chapter 1

Segmentação e recuperação de cenas em vídeo utilizando OpenCV

Rudinei Goularte, Tiago Henrique Trojahn

Abstract

The popularization of systems, applications and devices to produce, view and share video, increased the need to treat a large volume of multimedia data. In related areas (such as Multimedia Information Retrieval) a common requirement is the segmentation of a video into smaller, more manageable units, allowing the extraction of data and information essential to the proper perfomance of associated services. This short-course discusses the segmentation of videos into scenes, using current techniques exemplified in OpenCV. The interest into scenes is inherent to users, which are familiar to this concept, arousing the interest of services like Netflix and YouTube on this subject, attracting researchers and developers.

Resumo

A popularização de sistemas, aplicativos e dispositivos para produzir, exibir e compartilhar vídeo, fez surgir a necessidade de tratar um grande volume de dados multimídia. Nas áreas relacionadas (como Recuperação de Informação Multimídia) um requisito comum é a segmentação do vídeo em unidades menores e mais gerenciáveis, permitindo a extração de dados e informações essenciais para o bom funcionamento dos serviços associados. Este minicurso aborda a segmentação de vídeos em cenas, utilizando técnicas atuais exemplificadas em OpenCV. O interesse em cenas está presente de modo inerente nos usuários, familiares a esse conceito, o que leva também ao interesse de serviços como Netflix e YouTube nesse assunto, despertando o interesse de pesquisadores e desenvolvedores.

1.1. Introdução

Nos últimos anos houve um aumento na quantidade de dados multimídia disponíveis para acesso [Lu et al., 2011]. Isso, em parte, pode ser explicado pela proliferação de dispositivos de baixo custo para capturá-los e codificá-los. Além disso, o avanço da tecnologia

possibilitou o surgimento de uma grande variedade de meios que permitem ao usuário o uso de informações multimídia em qualquer lugar e a qualquer momento.

Atualmente, as pessoas podem ter acesso a conteúdos usando diferentes tipos de dispositivos e meios (notebooks, celulares, *Personal Digital Assistants*, WiFi, 3G e 4G, entre outros). Toda essa evolução criou ambientes heterogêneos [Bouyakoub and Belkhir, 2008], surgindo com isso desafios no tratamento dos dados, já que, geralmente, quando um dispositivo acessa um conteúdo multimídia para o qual não foi projetado, a experiência do usuário é insatisfatória [Magalhães and Pereira, 2004].

Além disso, a era digital trouxe outra importante característica: a interação do usuário com o conteúdo. Com os avanços da Web, as pessoas podem interativamente escolher diferentes caminhos de navegação, explorando variadas informações disponíveis, inclusive multimídia. Exemplos desse tipo de serviço são YouTube[1], Netflix[2] e Last.fm[3]. O avanço desses serviços fez com que, atualmente, usuários passassem a não somente acessar, mas, também, ativamente produzir conteúdo. Isso faz com que o volume de dados produzidos cresça contínua e rapidamente, agravando o problema da sobrecarga de informação: encontrar conteúdo de interesse em meio ao imenso volume de informações disponíveis [Toffler, 1984; Hu et al., 2011]. Tal problema vem sendo tratado por importantes áreas da Computação, como Recuperação de Informação [Baeza-Yates and Ribeiro-Neto, 1999], Recuperação de Informação Multimídia [Blanken et al., 2007], Recomendação e Personalização e Adaptação Multimídia [Lum and Lau, 2002; Mohan et al., 1999; Barrios et al., 2005].

No caso particular de vídeo digital, um fator comum é que os sistemas desenvolvidos nessas áreas necessitam que metadados sejam extraídos para representar o conteúdo de modo mais compacto e, também, servirem de alicerce para os serviços das áreas supracitadas. Entretanto, o processo de extração de metadados é complexo e envolve alto custo computacional [Blanken et al., 2007]. Por exemplo, é comum que vídeos sejam descritos por meio de suas características visuais e que para obtê-las sejam utilizadas técnicas de extração de características de imagens com relativo alto custo de processamento [Gonzalez and Woods, 2007]. A aplicação de tais técnicas no domínio de vídeos mostra-se um problema frente a grande quantidade de imagens a ser analisada. Consequentemente, o passo inicial dos sistemas citados é a segmentação de um vídeo em unidades menores e mais facilmente gerenciáveis, de modo a reduzir o volume de dados e facilitar o processamento [Lu et al., 2011].

Tradicionalmente um vídeo pode ser subdividido em quadros, tomadas e cenas. Quadros são imagens obtidas por uma câmera a uma taxa constante, formando o vídeo [Richardson, 2010]. Uma tomada pode ser definida como um conjunto de quadros capturados por uma única câmera e que representam uma ação contínua [Smeaton, 2007]. Uma cena, por sua vez, é um grupo de tomadas semanticamente relacionadas [Zhai and Shah, 2006]. Do ponto de vista do usuário, o mais interessante é que se obtenham cenas, já que elas contêm informações semanticamente mais relevantes. Além disso, diferente dos conceitos de quadro e tomada, o conceito de cena é familiar ao usuário comum. Por exemplo,

[1] http://www.youtube.com
[2] http://www.netflix.com
[3] http://www.lastfm.com

é usual ouvir as pessoas comentando a respeito de cenas de filmes a que assistiram ou afirmando que determinada cena de uma série de televisão que gostam é engraçada ou ainda perguntando sobre uma cena da novela do dia anterior.

A segmentação de vídeos digitais em cenas, entretanto, ainda é um campo de pesquisa sendo investigado e que apresenta muitos desafios. Isso porque é um processo de maior complexidade do que a segmentação em quadros ou tomadas, devido principalmente à subjetividade do conceito e à semântica envolvida [Hu et al., 2011; Zhu and Liu, 2008].

Este texto foi preparado para servir de material de apoio ao minicurso oferecido durante o XX Simpósio Brasileiro em Sistemas Multimídia e Web, realizado em novembro de 2014 em João Pessoa, Paraíba. O objetivo é introduzir conceitos e práticas de desenvolvimento para segmentação e recuperação de cenas em vídeos digitais. Das abordagens utilizadas para realizar segmentação em cenas, as baseadas em agrupamento de tomadas se destacam devido a fatores como penetração na comunidade científica, disponibilidade, custo computacional e desempenho [Chasanis et al., 2009; Sakarya and Telatar, 2010; Tapu and Zaharia, 2011]. As técnicas em estado da arte nesse contexto, segundo a literatura relacionada, utilizam duas abordagens não excludentes: características locais como modo de representação de quadros e tomadas no processo de segmentação em cenas; multimodalidade, utilizando características (locais ou não) advindas de mais de um tipo de mídia. Tais abordagens serão o foco deste minicurso.

As demais seções deste texto estão organizadas como segue. A Seção 1.2 apresenta além da biblioteca OpenCV, alguns dos fundamentos da área de segmentação de vídeo, tais como os conceitos de tomadas, cenas e características de vídeo tais como histogramas de cor e características locais. A Seção 1.3 discute a extração de quadros-chaves, uma importante técnica para a redução de volume de informação em segmentação de vídeo digital. A Seção 1.4 apresenta a abordagem de agrupamento de tomadas, uma das mais utilizadas para a segmentação em cenas devido a sua simplicidade e bons resultados. A Seção 1.5, por sua vez, apresenta uma abordagem multimodal para segmentação em cenas, potencialmente capaz de alcançar resultados superiores aos de técnicas baseadas em agrupamento de tomadas. Por fim, a Seção 1.6 descreve como podem ser comparadas diferentes técnicas de segmentação de vídeo, apresentando métricas simples e o conceito de "base confiável".

1.2. Biblioteca OpenCV & Conceitos Relacionados

Nesta seção, são discutidos ambos, a biblioteca *OpenCV*, utilizada como suporte ao minicurso, assim como alguns dos principais conceitos relacionados com segmentação de vídeo digital em cenas.

Além disso, a Seção 1.2.2 dedica-se a apresentar o conceito de vídeo digital e algumas de suas características básicas e a Seção 1.2.3 descreve a estrutura hierárquica presente em um vídeo. Por fim, as Seções 1.2.4 e 1.2.5 detalham os histogramas e as características locais, amplamente utilizados na área de segmentação devido ao baixo custo computacional (histogramas) e ao alto grau de discriminação (características locais). Por fim, dois exemplos de características locais, a SIFT e o SURF, são apresentadas nas Seções 1.2.6 e 1.2.7, respectivamente.

1.2.1. A Biblioteca OpenCV

A biblioteca OpenCV tem como objetivo incentivar o desenvolvimento de aplicações complexas voltadas para visão computacional em geral. Foi desenvolvida inicialmente pela Intel®em meados do ano 2000, sendo a mesma mantida atualmente pela Willow Garage[4] e também pela Itseez[5].

A OpenCV é uma biblioteca de código-aberto desenvolvida em C/C++ com suporte oficial a linguagens como C, C++, Java e Python. Além do suporte oficial, diversas outras bibliotecas, desenvolvidas por terceiros, podem ser utilizadas para estender a OpenCV a outras linguagens, como Ruby[6], C# e VisualBasic[7]. Algumas bibliotecas, além da OpenCV, oferecem suporte a recursos extras, como a JavaCV[8].

A biblioteca, além de uma ampla gama de linguagens de programação, também pode ser utilizada em diversas plataformas e sistemas operacionais, tais como o Microsoft®Windows™, FreeBSD, OpenBSD, Linux, OS X™, Android™e iOS™.

Um dos grandes destaques da biblioteca é a extensa documentação oficial, principalmente para as linguagens Python e C/C++, com a apresentação detalhadas das funcionalidades, tutoriais para diversas tarefas comuns e exemplos distribuídos juntamente com a biblioteca. Isso, aliado com o grande número de desenvolvedores e um amplo suporte da comunidade, tornam a OpenCV uma ferramenta bastante útil em diversos trabalhos na área.

Nesse sentido, a documentação da biblioteca[9] conta com a descrição detalhada das classes e funções dos diferentes módulos que compõe a OpenCV. Além disso, estão disponíveis [10] um amplo conjunto de tutoriais detalhados que apresentam diversos recursos da biblioteca, como a extração de histogramas, erosão e dilatação, derivadas de Sobel, *template matching*, entre outros.

Atualmente, está em desenvolvimento a versão 3.0 da OpenCV que visa, entre outros, prover suporte a tecnologias como o CUDA™e o OpenCL™, que objetivam aumentar a velocidade das aplicações ao permitir o uso de placas gráficas dedicadas. Com isso, espera-se que procedimentos mais complexos, atualmente com alto custo computacional, possam ser melhor aproveitadas em trabalhos da área.

Neste trabalho, utilizou-se a última versão estável disponibilizada no site oficial: 2.4.9. Para utilizá-la, faz-se necessário compilar e/ou instalar diversas dependências (tais como as bibliotecas de decodificação e arquivos jpeg, png e arquivos de vídeo), além de compilar a própria biblioteca OpenCV[11]. Caso seja utilizada uma distribuição Linux baseada em Debian, como o Ubuntu ou Mint, pode-se instalar as bibliotecas já compiladas e disponíveis nos repositórios da distribuição. Nota-se ainda que a versão disponível no repositório pode não ser a mais atual, mas as diferenças normalmente giram em torno de

[4] http://www.willowgarage.com/
[5] http://itseez.com/
[6] https://github.com/ruby-opencv/ruby-opencv
[7] http://www.emgu.com/wiki/index.php/Main_Page
[8] https://github.com/bytedeco/javacv
[9] http://docs.opencv.org/
[10] http://docs.opencv.org/doc/tutorials/tutorials.html
[11] http://docs.opencv.org/doc/tutorials/introduction/linux_install/linux_install.html

correções de pequenos bugs, podendo a mesma ser utilizada sem grandes prejuizos ao desenvolvedor.

Para exemplificar um programa escrito com a biblioteca OpenCV, o Código 1.1 apresenta um código-fonte de um programa capaz de decodificar um vídeo de nome **video.avi** e exibir seus quadros, um a cada milissegundo.

```
1  #include "opencv2/core/core.hpp"
   #include "opencv2/imgproc/imgproc.hpp"
3  #include "opencv2/highgui/highgui.hpp"
   #include <iostream>
5
   using namespace std;
7  using namespace cv;

9  int main(int argc, char* argv[]) {
       VideoCapture video("video.avi");
11     Mat quadro;
       while(1) {
13         bool temp = video.read(quadro);
           if(!temp) {
15             break;
           }
17         imshow("Video",quadro);
           waitKey(1);
19     }
       return 1;
21 }
```

Código 1.1: Exemplo de um programa escrito com a biblioteca OpenCV que exibe todos os quadros de um arquivo chamado de “video.avi”.

A explanação detalhada de cada procedimento relevante, implementado no código de exemplo mostrado anteriormente, é dada a seguir:

- Linhas 1 a 3 - Importação de algumas das dependências mais usadas da OpenCV.
- Linha 7 - Utilizada para especificar o namespace das funcionalidades da OpenCV, evitando que as chamadas ao método use tal namespace, como por exemplo *Mat* ao invés de *cv:Mat*.
- Linha 10 - Abre um sistema de captura de vídeo, armazenado localmente, chamado de “video.avi”.
- Linha 11 - Definição de uma variável, *quadro*, que será utilizada para armazenar cada um dos quadros do vídeo.
- Linha 14 - Decodifica um quadro do vídeo (inserindo-o na variável *quadro*), retornando um valor booleano indicando se a decodificação pode ou não ser feita com sucesso.
- Linhas 15 a 17 - Verifica se a decodificação pode ser feita com sucesso, encerrando o laço de repetição se o vídeo já foi completamente decodificado.

- Linha 19 - Exibe o quadro decodificado em uma janela com nome de “Video”, criada caso ainda não exista.

- Linha 20 - Interrupção de 1ms, efetivamente pausando o laço de repetição.

1.2.2. Vídeo Digital

Vídeo, do latim *videre* (ver), pode ser definido como um “conjunto de imagens estáticas que transmitem a sensação de movimento” [Chapman and Chapman, 2009]. Assim, um vídeo é formado por um conjunto de imagens estáticas, chamadas de quadros ou *frames*, que são apresentadas a uma taxa, normalmente constante, de tal modo que dão a ilusão de movimento.

A taxa requerida para que haja a percepção do movimento depende de diversos fatores tais como luminosidade, brilho e contraste tanto dos quadros como do ambiente, além do próprio estado fisiológico do espectador.

Um vídeo digital apresenta uma série de características, algumas das quais estão apresentadas a seguir [Richardson, 2010]:

- **Número de quadros por segundo:** um vídeo possui uma determinada taxa na qual os quadros são apresentados ao espectador. A taxa varia conforme diferentes padronizações entre países, modo de armazenamento, entre outros. Utiliza-se, usualmente, a forma abreviada fps, do Inglês *frames per second*, para se referir a essa taxa.

- **Resolução:** a resolução é o número de pixels na horizontal e na vertical. Um vídeo é dito de resolução A x B quando seus quadros possuírem A pixels na horizontal e B pixels na vertical.

- **Colorimetria:** assim como imagens, um dos detalhes mais importantes de um vídeo é a sua colorimetria. Vídeos e imagens podem ser de diferentes espaços de cores, como o *Red-Green-Blue* (RGB) ou o YCbCr, entre outros.

- **Detalhes de codificação:** um vídeo é geralmente codificado para que possa ser armazenado em um espaço de armazenamento tangível. Ao codificar um vídeo, surgem variáveis como o codificador empregado, taxa de compressão, taxa de *bits* por segundo, relação PSNR (*Peak Signal-to-Noise Ratio*) [Salomon et al., 2006], quantização, vetores de movimento, entre outros.

1.2.3. Estrutura hierárquica de um vídeo

Além dos detalhes da mídia, para a segmentação de vídeo é importante considerar ainda o relacionamento semântico existente em um vídeo. Os quadros, unidades fundamentais de um vídeo, podem ser agrupados em uma ou mais tomadas. Por fim, um conjunto de tomadas consecutivas pode ser agrupado formando uma cena. A Figura 1.1 ilustra as estruturas de um vídeo em diferentes níveis de abstração, partindo do quadro, passando pela tomada e cena, chegando no vídeo em si.

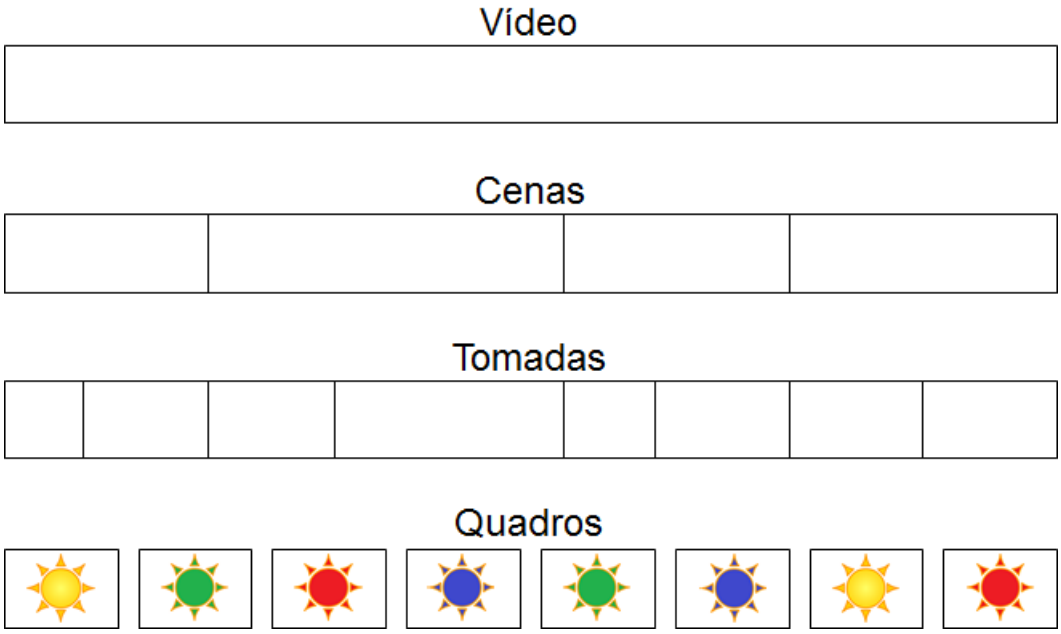


Figura 1.1: Estruturas formadoras de um vídeo, em diferentes níveis de abstração semântica. Adaptado de [Del Fabro and Böszörmenyi, 2013].

Uma tomada pode ser formalmente definida como um conjunto de imagens estáticas obtidas continuamente por uma única câmera [Koprinska and Carrato, 2001], representando uma ação contínua no tempo e espaço [Zhang et al., 2007]. A interrupção de uma tomada demarca o fim de uma e o começo da próxima tomada, determinando a chamada *transição de tomadas*. As transições de tomadas podem ser classificadas em transições abruptas ou graduais, cujas diferenças são descritas a seguir.

- **Transição Abrupta:** quando há uma mudança significativa entre dois quadros adjacentes, podendo ou não representar a troca do conteúdo semântico (por exemplo, demonstrar o mesmo cenário em diferentes ângulos).

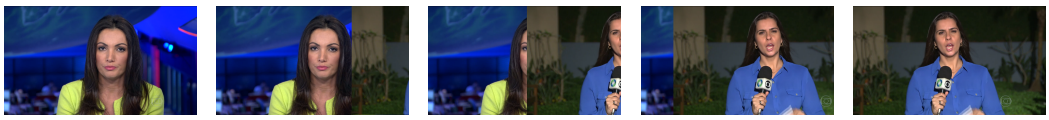
- **Transição Gradual:** quando alguns quadros de ambas as tomadas adjacentes são utilizadas para suavizar a transição entre as tomadas. São de três tipos básicos [Koprinska and Carrato, 2001]: dissolução, onde a imagem antiga é dissolvida em outra imagem; *fade in* e *fade out*, quando uma imagem é clareada ou escurecida gradativamente; *wipe*, quando uma imagem dá lugar à outra imagem através de um "empurrão".

As Figuras 1.2 e 1.3 exemplificam, respectivamente, uma transição abrupta e uma transição gradual, com o efeito *wipe*, entre duas tomadas.

Cena pode ser definida como um conjunto de tomadas semanticamente correlacionadas [Huang et al., 1998], sendo que o correlacionamento semântico pode ser encontrado por técnicas utilizando correlação entre tomadas [Rasheed and Shah, 2003; Chen and Li, 2010] ou dicionários visuais [Chasanis et al., 2009], entre outros.

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

Figura 1.2: Exemplo de duas tomadas consecutivas, filmadas sobre diferentes ângulos, com uma transição abrupta entre a Figura c) e d).



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

Figura 1.3: Exemplo de duas tomadas consecutivas, com o efeito de transição gradual presente nos quadros b), c), d) e e).

Por fim, destaca-se que a estrutura hierárquica apresentada nessa seção não é única. Por exemplo, pode-se agrupar um conjunto de cenas formando "capítulos". A divisão de um vídeo em quadros, tomadas e cenas, porém, é a mais utilizada em trabalhos de segmentação de vídeo em cenas [Del Fabro and Böszörmenyi, 2013].

1.2.4. Histogramas de cor

Segundo Marques [2011], o histograma de uma imagem é uma representação gráfica da frequência de cada nível de cor. A importância dos histogramas reside no fato de ser uma representação de uma imagem com volume de dados reduzido, tornando o processamento mais rápido.

Histogramas podem ser calculados sobre vídeos em escala-de-cinza ou coloridos, segmentando-se o mesmo em quadros e decompondo-se seus canais de cor Marques Filho and Vieira Neto [1999]. Caso o vídeo seja em escala-de-cinza, haverá 256 valores possíveis para os pixels, podendo-se, então, criar um histograma com 256 níveis. Porém, caso o vídeo seja de algum espaço de cor como o RGB ou HSV (*Hue-Saturation-Value*), o histograma resultante teria quase 17 milhões de possíveis cores a serem representadas (considerando todas as combinações entre cada nível de cor), 256 para cada canal de cor.

Assim, costuma-se quantizar cada quadro do vídeo [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). A Figura 1.4 ilustra o processo de quantização com 4 *bins* do cinza.

Um histograma 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

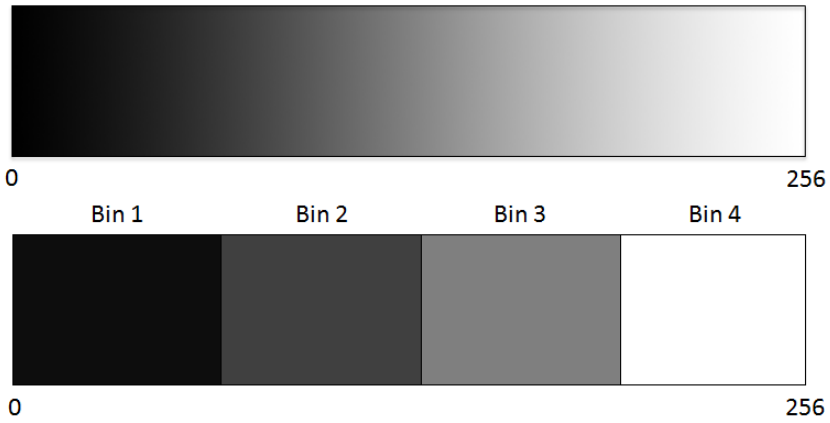


Figura 1.4: Exemplo de quantização dos 256 níveis da escala de cinza em 4 *bins*.

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, principalmente em degradês.

Um detalhe importante diz respeito aos valores em cada nível do histograma: caso sejam comparados dois histogramas de imagens de resoluções diferentes, o resultado obtido poderá ser incorreto. Tal erro reside do fato de que a quantidade de pixels em cada imagem (e, consequentemente, em seus histogramas) é diferente. Assim, para solucionar tal problema, geralmente aplica-se um processo de normalização dos valores. Neste processo, o somatório de todos os pixels do histograma é considerado como 1.0 (ou seja, 100% da imagem).

A comparação entre duas imagens, por meio de seus histogramas, é uma tarefa comum para muitas aplicações. Para poder comparar é necessário medir a similaridade entre as duas imagens, o que geralmente é feito utilizando-se medidas de similaridade (ou funções de distância) (1.2.5) [Blanken et al., 2007]. No caso de histogramas, duas das principais medidas de similaridade são a intersecção de histogramas e a diferença absoluta de histogramas [Marques, 2011]. A intersecção de histogramas e a diferença absoluta são descritas nas Equações 1 e 2.

$$d(H_1,H_2) = \sum_{I}^{N} \min(H_1(I),H_2(I)) \tag{1}$$

$$d(H_1,H_2) = \sum_{I}^{N} (H_1(I) - H_2(I))^2 \tag{2}$$

Onde H_1 e H_2 são dois histogramas, $H_1(I)$ e $H_2(I)$ é o número de pixels no I-nível de cor dos histogramas e N é o número total de níveis de cores dos histogramas H_1 e H_2.

Na intersecção de histogramas os valores de cada um dos níveis correspondentes de dois histogramas são comparados, gerando um valor que contenha o número de pixels

em comum, que estão no mesmo nível de cor, entre as duas imagens. Se os histogramas analisados passarem por um procedimento de normalização a intersecção será um valor no intervalo $[0,1]$, onde "zero"significa nenhuma intersecção e "um"significa histogramas idênticos.

Já no caso da diferença de histogramas, armazena-se o valor da diferença absoluta entre dois níveis correspondentes de cor de histogramas diferentes. Se os histogramas analisados passarem por um procedimento de normalização os valores obtidos pertencem ao intervalo $[0,2]$, onde zero significa "histogramas idênticos" e dois significa "nenhum nível de cor em comum".

Por fim, um histograma pode ser global ou local. Histogramas globais utilizam como informação todos os pixels da imagem como base para o cálculo do histograma. Já histogramas locais utilizam grupos de pixels, chamados de blocos, variando em número conforme o tamanho de cada bloco. Coimbra [2011], por exemplo, utilizou em seu trabalho histogramas locais com tamanho de bloco de 16 x 16 pixels. Histogramas locais são mais adequados para localizar objetos em uma imagem, pois reduzem a influência do plano de fundo [Ullman et al., 2002], enquanto que histogramas globais mostram-se mais eficientes que os histogramas locais para a segmentação em cenas [Coimbra, 2011] por considerarem a imagem como um todo.

Em termos de implementação, a biblioteca OpenCV disponibiliza um amplo suporte tanto à extração quanto a análise de histogramas. Nesse caso, utiliza-se a função *calcHist* que recebe como parâmetros, entre outros, o número de *bins* para cada canal de cor e o espaço de representação de cada canal para gerar o histograma. O resultado é armazenado em um tipo de dado chamado de *Mat*, utilizado par armazenar qualquer tipo de informação numérica multidimensional.

Como exemplo, o Código 1.2 apresenta uma função escrita com auxílio da OpenCV que demonstra a extração de um histograma de um quadro de vídeo, no espaço de cor RGB, com 16 *bins* para cada canal.

```
1  Mat calcularHistograma(Mat quadro) {
     float tamanho[] = {0,256};
3    const float* espaco[] = {tamanho, tamanho, tamanho};
     int canais[] = {0,1,2};
5    int binsR = 16;
     int binsG = 16;
7    int binsB = 16;
     int binsHistograma[] = {binsR, binsG, binsB};
9    Mat histograma;

11   calcHist(&quadro,1,canais,Mat(),histograma,3,binsHistograma,espaco);

13   return histograma;
   }
```

Código 1.2: Código fonte de uma função capaz de gerar o histograma de um quadro.

Além do cálculo do histograma, a OpenCV disponibiliza ainda uma série de métricas pré-definidas para o cálculo da dissimilaridade entre dois histogramas, através da função *compareHist*. São implementadas o método da correlação, Chi-Square, intersec-

ção e a distância de Bhattacharyya. Como exemplo, o Código 1.3 descreve uma função simples que retorna o valor de intersecção entre dois histogramas.

```
double compararHistogramas(Mat h1, Mat h2) {
  double similaridade = compareHist(h1,h2,CV_COMP_INTERSECT);
  return similaridade;
}
```

Código 1.3: Função desenvolvida com o auxílio da biblioteca OpenCV que retorna a intersecção entre dois histogramas.

Deve-se notar que outras métricas podem ser utilizadas, através da obtenção de cada nível de cor armazenada no histograma calculado.

1.2.5. Descritores Locais

Os diversos serviços de Recuperação de Informação baseados em vídeo necessitam, para operar, de uma representação computacional dos dados. Em muitos casos, essa representação é baseada em atributos visuais e obtida a partir da unidade mais básica do vídeo: o quadro, que é, na verdade, uma imagem. Tais atributos são conhecidos na literatura como *características* (do Inglês: *features*), e podem ser cor, textura, movimento, entre outros [Blanken et al., 2007]. Os histogramas, discutidos na Seção 1.2.4, são um exemplo de representação para a característica cor.

Assim, para extrair as características presentes em um quadro é necessário utilizar um descritor de imagem, capaz de gerar uma caracterização compacta dos dados de modo a facilitar o processamento mas, ao mesmo tempo, representativa do conteúdo.

Um descritor de imagem é composto por um algoritmo de extração para codificar características da imagem em vetores de características e por uma medida de similaridade para comparar duas imagens. A medida de similaridade é uma função de comparação, que fornece o grau de similaridade para um dado par de imagens representadas por seus vetores de características e é geralmente definida como uma função inversa à distância (por exemplo, Euclidiana), isto é, quanto maior o valor da distância, menos similares são as imagens [Torres and Falcão, 2006].

A seguir são apresentadas definições de imagem, vetor de características e descritor de imagem, segundo Torres and Falcão [2006].

Uma imagem I é um par $(D_I, \vec{I})$, onde:

- D_I é um conjunto finito de pixels (pontos em N^2, isto é, $D_I \subset N^2$) e
- $\vec{I} : D_I \rightarrow R^n$ é uma função que associa para cada pixel p em D_I um vetor $\vec{I}(p) \in R^n$ (por exemplo, $\vec{I}(p) \in R^3$ quando uma cor no sistema RGB (Red, Green, Blue) é associada a um pixel).

Um vetor de características $\vec{v}_I$ de imagem I pode ser visto como um ponto no espaço R^n: $\vec{v}_I = (v_1, v_2, ..., v_n)$, onde n é a dimensão do vetor.

Exemplos de possíveis vetores de características são histogramas de cor, momentos de Zernike e filtros de Gabor. Eles codificam propriedades da imagem, tais como

cor, forma e textura, respectivamente. Vale ressaltar que diferentes tipos de vetores de características podem requerer diferentes funções de similaridade.

Um descritor de imagem D é definido como uma tupla $(\varepsilon_D, \delta_D)$, onde:

- $\varepsilon_D : \{I\} \rightarrow R^n$ é uma função que extrai um vetor de características $\vec{v}_I$ de uma imagem I.
- $\delta_D : R^n \times R^n \rightarrow R^n$ é uma função de similaridade (por exemplo, baseada na distância métrica) que computa a similaridade entre duas imagens como o inverso da distância entre seus correspondentes vetores de características.

A Figura 1.5 ilustra o uso de um descritor simples D para computar a similaridade entre duas imagens I_A e I_B. Primeiramente, o algoritmo de extração ε_D é usado para computar os vetores de características $\vec{v}_{IA}$ e $\vec{v}_{IB}$ associados às imagens. Então, a função de similaridade δ_D é usada para determinar o valor da similaridade d entre as imagens. Vale ressaltar que múltiplos descritores podem ser combinados em um descritor complexo, o qual é capaz de codificar várias propriedades da imagem ao mesmo tempo (Torres et al., 2005).

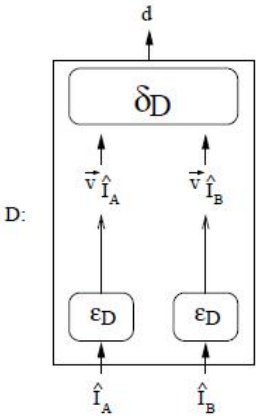


Figura 1.5: Um descritor simples D para computar a similaridade entre imagens. Retirado de Torres and Falcão [2006].

Um método recente na literatura para descrição de imagens, que vem sendo aplicado com sucesso em tarefas de recuperação, são os descritores locais, que consistem na extração de características por meio da identificação e descrição de pontos de interesse (do Inglês: *Points-of-Interest* - PoI). Embora existam diversas técnicas de descrição local de imagens, todas se baseiam em três passos principais, explicados a seguir, segundo Bay et al. [2006].

- Pontos de interesse são selecionados em alguns locais da imagem, tais como cantos ou junções-T. A propriedade mais valiosa de um detector de pontos de interesse é a repetitividade, isto é, se é confiável para encontrar os mesmos pontos de interesse em diferentes condições de visualização.

- A vizinhança de cada ponto de interesse é representada por um vetor de características. Esse descritor deve ser diferenciado e, ao mesmo tempo, robusto a ruídos, detecção de erros e deformações geométricas e fotométricas.

- Os descritores de diferentes imagens são comparados. A comparação é frequentemente baseada na distância entre vetores, por exemplo, distância Euclidiana. A dimensão do descritor tem um impacto direto no tempo de execução deste passo.

Dois exemplos de descritores locais, considerados estado da arte e que seguem os passos descritos anteriormente na busca por similaridade entre imagens, são o SIFT (*Scale Invariant Feature Transform*) e o SURF (*Speeded Up Robust Features*), apresentados a seguir.

1.2.6. SIFT

O método SIFT transforma uma imagem em uma coleção de vetores de características locais, sendo cada um deles invariante a translação, escala e rotação e parcialmente invariante a mudanças na iluminação ou projeção 3D [Lowe, 1999]. Tais vetores estão localizados tanto no domínio espacial quanto no domínio das frequências, reduzindo a probabilidade de perturbações por oclusão, desorganização ou ruídos [Lowe, 2004].

Para a extração das características é utilizada uma abordagem de filtragem em cascata, na qual as operações mais caras são aplicadas apenas em locais que passam por um teste inicial. A seguir são apresentados, segundo Lowe [2004], os quatro principais estágios de computação usados para gerar o conjunto de características de uma imagem.

1. Detecção de extremos escala-espaço. O primeiro estágio de computação realiza buscas em todas as escalas e locais da imagem. Tal estágio é implementado utilizando uma função de diferença de Gaussianas para identificar potenciais pontos de interesse invariantes a escala e orientação.

2. Localização dos pontos chave. Em cada local candidato, um modelo detalhado é aplicado para determinar localização e escala. Pontos chave são selecionados com base em medidas de sua estabilidade.

3. Atribuição de orientação. Uma ou mais orientações são atribuídas a cada ponto chave baseadas nas direções de gradientes locais da imagem. Todas as futuras operações são realizadas nos dados da imagem que foram transformados com relação a orientação, escala e localização atribuídos para cada característica, provendo assim invariância a essas transformações.

4. Descritor de pontos chave. Os gradientes locais da imagem são medidos, na escala selecionada, na região ao redor de cada ponto chave. Eles são transformados em uma representação que permite níveis significativos de distorções locais de forma e mudanças na iluminação.

Um importante aspecto do método SIFT é que ele gera um grande número de vetores de características, de 128 dimensões, que cobrem a imagem em termos de escalas

e localizações. Uma imagem típica de 500x500 pixels dará origem a cerca de 2000 vetores (embora esse número dependa tanto do conteúdo da imagem quanto da escolha de diversos parâmetros). A quantidade de características é particularmente importante para o reconhecimento de objetos, pois, para uma identificação confiável de pequenos elementos em fundos desordenados, pelo menos três características de cada objeto devem apresentar correspondência [Lowe, 2004].

Porém, a alta dimensionalidade dos vetores obtidos com o SIFT mostra-se uma desvantagem no passo de comparação de descritores de diversas imagens [Bay et al., 2006].

A biblioteca OpenCV oferece ao usuário o suporte aos descritores SIFT, através da classe *SIFT*. Com ela, pode-se calcular facilmente os descritores de uma imagem de entrada de 8bits (por exemplo, em escala-de-cinza), do tipo *Mat*. Diversos outros parâmetros podem ser ajustados conforme a necessidade do usuário, tal como uma máscara (*Mat*) para limitar quais regiões da imagem devem ser processadas, uma lista pré-definida de pontos de interesse (caso o usuário tenha selecionado anteriormente alguns pontos de interesse), entre outros.

1.2.7. SURF

O método SURF possui detectores e descritores de pontos de interesse invariantes a escala e rotação. Isso porque essas propriedades apresentam um bom compromisso entre a complexidade das características e robustez para deformações comumente encontradas. Por sua vez, inclinação e perspectiva são considerados efeitos de segunda ordem, os quais são cobertos em algum grau pela robustez geral do descritor [Bay et al., 2006]. O algoritmo possui dois componentes principais: o detector de pontos de interesse e o descritor, sendo cada um deles sucintamente explicados a seguir, de acordo com Bay et al. [2006].

- Detector de pontos de interesse. É baseado na matriz de Hessian, devido ao seu bom desempenho em termos de tempo de computação e precisão. O determinante de Hessian é utilizado para selecionar a localização e a escala.
- Descritor. Possui dois passos principais:
    1. Atribuição de orientação. A fim de ser invariante a rotação, identifica-se uma orientação reproduzível para cada ponto de interesse. Para isso, são calculadas as respostas Haar-wavelet nas direções x e y, em uma vizinhança circular ao redor do ponto de interesse.
    2. Componentes do descritor. Consiste em construir uma região quadrada em torno do ponto de interesse, alinhada à orientação selecionada no passo 1 e extrair dela os descritores.

É importante destacar que, em geral, esse algoritmo utiliza vetores com apenas 64 dimensões (apesar de ser possível utilizar o descritor estendido com 128 dimensões), reduzindo, em relação ao SIFT, o tempo da computação das características e também da comparação entre descritores de imagens, ao mesmo tempo em que aumenta a robustez [Bay et al., 2006].

A biblioteca OpenCV, assim como para o SIFT, oferece ao usuário o suporte aos descritores SURF, através da classe *SURF*. Com ela, pode-se calcular facilmente os descritores de uma imagem de entrada de 8bits (por exemplo, em escala-de-cinza), do tipo *Mat*. Diversos outros parâmetros podem ser ajustados conforme a necessidade do usuário, tal como uma máscara (*Mat*) para limitar quais regiões da imagem devem ser processadas, uma lista pré-definida de pontos de interesse (caso o usuário tenha selecionado anteriormente alguns pontos de interesse), entre outros. Adicionalmente, o usuário também pode configurar o detector para extrair características com 68 ou 128 dimensões (descritor estendido).

1.3. Extração de quadros chave

A extração, ou seleção, de quadros-chave (do Inglês: *keyframe*) tem por objetivo encontrar conjuntos de quadros-chave que sejam capazes de representar adequadamente cada tomada. Tal seleção é de suma importância para a segmentação em cenas baseadas em agrupamento de tomadas, pois, depois de selecionados, são os quadros-chave que serão utilizados para comparar duas tomadas adjacentes e decidir se pertencem ou não à mesma cena. Com os quadros-chave pode-se então realizar o processamento apenas sobre tais quadros, e não sobre a toda a tomada, resultando em um significativo ganho de desempenho.

A seleção de quadros-chaves parte do pressuposto que quadros adjacentes tendem a apresentar conteúdo visual semelhante ou até mesmo idêntico [Jacobs et al., 1995]. Assim, caso seja utilizado apenas um conjunto restrito de quadros (os "quadros-chave"), o resultado obtido pode ser idêntico ou muito próximo ao resultado que seria obtido calculando-se a similaridade usando todos os quadros das tomadas. Para obter bons resultados é necessário que os quadros-chave sejam selecionados adequadamente. O problema da seleção de quadros-chave é exatamente como realizar tal seleção.

Em geral, a extração de quadros-chaves se divide em duas grandes abordagens: extrair um quadro-chave ou um conjunto de quadros-chave. A Seção 1.3.1 apresenta algumas técnicas para selecionar apenas um quadro chave e, a Seção 1.3.2, descreve algumas abordagens para seleção de um conjunto de quadros-chave.

1.3.1. Extração de um quadro-chave

Partindo do pressuposto que os quadros de uma mesma tomada são similares [Jacobs et al., 1995], pode-se concluir então que a escolha de algum desses quadros seja o bastante para representar o restante da tomada. Com isso, a tarefa de representar uma tomada recai em selecionar o quadro adequado para determinada tomada.

Três técnicas bem conhecidas para a seleção de quadros-chave são: selecionar o primeiro quadro, o último quadro ou o quadro mediano como quadro-chave [Gu et al., 2007; Li et al., 2001]. Tais técnicas são populares devido a sua simplicidade e ao baixo custo computacional, já que o único processamento envolvido é ordenar os quadros da tomada (tarefa trivial, haja visto que os quadros são geralmente armazenados ou decodificados sequencialmente).

Porém, selecionar um quadro-chave baseado unicamente em sua posição pode não

ser representativo: em uma mesma tomada, embora os quadros adjacentes não apresentem grande variação, um grande número de ações e transformações visuais pode ocorrer ao longo da tomada. Por exemplo, em uma mesma tomada, objetos podem desaparecer ou surgir, além das próprias alterações do cenário e da câmera.

Assim, procura-se desenvolver alguma forma de representação capaz de melhor representar uma tomada complexa, com potencialmente diversas transformações visuais. Nesse sentido, por exemplo, Souza and Goularte [2013] estimaram o "melhor quadro" para representar a tomada, que conteria o quadro mais similar aos demais quadros da tomada e, por isso, poderia ser mais significativo que simplesmente selecionar um quadro-chave.

Embora intuitivamente o método do "melhor quadro" devesse apresentar melhores resultados, Souza and Goularte [2013] não encontraram diferenças significativas entre esse método e aqueles baseados em posição quando aplicados em segmentação de cenas.

Em termos de implementação, o Código 1.4 apresenta o código-fonte de uma função capaz de escolher o "melhor quadro", ou seja, o quadro da tomada mais similar aos demais quadros da tomada. Nesse algoritmo, considera-se que os histogramas já foram extraídos como demonstrado na Seção 1.2.4 e agrupados em tomadas manualmente ou automaticamente.

```
Mat melhorQuadro(vector<Mat> tomada) {
  vector<int> numeroSimilares;
  for(int i = 0; i < tomada.size(); i++) {
    numeroSimilares.push_back(0);
  }

  for(int i = 0; i < tomada.size(); i++) {
    for(int j = i + 1; j < tomada.size(); j++) {
      if(compararHistogramas(tomada[i],tomada[j]) >= 0.95) {
        numeroSimilares[i] = numeroSimilares[i] + 1;
        numeroSimilares[j] = numeroSimilares[j] + 1;
      }
    }
  }

  int maiorValor = numeroSimilares[0];
  int maiorIndice = 0;

  for(int i = 1; i < numeroSimilares.size(); i++) {
    if(numeroSimilares[i] > maiorIndice) {
      maiorValor = numeroSimilares[i];
      maiorIndice = i;
    }
  }

  return tomada[maiorIndice];
}
```

Código 1.4: Função desenvolvida com a OpenCV capaz de retornar o quadro mais similar aos demais quadros da tomada.

1.3.2. Extração de um conjunto de quadros-chave

Ao selecionar um determinado quadro-chave para representar uma tomada alguns detalhes da mesma podem ser perdidas. Uma possível solução é selecionar um conjunto de quadros-chave para cada tomada [Liu et al., 2004; Mukherjee et al., 2007; Rathod and Nikam, 2013; Besiris et al., 2007; Rasheed and Shah, 2003, 2005]. O desafio para essas técnicas, além de selecionar quadros adequados para cada tomada, é o de reduzir a redundância entre o conjunto de quadros-chave selecionados.

As técnicas desenvolvidas são bastante variadas. Pode-se utilizar, por exemplo, um conjunto de quadros-chave formados pelo quadro inicial, médio e final da tomada. Rasheed and Shah [2003], por sua vez, propõem:

1. Selecionar o quadro médio da tomada e o adicioná-lo ao conjunto K de quadros-chave.

2. Percorrer os demais quadros da tomada e compará-los (usando a intersecção de histogramas de cor) com todos os quadros do conjunto K.

3. Caso o quadro candidato tenha um nível de similaridade abaixo de determinado limiar com todos os outros quadros-chave, ele é inserido no conjunto K.

Essa abordagem garante um conjunto restrito de quadros-chave, sendo eles diferentes entre si. Nota-se que o limiar de similaridade é de fundamental importância pois, se o valor for muito alto, muitos quadros-chaves serão selecionados e, caso o valor seja muito baixo, apenas o quadro médio seria selecionado.

Uma alternativa interessante ao método de Rasheed and Shah [2003] é a técnica desenvolvida por Trojahn and Goularte [2013]. Ela baseia-se na perspectiva de que, para alcançar uma seleção mais eficaz, é necessário calcular a similaridade entre os quadros de modo a garantir uma "representatividade mínima". Com isso a técnica garante que o quadro-chave candidato representa pelo menos um número significativo de quadros da tomada. O algoritmo é descrito a seguir:

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-chave, se nenhum quadro presente no conjunto de quadros-chave for similar ao quadro-chave candidato.

4. Repetem-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 quadros da 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 o valor obtido seja inferior à 95%.

```
vector<Mat> conjuntoQuadrosChave(vector<Mat> tomada) {
  vector<int> numeroSimilares;
  for(int i = 0; i < tomada.size(); i++) {
    numeroSimilares.push_back(0);
  }
  for(int i = 0; i < tomada.size(); i++) {
    for(int j = i + 1; j < tomada.size(); j++) {
      if(compararHistogramas(tomada[i],tomada[j]) >= 0.95) {
        numeroSimilares[i] = numeroSimilares[i] + 1;
        numeroSimilares[j] = numeroSimilares[j] + 1;
      }
    }
  }
  vector<Mat> quadrosChaves;
  while(true) {
    int maiorValor = numeroSimilares[0];
    int maiorIndice = 0;

    for(int i = 1; i < numeroSimilares.size(); i++) {
      if(numeroSimilares[i] > maiorIndice) {
        maiorValor = numeroSimilares[i];
        maiorIndice = i;
      }
    }
    if(maiorValor < tomada.size() * 0.2) {
      break;
    }
    Mat quadroCandidato = tomada[maiorIndice];
    numeroSimilares[maiorIndice] = 0;
    bool temp = true;
    for(int i = 0; i < quadrosChaves.size(); i++) {
      if(compararHistogramas(quadrosChaves[i],quadroCandidato) >= 0.95)
    {
        temp = false;
      }
    }
    if(temp) {
      quadrosChaves.push_back(quadroCandidato);
    }
  }
  return quadrosChaves;
}
```

Código 1.5: Função desenvolvida com a OpenCV capaz de retornar um conjunto de quadros-chave.

Além disso, 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 presentes na tomada. Assim, essa técnica permite a seleção de diversos quadros-chave (ou apenas um quadro-chave) complementares que são suficientemente dissimilares entre si e que, ao mesmo tempo, são representativos aos demais quadros da tomada.

Com a biblioteca OpenCV e as funções apresentadas na Seção 1.2.4 para a ex-

tração de um histograma (Código 1.2) e para comparar dois histogramas (Código 1.3), pode-se obter o resultado da técnica desenvolvida por Trojahn and Goularte [2013] com a função descrita no Código 1.5.

1.4. Segmentação em cenas baseada em agrupamento de tomadas

Uma das abordagens mais adotadas pela literatura para a segmentação em cenas é o agrupamento de tomadas. Tal abordagem é amplamente adotada pois, por definição, uma cena é formada por um conjunto de tomadas adjacentes, logo, agrupando tomadas consideradas "similares" obtém-se uma cena.

Considera-se que a segmentação em tomadas já tenha sido realizada, seja manualmente ou automaticamente. Em termos de implementação, embora no caso geral nem sempre as tomadas possam ser obtidas facilmente, na maioria dos casos a segmentação automática em tomadas apresenta bons índices de precisão e cobertura (do Inglês: *recall*) [Del Fabro and Böszörmenyi, 2013].

Além disso, ao invés de analisar todos os quadros das tomadas, as técnicas de segmentação em cenas geralmente utilizam quadros-chave para representar cada tomada. Neste trabalho, foi utilizada a técnica de seleção de um conjunto de quadros-chave desenvolvida por Trojahn and Goularte [2013], apresentada na Seção 1.3.2.

Após os quadros-chaves terem sido determinados para cada tomada, pode-se então proceder com o agrupamento das mesmas. A técnica aqui apresentada baseia-se no uso de descritores locais SIFT, apresentados na Seção 1.2.5. Para cada conjunto de quadros-chave, de cada tomada, extraem-se os vetores de características SIFT, formando uma representação para cada tomada do vídeo.

Para detectar as transições de cenas, analisa-se o gráfico do conjunto de valores de similaridade obtidos entre tomadas adjacentes. Nesse contexto, "vales" significam que a similaridade entre duas tomadas adjacentes mudou bruscamente, indicando uma transição.

Como medida de similaridade pode-se utilizar o processo conhecido como *matching*, onde calcula-se o número de correspondências (*matches*) entre os vetores de características de dois (conjuntos de) quadros-chave [Zhao et al., 2007].

É importante notar que a aplicação direta dessa abordagem implica em determinar quando a queda no número de *matches* é significativa. Pequenas variações, mesmo entre duas tomadas similares da mesma cena, são comuns devido ao aparecimento e desaparecimento de objetos, oclusão devido a movimentação da câmera ou dos personagens, etc. Assim, ao invés de adotar um limiar fixo, que poderia limitar a abordagem a apenas um determinado conjunto de vídeos, propõem-se utilizar um limiar dinâmico, calculado em tempo de execução de acordo com as características de cada vídeo de entrada. Tal algoritmo é descrito a seguir:

1. Identifica-se todas as reduções no número proporcional de *matches* do vídeo.

2. Remove-se as 10% maiores e menores reduções, considerados como *outliers* e removidos para não influenciar erroneamente o cálculo do limiar.

3. Calcula-se a média dos valores remanescente: esse será o valor a ser adotado como limiar para o vídeo.

Com isso, espera-se reduzir a dependência da técnica acerca do limiar: como o mesmo é determinado para cada vídeo, vídeos com diferentes características serão segmentados em cenas com limiares diferenciados.

Para exemplificar a execução do algoritmo de segmentação em cenas, considere o número de *matches* entre tomadas adjacentes apresentado na Figura 1.6.

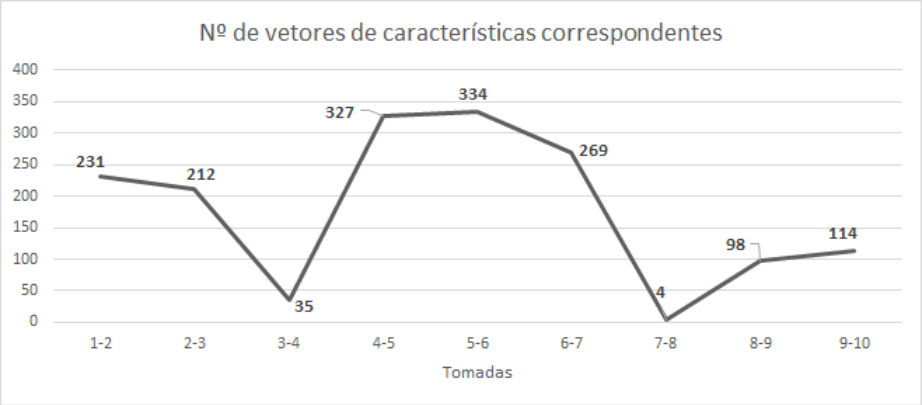


Figura 1.6: Exemplo de matches de um vídeo hipotético com 10 tomadas.

No vídeo hipotético apresentado na Figura 1.6, o algoritmo encontra 4 reduções no valores de número de *matches* entre 1-2 e 2-3, 2-3 e 3-4, 5-6 e 6-7 e, por fim, entre 6-7 e 7-8. Remove-se 10% das maiores e menores reduções encontradas (nesse caso específico, nenhuma redução é removida). Então, calcula-se a média da redução restante, no caso, igual a 131.5.

Com o valor do limiar calculado pode-se então proceder com a detecção de cenas. Caso uma redução seja seguida de um aumento, ambos significativos, uma transição de cena é detectada. Assim, no caso particular apresentado na Figura 1.6, as tomadas 3-4 seriam detectados como a única transição de cena do vídeo, já que a redução entre 2-3 e 3-4 é igual a 177 (superior ao limiar de 131.5) seguido por um aumento no número de *matches* de 292 (também superior ao limiar de 131.5).

Nota-se, por fim, que tal técnica não está limitada ao uso de descritores locais SIFT, podendo-se substitui-los facilmente por outros sem qualquer modificação na técnica. Por exemplo, histogramas poderiam ser utilizados, estimando-se a similaridade através da distância euclidiana ou intersecção de histogramas.

Em termos de implementação, como descrito na Subseção 1.2.6, a biblioteca OpenCV provê métodos para a seleção de pontos de interesse e também a extração dos descritores SIFT. Com os quadros-chaves determinados, pode-se então extrair os descritores dos mesmos. Com os descritores obtidos para cada tomada, pode-se utilizar algum algoritmo de *matching* entre dois conjuntos de descritores, como o FLANN (*Fast Library for Approximate Nearest Neighbors*) [Muja and Lowe, 2009], implementado pelo

OpenCV na classe *FLANN*. Com isso, pode-se utilizar a técnica descrita anteriormente e, dado o número de *matches* entre as tomadas, agrupá-las formando as cenas.

1.5. Segmentação em cenas baseada em multimodalidade

Um grande número de técnicas desenvolvidas para a segmentação em cenas utiliza-se apenas de informações visuais: em tais técnicas, um certo conjunto de características ou *features* são extraídas dos vídeos e as estruturas tais como tomadas e cenas são estimadas. Um exemplo de tal abordagem é a segmentação em cenas através do agrupamento de tomadas, apresentada na Seção 1.4.

Embora bastante populares, tais técnicas têm uma limitação importante: o fato de utilizarem-se apenas de informações visuais. Nota-se que, intuitivamente, uma cena é algum trecho ou segmento com alto relacionamento semântico, apresentando uma "ação" ou "evento" dentro de um contexto maior. Ao limitar a estimação ao conteúdo discernível visualmente, o relacionamento semântico pode ser incorretamente compreendido. Por exemplo, em um telejornal, um mesmo âncora pode, sucessivamente, apresentar notícias diferentes: visualmente, apenas uma notícia ou cena seria detectada, mesmo se o conteúdo de cada notícia fosse completamente diverso.

Uma alternativa para enfrentar tal limitação é utilizar uma abordagem conhecida como multimodalidade, onde o "modo" refere-se aos diferentes tipos de mídia que podem ser obtidos de um vídeo. Em geral, um vídeo, além do próprio fluxo visual, contém também uma faixa sonora e, eventualmente, algum tipo de legenda ou ainda o *closed caption*. Utilizando-se de modo combinado os dados contidos em tais mídias pode-se, então, obter uma segmentação em cenas mais relevante.

1.5.1. Fusão de Modalidades

Um importante aspecto acerca de técnicas multimodais de segmentação em cenas é a chamada "fusão de modalidades". A fusão de modalidades diz respeito às diferentes maneiras em que as informações extraídas das mídias são unidas ou fundidas.

Duas abordagens comuns para a fusão de modalidades são a fusão antecipada e a fusão tardia, apresentadas nas Seções 1.5.1.1 e a 1.5.1.2, respectivamente.

1.5.1.1. Fusão Antecipada

A fusão antecipada, do Inglês *early fusion*, é uma forma de fusão baseada na união das características antes do método de segmentação ser aplicado. Em tal abordagem, as características extraídas de cada tipo de modalidade (visual ou sonora, por exemplo) passam pelo procedimento de fusão: são colocadas em um mesmo espaço de características, representadas de modo único e combinado em vetores de características de mesmas estrutura e dimensão. Então, podem ser enviados para algum algoritmo responsável por identificar as transições de cenas. Um fluxograma que exemplifica a fusão antecipada é apresentada na Figura 1.7.

Essa estratégia de fusão também é conhecida como fusão a nível de características, já que se baseia na fusão direta das características extraídas de cada um dos modos

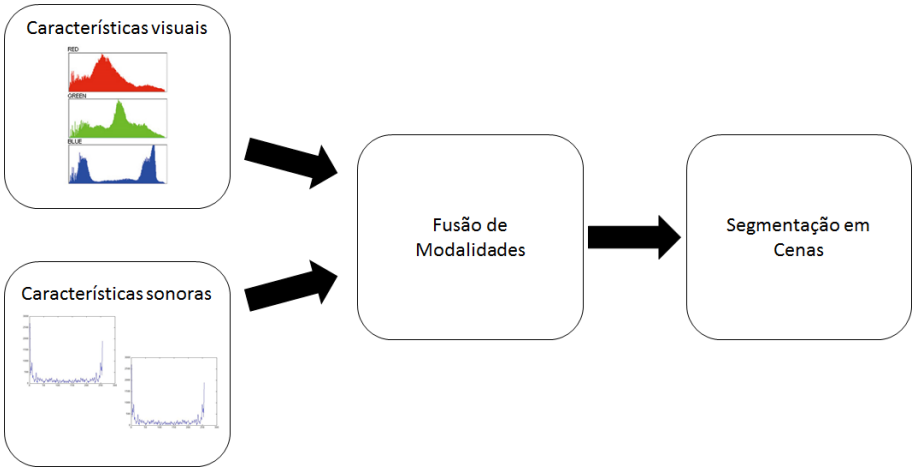


Figura 1.7: Fluxograma exemplificando a fusão antecipada de características visuais e sonoras.

utilizados, gerando também características como saída.

Um dos desafios da fusão antecipada é o como realizar a fusão de maneira adequada, já que, em geral, as características extraídas de diferentes modos possuem tamanhos, quantidade e espaços diferentes.

1.5.1.2. Fusão Tardia

A fusão tardia, do Inglês *late fusion*, é uma forma de fusão que ocorre após a execução da segmentação. Nesse tipo de fusão, as características extraídas dos diferentes modos, como áudio e vídeo, são analisadas independentemente, tendo como resultado uma possível segmentação em cenas para cada modo.

Após as possíveis segmentações terem sido calculadas, começa o procedimento de uní-las, em uma decisão fina, que combine o resultado das segmentações isoladas. A Figura 1.8 apresenta um fluxograma de exemplo para a fusão tardia.

A fusão tardia também é conhecida como fusão de decisões, já que sua entrada é um conjunto de decisões sobre as possíveis transições de cenas e a saída também é uma decisão, contendo um conjunto de transições de cenas ou similares.

Nota-se que a fusão tardia é amplamente utilizada devido à sua facilidade de aplicação e implementação. Destaca-se, porém, que a fusão tardia tende a ignorar o correlacionamento semântico entre os diferentes tipos de características, uma importante informação para a tarefa de segmentação em cenas.

1.5.2. Técnica Multimodal de Segmentação em Cenas

Neste tutorial adota-se a fusão tardia (*late fusion*) entre características visuais e sonoras. Tal escolha é justificada por: 1) a fusão tardia ser bem difundida, enquanto a fusão antecipada ainda necessita de mais investigação para obter bons resultados [Del Fabro and

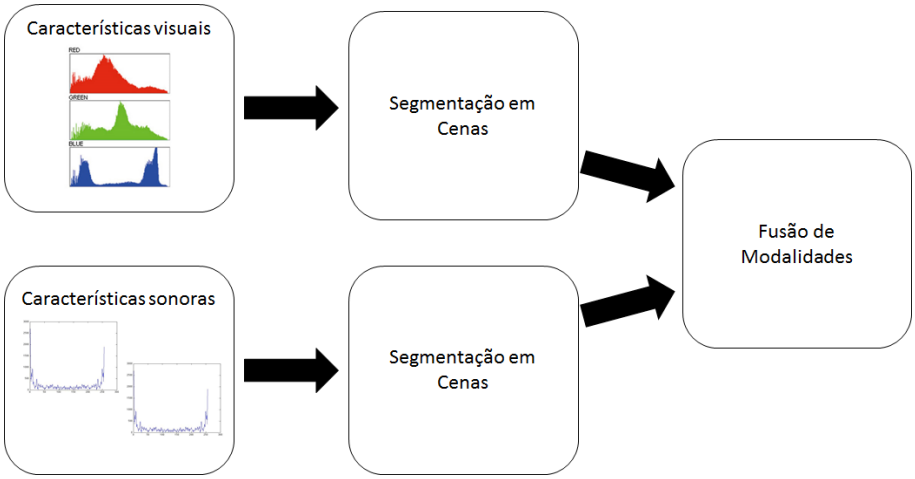


Figura 1.8: Fluxograma exemplificando a fusão tardia de características visuais e sonoras.

Böszörmenyi, 2013]; 2) características visuais e sonoras são bastante exploradas por técnicas de segmentação relacionadas, sendo, portanto, mais fácil para o leitor adaptar ou substituir as características específicas aqui utilizadas.

A segmentação em cenas através das informações visuais é obtida através do agrupamento de tomadas, implementada com o auxílio da biblioteca OpenCV, como descrito na Seção 1.4. Já a segmentação em cenas através das informações sonoras é baseada no trabalho de Lopes and Goularte [2013], extraíndo-se características por meio do descritor MFCC (*Mel-Frequency Cepstrum Coefficients*). A extração é realizada com o auxilio da biblioteca FFMPEG[12] e do *framework* MARSYAS[13].

A estratégia de fusão será baseada em um sistema de *ranking* para estimar as transições de cenas. Dadas as segmentações sonora e visual, caso alguma das duas detecte uma determinada transição, tal transição será adicionada ao conjunto final de transições. Efetivamente, isso significa que o resultado da fusão tardia será a união entre as transições das duas modalidades utilizadas. A seguir descreve-se como organizar as amostras de áudio para extração de características (1.5.2.1); como extrair as características sonoras (1.5.2.2) e a estratégia de fusão (1.5.2.3).

1.5.2.1. Quadro de Áudio

No caso visual, um descritor visual de características é aplicados à cada quadro-chave de um vídeo. O descritor de características detecta vários pontos de interesse em cada quadro-chave, e cada um desses pontos é descrito, dando origem a um vetor de características visuais. Portanto, um quadro-chave é descrito por um conjunto de vetores de características.

No caso do áudio, a situação é um pouco diferente. Um segmento de áudio pode

[12]https://www.ffmpeg.org/
[13]http://marsyas.info/

ser amostrado durante um período T arbitrário, gerando-se N amostras de áudio. O descritor de áudio atua sobre um subconjunto dessas amostras, gerando um único vetor de características, que é uma representação bastante compacta do áudio. Essa situação é representada na Figura 1.9. Nessa figura estão representados as N amostras de áudio para um determinado segmento de vídeo. Esse conjunto está dividido em três subconjuntos com M amostras de áudio cada. Para cada um desses subconjuntos, é aplicado um descritor de características sonoras, gerando um respectivo vetor de características sonoras.

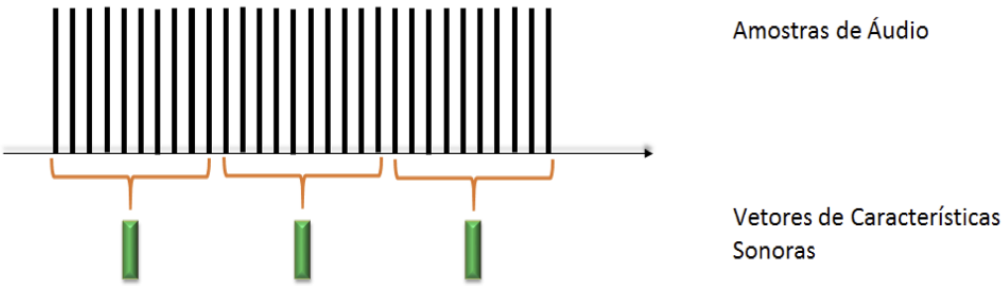


Figura 1.9: Descritores sonoro extraídos de um subconjunto de amostras de áudio.

Com o objetivo de criar uma representação, a qual denominaremos "quadro de áudio", que seja semelhante aos quadros de vídeo, pode-se dividir o intervalo entre dois quadros de vídeo em p partes, de modo que existirão p vetores de características para representar esse "quadro de áudio", que corresponde a um quadro de vídeo. Desse modo, cada quadro de áudio será representado por mais de um vetor de características, assim como o quadro de vídeo, facilitando a utilização das mesmas estratégias (visuais) para realizar a segmentação.

1.5.2.2. Extração de Características Sonoras

Utilizando uma ferramenta como a FFMPEG, é possível extrair o áudio do vídeo, gerando como saída um arquivo do tipo WAVE, com amostragem de 44100Hz, que será usado como entrada para a ferramenta MARSYAS.

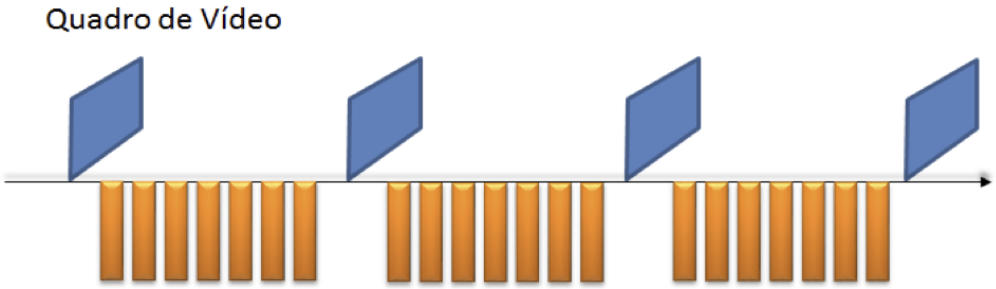


Figura 1.10: Descritores sonoro extraídos de um subconjunto de amostras de áudio.

A ferramenta MARSYAS pode então ser utilizada para computar os descritores de áudio. Esses descritores são computados usando janelas não sobrepostas com 256

amostras. Como os vídeos utilizados neste tutorial apresentam 24 quadros por segundo, existirão cerca de 1837 amostras de áudio entre cada quadro de vídeo. Como para cada uma das 256 amostras é calculado um vetor de características sonoras, existirão 7 vetores de características para representar cada quadro de áudio. Essa situação é ilustrada na Figura 1.10. Nessa figura, os retângulos azuis representam os quadros de um determinado vídeo. No intervalo entre eles, são computadas 7 amostras de áudio, representadas pelos retângulos laranja, que são utilizadas para representar um quadro de áudio.

Ao contrário da técnica visual, que fez uso de quadros-chave para reduzir a quantidade de informações a ser processada, aqui utilizou-se todos os quadros de cada tomada para a extração das características sonoras. Tal abordagem foi adotada pois segmentos de áudio tendem a ser muito diferentes, o que dificultaria a segmentação em cenas posteriormente.

1.5.2.3. Fusão Tardia Baseada em Regra

Nas Seções 1.4 e 1.5.2 foram apresentadas duas técnicas de segmentação de vídeos em cenas, sendo a primeira uma técnica que utiliza descritores de características visuais e a segunda descritores de características sonoras. Ambas as técnicas, individualmente, apresentam algumas dificuldades na detecção de cenas devido ao surgimento de falsos positivos e de falsos negativos.

Falsos positivos ocorrem quando a técnica detecta, erroneamente, um corte de cena. Falsos positivos podem ocorrer basicamente por dois motivos. O primeiro é a detecção de um corte verdadeiro, porém deslocado por algumas tomadas. Outro motivo é quando existe uma diferença considerável nas características sendo analisadas entre duas tomadas, sem que, contudo, haja realmente uma mudança de semântica entre as tomadas, não existindo assim um corte de cena verdadeiro.

Falsos negativos ocorrem, por exemplo, quando a mídia (visual ou sonora) utilizada pela técnica realmente não apresenta diferenças consideráveis entre as cenas. Sendo assim, a técnica falha em capturar a mudança da semântica, a qual nesses casos, só pode ser determinada com a análise de outras mídias presentes no vídeo.

Sendo assim, a utilização de mais de uma mídia - multimodalidade - pode ser benéfica. Neste tutorial, a fusão tardia foi a abordagem selecionada para combinar os resultados das técnicas visual e sonora. A decisão final da técnica multimodal segue as seguintes regras: 1) Se dois cortes de cenas detectados respectivamente pela técnica visual e pela sonora estão a menos de 3 tomadas de distância, eles são fundidos em um único corte, que é dado pela média entre eles; 2) Caso contrário, cada corte é mantido.

Com essa abordagem, falsos positivos da técnica visual e sonora podem ser fundidos em um único corte, o que diminui o número total de falsos positivos. Além disso, em algumas situações, essa fusão pode gerar verdadeiros positivos, quando a média entre eles fica correta. Por outro lado, verdadeiro positivos distantes (mais de 3 tomadas de distância uma da outra) provenientes das duas técnicas serão mantidos, aumentando o número em relação às técnicas individuais e melhorando a abrangência da técnica multimodal. Contudo, falsos positivos distantes também serão mantidos, o que pode afetar a precisão

da técnica.

1.6. Avaliação

Após implementado uma técnica de segmentação de cenas, é imprescindível realizar algum tipo de avaliação com o intuito de estipular a eficácia. Outro fator importante é que, após a avaliação ter sido realizada, pode-se comparar a técnica desenvolvida com outras técnicas.

A avaliação de técnicas de segmentação de vídeo em cenas requer dois dados básicos: um vídeo a ser segmentado e, também, uma base confiável (do Inglês *ground truth*). O vídeo é aquele que será submetido à técnica desenvolvida e cujos resultados serão avaliados. Por sua vez, a base confiável contém uma descrição completa do que seria esperado que o algoritmo retornasse.

No caso particular da segmentação de vídeo em cenas, o vídeo deve ter uma duração relevante com a presença de diversas transições de cenas. Já a base confiável pode ser obtida de diversas maneiras, sendo a manual preferível, mas bastante laboriosa.

Após processado o vídeo pela técnica desenvolvida, compara-se o resultado obtido com a base confiável, obtendo as seguintes possibilidades:

- **Verdadeiro positivo (VP):** Quando a técnica e a base confiável afirmam que há uma transição de cena.
- **Falso positivo (FP):** Quando a técnica afirma que há uma transição, sendo que tal transição não consta na base confiável.
- **Falso negativo (FN):** Quando a técnica afirma que não há uma transição, sendo que tal transição consta na base confiável.

Uma técnica deveria apresentar um número bastante alto de VP, mantendo os valores de FP e FN baixos. Idealmente, uma técnica apresentaria apenas VP, sem nenhum FP ou FN. Nota-se ainda que pode ser preferível a presença de falsos positivos sobre falsos negativos: pode-se adotar técnicas posteriores com o intuito de detectar transições errôneas e corrigi-las.

Com os três valores (VP, FP e FN) estimados, pode-se utilizar alguma métrica para descrever, sucintamente, o comportamento da técnica frente a determinado vídeo. Diversas métricas são descritas na literatura, tais como as curvas ROC (do Inglês *Receiver Operating Characteristics*) e o Erro Quadrático Médio (do Inglês *Mean Squared Error* - MSE) [Mood et al., 1974].

Três medidas amplamente utilizadas são a precisão (*Precision*), cobertura ou revocação (*Recall*) e o F-*measure* [Rijsbergen, 1979; Hua et al., 2002]. Quanto à F-*measure*, normalmente utiliza-se a variação conhecida como F_1, onde nem a precisão e nem a cobertura são priorizadas. A precisão avalia a proporção de acertos na classificação sobre o total de resultados. A cobertura, por sua vez, avalia a proporção de resultados corretos obtidos em comparação com o número total de resultados corretos presentes na base confiável e, por fim, a F_1 sintetiza os resultados de precisão e cobertura em um único valor. Formalmente, a precisão P é definida como:

$$P = \frac{vp}{vp + fp} \tag{3}$$

A cobertura C, por sua vez, é definida como:

$$C = \frac{vp}{vp + fn} \tag{4}$$

E, finalmente, a F_1 é definida como:

$$F_1 = 2 \cdot \frac{P \cdot C}{P + C} \tag{5}$$

Onde vp é verdadeiro positivo, fp é falso positivo e fn é falso negativo.

A escolha pelas métricas de precisão, cobertura e F_1 se devem ao fato de que diversos trabalhos na área também as adotarem como medida de avaliação de seus métodos, permitindo, assim, a comparação entre a técnica desenvolvida e outras presentes na literatura. A medida F_1, em especial, foi adotada pois, além de sintetizar os valores de precisão e cobertura em um único valor, tende a ser uma medida mais conservadora que uma média aritmética simples, por exemplo.

Como exemplo, seja um trecho de vídeo contendo dez transições de tomadas e uma transição de cena. Seja ainda uma técnica de segmentação em cenas que obtenha, para esse trecho, dez transições de cenas, uma para cada transição de tomada. Assim, essa técnica obteve 100% de cobertura, já que todas as transições de cenas foram encontradas (uma transição). Sua precisão, por outro lado, é bastante baixa: 10% (apenas um verdadeiro positivo e nove falsos positivos). Caso fosse realizada uma média simples entre a precisão e cobertura, a técnica obteria o valor de 55%, "maquiando" a baixa precisão obtida pela técnica.

Por outro lado, nas mesmas circunstâncias, a medida F_1 da técnica seria de apenas 18.18%, um resultado bastante inferior à média simples entre a precisão e cobertura. Desta forma, caso o algoritmo apresente resultados insatisfatórios em algum quesito (precisão e/ou cobertura), o valor F_1 obtido tenderá a se aproximar do valor mais baixo, caracterizando a medida F_1 como uma média ponderada.

Durante o mini-curso serão apresentadas avaliações práticas das técnicas apresentadas neste tutorial utilizando como medidas precisão, cobertura e F_1.

Referências

Baeza-Yates, R. and Ribeiro-Neto, B. *Modern Information Retrieval*. Addison Wesley, 1999.

Barrios, V. M. G., Mödritscher, F., and Gütl, C. Personalisation versus adaptation? a user-centred model approach and its application. In *Proceedings of I-KNOW*, pages 120–127, 2005. URL `http://i-know.tugraz.at/wp-content/uploads/2011/12/Barrios_paper.pdf`.

Bay, H., Tuytelaars, T., and Van Gool, L. Surf: Speeded up robust features. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*, ECCV'06, pages 404–417, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-33832-2, 978-3-540-33832-1. doi: 10.1007/11744023_32. URL http://dx.doi.org/10.1007/11744023_32.

Besiris, D., Laskaris, N., Fotopoulou, F., and Economou, G. Key frame extraction in video sequences: a vantage points approach. In *9th IEEE Workshop on Multimedia Signal Processing*, MMSP '07, pages 434–437, October 2007. URL http://dx.doi.org/10.1109/MMSP.2007.4412909.

Blanken, H. M., Blok, H. E., Feng, L., and de Vries, A. P. *Multimedia Retrieval*. Springer Berlin Heidelberg, 2007.

Bouyakoub, F. and Belkhir, A. Adams: An adaptation multimedia system for heterogeneous environments. In *New Technologies, Mobility and Security*, pages 1–5, Nov 2008. doi: 10.1109/NTMS.2008.ECP.15.

Chapman, N. P. and Chapman, J. *Digital Multimedia*. John Wiley & Sons, Inc., New York, NY, USA, 3 edition, March 2009. ISBN 0471983861.

Chasanis, V., Kalogeratos, A., and Likas, A. Movie segmentation into scenes and chapters using locally weighted bag of visual words. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, CIVR '09, pages 35:1–35:7, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-480-5. URL http://doi.acm.org/10.1145/1646396.1646439.

Chen, H. and Li, C. A practical method for video scene segmentation. In *3rd IEEE International Conference on Computer Science and Information Technology*, volume 9 of *ICCSIT '10*, pages 153–156, July 2010. URL http://dx.doi.org/10.1109/ICCSIT.2010.5564930.

Coimbra, D. Segmentação de cenas em telejornais: uma abordagem multimodal. Dissertação (mestrado em ciência da computação e matemática computacional), Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, São Paulo, Brasil, 2011. URL http://www.teses.usp.br/teses/disponiveis/55/55134/tde-28062011-103714.

Del Fabro, M. and Böszörmenyi, L. State-of-the-art and future challenges in video scene detection: a survey. *Multimedia Systems*, 19(5):427–454, 2013. ISSN 0942-4962. doi: 10.1007/s00530-013-0306-4. URL http://dx.doi.org/10.1007/s00530-013-0306-4.

Gonzalez, R. C. and Woods, R. E. *Digital Image Processing*. Prentice Hall, 3 edition, 2007.

Gu, Z., Mei, T., Hua, X.-S., Wu, X., and Li, S. Ems: Energy minimization based video scene segmentation. In *IEEE International Conference on Multimedia and Expo*, volume 1 of *ICME '07*, pages 520–523, July 2007. URL http://dx.doi.org/10.1109/ICME.2007.4284701.

Hu, W., Xie, N., Li, L., Zeng, X., and Maybank, S. A survey on visual content-based video indexing and retrieval. *IEEE Transactions on Systems, Man, and Cybernetics*, 41(6):797–819, November 2011. ISSN 1094-6977. URL http://dx.doi.org/10.1109/TSMCC.2011.2109710.

Hua, X.-S., Zhang, D., Li, M., and Zhang, H.-J. Performance evaluation protocol for video scene detection algorithms. In *Workshop on Multimedia Information Retrieval, in conjunction with 10th ACM Multimedia*, 2002.

Huang, J., Liu, Z., and Wang, Y. Integration of audio and visual information for content-based video segmentation. In *Proceedings of the International Conference on Image Processing*, volume 3 of *ICIP '98*, pages 526–529, 1998. URL http://dx.doi.org/10.1109/ICIP.1998.727252.

Jacobs, C. E., Finkelstein, A., and Salesin, D. H. Fast multiresolution image querying. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, volume 1 of *SIGGRAPH '95*, pages 277–286, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4. URL http://dx.doi.org/10.1145/218380.218454.

Koprinska, I. and Carrato, S. Temporal video segmentation: A survey. *Signal Processing: Image Communication*, 16(5):477–500, January 2001. ISSN 0923-5965. URL http://dx.doi.org/10.1016/S0923-5965(00)00011-4.

Li, Y., Ming, W., and Kuo, C.-C. Semantic video content abstraction based on multiple cues. In *IEEE International Conference on Multimedia and Expo*, ICME '01, pages 623–626, August 2001. doi: 10.1109/ICME.2001.1237797.

Liu, T., Zhang, X., Feng, J., and Lo, K. Shot reconstruction degree: a novel criterion for key frame selection. *Pattern Recognition Letters*, 25(12):1451–1457, September 2004. URL http://dx.doi.org/10.1016/j.patrec.2004.05.020.

Lopes, B. L. and Goularte, R. Multimodal late fusion bag of features applied to scene detection. In *Proceedings of the 19th Brazilian Symposium on Multimedia and the Web*, WebMedia '13, pages 15–22, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2559-2. doi: 10.1145/2526188.2526202. URL http://doi.acm.org/10.1145/2526188.2526202.

Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

Lowe, D. Object recognition from local scale-invariant features. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2 of *ICCV '99*, pages 1150–1157, 1999. doi: 10.1109/ICCV.1999.790410.

Lu, Y., Sebe, N., Hytnen, R., and Tian, Q. Personalization in multimedia retrieval: A survey. *Multimedia Tools and Applications*, 51(1):247–277, January 2011. ISSN 1380-7501. URL http://dx.doi.org/10.1007/s11042-010-0621-0.

Lum, W. Y. and Lau, F. C. M. A context-aware decision engine for content adaptation. *IEEE Pervasive Computing*, 1(3):41–49, July 2002. ISSN 1536-1268. doi: 10.1109/MPRV.2002.1037721. URL http://dx.doi.org/10.1109/MPRV.2002.1037721.

Magalhães, J. a. and Pereira, F. Using MPEG standards for multimedia customization. *Signal Processing: Image Communication*, 19(5):437–456, May 2004. ISSN 09235965. URL http://dx.doi.org/10.1016/j.image.2004.02.004.

Marques, O. *Practical Image and Video Processing Using MATLAB*. Wiley, IEEE Press, September 2011. ISBN 978-0-470-04815-3.

Marques Filho, O. and Vieira Neto, H. *Processamento Digital de Imagens*. Brasport, Rio de Janeiro, RJ, Brasil, 1 edition, 1999.

Mohan, R., Smith, J., and Li, C.-S. Adapting multimedia internet content for universal access. *Multimedia, IEEE Transactions on*, 1(1):104–114, Mar 1999. ISSN 1520-9210. doi: 10.1109/6046.748175.

Mood, A. M., Graybill, F. A., and Boes, D. C. *Introduction to the Theory of Statistics*. McGraw Hill, New York, NY, USA, 3 edition, June 1974.

Muja, M. and Lowe, D. G. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

Mukherjee, D., Das, S., and Saha, S. Key frame estimation in video using randomness measure of feature point pattern. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(5):612–620, May 2007. ISSN 1051-8215. URL http://dx.doi.org/10.1109/TCSVT.2007.895353.

Rasheed, Z. and Shah, M. Scene detection in hollywood movies and tv shows. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2 of *CVPR '03*, pages 343–348, June 2003. URL http://dx.doi.org/10.1109/CVPR.2003.1211489.

Rasheed, Z. and Shah, M. Detection and representation of scenes in videos. *IEEE Transactions on Multimedia*, 7(6):1097–1105, December 2005. ISSN 1520-9210. URL http://dx.doi.org/10.1109/TMM.2005.858392.

Rathod, G. I. and Nikam, D. A. An algorithm for shot boundary detection and key frame extraction using histogram difference. *International Journal of Emerging Technology and Advanced Engineering*, 3(8):155–163, August 2013. URL http://ijetae.com/Volume3Issue8.html.

Richardson, I. E. *The H.264 Advanced Video Compression Standard*. John Wiley & Sons Ltd, Chichester, 2 edition, 2010.

Rijsbergen, C. G. *Information Retrieval*. Butterworths, London, 2 edition, 1979.

Salomon, D., Motta, G., and Bryant, D. *Data Compression: The Complete Reference*. Springer, New York, NY, USA, 4 edition, 2006. ISBN 1846286026.

Smeaton, A. F. Techniques used and open challenges to the analysis, indexing and retrieval of digital video. *Information Systems*, 32(4):545–559, June 2007. ISSN 0306-4379. doi: 10.1016/j.is.2006.09.001. URL http://dx.doi.org/10.1016/j.is.2006.09.001.

Souza, T. T. and Goularte, R. Video shot representation based on histograms. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 961–966, New York, NY, USA, 2013. ACM. doi: 10.1145/2480362.2480547. URL http://doi.acm.org/10.1145/2480362.2480547.

Toffler, A. *Future Shock*. Bantam, 1 edition, 1984.

Torres, R. D. S. and Falcão, A. X. Content-based image retrieval: Theory and applications. *Revista de Informática Teórica e Aplicada*, 13:161–185, 2006.

Trojahn, T. H. and Goularte, R. Video scene segmentation by improved visual shot coherence. In *Proceedings of the 19th Brazilian Symposium on Multimedia and the Web*, WebMedia '13, pages 23–30, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2559-2. doi: 10.1145/2526188.2526206. URL http://doi.acm.org/10.1145/2526188.2526206.

Ullman, S., Vidal-Naquet, M., and Sali, E. Visual features of intermediate complexity and their use in classification. *Nature neuroscience*, 5(7):682–687, July 2002. ISSN 1097-6256. URL http://dx.doi.org/10.1038/nn870.

Zhai, Y. and Shah, M. Video scene segmentation using markov chain monte carlo. *IEEE Transactions on Multimedia*, 8(4):686–697, August 2006. ISSN 1520-9210. URL http://dx.doi.org/10.1109/TMM.2006.876299.

Zhang, L., Zhu, Z., and Zhao, Y. Robust commercial detection system. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, volume 1 of *ICMC '07*, pages 587–590, July 2007. URL http://dx.doi.org/10.1109/ICME.2007.4284718.

Zhao, W., Ngo, C., Tan, H., and Wu, X. Near-duplicate keyframe identification with interest point matching and pattern learning. *IEEE Transactions on Multimedia*, 9(5):1037–1048, 2007. ISSN 1520-9210. URL http://dx.doi.org/10.1109/TMM.2007.898928.

Zhu, S. and Liu, Y. Scene segmentation and semantic representation for high-level retrieval. *Signal Processing Letters, IEEE*, 15:713–716, 2008. ISSN 1070-9908. doi: 10.1109/LSP.2008.2002718.