Capítulo 5

Recuperação de Informação Multimídia em Big Data Utilizando OpenCV Python

Rudinei Goularte[1], Tiago Henrique Trojahn[2] e Rodrigo Mitsuo Kishi[3]

[1] Universidade de São Paulo. Instituto de Ciências Matemáticas e de Computação.

[2] Instituto Federal de São Paulo

[3] Universidade Federal de Mato Grosso do Sul.

rudinei@icmc.usp.br, tiagotrojahn@ifsp.edu.br, rodrigo.kishi@ufms.br

Abstract

The popularization of systems, applications and devices to produce, view and share multimedia, saw the need to treat a large volume of data arise. In related areas (such as Multimedia Big Data, Data Science and Multimedia Information Retrieval) a key step is commonly referred as Multimedia Indexing or Multimedia Big Data Analysis, where the aim is to represent multimedia content into smaller, more manageable units, allowing the extraction of data features and information essential to the proper performance of the associated services. This mini-course discusses current tools and techniques for indexing, extracting and processing of multimodal multimedia content. The techniques are exemplified in Python OpenCV over different content (like images, audio, text and video), leading to the interest of services like Netflix, Google and YouTube on this subject, attracting the interest of researchers and developers.

Resumo

A popularização de sistemas, aplicativos e dispositivos para produzir, exibir e compartilhar conteúdo multimídia fez surgir a necessidade de tratar um grande volume de dados. Nas áreas relacionadas (como Multimedia Big Data, Ciência de Dados e Recuperação de Informação Multimídia) um pré-requisito chave é comumente conhecido como Indexação Multimídia (ou Análise Multimídia em Big Data), onde o objetivo é representar o conteúdo em unidades menores e mais gerenciáveis, permitindo a extração de features dos dados e informações essenciais para o bom funcionamento dos serviços associados. Este minicurso aborda ferramentas e técnicas atuais para indexação, extração e processamento de conteúdo multimídia multimodal. As técnicas

são exemplificadas em OpenCV Python sobre diferentes conteúdos (imagens, áudio, texto e vídeo), levando ao interesse de serviços como Netflix, Google e YouTube nesse assunto, motivando pesquisadores e desenvolvedores.

5.1. Introdução

Nos últimos anos houve um aumento na quantidade de conteúdo multimídia disponível para acesso [Lu et al., 2011; Pouyanfar et al., 2018]. 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 e Belkhir, 2008; Baraldi et al., 2017; Pouyanfar et al., 2018], 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.

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: recuperar conteúdo de interesse em meio ao imenso volume de informações disponíveis [Hu et al., 2011; Toffler, 1984].

Tal problema vem sendo tratado por importantes áreas da Computação, como Recuperação de Informação [Baeza-Yates e Ribeiro-Neto, 2008], Recuperação de Informação Multimídia [Blanken et al., 2010] Recomendação e Personalização e Adaptação Multimídia [Lu et al., 2011; Pouyanfar et al., 2018] e, mais recentemente, por Ciência de Dados e *Multimedia Big Data Analisys* [Pouyanfar et al., 2018]. Um fator comum é que os sistemas desenvolvidos nessas áreas necessitam que dados (*features)* 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, comumente chamado de Indexação Multimídia, é complexo e envolve alto custo computacional [Blanken et al., 2010].

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 [Iwan e Thom, 2017]. Contudo, vídeo também possui características sonoras e textuais, que, juntamente com as visuais podem representar mais fielmente a informação sendo tratada. Essa faceta inerentemente multimodal de dados multimídia faz surgir a necessidade de se desenvolver métodos também multimodais de indexação, de modo a se obter uma única

[1] www.youtube.com

[2] www.netflix.com

[3] www.lastfm.com

representação (um único vetor de características) contendo informações de todas as modalidades (visual, sonora e textual) [Xie et al., 2017].

Os métodos consagrados na literatura para indexação multimídia são majoritariamente unimodais, permitindo extrair e representar uma única característica em um vetor de características. Isso é feito por meio de extratores de características bem conhecidos (como SIFT para imagens e MFCC para áudio), utilizando medidas de dissimilaridade [Blanken et al., 2010] entre os vetores de características resultantes para aferir pertinência, desigualdade e outras operações sobre os objetos/dados. Os métodos mais recentes, multimodais, empregam conceitos mais complexos, onde tanto a dissimilaridade obtida analisando-se os dados (*features*) brutos quanto a semântica latente entre os objetos analisados contribuem para o resultado final, melhorando a eficácia das tarefas e serviços [Pouyanfar et al., 2018]. Entre tais conceitos destacam-se a Fusão de Modalidades e os Dicionários de Palavras Multimodais (*Bag of Multimodal Words*), que vêm auxiliando a obter melhores resultados em diversas áreas relacionadas a Recuperação de Informação Multimídia (RIM) [Del Fabro e Boszormenyi, 2013; Pouyanfar et al., 2018].

5.2. OpenCV e Conceitos Básicos

Nesta seção, são discutidos ambos, a biblioteca *OpenCV*, utilizada como suporte ao minicurso, assim como alguns dos principais conceitos relacionados a recuperação de informação multimídia. A Seção 5.2.1 descreve a biblioteca OpenCV, composta de diferentes funcionalidades que auxiliam o desenvolvimento de aplicações voltadas a visão computacional e também utilizadas neste trabalho. Já a Seção 5.2.2 apresenta os principais conceitos relacionados.

5.2.1. A Biblioteca OpenCV

A biblioteca OpenCV[4] 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 mantida atualmente pela *OpenCV Foundation*.

OpenCV é de código-aberto (licença BSD) com suporte 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[5] e C#[6]. Além da variedade de linguagens de programação, a biblioteca também pode ser utilizada em diversas plataformas e sistemas operacionais, tais como 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 ao grande número de desenvolvedores e amplo suporte da comunidade tornam a OpenCV uma ferramenta bastante útil em diversos trabalhos na área.

[4] https://opencv.org/
[5] https://github.com/ruby-opencv/ruby-opencv
[6] http://www.emgu.com/wiki/index.php/Main_Page

Nesse sentido, a documentação da biblioteca conta com a descrição detalhada das classes e funções dos diferentes módulos que compõem a OpenCV. Além disso, está disponível 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.

Neste texto utiliza-se a versão estável 3.4 da OpenCV. Para sua instalação recomenda-se instalar antes o utilitário pip do Python. Exemplo: em ambiente Mac OS X:

>> easy_install pip

>> pip install –r setup.txt

Onde *setup.txt* é um arquivo texto contendo os nomes das bibliotecas a serem instaladas (uma por linha e sem vírgula), incluindo a OpenCV: numpy, scipy, opencv-contrib-python==3.4.0.12, python_speech_features, moviepy.

A Listagem 5.1 apresenta um exemplo simples de uso da biblioteca OpenCV via um script Python. Na linha 1 tem-se a importação da biblioteca.

```
1  import cv2
2
3  # Abre e carrega a imagem de entrada
4  imagem = cv2.imread("imagem.jpg")
5  # Converte a imagem para a escala de cinza
6  escala_cinza = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)
7  # Espelha a imagem (rotaciona) verticalmente
8  rotacionada = cv2.flip(escala_cinza, +1)
9  # Salva a imagem rotacionada como 'saida.jpg'
10 cv2.imwrite("saida.jpg", rotacionada)
```

Listagem 5.1. Exemplo de código Python utilizando OpenCV. Arquivo exemplo1.py

Na Listagem 5.1, os métodos *imread*, *cvtColor*, *flip* e *imwrite* pertencem à biblioteca OpenCV e realizam, pela ordem, a leitura de uma imagem para a memória, a conversão da imagem para escala de cinzas, o espelhamento da imagem e a escrita da imagem processada em arquivo. A documentação dos métodos assim como tutoriais da OpenCV podem ser encontrados em https://opencv-python-tutroals.readthedocs.io/en/latest/**.** Para executar o código da Listagem 5.1 basta salvar o arquivo texto correspondente (exemplo1.py) em um diretório e, via linha de comando/shell, digitar:

>> python exemplo1.py

O resultado será um arquivo, chamado *saída.jpg*. Um exemplo de imagem de entrada e correspondente saída para o código da Listagem 5.1 pode ser visualizado na Figura 5.1.

Figura 5.1. Imagem original (esquerda) e processada (direita).

5.2.2. Conceitos Relacionados

O presente minicurso se insere na área conhecida como Sistemas Multimídia, ou simplesmente Multimídia. O termo **Multimídia**, em Computação, pode ser definido como "a utilização simultânea de dois ou mais tipos diferentes de mídia, integradas e entregues por computador" [Havaldar e Medioni, 2009]. **Mídia**, nesse caso, pode ser entendida como "o meio pelo qual a informação é representada, apresentada, transmitida ou armazenada por sistemas computacionais" [Mandal, 2002].

A informação apresentada pelo computador é percebida pelos seres humanos quando estes utilizam seus canais sensoriais. Assim, uma informação percebida via o canal auditivo é dita ser da modalidade aural, do mesmo modo que uma informação percebida pelo canal visual é dita ser da modalidade visual. Logo, **modalidade** se refere à relação entre a mídia utilizada para comunicar a informação e o canal sensorial adequado para receber tal informação. **Multimodalidade**, por consequência, é a utilização conjunta de duas ou mais modalidades diferentes que, em Computação, tem como objetivo melhorar o desempenho em tarefas computacionais relacionadas ao processamento, representação ou recuperação de informações.

Uma área recente da Computação que faz uso intenso de multimodalidade é a **Recuperação de Informação Multimídia**. O objetivo desta área é recuperar dados textuais, sonoros, de imagens e de vídeo relacionados aos interesses do usuário e classificá-los de acordo com o grau de relevância relacionado à consulta do usuário [Blanken et al., 2010]. Atualmente o problema fundamental está em como habilitar ou melhorar a recuperação multimídia utilizando métodos baseados em conteúdo (não baseados em texto). Métodos baseados em conteúdo são necessários quando anotações textuais são inexistentes ou incompletas. Além disso, métodos baseados em conteúdo potencialmente podem melhorar a acurácia da recuperação quando anotações textuais estão presentes fornecendo entendimento adicional sobre as coleções de mídias [Blanken et al., 2010; Pouyanfar et al., 2018].

5.3. Extração de características

Um sistema de recuperação de informação multimídia deve ser capaz de indexar dados de sua coleção pelos seus respectivos conteúdos. Graças ao grande volume de dados e à complexidade das informações, a indexação é realizada através de alguma propriedade discriminatória, chamada **característica** [Atrey *et al.*, 2010], termo advindo da língua

inglesa - ***feature***. As características comumente se dividem três categorias, de acordo com seu nível de abstração [Martinet e Sayad, 2012]: baixo, médio e alto nível semântico. Características de baixo nível semântico são leituras numéricas computadas diretamente dos dados brutos, sem que haja o emprego de conhecimento externo, aprendizado de máquina ou análise estatística de outros documentos. Exemplos de características de baixo nível semântico são informações de cor de uma imagem ou de intensidade de uma amostra de áudio.

Segundo Martinet e Sayad (2012), as características de médio nível semântico são definidas como leituras numéricas ou simbólicas produzidas por uma etapa de refinamento semântico em características de baixo nível semântico. Um exemplo de característica de médio nível semântico é o *Bag-of-Features* (BoF), que é uma extensão do modelo *Bag-of-Words* (BoW) da área de Processamento de Linguagem Natural. Já as características de alto nível semântico são informações sobre o significado do conteúdo, diretamente interpretáveis por humanos. Elas se apresentam, costumeiramente, através de texto descrevendo o conteúdo. Uma imagem, por exemplo, pode ter a ela associados os rótulos "carro" e "moto", indicando que tal imagem ilustra um carro e uma moto. Estes rótulos podem também representar conceitos não materiais como, por exemplo, "raiva" ou "tranquilidade". As características de alto nível semântico podem ser geradas por anotação manual ou automaticamente por análise computacional do conteúdo (a partir de características de baixo/médio nível semântico), sujeita à lacuna semântica [Smeulders et al., 2000].

As características de baixo e de médio nível semântico extraídas de conteúdo multimídia são codificadas em vetores numéricos n-dimensionais, chamados **vetores de características** [Martinet e Sayad, 2012]. Os vetores de características, representações obtidas através de processamento e análise computacional, servem para realizar comparação entre vídeos e se caracterizam como representações compactas e relevantes do conteúdo. O processo de indexar automaticamente uma base de vídeos por suas características é chamado **extração de características** [Blanken et al., 2010]. A uma ferramenta que computa um vetor de características de um canal de informação é dado o nome de detector ou extrator de características [Tuytelaars e Mikolajczyk, 2008].

A extração de características de alto nível semântico está fora do escopo deste texto. Assume-se que tais características, na forma de textos descritivos de conteúdo ou rótulos de conteúdo, podem estar presentes e podem ser utilizados como entrada para um sistema de RIM.

5.3.1. Características de Baixo Nível Semântico

Características Visuais

Características visuais de baixo nível semântico são extraídas de imagens ou de quadros de vídeo, que em última análise são também imagens. Características de imagens estáticas se dividem em globais e locais, de acordo com seu escopo de representação. As características globais se referem à imagem inteira e normalmente são compactas. Uma das mais populares características globais de comparação de conteúdo visual é o histograma global de cor [Stockman e Shapiro, 2001].

Um histograma de cor de uma imagem é uma contagem dos pixels, por cor, de todos os pixels dessa imagem. Ele pode ser implementado como um vetor

unidimensional onde cada índice corresponde a um intervalo de cores e o conteúdo de cada célula corresponde ao total de pixels da imagem que se encaixam no intervalo relacionado àquele índice. A obtenção de histogramas de cor tem baixo custo computacional e eles são relativamente invariantes à diferenças de translação, rotação centralizada no eixo de captura, pequenas rotações não centralizadas no eixo de captura, mudanças de escala e oclusão parcial. Como os histogramas de cor desconsideram informação espacial dos pixels, eles falham quando precisam diferenciar imagens muito diferentes com cores semelhantes. Além da indiferença à informação espacial, histogramas de cor falham ao comparar duas imagens com conteúdo parecido, mas que foram capturadas em condições de iluminação muito diferentes [Lu et al. 2001].

Uma característica local, diferentemente de uma global, se refere a um subconjunto de uma imagem. Este subconjunto pode ser um ponto, região ou mesmo um segmento de aresta contendo um padrão que o diferencia de sua vizinhança [Tuytelaars e Mikolajczyk, 2008]. Espera-se, com a extração de características locais, representar estruturas distintivas de uma imagem. De acordo com Grauman e Leibe (2011), um bom extrator de características locais deve garantir repetibilidade e precisão, extraindo sempre as mesmas características de duas imagens distintas que exibem o mesmo objeto e, ao mesmo tempo, oferecer distintividade, a habilidade de distinguir dois objetos diferentes. Geralmente tais métodos operam seguindo uma série de passos: detectando regiões distintivas na imagem, chamadas de **pontos de interesse**, normalmente relativas a arestas, cantos e junções T; realizando operações com os valores ao redor de tal ponto para torná-lo robusto a variações na escala, rotação e iluminação; transformando esses valores em um vetor de características.

Existem hoje diversos extratores locais populares, como SURF (*Speeded up Robust Features*), SIFT (*Scale-Invariant Feature Transform*), FAST (*Features from Accelerated Segment Test*), BRIEF (*Binary Robust Independent Elementary Features*) e ORB (*Oriented FAST and Rotated BRIEF*). Os dois primeiros estão patenteados, sendo seu uso livre para propósitos acadêmicos. O último é uma alternativa aberta suportada pela biblioteca OpenCV. Detalhes sobre esses e outros métodos podem ser encontrados no trabalho de Leng et al. (2019). Neste texto, a título de ilustração, iremos abordar em um pouco mais de detalhes o método SIFT.

O método de extração de características locais SIFT, proposto por Lowe (2004), é um dos mais conhecidos atualmente. O método SIFT foi apresentado como uma combinação do detector de regiões de interesse por diferenças de gaussianas, do inglês *Difference-of-Gaussians* (DoG), e de uma técnica para produzir um vetor de características. As características extraídas pelo método SIFT são invariantes à escala e à rotação.

O detector de regiões de interesse baseia-se no fato de que pontos de máximo de laplacianas de gaussianas, do inglês *Laplacian-of-Gaussians* (LoG), são equivalentes à arestas e/ou cantos de uma imagem. O cálculo de LoG em uma imagem é computacionalmente custoso e, por isso, o SIFT usa a DoG como uma aproximação para a LoG. A DoG é basicamente uma subtração envolvendo uma imagem e uma versão desfocada dela mesma. Para cada imagem, o SIFT computa três cópias redimensionadas sucessivamente, onde cada cópia tem dimensões correspondentes à um quarto da cópia anterior. As diferentes escalas são chamadas oitavas e sua obtenção compõe a estratégia responsável pela invariância à escala. Para cada oitava são

computadas quatro cópias suas com desfoque gaussiano, aumentado sequencialmente em cada cópia. A DoG é, então, computada entre as imagens de cada oitava e a imagem seguinte com maior desfoque da mesma oitava. Os pontos aproximados (pontos-chave) de máximo e mínimo são encontrados, realizando uma busca em vizinhanças 3×3×3 na matriz tridimensional resultante da combinação das matrizes bidimensionais DoG.

A quantidade de pontos chave produzida é grande e muitos deles não correspondem a regiões interessantes. Por este motivo, são realizadas duas etapas de filtragem, rejeitando pontos com baixo contraste e depois aqueles com alta resposta de aresta em uma única direção usando determinante do hessiano, do inglês *Determinant of Hessian* (DoH). Depois das etapas de filtragem, inicia-se a construção do vetor de características, ao se determinar a orientação mais proeminente dos gradientes de direção em torno de cada ponto-chave. Essa orientação é atribuída ao ponto chave e todo cálculo posterior será relativo a ele, permitindo a invariância à rotação. A orientação é obtida através de um histograma de 36 posições, onde cada posição cobre 10 graus e tem como valor a soma das magnitudes dos gradientes. O maior pico, assim como os outros picos com valor de pelo menos 80% dele, são transformados em pontos chave individuais onde a orientação é dada pelo valor da posição daquele pico no histograma.

O último passo do método SIFT consiste em gerar uma assinatura para cada ponto chave, que é o próprio vetor de características. O vetor de características é obtido pela inspeção de uma janela 16×16 em volta do ponto chave. Essa janela é dividida em dezesseis janelas 4×4 e cada janela produz um histograma de 8 posições de magnitudes de gradientes, onde cada posição do histograma cobre 45 graus. O conteúdo adicionado a cada posição do histograma também é proporcional à distancia do ponto chave. Fazendo isso para cada uma das dezesseis regiões, um vetor de características de 16×8 = 128 bytes é produzido. Esse vetor de características é normalizado, dele é subtraída a orientação do ponto chave e por fim é limiarizado para 0:2. A subtração da orientação do ponto chave serve para prover independência à rotação. Já a limiarização provê independência à diferenças de iluminação. Um exemplo de extração de pontos chave pode ser visto na Figura 5.2. A imagem à esquerda corresponde à imagem original. A imagem à direita corresponde à imagem processada, indicando-se por meio de círculos coloridos os pontos-chave (*keypoints*) detectados. A literatura da área refere-se a esse processo como detecção de pontos-chave. Embora a documentação da OpenCV refira-se ao processo como extração de pontos-chave, são o mesmo processo.

Figura 5.2. Exemplo de Extração de Pontos-chave.

```
1  import numpy as np
2  import cv2
3
4  # Carrega a imagem de entrada
5  imagem = cv2.imread('pisa.jpg')
6
7  # Converte a imagem para a escala de cinza
8  escala_cinza = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)
9
10 # Cria o extrator de caracteristicas SIFT
11 sift = cv2.xfeatures2d.SIFT_create()
12
13 # Detecta os pontos-chave da imagem
14 keypoints, descritores = sift.detectAndCompute(escala_cinza, None)
15
16 # Desenha os pontos-chave como circulos na imagem 'escala_cinza'
17  imagem_saida = cv2.drawKeypoints(escala_cinza, keypoints, imagem)
18  # Salva a imagem com os pontos-chave como 'pontos_chave_SIFT.jpg
19   cv2.imwrite('pontos_chave_SIFT.jpg', imagem_saida)
```

Listagem 5.2. Exemplo de código para extração de pontos-chave.

Um exemplo de código em OpenCV Python para detecção/extração de pontos-chave pode ser encontrado na Listagem 5.2. Nesse exemplo utilizou-se o detector fornecido pela implementação OpenCV do extrator de características SIFT. Percebe-se, na linha 14, que o método *detectAndCompute* devolve tanto os pontos-chave (*keypoints*) quanto os vetores de características SIFT (*descritores*).

Características Aurais

As características aurais (sonoras), de maneira análoga às características visuais, representam propriedades específicas do sinal de áudio em um formato compacto, possibilitando a comparação com outros sinais de áudio. Características como momentos de silêncio, amplitude do sinal e frequência são bastante comuns e triviais de serem obtidas (com o auxílio de ferramentas como SoundForge[7] e Audacity[8]) e armazenadas em um vetor de características. Apesar de úteis, tais características isoladamente limitam a realização de tarefas importantes como detecção e reconhecimento de fala e similares com maior grau de complexidade. Para tais tarefas, características mais robustas tem sido empregadas, como *Linear Predictive Coding* (LPC), *Linear Predictive Cepstral Coefficients* (LPCC), *Linear Frequency Cepstral Coefficients* (LFCCs) and *Exponential Frequency Cepstral Coefficients* (EFCCs) [Vestman et al., 2018].

Um dos mais famosos extratores de características aurais, o *Mel Frequency Cepstral Coefficients* (MFCC), foi introduzido por Davis e Mermelstein (1980) e era destinado à tarefa de reconhecimento de palavras monossilábicas. Mais tarde, o MFCC se mostrou aplicável em tarefas como o reconhecimento de locutores, integrando o

[7] https://www.magix.com/br/musica/sound-forge/

[8] https://www.audacityteam.org/

estado da arte em análise de áudio atual [Yang et al., 2019]. O MFCC foi desenvolvido para mimetizar o comportamento da percepção de fala humana.

A frequência percebida por um humano é diferente do que é emitido pela fonte. A mimetização é realizada pelo uso da Escala Mel [Stevens et al., 1973], uma função da frequência para valores mel, que são unidades obtidas pela percepção por ouvintes de frequências consideradas como iguais, em distância, uma da outra. A extração de características MFCC é iniciada pela divisão do sinal de áudio em quadros. O sinal nestes quadros é convertido para o domínio de frequências através de uma transformada discreta de Fourier e um espectro de frequências mel é computado sobre estas frequências. O espectro de frequências mel é submetido a uma função logarítmica, para imitar a percepção humana de volume, também logarítmica. O próximo passo consiste na remoção de características dependentes do locutor por meio do cálculo de coeficientes cepstrais. Este passo se baseia no fato de que a fala humana pode ser modelada por um modelo fonte-filtro que descreve a produção de fala como um sinal e um filtro. O cepstro tem como objetivo suprimir o sinal fonte, mantendo apenas o sinal de filtro. Ele é obtido pela aplicação de uma transformada discreta de cosseno no sinal, mantendo os coeficientes de ordem baixa, que apresentam a resposta de frequência do trato vocal. Os coeficientes de ordem alta que apresentam a resposta de frequência do sinal de origem são descartados. Os vetores MFCC típicos mantém 13 coeficientes cepstrais.

```
import numpy as np
from python_speech_features import mfcc
import scipy.io.wavfile as wav
# Abre o audio de entrada 'audio.wav', retornando seu sinal e o valor da taxa de frequencia em Hz
(sinal, taxa_frequencia) = wav.wave("audio.wav")
# Extraem descritores mfcc do audio de entrada.
# Cada descritor será gerado sobre 0.1s de áudio, usando uma janela deslizante com deslocamento de 0.05s. Os 13 primeiros ceptrais serão retornados
caracteristicas = mfcc(sinal, taxa_frequencia, winlen=0.1, winstep=0.05, numcep=13)

# Imprime as características encontradas no áudio de entrada
for c in caracteristicas:
   print(c)
```

Listagem 5.4. Exemplo de código para extração de característica aurais MFCC.

Apesar de amplamente utilizado, o MFCC possui limitações e é passível de melhora (Sharma e Ali, 2015). Uma das mais problemáticas é a falta de significado de seus valores, já que apenas os dois primeiros coeficientes possuem significado determinado. O primeiro é a média da energia de todas as bandas de frequência e o segundo é o balanço de componentes de baixa e alta frequência do quadro de sinal. Os outros 11 coeficientes contém detalhes do espectro. O MFCC também é bastante sensível à presença de ruído no sinal, enfrentando uma degradação de performance severa nesse tipo de situação. Outro problema é a presunção de que os coeficientes

cepstrais da frequência fundamental são mais baixos do que dos componentes de frequência da mensagem linguística, que para alguns locutores do sexo feminino é falsa.

A listagem 5.4 traz um exemplo de código para a extração de características MFCC. Nesse exemplo, do áudio de entrada (*audio.wav*) gera-se um sinal (método *wave*) e desse sinal serão extraídas as características MFCC (método *mfcc*).

5.3.2. Características de Médio Nível Semântico

Bag of Words

O *Bag of Words* (BoW) é um modelo que gera uma representação simplificada de um texto [Aggarwal e Zhai, 2012], representado por um vetor que contém a contagem das ocorrências de suas palavras de acordo com um dicionário. Com isso, dois documentos podem ser comparados por seus vetores BoW gerados com base em um dicionário comum, usando alguma medida de distância entre vetores. Formalmente, o modelo consiste em obter uma representação de um documento $D = (t_1,t_2,...,t_p)$ no qual p é o tamanho do vocabulário e ti é a relevância do termo ti no documento D [Salton e Buckley, 1988]. Cada termo no modelo BoW pode se referir a uma única palavra, a duas palavras (bigrama) ou até mesmo uma frase completa. Já a relevância do termo pode ser calculada de diversas formas, incluindo a frequência do termo (do inglês *term frequency* - *tf*) ou a frequência inversa do termo (do inglês *inverse term frequency* - *idf*) [Ko, 2015].

Considerando $f_{t,d}$ como a frequência do termo t no documento d, a frequência do termo (*tf*) pode ser obtida de diversas formas [Aggarwal e Zhai, 2012], como a definida a seguir:

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{5.1}$$

Ao invés de contar o número de ocorrências de cada termo no documento, a frequência inversa do termo (*idf*) busca medir o número de documentos de um *corpus* em que o termo ocorre, separando palavras comuns e que aparecem na maioria dos documentos ou palavras raras que aparecem em poucos documentos. Considerando D o conjunto de documentos do corpus textual e $|D|$ o número de documentos em D e $|d \in D: t \in d|$ o número de documentos que possuem o termo t, a frequência inversa do termo $idf(t;D)$ pode ser calculada como:

$$idf(t,D) = \log \frac{|D|}{1+|d \in D: t \in d|} \tag{5.2}$$

Além do cálculo da frequência de um termo, um importante aspecto na aquisição do conhecimento por informações textuais não-estruturadas é a etapa de pré-processamento [Rezende et al., 2011], no qual os termos de entrada são tratados e padronizados, preservando as principais características do texto de entrada. Exemplos de abordagens de pré-processamento amplamente utilizadas são a remoção de palavras muito comuns, chamadas de *stop words*, padronização do texto para minúsculas, radiciação (do inglês *stemming*) e lematização (do inglês *lemmatization*), entre outros.

Nesse sentido, Uysal e Gunal (2014) reportam uma avaliação de diferentes técnicas de pré-processamento e seu impacto em diversos *corpus* textuais diferentes.

Bag of Features

Os primeiros trabalhos utilizando o método *Bag of Features* (BoF), análogo do BoW aplicado em outros tipos de informação, são da área de Visão Computacional e datam da década de 2000. Neles, os textos foram substituídos por imagens e as palavras por *textons*, vetores que representam texturas [Leung e Malik, 2001]. Mais tarde, Csurka et al. (2004) propuseram o *Bag-of-Keypoints*, também destinado à comparação de imagens, que utilizava centroides de grupos de pontos chave SIFT no lugar dos *textons*. Muitos autores utilizam o termo *Bag-of-Visual-Words* (BoVW) quando referenciam um BoF orientado a conteúdo visual, já que no método, os padrões distintivos representados pelos centroides são análogos às palavras de um texto. O modelo BoF também foi estendido para outras naturezas de dados como, por exemplo, dados aurais, formando um *Bag-of-Aural-Words* (BoAW) [Carletti et al., 2013].

Apesar de existirem variações em determinadas aplicações, é possível enquadrar os métodos BoF num arcabouço genérico para comparar dois ou mais documentos[9] de qualquer natureza. Uma descrição em alto nível deste arcabouço para comparação de dois documentos é apresentada no Algoritmo 1.

Algoritmo 1 – Arcabouço da técnica de comparação de documentos por BoF

COMPARA_BOF(a,b,k,d): recebe um par de documentos a e b, um tamanho de dicionário k e uma distância de similaridade entre vetores como entrada. Devolve um valor d_{bof} que corresponde à similaridade entre a e b.

1: Extraia múltiplos vetores de características representativos de a e de b, e armazene-os em conjuntos de vetores de características v_a e v_b, respectivamente;
2: $D \leftarrow v_a \cup v_b$
3: Agrupe os elementos do conjunto D em k grupos, obtendo um conjunto de grupos W;
4: Calcule, para cada grupo $w \in W$, um vetor de características que melhor represente os elementos de w. O conjunto resultante destes vetores é o dicionário *Dict*;
5: Calcule os histogramas h_a e h_b, calculando as ocorrências de v_a e de v_b, respectivamente, em *Dict*;
6: **devolva** $\delta(h_a,h_b)$.

As Linhas 1 e 2 do Algoritmo 1, descrevem a extração de características dos dois documentos e sua união em um conjunto único. A justificativa desta etapa é a criação de um conjunto que possua todas as características presentes em qualquer um dos documentos, para permitir a identificação de padrões comuns em seguida. O passo descrito na Linha 3 do algoritmo serve para encontrar padrões entre todos os vetores de características de a e de b. Espera-se que os vetores de características de documentos diferentes, mas que representam um padrão semelhante em ambos os documentos, sejam atribuídos a um mesmo grupo. A instrução presente na Linha 4 do algoritmo é

[9] "Documento" como sinônimo de objeto multimídia, que pode inclusive ser um documento textual, mas também pode ser uma imagem ou vídeo.

responsável por definir uma representação única para cada um dos padrões identificados no passo anterior. A determinação do vetor de características que melhor representa um elemento de um grupo normalmente se dá através do cálculo de um centróide ou de um medóide. O conjunto de representações únicas corresponde a um dicionário de padrões encontrados nos documentos. Na etapa descrita na Linha 5 é realizada uma contagem de quantos vetores de características, do total de um documento, são mais próximos a cada um dos elementos de *Dict*. O resultado dessa contagem é o histograma de ocorrência de padrões. Por fim, a Linha 6 contém o cálculo de distância entre os vetores de ocorrência de padrões. É esperado que, nesta etapa, documentos com padrões semelhantes sejam considerados semelhantes.

O Algoritmo 1 pode ser facilmente adaptado para aplicação em sistemas RIM, onde o objetivo é recuperar os documentos similares a um documento dado como entrada. Para isso, basta gerar o dicionário de uma base de documentos, por exemplo, uma base de imagens. Em seguida, deve-se gerar os histogramas representativos de cada documento da base. Esses histogramas formam então a base de dados para comparação. Ao receber um documento de entrada, o sistema usa o dicionário para gerar o histograma representativo deste documento, comparando o mesmo com todos os histogramas da base. Os histogramas considerados similares – acima de um determinado limiar para alguma medida de similaridade – são selecionados para indicar os documentos/imagens a serem devolvidosb3t4schin

A Seção 5.5 apresenta uma discussão e exemplos de como gerar *Bags-of-features* e dicionários de modalidades diferentes, fundindo as características para que sejam utilizadas em uma tarefa específica de RIM. A Fusão de Modalidades é discutida na Seção 5.4.

5.4. Fusão de Modalidades

O emprego de multimodalidade em um sistema exige a definição de algum tipo de estratégia para combinar a informação proveniente de diferentes fontes. A combinação da informação de diferentes modalidades é chamada fusão multimodal. De acordo com Atrey *et al.* (2010), a fusão multimodal é realizada, geralmente, em dois níveis. Um destes níveis é a fusão tardia, também conhecida como fusão no nível de decisão. O arcabouço de fusão tardia é apresentado na Figura 5.3, inspirada em Atrey *et al.* (2010).

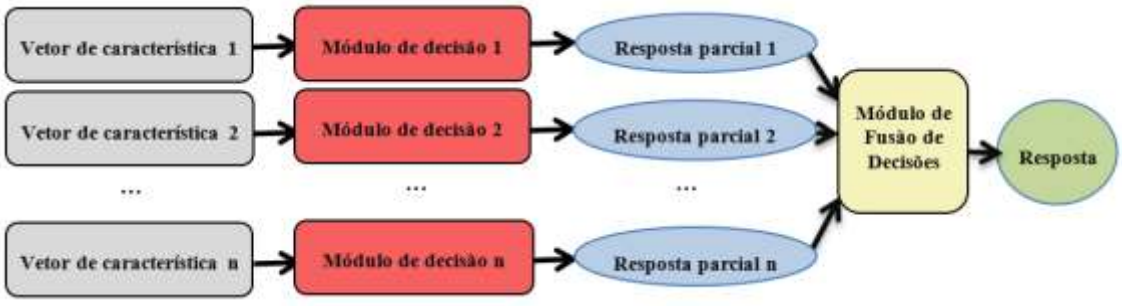

Figura 5.3. Fusão Tardia

Na fusão tardia, há uma etapa de decisão para cada um dos *n* vetores de características. Estas *n* decisões individuais são realizadas por *n* módulos de decisão, representados por retângulos vermelhos na Figura 5.3. As *n* etapas de decisão produzem *n* respostas parciais à tarefa, ilustradas na Figura 5.3 pelas elipses azuis. As respostas

parciais são então combinadas em uma resposta consensual por um módulo de fusão de decisões, representado na Figura 5.3 pelo retângulo amarelo.

Uma das vantagens da fusão tardia é a facilidade em combinar informações heterogêneas já que cada diferente representação proveniente de uma diferente modalidade dará origem a uma resposta parcial. Combinar as respostas parciais é relativamente simples considerando que todas as respostas têm a mesma representação. A fusão tardia também oferece relativa facilidade de inclusão/remoção de modalidades no processo pelo mesmo motivo. Outra vantagem da fusão tardia é a possibilidade de escolha de diferentes métodos de decisão para as diferentes modalidades. É possível, não somente escolher o método mais adequado para cada modalidade, como também parametrizá-lo para otimizar o aproveitamento da informação unimodal. A fusão tardia, no entanto, é incapaz de utilizar qualquer correlação existente entre as diferentes modalidades antes de uma etapa de decisão, podendo levar à perda de informação importante para esta etapa. Além disso, por requerer múltiplas etapas de decisão individuais, torna-se computacionalmente cara.

O segundo nível de fusão multimodal existente é a fusão prévia, também conhecida como fusão no nível de características. A fusão prévia é ilustrada na Figura 5.4, inspirada em Atrey et al. (2010).

Figura 5.4. Fusão Prévia

Em uma abordagem fusão prévia, há uma combinação de *n* vetores de características, produzindo um novo vetor de características que contém uma combinação da informação dos vetores de características de origem. Este vetor com informação combinada, chamado vetor de características fundidas, é ilustrado na Figura 5.4 por um retângulo azul. A ferramenta que efetua esta combinação é chamada módulo de fusão e é representada pelo retângulo de cor amarela da Figura 5.4. O vetor de características fundidas é então fornecido ao módulo de decisão, que produz a resposta esperada de acordo com a tarefa de vídeo. Em um sistema de segmentação temporal de vídeo, por exemplo, o módulo de decisão utiliza o vetor de características fundidas para encontrar as fronteiras entre os segmentos de vídeo. O módulo de decisão e a resposta produzida por ele são representados, respectivamente, pelo retângulo vermelho e pela elipse de cor verde na Figura 5.4.

A fusão prévia possibilita a identificação e o uso de correlação entre as diferentes características em uma etapa inicial, viabilizando a extração de melhores recursos para a execução da tarefa [Atrey *et al.*, 2010]. Outra vantagem da fusão prévia é a necessidade de uma única etapa de decisão que utiliza o vetor de características fundidas como entrada, dispensando o processamento individual dos múltiplos vetores unimodais. Isso se traduz em um ganho de eficiência, caso o vetor de características

fundidas seja pequeno em relação ao tamanho de todos os vetores de características unimodais juntos. Por fim, a complexidade em parametrizar e ajustar um único método de decisão é consideravelmente menor do que fazer o mesmo para *n* modalidades.

Apesar de suas vantagens, a fusão prévia é difícil de ser implantada. Um dos motivos é a dificuldade de identificação de correlação entre características. Isso acontece porque as características de diferentes modalidades tem diferentes representações. Um vetor de características SIFT, por exemplo, carrega magnitudes de gradientes em suas células enquanto um vetor de características MFCC carrega coeficientes que caracterizam um espectro sonoro. Outro motivo é a dificuldade de identificação de correlação entre características intermodais não sincronizadas temporalmente. Um padrão visual, por exemplo, pode ser complementar a um padrão aural que ocorre em um instante de tempo diferente. Por fim, ainda não se conhece um modelo adequado para representação única de informação combinada proveniente de múltiplas modalidades que seja compacto e, ao mesmo tempo, capaz de representar adequadamente o conteúdo. Um exemplo de método de fusão prévia é discutido na Seção 5.5.

5.5. Dicionários Multimodais e Representação de Fusões

Este seção discute um método eficaz para fusão prévia multimodal. Fusão prévia implica na obtenção de representações únicas (um único vetor de características) advindas de múltiplas representações de informações de uma coleção, de tal forma que estas representações únicas contenham informações relevantes provenientes das múltiplas representações originais. Estas representações únicas são ditas fundidas. O método discutido aqui é independente de domínio de aplicação e é flexível em relação à quantidade de modalidades a serem fundidas. Além disso, diferente da maioria dos trabalhos similares atualmente encontrados na literatura, é capaz de produzir representações que contém, simultaneamente, padrões unimodais e multimodais.

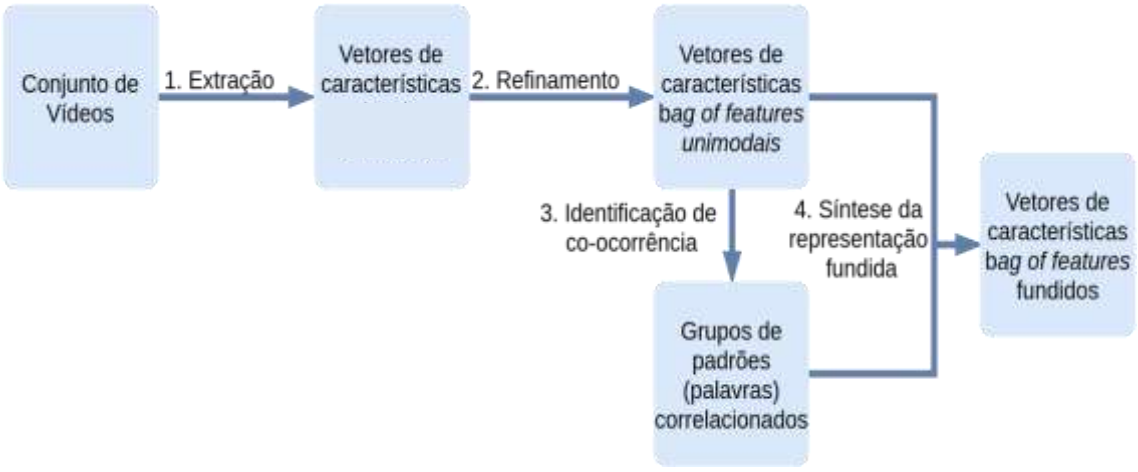

Figura 5.5. Visão geral do método de fusão prévia

A Figura 5.5 apresenta uma visão em alto nível dos passos do método, o qual é composto por quatro etapas. A primeira delas (Extração) determina múltiplos processos de produção de vetores de características, um por modalidade. A segunda etapa (Refinamento) se dá pela produção, à partir das características unimodais brutas, de representações comuns por meio de características de médio nível semântico (BoF). Na terceira etapa (Identificação de co-ocorrência) é realizada uma análise das

características unimodais BoF em busca de correlação inter/intra modalidades. Por fim, na última etapa (Síntese da representação fundida), uma representação única é produzida, a partir das as representações de médio nível semântico das correlações descobertas entre elas.

Usaremos vídeo como fonte de informação por conveniência. Explica-se: vídeo é multimodal por natureza, possuindo tanto fontes de informações visuais (imagens e texto) quanto aurais (áudio), sendo propício para exemplificar fusão multimodal assim como para aplicar o método a alguma tarefa de análise de dados (de vídeo no caso). Deve-se, contudo, estar claro que o método é independente dos tipos de fontes de informação.

Iremos tomar como exemplo a tarefa de segmentação temporal de vídeo em cenas, uma tarefa ainda em aberto na área [Kishi, R., Trojahn, T. e Goularte, R., 2019]. A segmentação temporal de vídeo em cenas consiste em definir onde, temporalmente, estão as fronteiras entre as cenas deste vídeo. Uma cena é definida como um conjunto de tomadas adjacentes semanticamente relacionadas [Saraceno e Leonardi, 1997] onde, tomadas, por sua vez, são sequências de quadros capturados ininterruptamente por uma única câmera [Koprinska e Carrato, 2001]. Considerando que a maioria das técnicas de segmentação de vídeo em cenas é baseada no agrupamento de tomadas adjacentes por sua semântica [Del Fabro e Boszormenyi, 2013], a extração da semântica das tomadas de um vídeo caracteriza-se como etapa fundamental neste processo, o que pode ser alcançado de modo latente por métodos de dicionários multimodais.

5.5.1. Extração de Características

A etapa 1 do processo ilustrado na Figura 5.5 corresponde à extração de características. Para o caso do vídeo a extração ocorre nas modalidades visual e aural, extraindo-se *features* de baixo nível das imagens (SIFT) e do áudio (MFCC). Esta etapa ocorre conforme discutido na Seção 5.3.1.

Como o volume de informações em vídeo é muito alto, o número de vetores de características extraídas também é alto. Comumente limita-se a quantidade de vetores escolhendo-se um subconjunto representativo das informações antes da extração. Quando se trata da extração de características da modalidade visual, é realizada a seleção de um ou mais quadros chave para representar o conteúdo de cada trecho de um determinado vídeo ou até mesmo de todo o vídeo, dependendo da tarefa em questão. Um quadro chave é um dos quadros que compõem a imagem em movimento do vídeo, escolhido para representar da melhor forma possível o conteúdo de todos os outros quadros restantes daquele segmento.

```
1  def sift(frame):
2    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
3    sift = cv2.xfeatures2d.SIFT_create()
4    kp, des = sift.detectAndCompute(gray, None)
5    return des
6
7  def extrairSIFT(video, keyframes):
8    cap = cv2.VideoCapture(video)
9    frameNumber = 0
10   featuresVisuais = []
11
12   while(cap.isOpened()):
13     ret, frame = cap.read()
14     if not ret:
15       break
16
17     if frameNumber in keyframes:
18       print("Extracting SIFT from keyframe %d" % frameNumber)
19       featuresVisuais.append(sift(frame))
20
21     frameNumber = frameNumber+1
22   cap.release()
23   return featuresVisuais
```

Listagem 5.5. Método geral para extração de características SIFT de uma tomada de vídeo.

A Listagem 5.5 apresenta um exemplo de código Python para extração de características visuais SIFT de um vídeo. O método *extrairSIFT* (linha 7) recebe como entrada o vídeo a ser processado (*video*) e uma lista contendo os números sequenciais dos quadros de vídeo que são quadros-chave (*keyframes*). Esta lista é obtida selecionando o quadro mediano de cada tomada do vídeo. Por exemplo, supondo que a tomada 1 comece no quadro 0 do vídeo e termine no quadro 100, o quadro-chave representativo da tomada 1 será o quadro 50. Percorre-se então o vídeo extraindo-se seus quadros (linhas 12 e 13) um a um, armazenando a imagem resultante (*frame*, linha 13) e buscando-se seu número sequencial na lista de quadros-chave, caso o quadro atual esteja presente na lista de quadros-chave (linha 17), a imagem correspondente é processada para se extrair os vetores de características SIFT. Para isso chama-se o método *sift* (linha 19) passando como argumento a imagem (*frame*). O resultado para cada quadro/imagem, isto é, os vetores de características, é armazenado na lista *featuresVisuais* (linha 19). O método *sift* (linha 1), por sua vez, transforma a imagem para tons de cinza (linha 2), cria o extrator de características (linha 3) e computa os pontos-chave e os vetores de características (*kp* e *des*, linha 4), retornando os vetores (*des*, linha 5).

A mesma ideia se aplica, de maneira análoga, à modalidade aural de vídeos. É escolhido um conjunto de amostras que melhor representam o conteúdo aural de todo o vídeo/segmento de vídeo e que seja compacto o suficiente para possibilitar e/ou agilizar o processamento.

```
1  def extrairMFCC(video, segTomadas):
2      audio = VideoFileClip(video).audio
3      fps = VideoFileClip(video).fps
4      features = []
5
6     for tomada in segTomadas:
7        inicio = math.ceil(tomada[0]/fps)
8        fim = math.floor(tomada[1]/fps)
9        audio_tomada = audio.subclip(inicio, fim).to_soundarray()
10       mfcc_tomada = mfcc(audio_tomada, 44100)
11       features.append(mfcc_tomada)
12
13    return features
```

Listagem 5.6. Método geral para extração de características MFCC de um segmento de vídeo.

A Listagem 5.6 apresenta um método/função em Python que recebe como entrada um vídeo (*video*) e um arquivo contendo as transições de tomadas do vídeo (*segTomadas*) no formato *Comma Separated Values* (CSV), onde o primeiro valor de cada linha representa o quadro inicial da tomada e o segundo valor representa o quadro final da tomada. Por exemplo,

0, 123

124, 357

significa um vídeo com duas tomadas, onde a primeira tomada termina no quadro 123 e a segunda tomada começa no quadro 124. O segmento de vídeo, então, para o qual se irá gerar uma representação MFCC é uma tomada. Como as características aurais são extraídas do áudio em um intervalo de tempo, o intervalo de cada tomada foi calculado para servir de parâmetro para o método de extração. O cálculo do intervalo para cada tomada (linha 6) divide o número do quadro inicial e final da tomada pela taxa de quadros do vídeo (*fps*). Assim, tem-se que as variáveis *inicio* e *fim* determinam o intervalo de tempo da tomada (linhas 7 e 8). Por exemplo, para uma taxa de quadros igual a 25fps e sendo a tomada 1 compreendida entre os quadros 0 e 123, temos que o intervalo de tempo da tomada 1 inicia no segundo 0 do vídeo e que termina no segundo 4,92. O método OpenCV *audio.subclip* (linha 9) determina o segmento de áudio, enquanto o método *to_soundarray* retorna as informações sobre o segmento (veja Listagem 5.4) necessárias para a aplicação do método *mfcc* que procederá a extração de características MFCC que irão representar a

tomada (linha 10). Cada representação de tomada é armazenada em uma lista (*features*, linha 11).

5.5.2. *Bag-of-Features* e Dicionários

A etapa seguinte do método de fusão exemplificado (etapa 2 na Figura 5.5) tem duas funções principais: realizar um refinamento semântico na semântica latente presente nos vetores de características locais em uma representação concisa e semanticamente representativa e, também, dar início ao processo transferência das informações unimodais representadas de maneira heterogênea para um espaço de representação comum. Estes dois objetivos são alcançados computando vetores de características BoF para cada modalidade, à partir dos vetores de características locais. Converter as diferentes representações unimodais para um espaço de representação comum resolve o dilema da heterogeneidade de representação de características de diferentes extratores de características, permitindo a comparação direta entre representações de diferentes modalidades. A Figura 5.6 ilustra a primeira parte do processo de computação de vetores de características BoF unimodais.

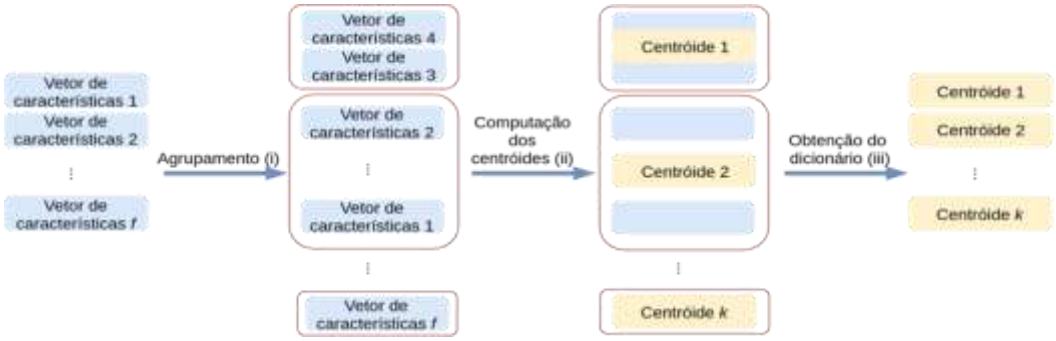

Figura 5.6. Obtenção de características BoF e do Dicionário.

O conjunto com todos os *f* vetores de características locais provenientes de um extrator de características passa, inicialmente, por um processo de agrupamento pelo método *k-means*. Os *k* grupos produzidos indicam a existência de padrões naquela modalidade. Como exemplo, espera-se que vetores de características MFCC que representam o som da voz de um locutor específico sejam mais próximos um do outro do que de vetores de características MFCC que representam a voz de um segundo locutor com voz distinta e, desta forma, sejam atribuídos a um mesmo grupo pelo processo de agrupamento.

Cada grupo/padrão é então representado por um centróide, que é um vetor de características sintetizado com as médias dos vetores de características que compõem aquele grupo. O conjunto com os *k* centróides produzidos neste processo é chamado dicionário de características e a cada um destes centroides é dado o nome de **palavra unimodal** (ou simplesmente palavra), em analogia ao modelo BoW que emprega um dicionário de palavras textuais. Além disso, costuma-se explicitar na terminologia a modalidade de origem da palavra (*e.g* palavra visual/aural).

A Listagem 5.7 apresenta um exemplo de código em OpenCV Python para a construção de dicionários de palavras unimodais. O método *criar_dicionario* recebe como parâmetro uma lista do tipo *bag-of-features* contendo as características de todas as

tomada (*f_tomadas*, linha 1), vinda, por exemplo, do método apresentado na Listagem 5.5. Nesse caso, são então *bags* de características visuais. Além disso, o método recebe também como parâmetro o tamanho do dicionário que ser pretende gerar (*numero_centroides*). Esse tamanho corresponde ao número de grupos que se pretende gerar ao se aplicar o algoritmo de agrupamento *k-means* (parâmetro k do *k-means*). Entre as linhas 3 e 5 da Listagem 5.7, para cada tomada presente na lista *f_tomadas*, são recuperados os vetores de características que, por sua vez, são adicionados a um espaço comum de características (*data*, linhas 5 e 6). Esse espaço comum será então utilizado para se gerar os k agrupamentos. Para tanto se utiliza o método OpenCV *cv2.kmeans* (linha 8), sendo que os centroides de cada grupo (palavra visual, nesse caso) são retornados e armazenados na lista *center* (linha 8). A lista *center* corresponde ao dicionário de palavras visuais de tamanho k.

```
1 def criar_dicionario(f_tomadas, numero_centroides):
2     data = []
3     for tomada in f_tomadas:
4         for feat_vet in tomada:
5             data.append(feat_vet)
6     data = np.float32(np.asarray(data))

7     crit = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER,
              100, 0.1)
8     ret, label, center = cv2.kmeans (data, numero_centroides, None, crit, 10,
                                       cv2.KMEANS_RANDOM_CENTERS)
9     return center
```

Listagem 5.7. Exemplo em OpenCV Python de construção de um dicionário de palavras unimodais.

A etapa seguinte, e final, da computação de vetores de características BoF é ilustrada na Figura 5.7, que ilustra a contagem de ocorrências das palavras do dicionário de características em cada segmento de informação, no exemplo cada tomada de vídeo. Isso é feito comparando cada vetor de características proveniente de cada tomada de um vídeo com todas as palavras do dicionário de características. É contabilizada, para cada um destes vetores, uma ocorrência da palavra que mais se assemelha àquele vetor de características em questão. O conjunto de contagens de ocorrências das palavras em um vídeo é armazenado em um vetor chamado histograma de ocorrência de palavras (ou vetor de características BoF) que é considerado uma representação de médio nível semântico deste vídeo. Há um mapeamento entre o índice da célula de um vetor de características BoF de uma modalidade e cada palavra daquela modalidade (por exemplo, a terceira célula de qualquer vetor de características BoF é relacionado à palavra w_3, s.p.g), permitindo a comparação entre vetores através de uma medida de similaridade/distância entre vetores unidimensionais. A intuição por trás da representação pelo modelo BoF é a de que vídeos com histogramas de ocorrência semelhantes devem apresentar padrões semelhantes naquela modalidade e, assim, portarem semântica semelhante.

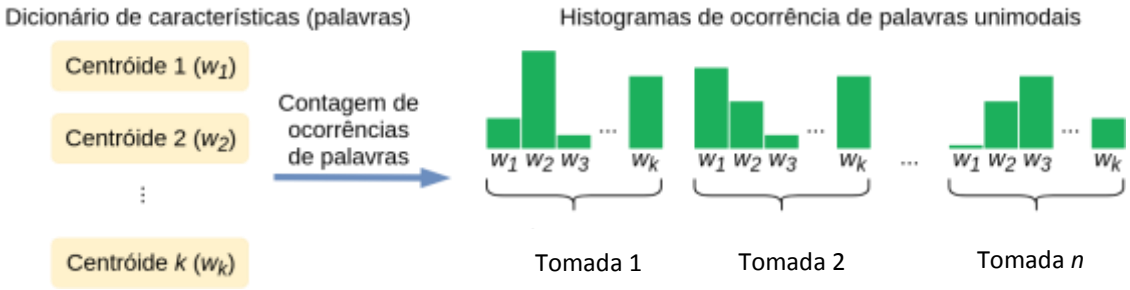

Figura 5.7. Geração dos histogramas representativos dos dados – vetores de características BoF.

```
1  def gerar_histogramas(tomadas, dicionario):
2    histograms = []
3    for tomada in tomadas:
4       histograms.append(np.zeros(len(dicionario)))

5     for i in range(len(tomadas)):
6       for feature in tomadas[i]:
7           indice = encontrar_match(feature, dicionario)
8           histograms[i][indice] = histograms[i][indice]+1
9        histograms[i] = histograms[i]/sum(histograms[i])
10     return histograms

11 def encontrar_match(feature, dicionario):
12     dist = []
13     for palavra in dicionario:
14        dist.append(distance.euclidean(feature, palavra))
15     return dist.index(min(dist))
```

Listagem 5.8. Exemplo em OpenCV Python de geração de histogramas representativos de tomadas.

A Listagem 5.8 exemplifica, em Python OpenCV, como é possível gerar os histogramas de frequências das tomadas ilustrados na Figura 5.5. O dicionário de características (Figura 5.5) equivale a *dicionário*. As características de cada tomada (*feature*) são comparadas às palavras no dicionário por meio do método *encontrar_match* (linhas 7 e 11), que usa distância euclideana (linha 14) para calcular a similaridade entre as palavras e as *features*.

Apesar dos vetores de características BoF de diferentes modalidades possuírem contagens de padrões em suas células e, possivelmente, possuírem mesmos comprimentos, os padrões representados por eles em células de mesmo índice são diferentes e provavelmente sequer relacionados. Uma comparação direta entre vetores de características BoF de diferentes modalidades não tem utilidade aparente. É necessário um processamento adicional para concluir a transferência das informações

heterogêneas unimodais para um espaço de representação comum. Isto pode ser realizado detectando-se co-ocorrência de padrões.

5.5.3. Detecção de co-ocorrência

A detecção de co-ocorrência completa o processo que permite a comparação entre informações unimodais representadas de maneira heterogênea e ainda lida com outro dilema da fusão prévia: a falta de sincronismo temporal entre informações de diferentes modalidades.

Esta etapa tem como entrada os vetores de características BoF produzidos para as diferentes modalidades na etapa descrita e gera como saída um mapeamento dos padrões correlacionados indicando quais deles, em conjunto, participam da expressão de um padrão específico presente no conjunto de vídeos. Inicialmente converte-se os histogramas de ocorrência de todas as palavras de cada modalidade em cada tomada (histograma de ocorrências por tomada) em histogramas de ocorrência de cada palavra em todas as tomadas (histograma de ocorrências por palavra).

A conversão é realizada por meio de uma transposição da matriz formada pela sobreposição dos histogramas de ocorrências de palavras em cada tomada. Em OpenCV Python isso pode ser obtido por meio do método *transpose*:

```
...
histogramasVisuais = gerar_histogramas(featuresVisuais, dicionarioVisual)
histogramasVisuais = np.asarray(histogramasVisuais).transpose()
...
```

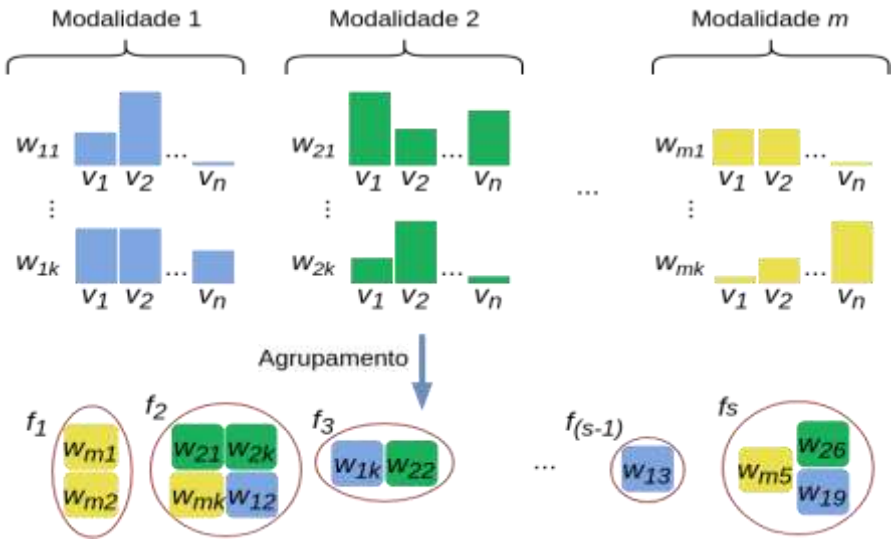

Figura 5.8. Agrupamento de palavras unimodais com base em seus padrões de ocorrência em diferentes vídeos. Cores iguais indicam que se trata de informações de uma mesma modalidade.

Os histogramas de ocorrência por palavra contêm o padrão de ocorrência daquela palavra em diferentes tomadas. Dessa forma, o cálculo de

similaridade/dissimilaridade entre histogramas de ocorrências por palavra corresponde à comparação de palavras em relação às suas ocorrências nas tomadas do conjunto. A computação dos histogramas de ocorrência por palavra de todas as modalidades permite a comparação, relativa ao padrão de ocorrência, entre palavras de diferentes modalidades. Como palavras correspondem a padrões nos vídeos/tomadas, ao comparar palavras de diferentes modalidades está se comparando, essencialmente, os padrões que as originaram. Baseando-se neste fato, o processo de detecção de co-ocorrência se completa pela etapa ilustrada na Figura 5.8.

A etapa ilustrada na Figura 5.8 é efetuada pela aplicação do método *k-means*, agrupando as palavras w_{ij} pela distância entre seus histogramas de ocorrência por palavra, onde i é o índice de uma modalidade e j é o índice de cada palavra da modalidade i, com $1 \leq i \leq m$ e $1 \leq j \leq k$. Os grupos f_o, com $1 \leq o \leq s$ produzidos neste processo contém palavras (padrões unimodais) que ocorrem em quantidades semelhantes em determinados vídeos. Ao determinar estes grupos, espera-se identificar padrões unimodais que se complementam com o objetivo de expressar um conceito semântico. Um exemplo de padrões com essa característica pode ser visto na Figura 5.9.

Figura 5.9. Exemplo de co-ocorrência de padrões aurais/visuais em diferentes vídeos.

Os dois quadros presentes na Figura 5.9, são provenientes de vídeos diferentes e contém padrões visuais semelhantes. Além disso, há padrões aurais – sons de pinguins – que também co-ocorrem nos mesmos dois vídeos. Em um processo de agrupamento como o recém descrito, estes padrões visuais e aurais provavelmente seriam atribuídos a um mesmo grupo, já que os padrões visuais presentes na representação gráfica de um pinguim são, na *BBC Dataset* (conjunto de informações utilizado), sempre acompanhados de sons de pinguins. Terminado o processo de computação de grupos de palavras unimodais correlacionadas, é necessário gerar uma representação única para cada grupo.

```
1  def deteccao_coocorrencia(hist_modalidades, numero_grupos):
2      hist_modalidades = np.float32(np.asarray(hist_modalidades))

3      crit = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER,
                100, 0.1)
4      ret, label, center = cv2.kmeans(hist_modalidades, numero_grupos, None,
                                       crit, 10, cv2.KMEANS_RANDOM_CENTERS)
5      grupos = []
6      for i in range(numero_grupos):
7          grupos.append([])

8      for i in range(len(label)):
9          grupos[label[i][0]].append(hist_modalidades [i])

10     histogramasGrupos = []
11     for grupo in grupos:
12         histogramasGrupos.append(maxPooling(grupo))

13     return histogramasGrupos

14 def maxPooling(grupo):
15     if len(grupo) == 1:
16         return grupo[0]
17     vetor = []
18     for i in range(len(grupo[0])):
19         vetor.append(max(coluna(grupo, i)))
20     return vetor
```

Listagem 5.9. Exemplo em OpenCV Python da detecção de co-ocorrência.

A Listagem 5.9 ilustra como implementar em OpenCV Python a detecção de co-ocorrência discutida nesta subseção. Os histogramas de palavras unimodais unidos em um único espaço de características (hist_*modalidades*) são entrada para o método *deteccao_coocorrencia*. As palavras unimodais são agrupadas utilizando o algoritmo kmeans gerando grupos compostos, contendo combinações de palavras visuais e aurais (linhas 3 e 4). O método *cv2.kmeans*, que implementa o algoritmo k-means, devolve, além dos centróides, uma lista de índices (*label*, linha 4). Tais índices permitem identificar em que agrupamento está cada palavra unimodal de entrada presente em *hist_modalidades*. Assim, é possível percorrer *hist_modalidades* colocando as palavras em grupos de fato (*grupos*), pois o método *cv2.kmeans* não retorna os grupos, apenas os índices. Agora, com os agrupamentos de palavras multimodais criados, pode-se completar a fusão gerando-se uma representação única, fundida. Isso é feito utilizando-se o método *maxPooling* (linhas 12 e 14), cujo detalhamento, entre outros métodos de pooling, está discutido na Seção 5.5.4.

5.5.4. Representação Única

A etapa final do método de fusão de características exemplificado lida com o dilema restante da fusão prévia: produz uma representação única para a expressão de múltiplos padrões correlacionados, mantendo o significado original das representações originais, ao mesmo passo em que descarta detalhes irrelevantes. Tal representação única permite pós processamento com o objetivo de melhor aproveitar a semântica na tarefa fim. A representação única da presença de padrões correlacionados é produzida através da técnica de *pooling*, bastante comum na área de Visão Computacional [Boureau et al., 2010]. De acordo com Boureau et al. (2010), o *pooling* produz, a partir das saídas de diferentes extratores de características, uma representação única combinada que preserva informação relevante à uma determinada tarefa, enquanto descarta detalhes irrelevantes.

A Figura 5.10 ilustra a aplicação do *pooling* na fusão de características, com uma visão detalhada dos histogramas que representam as palavras do grupo f_1.

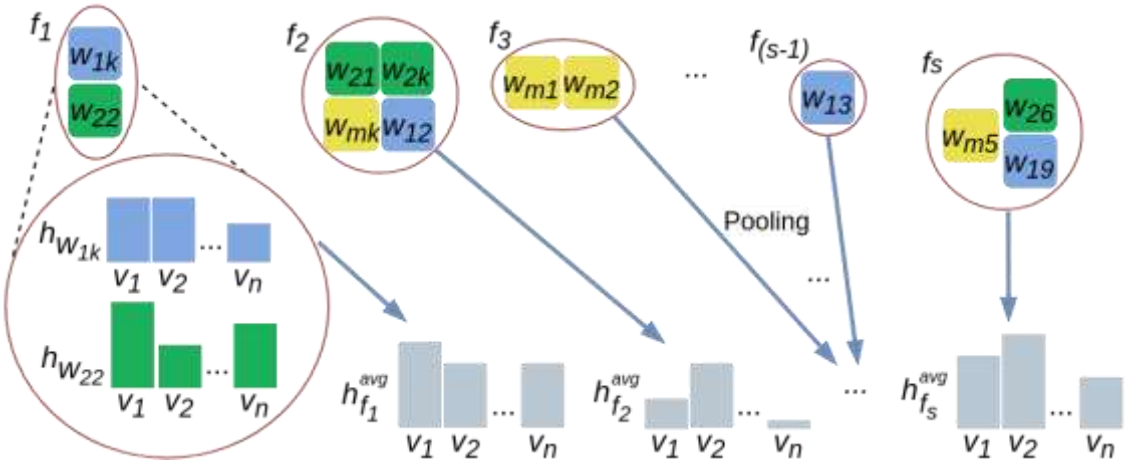

Figura 5.10. Exemplo do processo de *pooling*.

Considerando os grupos $f_1 \ldots f_s$, compostos por palavras unimodais, produzidos pela etapa descrita na Seção 5.5.3, o *pooling* ocorre através da aplicação, para cada grupo $\boldsymbol{f} \in \{\boldsymbol{f_1}, \ldots, \boldsymbol{f_s}\}$, de uma função que toma como entrada o conjunto dos histogramas de ocorrência em vídeos h_w de cada palavra unimodal w que compõe $\boldsymbol{f}$ nos vídeos $v_1,\ldots, v_n$ e produz como saída um histograma único h^{avg} que representa $\boldsymbol{f}$.

Há duas estratégias de *pooling* adequadas nesse tipo de fusão, que diferem em como as ocorrências do padrão correspondente a um determinado grupo serão contabilizadas: *average pooling* e *max pooling*. Ambas as estratégias podem, ainda, ser modificadas utilizando uma heurística de ponderação para aprimorar os histogramas de ocorrências de padrões produzidos pela fusão.

O *average pooling*, conforme sugere o nome, determina que seja computada uma média das ocorrências de cada palavra unimodal em cada vídeo. Os valores das células do histograma de ocorrências $\boldsymbol{h_f^{avg}}$, do *average pooling* do grupo $\boldsymbol{f}$ nos vídeos $v_1,\ldots, v_n$, são calculados usando a equação:

$$h_f^{avg}[v_i] = \frac{\sum_{w_j \in f} h_{w_j} v[i]}{|f|} \tag{5.3}$$

onde $\boldsymbol{h_{w_j}}$ é o histograma da palavra $\boldsymbol{w_j}$ que compõe o grupo $\boldsymbol{f}$.

O *max pooling*, também como sugere o nome, determina que sejam utilizados os máximos dentre todas as ocorrências de palavras unimodais que compõem um determinado grupo em cada vídeo. Os valores das células do histograma de ocorrências *max pooling* $\boldsymbol{h_f^{max}}$, do grupo f nos vídeos v_1,..., v_n são calculados usando a equação:

$$\boldsymbol{h_f^{max}[v_i] = \max_{w_j \in f} h_{w_j} v[i]} \tag{5.4}$$

onde $\boldsymbol{h_{w_j}}$ é o histograma da palavra $\boldsymbol{w_j}$ que compõe o grupo $\boldsymbol{f}$.

O *max pooling*, em contraste com o *average pooling*, produz um histograma de ocorrências de padrões baseado na parte mais representativa das ocorrências de palavras unimodais que compõem cada padrão. Desse modo, *o max pooling* é capaz de representar ocorrências heterogêneas com contagens semelhantes de um mesmo padrão, desde que haja uma palavra unimodal componente deste padrão sendo expressa de maneira semelhante em todas estas ocorrências.

Considerando o mesmo exemplo apresentado para o *average pooling*, diferentes expressões de explosões podem possuir conteúdo visual distinto e mesmo assim apresentarem histogramas de ocorrência *max pooling* semelhantes, desde que seu conteúdo aural seja semelhante e mais representativo do que o visual. Apesar de mais flexível, o *max pooling* é mais suscetível a erros de representação já que o valor final de contagem de ocorrências pode ser originado de qualquer uma das palavras unimodais que compõem um determinado padrão.

Ponderação

Dentre as virtudes do método de fusão proposto estão a preservação do significado original nos valores componentes dos vetores de características fundidos, bem como a rastreabilidade das origens destes valores. Graças a estas virtudes, é possível aprimorar os vetores de características fundidas identificando situações onde duas ou mais fontes de informação apresentam padrões de ocorrências semelhantes de palavras unimodais.

Duas ou mais fontes de informação indicando a existência de um padrão em um vídeo, intuitivamente, aumenta a confiança na existência daquele padrão. Pode-se então expressar a menor probabilidade de erro na identificação de um padrão ponderando sua contagem no histograma de ocorrências. O destaque destes padrões mais confiáveis aumenta o poder discriminatório das medidas de distância entre histogramas que poderão vir a ser empregadas nas ferramentas de análise. A abordagem proposta para realizar a ponderação se dá pela aplicação da seguinte equação nos histogramas de ocorrências fundidos $\boldsymbol{h_f^{fused}}$ dos grupos $\boldsymbol{f} \in \{\boldsymbol{f_1}, \ldots, \boldsymbol{f_s}\}$, produzindo histogramas de ocorrências ponderados $\boldsymbol{h_f^{weighted}}$:

$$\boldsymbol{h_f^{weighted}[v]} = \begin{cases} \boldsymbol{h_f^{fused}[v]\ x\ \partial}, & \textbf{se} \quad |\boldsymbol{f}| > \mathbf{1} \\ \boldsymbol{h_f^{fused}[v]}, & \textbf{caso contrário} \end{cases} \tag{5.5}$$

onde *v* é um dos vídeos em $\{v_1,..., v_n\}$ e $\boldsymbol{\partial}$ é o fator de ponderação. Os histogramas $\boldsymbol{h_f^{fused}}$ podem ser histogramas produzidos por *average pooling* ou *max pooling*.

Conversão em histogramas por tomada

O fim do método se dá pela conversão dos histogramas de ocorrências de cada palavra nas tomadas em histogramas de ocorrência de todas as palavras em cada tomada.

Os histogramas de ocorrência $\boldsymbol{h_v^{fused}}$, por tomada $\boldsymbol{v} \in \{\boldsymbol{v_1}, \ldots, \boldsymbol{v_n}\}$ são obtidos facilmente a partir de histogramas de ocorrência $\boldsymbol{h_f^{fused}}$ por palavra $\boldsymbol{f} \in \{\boldsymbol{f_1}, \ldots, \boldsymbol{f_n}\}$, uma vez que o histograma $\boldsymbol{h_v^{fused}}$ de tamanho *n* da tomada *i*, é formado pelos valores contidos nas posições *i* dos *n* histogramas $\boldsymbol{h_f^{fused}}$. Os histogramas $\boldsymbol{h^{fused}}$podem ser histogramas produzidos por *average pooling*, *max pooling* ou ponderação. Estes histogramas são o produto final da fusão de características e servirão de entrada para uma técnica-fim, como uma técnica de segmentação de vídeo em cenas.

5.6. Avaliação

A avaliação de sistemas de RIM normalmente implica em analisar e medir comparativamente um ou mais sistemas por meio de metodologia bem estabelecida. A metodologia compreende definir: o tipo de avaliação; uma base dados anotada (*groundtruth*) e medidas de avaliação. As avaliações podem ser realizadas com ou sem a participação direta do usuário do sistema. Neste texto estamos interessados nos métodos sem a participação do usuário. Tanto as medidas de avaliação quanto a base de dados devem ser adequadas para julgar a eficácia de um dado sistema e estão intimamente ligadas à tarefa que se deseja realizar, variando de tarefa para tarefa [Blanken et al., 2010].

O *groundtruth* utilizado para essa tarefa é composto pela a BBC *Planet Earth Dataset*[10], referenciada de agora em diante simplesmente como BBC *dataset* ou base BBC. Tal base contém 11 vídeos do documentário BBC *Planet Earth*, totalizando mais de 8 horas de vídeo, 4913 tomadas e 670 cenas. Além dos vídeos existe a anotação textual indicando os limites (cortes ou transições) entre cenas (segmentação ideal), possibilitando a comparação com segmentações geradas por técnicas automáticas.

As métricas, por sua vez, devem medir o quão distante o resultado de uma técnica de segmentação está da segmentação ideal (*groundtruth*). Nesse sentido, a literatura reporta diversas métricas diferentes, sendo as mais utilizadas pelos pesquisadores descritas a seguir nesta seção.

As métricas mais amplamente utilizadas para a tarefa em questão são a Precisão e a Abrangência. A Precisão (do inglês *precision* - *P*) e a Abrangência (do inglês *Recall*

[10]http://imagelab.ing.unimore.it/imagelab/page.asp?IdPage=5

- *R*) [Rijsergen 1979] são originárias da área de recuperação de informação. No contexto de segmentação em cenas, como cada tomada é ou não é de transição, o conjunto resposta do segmentador inclui os seguintes casos: Verdadeiro Positivo (*vp*), quando uma tomada retornada como sendo de transição corresponde de fato a uma transição no *groundtruth*; Falso Positivo (fp), quando uma tomada retornada como sendo de transição não corresponde a uma transição de fato; Verdadeiro Negativo (vn), é uma tomada não retornada pelo segmentador e que de fato não é de transição; Falso Negativo (fn), é uma tomada não predita pelo segmentador mas que, na realidade, é de transição. Assim, Precisão *P* e Abrangência *R* são definidas como:

$$P = \frac{vp}{vp + fp} \tag{5.6}$$

$$R = \frac{vp}{vp + fn} \tag{5.7}$$

A Precisão e Abrangência avaliam duas características diferentes de um segmentador qualquer. Caso um algoritmo obtenha elevada Precisão, significa dizer que a maioria das transições detectadas pelo algoritmo são transições verdadeiras, ou seja, o número de falsos positivos obtido é baixo. Mas isso não implica que todas as transições verdadeiras foram detectadas. Por outro lado, caso o algoritmo obtenha uma elevada Abrangência, significa que a maioria das transições que existem na base confiável foram detectadas, indicando um baixo número de falsos negativos. Mas não implica que somente transições verdadeiras foram detectadas.

Como tanto *P* quanto *R* medem diferentes aspectos da segmentação em si, um modo de agrupá-los em um valor único é por meio da Medida F ou F-score [Rijsergen 1979]. A Medida F é uma média harmônica entre os valores *P* e *R* que apresenta como vantagem o fato de ser mais conservadora que a média aritmética simples entre *P* e *R*, podendo inclusive dar maior prioridade tanto a *P* quanto a *R* em variantes tais como $F_{0.5}$ ou F_2. A métrica Medida F $\boldsymbol{F\beta}$ é definida como:

$$\boldsymbol{F_\beta} = (1 + \beta^2).\frac{\mathrm{P}\,.\,\mathrm{A}}{(\beta^2.\ \mathrm{P}) + \mathrm{A}} \tag{5.8}$$

Onde *b* é usado para dar prioridade à Precisão (valores abaixo de um) ou à Abrangência (valores acima de um). Caso $b = 1$ (F_1), não há prioridade qualquer, sendo a mesma doravante chamada F_{PR}, definida como:

$$\boldsymbol{F_{PR}} = 2\,.\frac{\mathrm{P}\,.\,\mathrm{A}}{\mathrm{P} + \mathrm{A}} \tag{5.9}$$

Embora amplamente utilizadas em trabalhos seminais na área [Del Fabro and Boszormenyi, 2013], as métricas de Precisão e Abrangência possuem limitações quanto a segmentação em cenas. Nesse sentido, a Figura 5.11 ilustra duas técnicas que obtém, para um *groundtruth* com duas transições (indicado como *Confiável* na Figura 5.11),

diferentes segmentações. Note que os valores de *P*, *R* e F_{PR} da **Técnica 1** (20%, 50% e 28% respectivamente), são superiores aos da **Técnica 2** (0%, 0% e 0% respectivamente), embora intuitivamente a segunda tenha obtido uma segmentação mais próxima da desejada (apenas deslocada por uma tomada).

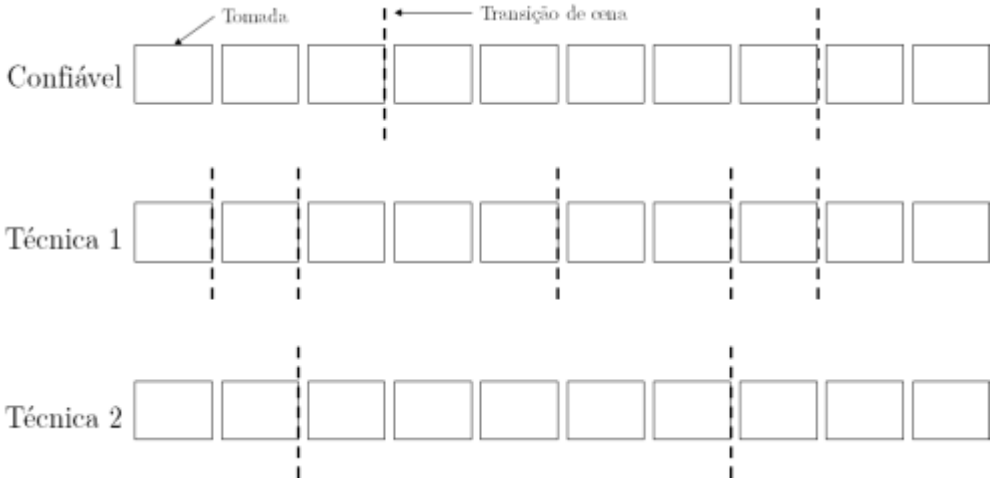

Figura 5.11. Ilustração de uma segmentação em cenas segundo o *groundtruth* (Confiável) e de duas técnicas hipotéticas para um determinado vídeo com 10 tomadas. Os retângulos representam as tomadas e a as linhas verticais denotam as transições de cenas.

Por tal motivo, diversos pesquisadores costumam adotar uma janela de tolerância, normalmente baseada em um número de quadros de vídeo, em tempo ou em número de tomadas [Baraldi et al., 2015]. Por exemplo, [Rasheed e Shah, 2003) especificam que uma transição de cena é considerada correta se estiver até dez segundos da transição confiável, enquanto Hanjalic et al. (1999) usaram três tomadas de tolerância. Tal prática, porém, causa inconvenientes tais como a dificuldade de comparar técnicas que usaram janelas de tolerância diferentes, além de ocultar pequenas diferenças que diferentes técnicas poderiam gerar. Del Fabro e Boszormenyi (2013) citam como outra possível limitação das métricas a dificuldade de medir quando uma cena é parcialmente detectada, quando o início e fim da cena não estão alinhados.

Nesse sentido, Vendrig e Worring (2002) definiram duas métricas específicas para a segmentação em cenas, conhecidas como Cobertura (do inglês *Coverage* - *C*) e Transbordamento (do inglês *Overflow* - *O*). A Cobertura procura medir quantas tomadas pertencentes à mesma cena foram corretamente agrupadas. Já o Transbordamento mede quantas tomadas, mesmo não pertencentes à mesma cena, foram incorretamente agrupados em uma mesma cena. As medidas de Cobertura (*C*) e Transbordamento (*O*) são definidas como:

$$C(x_t) = \frac{max_{j=0\ldots n}\#(y_j)}{\#(x_t)} \tag{5.10}$$

$$C(x_t) = \frac{\sum_{j=0}^{n} \#(y_j \backslash x_t).\min(1, \#(y_j \cap x_t))}{\#(x_{t-1}) + \#(x_{t+1})} \quad (5.11)$$

Sendo $\boldsymbol{x_t}$ o conjunto de tomadas que forma a cena confiável de índice *t*, y_j o conjunto de tomadas que forma a cena de índice *j* obtida pelo algoritmo segmentador, #(*x*) o operador que retorna a quantidade de tomadas de uma determinada cena *x* e #($y_j \backslash x_t$) o número de tomadas da cena detectada que não se encontram também na cena confiável (ou, em outras palavras, a operação de diferença entre y_j e x_t). É importante ressalvar que, ao contrário das métricas *P*, *R* e *C*, no qual o valor desejado é igual a 1 (100%), o valor desejado de Transbordamento (*O*) é igual a zero (0%), que corresponde ao caso de que nenhuma tomada de uma cena adjacente foi erroneamente agrupada na cena detectada.

Para obter o valor de *C* ou *O* de um dado vídeo *V*, a média ponderada é calculada de acordo com o número de tomadas do vídeo. Assim, sendo #(*Vs*) o número total de tomadas e #(*Vxj*) o número de cenas confiáveis do vídeo *V* , o cálculo da Cobertura *C*(*V*) e do Transbordamento *O*(*V*) são dados por:

$$C(\mathrm{V}) = \sum_{t=0}^{\#(Vx_j)-1} C(x_t).\frac{\#(x_t)}{\#(V_\sigma)} \quad (5.12)$$

$$O(\mathrm{V}) = \sum_{t=0}^{\#(Vx_j)-1} O(x_t).\frac{\#(x_t)}{\#(V_\sigma)} \quad (5.13)$$

Assim como no caso de Precisão e Abrangência, os valores de Cobertura e Transbordamento podem ser agrupados por meio da média harmônica Medida F, doravante representada como F_{CO}. É importante ressalvar que o valor do cálculo de Transbordamento utilizado na métrica F_{CO} é igual a 1 - *O*, visto a característica do valor desejado de *O* ser igual a zero.

Uma das principais vantagens das medidas de Cobertura/Transbordamento sobre a Precisão/Abrangência é não requerer a especificação de um parâmetro de tolerância, que pode interferir na interpretação dos resultados da técnica avaliada, facilitando inclusive a comparação com outras técnicas desenvolvidas. Outro benefício é a de que a métrica avalia o *quanto* uma cena foi detectada, sendo, portanto, mais robusta a cenas parcialmente detectadas [Del Fabro e Boszormenyi, 2013].

Contudo, a formulação da métrica de Transbordamento definida anteriormente contém uma limitação importante caso a segmentação a ser avaliada possua alto índice de subsegmentação (do inglês *undersegmentation*), quando uma cena detectada engloba

diversas cenas do *groundtruth* (que deveriam ter sido detectadas), resultando em valores inválidos. Para ilustrar tal limitação, considere um vídeo formado de 20 tomadas e 6 cenas (5 transições) e o resultado obtido por uma técnica de segmentação com 4 cenas (3 transições) sobre o mesmo vídeo. A Figura 5.12 ilustra o vídeo e as duas segmentações, confiável e detectada, mencionados.

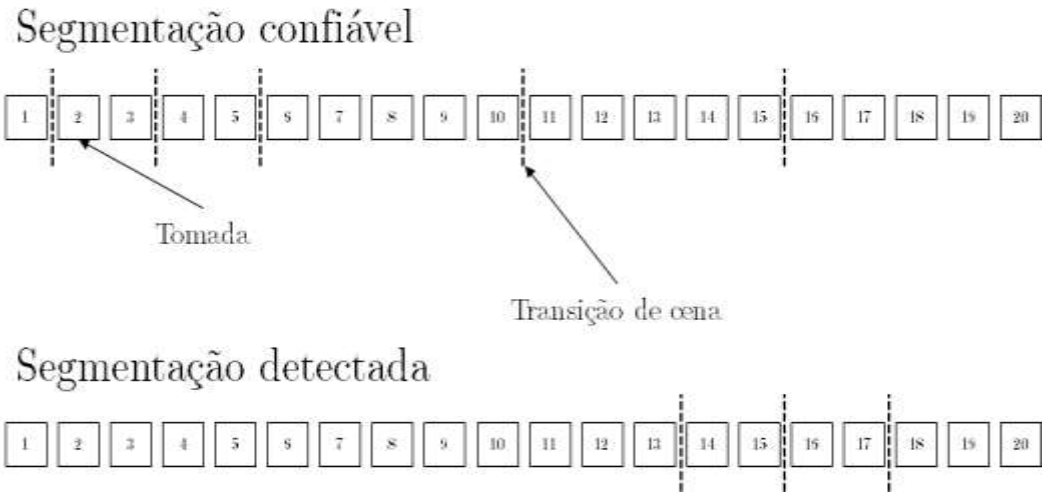

Figura 5.12. Exemplo de segmentação confiável versus segmentações geradas por técnicas hipotéticas.

Embora a segmentação detectada obtenha uma Cobertura válida de 80%, já que apenas 4 tomadas (20%) do *groundtruth* não são cobertas pela respectiva cena detectada (tomadas 14 a 17), o valor de Transbordamento é de 135%, um valor claramente inválido que excede o intervalo válido da métrica. Consequentemente, nesse caso, o valor obtido para F_{CO} (-124%) também é inválido. Para evitar tal problema, Han e Wu (2011) apresentam uma formulação alternativa para a métrica de Transbordamento, na qual se normaliza o valor para o intervalo válido [0,1]. Tal métrica, doravante chamada de *NO* (*New Overflow*), é definida como:

$$NO(x_t) = 1 - \frac{\#(x_t)}{\sum_{j, X_t \cap y_j \neq 0} \#(y_j)} \tag{5.14}$$

A principal melhoria da formulação da métrica *NO* sobre a métrica *O* é o denominador da divisão, formado pela soma do número de tomadas de todas as cenas detectadas que possuem alguma intersecção com a cena do *groundtruth* sendo analisada. Tal modificação significa que o denominador sempre será maior ou igual (melhor caso) que o numerador o que, associado com a subtração da equação, garante que o valor *NO* será um valor válido no intervalo [0,1]. Também pode-se calcular a média harmônica Medida F de maneira semelhante à métrica F_{CO}, usando o valor 1-*NO*, sendo a mesma doravante representada como F_{CNO}. Graças a tal melhoria, no exemplo hipotético ilustrado na Figura 5.12 o valor obtido de *NO* é aproximadamente 53% (0.53) e seu valor F_{CNO} é aproximadamente 59% (0.59), ambos os valores contidos no intervalo [0,1].

5.7. Considerações Finais

Este texto, parte do minicurso apresentado durante o WebMedia 2019, discutiu como realizar indexação multimídia utilizando técnicas de fusão multimodal e de detecção de co-ocorrência. Os conceitos foram apresentados e discutidos assim como exemplos de implementação em OpenCV Python. As técnicas e métodos discutidos são de interesse de uma variedade de aplicações que lidam com grandes volumes de dados não estruturados (Big Data, Recuperação de Informação Multimídia, etc.) pois, de modo eficaz, reduzem o volume de dados necessário para representar as informações mantendo representatividade. Em particular, o método apresentado de fusão prévia e de representação dos dados fundidos se mostra uma boa alternativa para tarefas relacionadas a RIM, como a segmentação de vídeo em cenas. Outras tarefas podem se beneficiar do método, como classificação de vídeos e localização de objetos.

Como limitações, a técnica de fusão apresentada utiliza apenas dois tipos de características, uma visual e outra aural, apesar de ser possível estender a técnica para utilizar também outros tipos de características, como texto. Além disso, por simplicidade, a seleção de quadros-chave (modalidade visual) apresentada no minicurso é realizada de modo ingênuo, selecionando-se o quadro mediano de cada tomada. Essa abordagem pode ser melhorada selecionando-se um conjunto de quadros-chave, o que seria mais representativo. A implementação do método de *pooling* não levou em consideração a possibilidade de se fazer ponderação, quê tem potencial para melhora de representatividade e de eficácia, consequentemente.

Currículo Resumido dos Autores

Rudinei Goularte possui graduação em Ciência da Computação pela Universidade Federal de Mato Grosso do Sul (1995). Possui mestrado (1998), doutorado (2003) e livre-docência (2011) pela Universidade de São Paulo, campus São Carlos, todos em Ciência da Computação. Atualmente é professor associado do ICMC/USP em regime de dedicação integral à docência e à pesquisa, atuando também como orientador pleno de mestrado e doutorado. Atua como consultor *ad hoc* da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq). Desenvolve pesquisa em Multimídia nas linhas: codificação de vídeo digital, vídeo 3D, recuperação de informação multimídia e análise multimodal.

Tiago Henrique Trojahn possui graduação em Ciências da Computação pela Universidade Federal de Pelotas (2010), mestrado (2014) e doutorado (2019) em Ciência da Computação e Matemática Computacional pela Universidade de São Paulo, campus São Carlos. Atualmente, é professor do Instituto Federal de São Paulo, campus São Carlos, em regime de dedicação exclusiva. Possui interesse nas áreas de segmentação de vídeo digital, multimodalidade, personalização e adaptação de conteúdo e desenvolvimento para web.

Rodrigo Mitsuo Kishi possui bacharelado em Ciência da Computação pela Universidade Federal de Mato Grosso do Sul (2007) e mestrado em Ciência da Computação pela Universidade Federal de Mato Grosso do Sul (2010). É professor da

Universidade Federal de Mato Grosso do Sul desde 2011. Cursa, atualmente, doutorado em Ciência da Computação na Universidade de São Paulo (USP). Tem experiência na área de Ciência da Computação, com ênfase em Sistemas Multimídia, Bioinformática, Teoria dos Grafos e Análise de Algoritmos.

Referências

Aggarwal C.C., Zhai C. A Survey of Text Classification Algorithms. In: Aggarwal C., Zhai C. (eds) Mining Text Data. Springer, Boston, MA, 2012. p. 163–222. ISBN 978-1-4614-3223-4. Disponível em: https://doi.org/10.1007/978-1-4614-3223-4_6.

Atrey, P. K.; Hossain, M. A.; Saddik, A. E.; Kankanhalli, M. S. Multimodal fusion for multimedia analysis: a survey. Multimedia Systems, v. 16, n. 6, p. 345–379, Nov 2010. ISSN 1432-1882.

Baeza-Yates, R and Ribeiro-Neto, B., Modern Informatin Retrieval, Addison-Wesley, 2008.

Baraldi L, Grana C, Cucchiara R (2017) Recognizing and Presenting the Storytelling Video Structure with Deep Multimodal Networks. IEEE Transactions on Multimedia 19(5):955{968, DOI 10.1109/TMM.2016.2644872, URL http://ieeexplore.ieee.org/document/7797131/

Blanken, H. M., Vries, A. P., Blok, H. E. and Feng, L., Multimedia Retrieval, Springer, 2010.

Boureau, Y.; Bach, F.; Lecun, Y.; Ponce, J. Learning mid-level features for recognition. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. San Francisco, CA, USA: IEEE, 2010. p. 2559–2566. ISSN 1063-6919

Bouyakoub, F. M. and Belkhir, A. (2008) "AdaMS: Na Adaptation Multimedia System for Heterogeneous Environments", In: New Technologies, Mobility and Security (NTMS), p. 1-5.

Carletti, V.; Foggia, P.; Percannella, G.; Saggese, A.; Strisciuglio, N.; Vento, M. Audio surveillance using a bag of aural words classifier. In: 2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance. Krakow, Poland: IEEE, 2013. p. 81–86.

Chasanis, V., Kalogeratos, A. and Likas, A. (2009) "Movie segmentation into scenes and chapters using locally weighted bag of visual words", International Conference on Image and Video Retrieval, p.35:1-35:7.

Csurka, G.; Dance, C. R.; Fan, L.; Willamowski, J.; Bray, C. Visual categorization with bags of keypoints. In: In Workshop on Statistical Learning in Computer Vision, ECCV. Czech Republic: Springer, 2004. p. 1–22.

Davis, S.; Mermelstein, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Transactions on Acoustics, Speech, and Signal Processing, v. 28, n. 4, p. 357–366, Aug 1980. ISSN 0096-3518

Del Fabro, M., Boszormenyi, L. (2013) State-of-the-art and future challenges in video scene detection: a survey. Multimedia Syst 19(5):427–454. https://doi.org/10.1007/s00530-013-0306-4

Grauman, K.; Leibe, B. Visual Object Recognition. 1st. ed. United States: Morgan & Claypool Publishers, 2011. ISBN 1598299689, 9781598299687

Han, B.; Wu, W. Video scene segmentation using a novel boundary evaluation criterion and dynamic programming. In: IEEE International Conference on Multimedia and Expo. [s.n.], 2011. p. 1–6. ISSN 1945-7871. Disponível em: https://ieeexplore.ieee.org/document/6012001/

Hanjalic, A.; Lagendijk, R. L.; Biemond, J. Automated high-level movie segmentation for advanced video-retrieval systems. IEEE Transactions on Circuits and Systems for Video Technology, v. 9, n. 4, p. 580–588, June 1999. ISSN 1051-8215.

Havaldar, P.; Medioni, G. Multimedia Systems: Algorithms, Standards, and Industry Practices. Cengage Learning; 2009. ISBN: 1418835943.

Hu, W., Xie, N., Li, L., Zeng, X. and Maybank, S. (2011) "A survey on visual content-based video indexing and retrieval". In: IEEE Transactions on Systems, Man and Cybernatics, v. 41, p. 797-819.

Iwan, L. and Thom, J. A. 2017. Temporal video segmentation: detecting the end-of-act in circus performance videos. Multimedia Tools and Applications 76, 1 (jan 2017), 1379–1401. https://doi.org/10.1007/s11042-015-3130-3

Kishi, R. M., Trojahn, T. H., Goularte, R. 2019. Correlation based feature fusion for the temporal video scene segmentation task. Multimedia Tools and Applications 78, 1 (jun 2019), 15623–15646. https://doi.org/10.1007/s11042-018-6959-4

Ko, Y.A. A new term-weighting scheme for text classification using the odds of positive and negative class probabilities. Journal of the Association for Information Science and Technology, Wiley-Blackwell, Hoboken, NJ, USA, v. 66, n. 12, p. 2553–2565, jan. 2015. ISSN 2330-1643. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.23338.

Koprinska, I.; Carrato, S. Temporal video segmentation: A survey. Signal Processing: Image Communication, v. 16, n. 5, p. 477 – 500, 2001. ISSN 0923-5965. Disponível em: http://www.sciencedirect.com/science/article/pii/S0923596500000114.

Leung, T.; Malik, J. Representing and recognizing the visual appearance of materials using three-dimensional textons. International Journal of Computer Vision, v. 43, n. 1, p. 29–44, 2001. ISSN 1573-1405. Disponível em: http://dx.doi.org/10.1023/A:1011126920638.

Leng, C.; Zhang, H.; Li, B.; Cai, G.; Pei, Z.; He, L. "Local feature descriptor for image matching: A Survey", IEEE Access, vol. 7, p. 6424-6434, 2019.

Lowe, D. G. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, v. 60, n. 2, p. 91–110, Nov 2004. ISSN 1573-1405. Disponível em: <https://doi.org/10.1023/B:VISI.0000029664.99615.94>

Lu, C.; Drew, M. S.; Au, J. Classification of summarized videos using hidden markov models on compressed chromaticity signatures. In: Proceedings of the Ninth ACM International Conference on Multimedia. New York, NY, USA: ACM, 2001. (MULTIMEDIA '01), p. 479–482. ISBN 1-58113-394-4.

Lu, Y., Sebe, N., Hytnen, R. and Tian, Q. (2011) "Personalization in multimídia retrieval: A survey". In: Multimedia Tools and Applications, v.51, p. 247-277.

Mandal, M. K. Multimedia Signals and Systems. Kluwer Academic Publishers, 2002. ISBN: 1402072708.

Martinet, J.; Sayad, I. E. Mid-level image descriptors. In: MA, Z. (Ed.). Intelligent Multimedia Databases and Information Retrieval: Advancing Applications and Technologies. USA: IGI Global, 2012. p. 46–60.

Pouyanfar, S., Yang, Y., Chen, S., Shyu, M. and Iyengar, S. S. 2018. Multimedia Big Data Analytics: A Survey. ACM Comput. Surv. 51, 1, Article 10 (January 2018), 34 pages. DOI: https://doi.org/10.1145/3150226

Rasheed, Z.; Shah, M. Scene detection in hollywood movies and tv shows. In: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings. Madison, WI, USA: IEEE, 2003. v. 2, p. II–343. ISSN 1063-6919.

Rezende, S. O.; Marcacini, R. M.; Moura, M. F. O uso da mineração de textos para extração e organização não supervisionada de conhecimento. Revista de Sistemas de Informação da FSMA, v. 7, p. 7–21, 2011. ISSN 19835604. Disponível em: http://www.fsma.edu.br/si/7edicao.html.

Rijsbergen, C. J. V. Information Retrieval. 2nd. ed. Newton, MA, USA: ButterworthHeinemann, 1979. ISBN 0408709294.

Sakarya, U. and Telatar, Z. (2010) "Video scene detection using graph-based representations" In: Signal Processing – Image Communication, v.25, p. 774-783.

Salton, G.; Buckley, C. Term-weighting approaches in automatic text retrieval. Information Processing and Management: an International Journal, Pergamon Press, Inc., Tarrytown, NY, USA, v. 24, n. 5, p. 513–523, ago. 1988. ISSN 0306-4573. Disponível em: https://dx.doi.org/10.1016/0306-4573(88)90021-0.

Saraceno, C.; Leonardi, R. Audio as a support to scene change detection and characterization of video sequences. In: 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing. Munich, Germany: IEEE, 1997. v. 4, p. 2597–2600 vol.4. ISSN 1520-6149.

Sharma, D.; Ali, I. A modified mfcc feature extraction technique for robust speaker recognition. In: 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI). Kochi, India: IEEE, 2015. p. 1052–1057

Smeulders, A.; Worring, M.; Santini, S.; Gupta, A.; Jain, R. Content-based image retrieval at the end of the early years. IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Computer Society, Washington DC, USA, v. 22, p. 1349-1380. ISSN 01628828. Disponível em: https://dl.acm.org/citation.cfm?id=357871.357873

Stevens, S. S.; Volkmann, J.; Newman, E. B. A scale for the measurement of the psychological magnitude pitch. The Journal of the Acoustical Society of America, v. 8, n. 3, p. 185–190, 1937. Disponível em: <http://scitation.aip.org/content/asa/journal/jasa/8/3/10.1121/1.1915893>.

Stockman, G.; Shapiro, L. G. Computer Vision. 1st. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. ISBN 0130307963.

Tapu, R. and Zaharia, T. (2011) "High level video temporal segmentation", International Conference on Advances in Visual Computing, p. 224-235.

Toffler, A., Future Shock, Bantam, 1984.

Tuytelaars, T. and Mikolajczyk, K. (2008) "Local invariant feature detectors: a survey". In: Foundations and Trends in Computer Graphics and Vision, v. 3, p. 177-280.

Uysal, A. K.; Gunal, S. The impact of preprocessing on text classification. Information Processing and Management: an International Journal, Pergamon Press, Inc., Tarrytown, NY, USA, v. 50, n. 1, p. 104–112, jan. 2014. ISSN 0306-4573. Disponível em: <https://dx.doi.org/10.1016/j.ipm.2013.08.006.

Vendrig, J.; Worring, M. Systematic evaluation of logical story unit segmentation. IEEE Transactions on Multimedia, v. 4, n. 4, p. 492–499, Dec 2002. ISSN 1520-9210.

Vestman, V.; Gowda, D.; Sahidullah, M.; Alku, P.; Kinnunen, T. Speaker recognition from whispered speech: A tutorial survey and an application of time-varying linear prediction, Speech Communication, v. 99, 2018, p. 62-79. ISSN 0167-6393, https://doi.org/10.1016/j.specom.2018.02.009.

Xie L, Shen J, Han J, Zhu L, Shao L (2017) Dynamic multi-view hashing for online image retrieval. In: Proceedings of the 26th international joint conference on artificial intelligence, IJCAI'17, pp 3133–3139. AAAI Press. http://dl.acm.org/citation.cfm?id=3172077.317232.

Yang, L.; Wang, Y.; Dunne, D.; Sobolev, M.; Naaman, M.; Estrin, D. More than just words: Modeling non-textual characteristics of podcasts. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining. New York, NY, USA: ACM, 2019. (WSDM '19), p. 276–284. ISBN 978-1-4503-5940-5. Disponível em: <http://doi.acm.org/10.1145/3289600.3290993>

Zhu, S. and Liu, Y. (2008) "Scene segmentation and semantic representation for high-level retrieval". In: IEEE Signal Processing Letters, v. 15, p. 713-716.