Ministério da Educação
Universidade Tecnológica Federal do Paraná
Pró-Reitoria de Pesquisa e Pós-Graduação

Relatório Final de Atividades

Reconhecimento de objetos a partir de sequências de vídeos

vinculado ao projeto

Métodos para análise, caracterização e classificação de bioimagens.

Thullyo Radeli Castilho

Voluntário

Engenharia de Computação

Data de ingresso no programa: 10/2013

Prof(º). Dr(º). Fabrício Martins Lopes

Área do Conhecimento: 1.03.00.00-7 - ciência da computação

CAMPUS CORNÉLIO PROCÓPIO, 2014

THULLYO RADELI CASTILHO
FABRÍCIO MARTINS LOPES

RELATÓRIO FINAL DE INICIAÇÃO TECNOLÓGICA

Relatório Técnico do Programa de Iniciação Tecnológica da Universidade Tecnológica Federal do Paraná.

CORNÉLIO PROCÓPIO, 2014

SUMÁRIO

INTRODUÇÃO	**2**
REVISÃO BIBLIOGRÁFICA	**2**
MATERIAIS E MÉTODOS	**7**
RESULTADOS E DISCUSSÕES	**10**
CONCLUSÕES	**11**
REFERÊNCIAS	**12**

INTRODUÇÃO

Informações podem ser geradas e transmitidas de diversas maneiras, uma forma muito comum são os vídeos digitais [1]. Eles são utilizados para diversos fins, mas em praticamente todo tipo de uso, há a sua geração em grandes quantidades e, consequentemente, uma significativa parte precisa ser armazenada e/ou processada, a exemplo de *websites* de entretenimento, como o *YouTube*, que recebe aproximadamente cem horas de vídeo a cada minuto [2], os quais podem estar diversos formatos[1], dificultando ainda mais o seu processamento.

Essa forma de armazenamento de informações é extremamente utilizada não somente para entretenimento, mas também e, principalmente, para segurança, desde a vigilância de casas e prédios, até a sua monetização com propagandas, como ocorre normalmente nos programas de televisão.

Há muitas vezes a necessidade de se extrair certos dados específicos, podendo ser numéricos, como no caso da simples detecção de uma tonalidade, ou algo mais complexo, como o reconhecimento de formas visuais, em que se pode exemplificar as ferramentas do tipo *OCR* (*Optical Character Recognition* – Reconhecimento Ótico de Caracteres), a qual se baseia em padrões de formato das letras de uma determinada base de dados, comparando-os com os encontrados nas imagens em que se deseja obter o reconhecimento dos caracteres [3].

De extrema importância para a segurança de grandes eventos, como nas olimpíadas, a vídeo vigilância gera uma gigantesca quantidade de imagens a serem visualizadas constantemente, o que torna a visualização de todos os pontos de vídeos ao mesmo tempo, em muitos casos, inviável.

Ao se considerar tal quantidade de informações geradas, faz-se necessária a automação da extração de possíveis informações ou características de interesse, como o reconhecimento de um fugitivo por uma câmera de segurança pelo seu rosto, o qual segue um processo similar ao *OCR*, utilizando-se de *softwares* para reconhecimento de faces em imagens [3,4].

Mas para se obter a automatização do processamento de tais vídeos, faz-se necessário uma grande quantidade de processamento computacional devido à enorme quantidade de imagens envolvidas, requerendo assim um método eficiente e robusto para a redução do tempo de processamento.

Nesse contexto, o objetivo do presente trabalho foi estudar um método de escolha e extração de *frames* de interesse de um determinado vídeo, por comparação de histogramas, de forma a reduzir a quantidade de dados (imagens) que seriam processadas, reduzindo-se significativamente o total de processamento necessário.

REVISÃO BIBLIOGRÁFICA

Espaços de cor. A cor é o resultado da percepção da luz pelos olhos, sendo a maioria das cores visíveis podendo ser representadas pela combinação de luzes monocromáticas nos comprimentos de onda do azul, vermelho e verde. Imagens coloridas são armazenadas nestas três componentes primárias, formando assim um espaço de cor. Os espaços mais comuns são o RGB e o YCbCr [3, 4].

O espaço RGB é formado pela sensação da soma ponderada dos componentes vermelho (R), verde (G) e azul (B), sendo este espaço normalmente utilizado para mostrar imagens coloridas na tela de computadores.

Já o YCbCr é utilizado para representação de vídeos digitais, sendo o cálculo de suas componentes com base no RGB apresentado abaixo:

Para todo pixel $p \in DI$, $0 \leq R(p) \leq 255$, $0 \leq G(p) \leq 255$, e $0 \leq B(p) \leq 255$ temos:

$$Y(p) = 0.257R(p) + 0.504G(p) + 0.098B(p) + 16 \quad (1)$$

$$Cr(p) = 0.439R(p) - 0.368G(p) - 0.071B(p) + 128 \quad (2)$$

$$Cb(p) = -0.148R(p) - 0.291G(p) + 0.439B(p) + 128 \quad (3)$$

onde $0 \leq Y(p) \leq 255$, $0 \leq Cb(p) \leq 255$, e $0 \leq Cr(p) \leq 255$.

$$R(p) = 1.164(Y(p) - 16) + 1.596(Cr(p) - 128) \quad (4)$$

$$G(p) = 1.164(Y(p) - 16) - 0.813(Cr(p) - 128) - 0.392(Cb(p) - 128) \quad (5)$$

$$B(p) = 1.164(Y(p) - 16) + 2.017(Cb(p) - 128) \quad (6)$$

Imagens digitais. Enquanto uma imagem gravada em um filme pode ser representada eletronicamente por uma onda analógica contínua, a imagem digital é representada por valores digitais obtidos a partir de amostras da forma analógica [3]. Uma imagem digital monocromática é representada por um conjunto de elementos chamados de *pixels* (*picture elements* ou elementos de imagem) que contém valores representando o nível de intensidade de cinza de cada ponto da imagem, que são armazenados juntos, formando um mapa de *bits*, o qual reproduz a imagem digitalmente, conhecido como *bit-map* o qual pode ser observado na Figura 1.

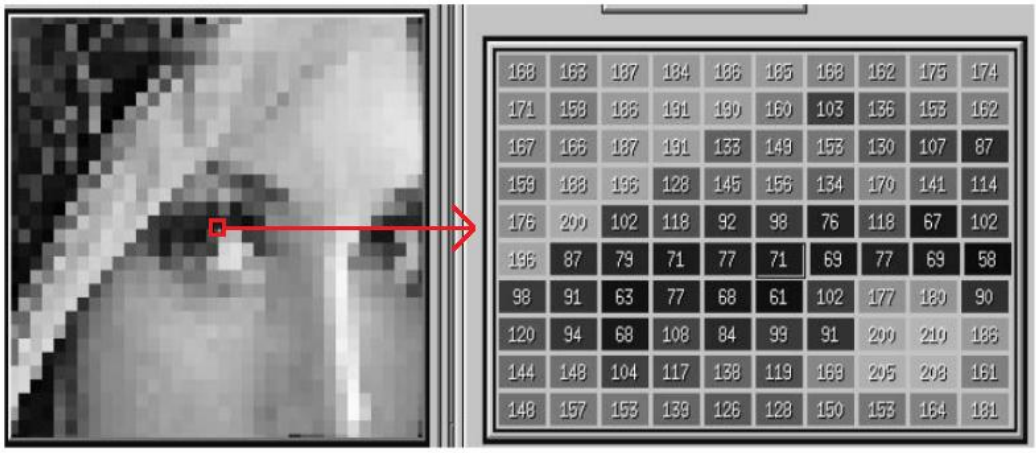


Figura 1. Ilustração do mapa de bits de uma imagem digital monocromática em uma região de interesse de 10x10 pixels [3].

Cada pixel pode ser representado como uma função *f(x,y)*, como pode ser visto na Figura 2. Sendo *x=0...m-1* e *y=0...n-1*, onde m é o total de *pixels* na horizontal da imagem e *y* o total de *pixels* na vertical. Para cada par ordenado há um valor L que é o nível de cinza daquele ponto(*pixel*), onde $L_{min} \leq f(x,y) \leq L_{max}$ *e* $L = L_{max} - L_{min} + 1$ [1].

Figura 2. Imagem digital como uma função *f(x,y)* [3].

Vídeos digitais. Vídeos são compostos por sequências de imagens, chamadas de *frames* que representam efetivamente os seus aspectos visuais podendo ou não tais imagens ser acompanhadas por sons ou músicas. A composição básica é geralmente uma gravação de imagens sequenciais (*frames*) distribuídas para exibição em uma determinada quantidade por segundo quando com o intuito de representar movimento [5,6].

Segue-se o mesmo princípio das animações de desenho animado, onde um mesmo objeto ou personagem é reproduzido dezenas de vezes por segundo (*frames* por segundo) com pequenas alterações em sua movimentação à cada quadro, de forma ordenada, produzindo assim a sensação de movimento do mesmo.

Para o armazenamento de tais vídeos, devido principalmente ao advento da internet, faz-se uso de técnicas de compressão e descompressão para se reduzir o seu tamanho, onde são usados os chamados *codecs* (compressores/descompressores), os quais são compostos de algoritmos matemáticos que executam tanto a compressão dos arquivos como posteriormente a sua descompressão. Os formatos (*codecs*) mais utilizados são *H264* e o *MPEG2*, em que estes se diferenciam pelos algoritmos utilizados, resultando em um tamanho final do arquivo diferente, de acordo com uma maior ou menor taxa de compressão. No presente trabalho utilizou-se para o *dataset* criado o *codec* MPEG2, o qual é explicado a seguir [6].

Tal *codec* se baseia em técnicas para reduzir a qualidade da imagem de modo imperceptível, aliando-se ainda à outras que não afetam a qualidade final da imagem, no qual se utiliza principalmente o método de eliminação de redundâncias na informação.

O algoritmo funciona em quatro principais etapas: redundância temporal, que consiste no aproveitamento da similaridade existente entre quadros sucessivos que formam um movimento (imagem dinâmica), assim ao invés de se enviar os dados de dois ou mais quadros completos, envia-se apenas a informação completa de um quadro *n* base e um vetor de movimento do quadro *n+1*; redundância espacial, a qual consiste na semelhança dos *pixels* adjacentes de uma imagem, como em um quadro azul com animal no centro, conseguindo-se assim uma boa compactação pela uniformização (diminuição da frequência espacial) da cor de fundo devido a baixa sensibilidade do olho humano para altas frequências espaciais, tornando assim a compressão quase imperceptível, como pode ser observado na Figura 3; após as duas primeiras etapas, são executados algoritmos de código de comprimento variável, os quais compactam o código do vídeo pela substituição de componentes repetitivos por palavras-chave; por fim há também um *buffer*, o qual serve para controlar o fluxo de dados do

MPEG2, em alguns casos implementado fisicamente nos hardwares de computadores para uma melhor otimização do desempenho de reprodução das mídias [4, 5, 6].

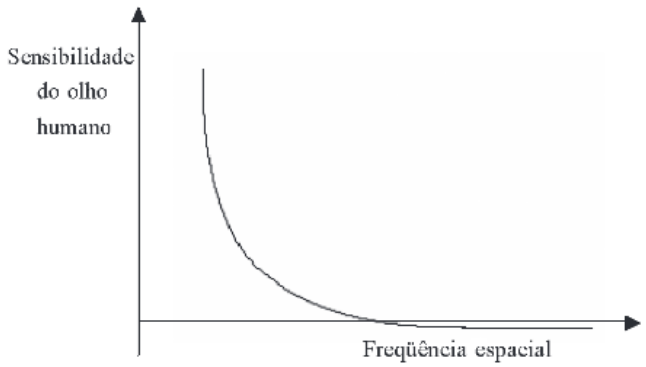


Figura 3. Curva de sensibilidade do olho humano com relação à frequência espacial de uma imagem. O olho humano deixa de perceber diferenças para grandes frequências [8].

Histogramas. De acordo com [3,4], o histograma de uma imagem digital com níveis de intensidade da sua cor no intervalo *[0, L-1]*, sendo que para o presente trabalho *L=256* como pode ser observado na Figura 4, é uma função discreta $f(i_n)=m_n$, onde i_n é um *n-ésimo* valor de intensidade e m_n a quantidade de pixels da imagem com a referida intensidade i_n.

$$[0, 255] = [0, 15] \cup [16, 31] \cup .... \cup [240, 255]$$
$$\mathbf{range} = \mathbf{bin}_1 \cup \mathbf{bin}_2 \cup .... \cup \mathbf{bin}_{n=15}$$

Figura 4. Composição de um histograma por *bins*, dado um range (intervalo) de cor [4].

Uma forma gráfica de representação de um histograma é mostrada na Figura 5, onde cada cor representa uma quantidade de pixels com determinada intensidade *L*, sendo, para um *L=256* níveis de cinza, um *bin* ($b_{i=1}...b_{16}$) representaria um intervalo de 16 níveis de cinza.

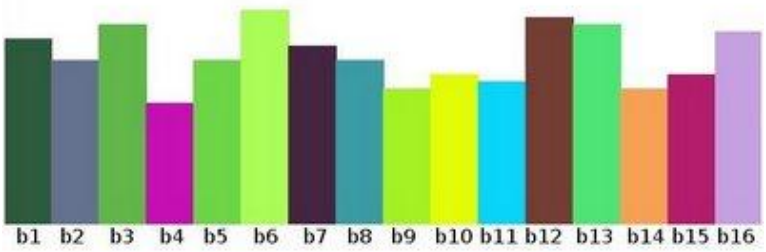


Figura 5. Representação da composição de um histograma por *bins* [11].

Distância Euclidiana. Em matemática, distância euclidiana (ou distância métrica) é a distância entre dois pontos, que pode ser provada pela aplicação repetida do teorema de Pitágoras [3,4]. Aplicando essa fórmula como distância, o espaço euclidiano torna-se um espaço métrico. A distância euclidiana entre os pontos, que para o presente projeto serão os *bins*, num espaço euclidiano n-dimensional, é definida como:

$$\sum_{i=0}^{n} \sqrt{(b_i - b_{i+1})^2} \qquad (7)$$

onde:

- *n* é o número de *bins*, em todos os histogramas calculados utilizaram-se 256 (0 à 255 níveis de cinza).

-b_i é o valor do *bin* de número *i*.

SIFT. O algoritmo *Scale-Invariant Feature Transform* é composto por duas partes distintas: o detector e o descritor. O detector é baseado em cálculos de diferença de Gaussianas e o descritor utiliza histogramas de gradientes orientados para descrever a vizinhança local dos pontos de interesse, localizados durante a primeira parte do algoritmo [7].

O algoritmo começa pela criação de uma pirâmide de imagens a partir da que está sendo processada, dividindo-a em oitavos(frações) que, por sua vez, são divididos em intervalos de imagens. Cada oitavo é formado por um número de imagens com a mesma dimensão, sendo que de um oitavo para o seu seguinte, a dimensão das imagens são reduzidas pela metade. Cada intervalo gerado é suavizado através de um filtro gaussiano $G(x, y, \sigma)$, o qual se aplica a cada imagem subsequente do intervalo, por um fator *k*, gerando $G(x,y, k\sigma)$.

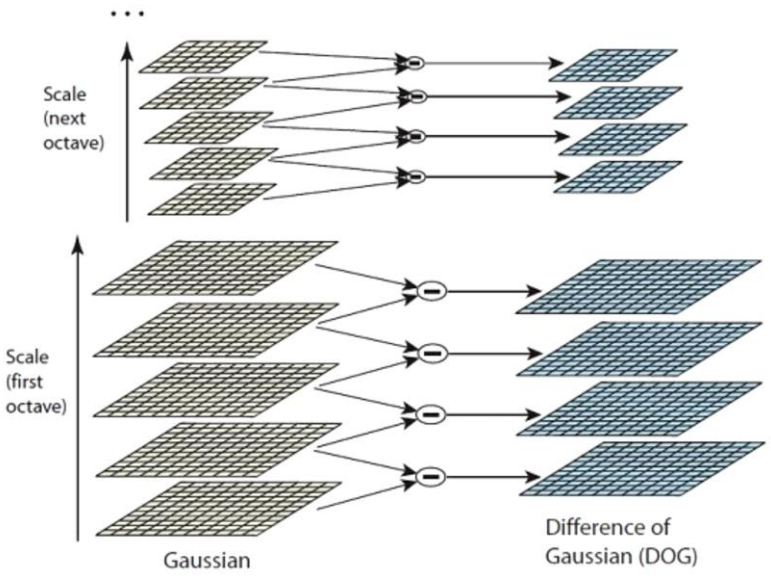


Figura 6. Diagrama esquemático da Diferença de Gaussianas [7].

Depois de obtidas as miniaturas, criam-se as imagens chamadas Diferenças das Gaussianas, ou DOG, do inglês *Diference of Gaussians*, como visto na Figura 6. Para isso, subtraem-se as imagens consecutivas na pirâmide. Usa-se as diferenças das gaussianas porque são uma boa aproximação do Laplaciano da Gaussiana [7, 8]. Para obter realmente invariância à escala, seria preciso utilizar o Laplaciano da Gaussiana [7], o que é muito custoso. Trabalhar com a Diferença das Gaussianas é uma forma bem mais eficiente de se obter invariação à escala.

O próximo passo é detectar os *keypoints* nas DOG's, o que é feito determinando se ele é um ponto extremo, comparando-o com seus vizinhos da sua imagem e nas imagens acima e abaixo(da pirâmide). Depois de calculados os pontos extremos de uma imagem, eliminamos os pontos ruins, os quais são aqueles que possuem baixo contraste ou estão localizados em uma aresta. Após identificados bons *keypoints*, determina-se orientação principal de cada um. Para isso, calculamos os gradientes, m(x; y), e orientações, (x; y), de cada pixel.

Posteriormente, dentro de uma área ao redor do *keypoint*, encontram-se as orientações e gradientes dos pixels da área e acumulamos os valores dos gradientes de acordo com a orientação em um histograma de 36 posições. Nesse caso, cada posição do histograma corresponde a um intervalo de 10º, assim, o primeiro intervalo será de 0º à 9º, o segundo de 10º à 19º, etc..

Determina-se então o ponto máximo do histograma e atribuímos essa orientação ao *keypoint*, e, além disso, para cada valor do histograma que esteja a 80% do valor máximo, criamos um novo *keypoint* no mesmo local, mas com outra orientação.

Finalmente, criamos um descritos para cada *keypoint*. Esse descritor é formado através de uma janela 16x16 ao redor do ponto e criamos novamente um histograma das orientações para essa área. Porém, essa área é dividida em 16 áreas 4x4 e é calculado o histograma para cada subaérea, sendo que neste caso, o histograma vai conter oito intervalos, onde cada um corresponde a 45^o. Cada entrada nos histogramas é suavizada por uma janela gaussiana centralizada no *keypoint*, para que pontos mais perto possuam mais influência. Por fim, o descritor do *keypoint* é formado pelos histogramas unidos em um só vetor de *8x4x4 = 128* dimensões.

Classificadores. Em reconhecimentos de padrões, classificador é um conjunto ordenado de categorias relacionadas usadas para agrupar dados de acordo com suas similaridades. Um classificador é, portanto, um algoritmo que dada várias entradas irá agrupá-las através do reconhecimento de suas características em comum únicas, onde para se chegar a um bom classificador faz-se uso de técnicas como o treinamento [4,9].

Através da visualização das características encontradas para uma determinada imagem, pode-se concluir, considerando um exemplo em que ela possua apenas um tomate e uma cenoura, uma forma precisa de se diferenciar tais vegetais é se utilizar das diferenças entre suas componentes RGB. Mas como se pretende utilizar resultados que sejam válidos também para outras imagens, que por sua vez podem conter diversas diferenças, há a necessidade de se encontrar um padrão comum à maioria das imagens que contenham tais objetos de interesse, o que é normalmente feito através da extração de características e posterior classificação de um conjunto de imagens, a este processo denomina-se treinamento [4].

***Framework* Weka.** O Weka (*Weikato Environment for Knowledge Analysis*) é uma coleção de algoritmos para aprendizado de máquina [10]. O framework possui um tipo de arquivo único para utilização, chamado de *arff*, o qual é constituído de um arquivo de texto ASCII para descrever um conjunto de atributos, os quais serão utilizados posteriormente por classificadores gerando-se os resultados finais.

Validação Cruzada. A divisão entre diversos conjuntos de dados, alguns para treinamentos e outros para testes se faz necessária para evitar resultados falsos, os quais seriam obtidos ao se testar os classificadores sobre a mesma base de dados sobre as quais eles extraíram suas informações iniciais. Considerando-se que o estudo seja sobre um conjunto grande de dados o suficiente para ser dividido por *n* partes mais ou menos iguais entre dados para treinamento e teste e, ainda gerar bons resultados, este processo resultaria em uma validação cruzada de *n*-vias[4].

Biblioteca OpenCV. O OpenCV é um projeto *open source*, ou seja, possuí código fonte livre, que contem dezenas de algoritmos para visão computacional já implementados. É também uma biblioteca com diversas funções para manipulação básica de imagens e vídeos, desde o carregamento de arquivos até a conversão entre um formato e outro [11]. Entre as funções encontradas na biblioteca, há as de conversão de imagens coloridas para a escala de cinza, o que foi extremamente utilizado no presente trabalho, além de funções mais complexas, como a implementação do algoritmo SIFT na linguagem C++ utilizado pelo método desenvolvido.

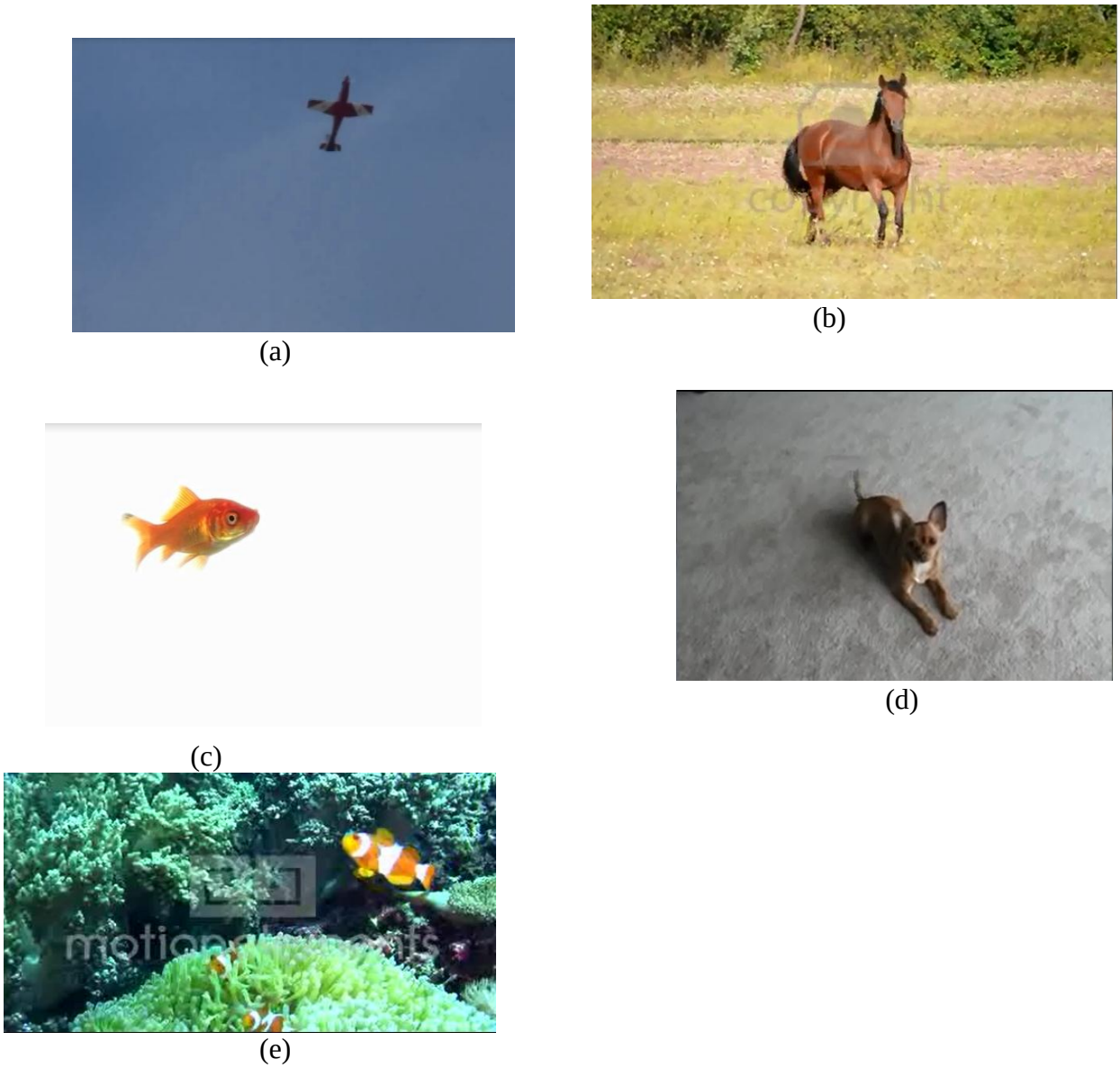
MATERIAIS E MÉTODOS

Materiais utilizados. Para a implementação do projeto proposto, foi utilizada a linguagem de programação C++ [12,13], fez-se também o uso da *IDE – Integrated Development Framework* ou Ambiente de Desenvolvimento Integrado Visual Studio 2013. O projeto foi desenvolvido para ser multi-plataforma, onde se utilizou o *framework* wxWidgets [14] para a

construção das interfaces gráfica e com o sistema operacional, proporcionando portabilidade ao código gerado, mesmo este gerado em ambiente Windows®.

O *hardware* utilizado tanto para o desenvolvimento como para processamento dos dados foi um computador do tipo desktop, o qual possui processador Intel Core i7, 16GB de memória RAM, HD de 1TB e monitor quadrado de 17’’.

Os vídeos utilizados consistem de um *dataset* criado por este trabalho, o qual conta com 70 vídeos sem direitos autorais (domínio público), que foram encontrados através de pesquisas pela internet por meio de buscadores, como o *Google*. A Figura 7 apresenta amostras representativas das classes de vídeos utilizadas.



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

Figura 7. Imagens representativas das classes de vídeos utilizadas: (a) Classe 1, com vídeos de aviões, (b) Classe 2, com vídeos de cavalos, (c) Classe 3, com vídeos de peixes dourados, (d) Classe 4, com vídeos de chihuahuas, (e) Classe 5, com vídeos de peixes-palhaço.

Tipo de pesquisa utilizada. A pesquisa realizada, de acordo com [3], foi de natureza

aplicada e descritiva objetivando o registro e a análise das características pertinentes ao processamento de vídeos, mais especificamente à extração de frames-chave por meio de um estudo de caso. O estudo de caso se deu através dos algoritmos já descritos, que foram executados tendo como entradas os vídeos já convertidos pelo mesmo programa para preto e branco.

Método estudado. O método desenvolvido consistiu de sete etapas principais conforme a Figura 8, cada etapa é explicada a seguir:

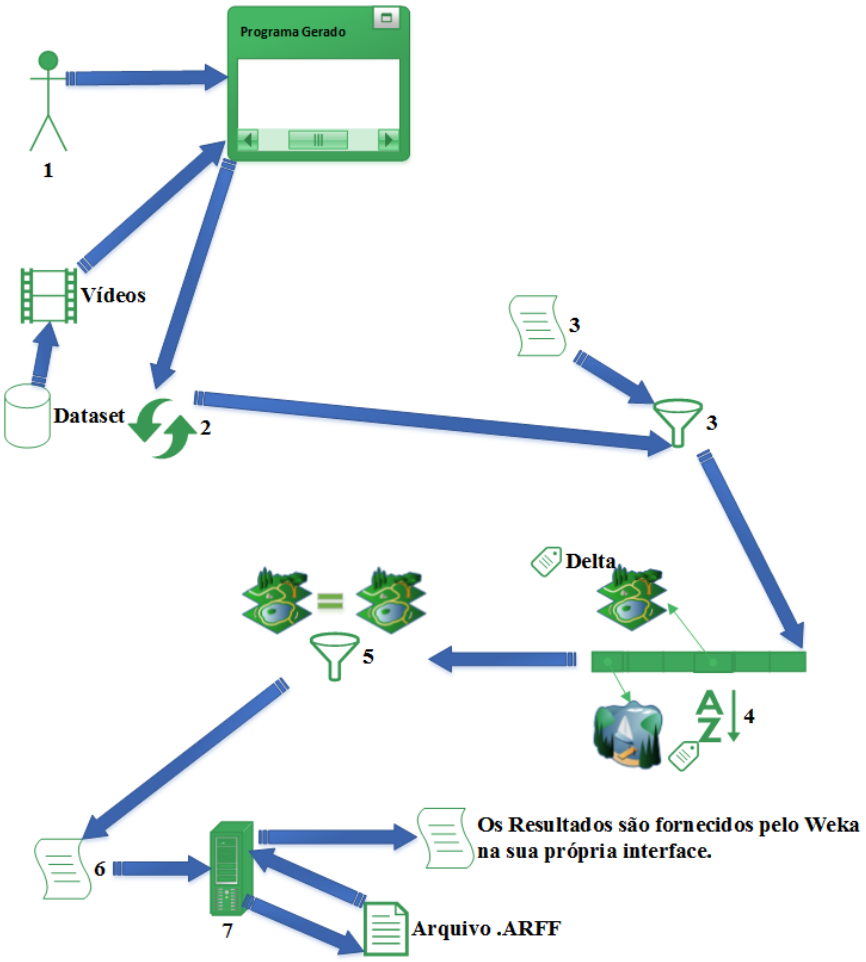


Figura 8. Diagrama esquemático do método desenvolvido.

Com o intuito de se obter um bom desempenho para a extração de *frames* únicos dos vídeos do *dataset* criado, foram estudados algoritmos tanto para o processamento como para a seleção das imagens mais diferentes, considerando como fator de comparação a primeira imagem a aparecer no vídeo.

Após o usuário informar ao programa desenvolvido a pasta onde o vídeo se encontra em (1), caso seja um arquivo válido, o usuário é solicitado a informar um valor inteiro *x*, o qual será utilizado para a definição da quantidade de frames intermediários que serão selecionados do vídeo em questão pelo processo (3). Após a validação do vídeo escolhido, este é convertido na etapa (2), caso seja colorido, para a escala de cinza (apenas um único valor inteiro para a descrição da cor), o que proporciona um ganho de desempenho considerável ao se trabalhar com um terço dos dados de uma imagem colorida (vermelho, verde e azul) [4].

Com o vídeo já convertido para a escala de cinza, prossegue-se à etapa (3) em que é realizada a comparação por distância euclidiana entre os histogramas, de todos os *frames* subsequentes ao primeiro, em relação ao histograma do primeiro *frame*. Cada comparação gera um valor, chamado aqui de *delta*, o qual representa distância entre os histogramas como uma forma de diferença entre as imagens comparadas. Todo o processo de cálculo dos histogramas e posterior comparação é realizado conforme o algoritmo.

Após o cálculo dos deltas, na etapa (4) há a ordenação em um vetor de todos os frames do vídeo, os quais são armazenados junto aos seus respectivos deltas. Posteriormente se faz a ordenação decrescente de tal vetor pelo valor dos deltas e são mantidos apenas os *x* frames do vetor. Com estes *frames*, o algoritmo prossegue agora comparando-os 2 a 2, de forma a filtrar aqueles repetidos durante a pré-seleção.

Já selecionados, há o processamento dos dados pelo SIFT gerando um arquivo *ARFF* com o qual após a detecção desses postos-chave, ou *keypoints*, em uma imagem, criamos um descritor para cada ponto e podemos, então, realizar comparações entre pontos. Dessa forma, é possível fazer um casamento, *matching*, entre diferentes imagens contendo o mesmo objeto que depois é processado pelo Weka.

RESULTADOS E DISCUSSÕES

O resultado do presente trabalho pode ser observado na Figura 9, o qual é o programa gerado ao final da iniciação tecnológica, que contém todos os algoritmos descritos para o método proposto de estudo.

Após o processamento dos *frames*-chave pelo SIFT, foi gerado um arquivo *arff* com os descritores de características, os quais descrevem as características dos *keypoints* detectados durante uma etapa anterior do algoritmo [7], procedendo-se ao processamento de tal arquivo pela ferramenta Weka, utilizando-se de classificadores [10], os quais foram treinados com os dados extraídos pelo *SIFT*, de forma a se encontrar um padrão para o reconhecimento das classes de vídeos estudadas, possibilitando a análise de desempenho do *matching* obtido por tais classificadores, obtendo assim os resultados apresentados na Tabela 2.

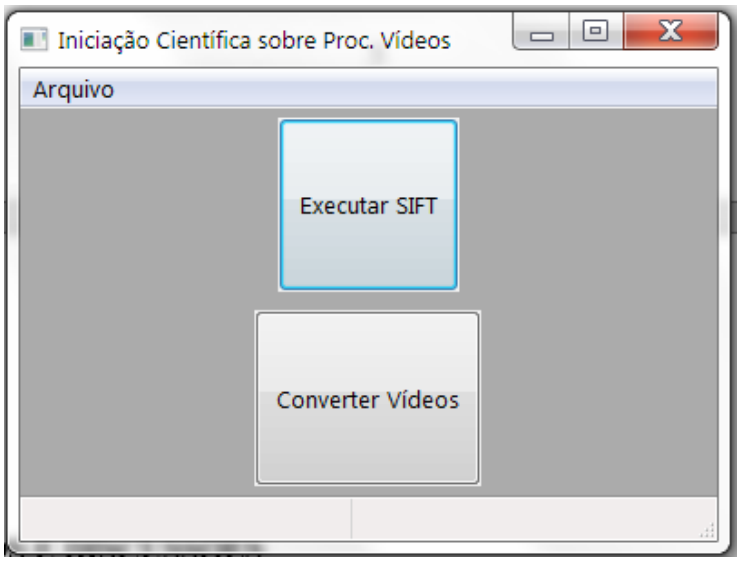


Figura 9. Tela inicial do programa final gerado.

Tabela 1. Quantidade de *keypoints* detectados pelo algoritmo SIFT para as diferentes classes de vídeos utilizadas.

	Classes enumeradas conforme já explicitado.				
	1	2	3	4	5
Quantidade de *keypoints* detectados	57	620	800	250	1200

Tabela 2. Resultados obtidos por meio da ferramenta Weka [9].

Classificadores (em ordem de precisão)	Verdadeiros Positivos	Falsos positivos	Precisão
Ramdom Forest	0,413	0,023	0,385
NaiveBayes	0,352	0,046	0,327
J48	0,322	0,052	0,301
VotedPerceptron	0,264	0,281	0,085
SMO	0,128	0,353	0,056

Como se pode observar na Tabela 2, os cinco classificadores apresentaram relativamente pouca quantidade de verdadeiros positivos, o que indica uma falha na eficiência dos mesmos, seja por causa do treinamento com imagens insuficientes, seja pela má qualidade do *dataset* ou ainda devido a parâmetros fixos estabelecidos.

Quanto aos parâmetros, estes foram definidos como base da metodologia proposta, os quais, se configurados incorretamente, poderiam interferir seriamente nos resultados, sendo eles: a definição de uma quantidade *x* de frames baixa demais a serem selecionados, o que poderia resultar em um treinamento fraco e, o estabelecimento do primeiro frame do vídeo a ser sempre considerado como um frame chave, o que em certas condições poderia ocasionar, como no caso de um primeiro frame de cores apenas pretas, a seleção de possivelmente todos os frames subsequentes aos primeiro, gerando-se assim resultados díspares, já que apenas parte do vídeo foi considerada para o treinamento dos classificadores.

CONCLUSÕES

Ao se considerar os desafios inerentes ao processamento de vídeos, como a quantidade de parâmetros a serem considerados, principalmente devido às peculiaridades de cada um, que se relaciona diretamente com o *dataset* criado, pode-se concluir que o presente método estudado mostrou-se extremamente versátil, apesar das suas diversas deficiências, principalmente se considerarmos que tal abordagem não considera aspectos específicos de cada vídeo, o que possibilita sua aplicação nos mais diferentes tipos de vídeos existentes.

Como forma de aprimorar o presente método, poderia se considerar uma melhor forma de pré-seleção das imagens do vídeo, pois a atual abordagem considera como fator principal o primeiro *frame*, o que se provou pouco eficiente em termos de resultados, devido ao fato de que as imagens selecionadas de forma pouco precisa

acabam por impactar todos os processos subsequentes do método.

Além disso, faz-se necessária a observação de que o *dataset* utilizado foi relativamente pequeno, possuindo apenas setenta vídeos no total, dividido em cinco classes de animais diferentes, além disso, os vídeos em si não apresentam total padronização em termos da compressão ou categorias utilizadas, o que, se melhorado, poderia impactar significativamente de forma positiva nos resultados.

Dados os resultados do presente trabalho, verifica-se a possibilidade de sua futura continuação e expansão. Os possíveis pontos a se considerar para uma futura melhora seriam: aumentar e padronizar o *dataset*, de forma a melhorar a qualidade dos dados estudados; formular um método alternativo para a comparação dos *frames* na durante a primeira etapa de seleção, e, desenvolver um método para a eliminação de imagens sem conteúdos significativos, como é o caso de imagens puramente pretas ou brancas.

REFERÊNCIAS

[1] NETTO, A. A. de O. **Metodologia da Pesquisa Científica:** Guia prático para apresentação de trabalhos acadêmicos. 3ª ed. Editora Atual. Florianópolis: Axcell Books, 2008.

[2] GOOGLE. **Estatísticas do *youtube***. YouTube – Google Corp., Disponível em: <http://www.cs.waikato.ac.nz/ml/weka/>. Acesso em 14 set. 2014.

[3] GONZALEZ, R. C. **Processamento de Imagens Digitais**. 3ª ed. São Paulo: Edgard Blucher, 2000.

[4] PARKER, J. R. ***Algorithms for image processing and computer vision***. 2ª ed. Indianapolis: Wiley Publishing, Inc., 2011.

[5] MORIMOTO, H. M.; Santos, T. T. **Segmentação, Indexação e Recuperação de Vídeo Utilizando OpenCV**. WVC: IV workshop de Visão Computacional, Baurú, 2008.

[6] FILHO, W. P. P. **Multimídia:** conceitos e aplicações. 2ª ed. Rio de Janeiro: LTC, 2011.

[7] LOWE, D. G. ***Distinctive image features from scale-invariant keypoints*****.** *International Journal of Computer Vision,* 60, 2, pp. 91-110, 2004.

[8] CESAR, R. M.; Costa, L. F. ***Shape classification and analysis:*** *theory and practice*. 2ª ed. Flórida: CRC Press, 2009.

[9] BISHOP, C. M. ***Pattern recognition and machine learning***. Nova Iorque: Springer Science, 2006.

[10] WAIKATO, U. O. ***Weka data mining software in java***. Weka – The University of Waikato, Disponível em: <http://www.cs.waikato.ac.nz/ml/weka/>. Acesso em 02 set. 2014.

[11] ITSEEZ. ***Open source computer vision library***. OpenCV – Itseez, Disponível em: <http://opencv.org/documentation.html>. Acesso em 02 set. 2014.

[12] DEITEL, H. M.; DEITEL, P.J. **C++ como programar**. 5ª ed. São Paulo: Pearson Education do Brasil, 2006.

[13] MIZRAHI, V. V. **Treinamento em linguagem C++**. 2ª ed.São Paulo: Pearson Education do Brasil, 2006.

[14] WXWIDGETS. ***Cross-platform GUI library***. wxWidgets – Open Source Project. Disponível em: <http://www.wxwidgets.org/docs/book/>. Acesso em 02 set. 2014.