

Detecção de Imagens Parecidas

Tiago Rodrigues
Universidade Federal de Minas Gerais
tiagorm@dcc.ufmg.br

Bruno Santos
Universidade Federal de Minas Gerais
bnsantos@dcc.ufmg.br

1. Introdução

A detecção de imagens parecidas é um problema amplamente estudado devido a sua importância. Como por exemplo, não é desejável realizar buscas por imagens e só encontrar imagens parecidas nos resultados das máquinas de buscas. Outro fator de interesse em algoritmos de detecção de imagens parecidas é o fato de poder identificar imagens que violam os direitos autorais e se encontram espalhadas na Web. Através de imagens parecidas ainda se pode detectar spam em imagens, já que este tipo de imagem geralmente são criadas baseadas em modelos existentes. Enfim, todas estas aplicações necessitam de algoritmos para detecção de imagens parecidas em larga escala. É um problema bastante difícil e que apresenta uma série de desafios, como por exemplo, a necessidade de se obter uma representação eficiente e compacta para as imagens, além de obter uma alta taxa de acerto e ser bastante eficiente.

Podemos considerar que duas imagens são parecidas se elas são iguais ou muito próximas. No entanto, a definição do termo imagens parecidas é muito subjetiva pois depende de quais variações fotométricas e geométricas são aceitáveis. As imagens parecidas são agrupadas em 3 categorias na literatura:

- **Cena:** Diferem na adição, oclusão ou movimento de objetos em primeiro plano.
- **Câmera:** Variações do dispositivo de captura, como ângulo de visão.
- **Imagem:** variações devido a operações de edição digital, como cor, contraste, alteração da resolução.

O presente trabalho visa a detecção de imagens parecidas na categoria de *Imagem*. Construímos uma base de dados realizando uma série de transformações em um conjunto de 10 imagens selecionadas aleatoriamente. A figura 1 mostra um exemplo de um grupo de imagens parecidas após as transformações aplicadas. Abaixo segue uma lista com alguns exemplos de transformações aplicadas:

- **Rotação:** 90°, 180° e 270°.
- **Escala:** aumenta ou diminui a imagem em 2, 4 e 8 vezes.

- **Contraste:** aumenta ou diminui.
- **Equalizar**
- **Normalizar**

Após a criação da base de dados extraímos algumas características para criar uma representação mais compacta da imagem e realizar a detecção. As características extraídas são armazenadas em um vetor de características, que por analogia pode ser considerado como a assinatura ou impressão digital da imagem. Algumas das características extraídas foram:

- Número de cores únicas.
- Média de cada canal, RGB e HSV.
- Histograma de cores, 64 bins.
- Momento de cor RGB.
- Quantidade de pixels nas bordas detectadas usando Canny.
- Quantidade de corners detectados usando Harris.

Depois de representar as imagens de uma forma mais compacta, o passo seguinte foi utilizar uma técnica de classificação para detectar a qual grupo de imagens parecidas uma determinada imagem pertence. Uma segunda abordagem utilizada foi comparar par a par duas imagens e utilizar o classificador para prever se ambas são parecidas ou diferentes.

O restante do artigo está organizado da seguinte forma: na seção 2 apresentamos a revisão bibliográfica do estado-da-arte em detecção de imagens parecidas e problemas relacionados. Na seção 3 explicamos nossa metodologia. Na seção 4 explicamos o desenvolvimento do trabalho, dando detalhes da realização de cada etapa da metodologia proposta. Na seção 5 apresentamos e discutimos os resultados obtidos, e na seção 6 apresentamos nossa conclusão e indicamos possíveis direções de trabalhos futuros.

2. Revisão Bibliográfica

No trabalho [16] os autores propõem um sistema para detecção de imagens parecidas e recuperação de



Figura 1. Exemplo de grupo de imagens parecidas da categoria *Imagem*

sub-imagens. O trabalho é motivado pela procura de imagens possivelmente violadas de direitos autorais e detecção de imagens forjadas. Definem uma representação específica para as imagens utilizando descritores locais que, segundo os autores, conseguem fazer um parelhamento de alta qualidade mesmo quando as imagens sofrem várias transformações. Utilizaram o PCA-SIFT [15] para extrair os descritores locais e fizeram uma verificação geométrica nas características encontradas. Ao invés de utilizar uma aproximação de similaridade ad-hoc, implementaram um hashing sensível a localidades [12] para indexar os descritores locais e lidar com o grande número de características extraídas das imagens, permitindo examinar uma pequena fração da base de dados. Também construíram índices para otimizar o acesso a discos e a busca pelas requisições dos descritores locais em um passo único, permitindo trabalhar com grandes coleções de imagens em tempo real. O sistema proposto foi quase perfeito na detecção das imagens, conseguindo uma precisão de 100% e uma revocação de 99,85%. No entanto a base de testes foi criada a partir de uma série de 40 modificações feitas em um conjunto de imagens de uma galeria de arte, o que pode ter influenciado na altíssima precisão conseguida.

Em [11] é proposta uma abordagem para detecção de imagens parecidas onde fazem uma adaptação de técnicas de agrupamento de textos duplicados combinando com técnicas de SIFT. Em um trabalho anterior [9] propuseram uma abordagem de identificação de pares de imagens parecidas. Segundo os autores, existem na literatura diversos métodos propostos para descritores locais, como o SIFT, PCA-SIFT, SURF [13] e GLOH [18], mas não há um consenso sobre qual deles é melhor para o problema da detecção de imagens parecidas. Escolheram o PCA-SIFT pois esse se mostrou mais compacto e mais com acurácia mais alta do que outros [15, 16, 10]. Para avaliar a metodologia foram utilizadas três bases de dados, uma criada artificialmente, outra a partir de coleções disponíveis na Web, e uma terceira criada a partir de buscas na Web. A metodolo-

gia sugerida no trabalho é bem interessante, combinando técnicas de visão computacional com métodos de agrupamento de textos, e apresentou resultados satisfatórios com 3 grandes e diferentes coleções de dados. Como trabalhos futuros, os autores pretendem avaliar e comparar outras técnicas de agrupamento mais predominantes como o k-means.

Outra metodologia para o problema da detecção de imagens parecidas foi apresentado em [14]. Primeiramente os autores apresentaram um modelo simples que lida com mudanças entre duas fotografias de aproximadamente uma mesma cena e propuseram uma nova classificação para os diferentes tipos de imagens parecidas. Elas foram classificadas em termos da cena (o que é fotografado), da câmera (o dispositivo de captura) e da imagem (a representação digital). Depois os autores apresentaram uma metodologia para automaticamente diferenciar entre imagens não-identicas e muito similares, baseados na combinação de características de baixo nível, detecção de objetos e conhecimento do domínio. Utilizaram para os testes uma base de 255 fotografias, que foram etiquetadas par a par por 10 pessoas diferentes, mostrando que decidir se duas imagens são duplicatas é um problema muito subjetivo, já que somente 43% dos pares tiveram 100% de concordância. Fizeram os testes em dois conjuntos extraídos da base: um com as imagens que tiveram 100% de concordância (64% de precisão e 97% de revocação) e outro com 100 pares com diferentes níveis de concordância (52% de precisão e 70% de revocação), obtendo resultados não muito precisos.

No trabalho [10] foi utilizada a técnica PCA-SIFT para detectar imagens parecidas, onde os autores propõem uma poda nos descritores da imagem utilizados no SIFT e PCA-SIFT para ganhar em eficiência sem perder exatidão do algoritmo. Os resultados obtidos foram satisfatórios como uma pequena perda na precisão mas grande economia de memória, e o algoritmo rodou para um acervo de 100.000 imagens, o que não é possível no PCA-SIFT.

O objetivo do trabalho [19] é a comparação de técnicas de detecção de imagens, como por exemplo, comparar métodos como PCA-SIFT e Dynamic Partial Func-

tions contra métodos que utilizam uma combinação de descritores locais e *hash-based counting*. A motivação deste trabalho surgiu dos primeiros resultados de uma pesquisa no Google Image ¹ por "miles davis kind of blue", que retornaram a mesma imagem da capa do álbum com pequenas alterações. Além da comparação destas técnicas o artigo também realiza uma pesquisa a cerca das variações que podem ser encontradas nas imagens da web. Mostraram que o PCA-SIFT tem a maior precisão mas perde em eficiência e não pode ser utilizado na prática para pesquisas web.

O artigo [8] propoe duas novas complexas medidas de similaridade que são inspiradas pelos sistemas de pesquisa de imagem. E estas métricas propostas podem ser eficientemente computadas usando um método de min-Hash modificado sem apresentar nenhum custo computacional adicional. A primeira métrica representa a imagem como um conjunto ponderado de palavras visuais (idf - "inverse document frequency"). A função de similaridade é um conjunto ponderado de interseções dos conjuntos de palavras visuais, palavras com peso menos contribuem menos para a função de similaridade. E um segundo passo depois da ponderação tf-idf, propõe-se estender a medida de similaridade para computar um histograma ponderado das interseções, o qual é capaz de levar em conta a frequência do termo. Os resultados foram obtidos utilizando a base de dados TrecVid 2006 e a base de dados da Universidade de Kentucky. Como resultados o artigo comprova que as técnicas propostas é capaz de detectar imagens com altos valores de similaridade em um tempo proporcional ao número de imagens na base de dados. O histograma ponderado das interseções é a melhor medida de similaridade (das três analisadas no artigo) tanto na qualidade dos resultados como na eficiência.

O trabalho [17] extrai características das imagens para detectar um certo padrão nas imagens utilizadas como spam, além de previamente realizar um estudo sobre as modalidades de imagens utilizadas em spam e constatar que existe um certo padrão nas imagens utilizadas como spam. Após extrair as características as comparações são feitas via o classificador SVM. As características extraídas por este artigo são divididas em características de cor, características de textura e características de forma. Das características de cor extraídas são: média dos canais do RGB e do HSV, histograma de cores, "Color Moment" e vetor de coerência de cores. De textura foram extraídas as seguintes características: Auto-correlação, frequência de arestas, "primitive length" e matrizes de co-ocorrência. As características de forma utilizadas são: Momento geométrico, ecentricidade e Legendre e Zernike momento. O método proposto pelo presente trabalho obteve uma precisão de cerca de 95% em todos os casos.

3. Metodologia

Baseando-se nos estudos realizados dos trabalhos relacionados na revisão bibliográfica, percebemos que a grande maioria segue algumas etapas básicas para se resolver o problema da detecção de imagens parecidas. Assim, propomos neste trabalho uma abordagem diferente, mas seguindo as mesmas etapas básicas para resolver o problema, as quais discutimos com mais detalhes em seguida:

- **Criação da base de dados:** Primeiramente vamos construir uma base de dados para trabalhar e testar o mecanismo de detecção de imagens parecidas. Para reduzir o escopo do problema, vamos trabalhar apenas com a categoria de imagens parecidas *Imagens*, que são aquelas que variam em decorrer de mudanças digitais aplicadas às imagens como cores e resolução, por exemplo. A partir de uma base de 5,304 imagens no formato PNG disponível na Web ², selecionamos aleatoriamente algumas imagens para aplicar transformações e construir nosso conjunto de testes. Mais detalhes serão explicados na seção 4.1.
- **Extração de características das imagens:** A característica de uma imagem é um pedaço de informação que é útil para alguma tarefa computacional. Não é viável utilizar todos os pixels das imagens para se detectar as que são parecidas, por motivos de desempenho e espaço de armazenamento. Assim, algumas características serão extraídas das imagens e armazenadas em um vetor que pode ser comparado a uma assinatura daquela imagem. Com os vetores podemos passar para o próximo passo, que trata da detecção das imagens parecidas.
- **Uso de classificação para detectar imagens parecidas:** Depois de construir um vetor de características que representa uma imagem, o passo seguinte será a utilização de um classificador para detectar as imagens parecidas. A tarefa de classificação consiste em 2 etapas: treino e teste. No treino, parte da base de dados é passada para o classificador, que vai criar um modelo a partir do que "aprendeu" com os exemplos. No teste será passado uma outra parte dos dados, sem falar qual é a classe que cada imagem pertence, para o classificador prever qual é a classe de cada imagem do teste a partir do modelo que "aprendeu" no treino. No nosso mecanismo de detecção vamos criar duas abordagens para resolver o problema. Na primeira, a cada conjunto de imagens parecidas será atribuída uma classe. Na segunda, vamos comparar pares de imagens e classificar em duas classes: parecidas e diferentes.

¹ <http://images.google.com/>

² <http://pascallin.ecs.soton.ac.uk/challenges/VOC/databases.html>

- **Análise dos resultados obtidos:** Por último iremos fazer uma análise dos resultados obtidos pelo mecanismo utilizado para detectar imagens parecidas. Com o resultado do classificador poderemos comparar a eficiência do método proposto. Para isto, vamos utilizar duas métricas padrões de recuperação de informação: precisão e revocação.

Na próxima sessão vamos discutir sobre o desenvolvimento do trabalho, explicando em mais detalhes cada etapa da metodologia proposta.

4. Desenvolvimento

Nesta seção descreveremos como foi feito o desenvolvimento da metodologia proposta na seção anterior.

4.1. Criação da base de dados

Para criar nossa base de dados para testar nosso mecanismo de detecção de imagens duplicadas, utilizamos um conjunto de imagens disponíveis na Web [2]. A base de dados do “*The 2006 PASCAL Visual Objects Classes Challenge (VOC2006)*” [1] é composta por 5,304 imagens coletadas de fotógrafos profissionais, do sistema de compartilhamento de fotografias digitais Flickr [3] e da base de dados da Microsoft Research. São imagens de bicicletas, ônibus, gatos, carros, vacas, cachorros, cavalos, motocicletas, pessoas e ovelhas. Todas estão no formato PNG.

Os conjuntos de imagens duplicadas foram criados após aplicar uma série de transformações em imagens selecionadas aleatoriamente dentro dessas 5,304 imagens obtidas. Selecionamos 10 imagens para aplicar as transformações usando o programa Image Magick [4], seguindo a metodologia do artigo [16].

Foram aplicadas um total de 43 transformações em cada uma das 10 imagens selecionadas. Segue a lista das transformações:

- **Rotacionar:** rotaciona a imagem em 90°, 180° e 270°.
- **Escalar:** aumenta e diminui a imagem em 2, 4 e 8 vezes o tamanho original. Depois redimensiona a imagem para o tamanho original.
- **Contraste:** aumenta e diminui o contraste utilizando os valores padrões do programa Image Magick.
- **Cortar:** cortar 5, 10, 20 e 30% da imagem, redimensionando depois ao tamanho original e preservando o centro da imagem.
- **Despeckling:** aplica a função do programa Image Magick com os valores padrões.
- **Redução:** diminui o tamanho da imagem para 10, 20, 30, 40, 50, 70 e 90% do tamanho original.
- **Coloração:** aumenta em 10% os valores das tonalidades de vermelho, verde e azul.
- **Moldura:** insere uma moldura na imagem de 10% do tamanho da figura original. Utiliza 4 cores aleatórias.
- **Distorção:** aplica a função do programa Image Magick com os valores padrões.
- **Embaçar:** aplica a função do programa Image Magick com os valores padrões.
- **Equalizar:** aplica a função do programa Image Magick com os valores padrões.
- **Normalizar:** aplica a função do programa Image Magick com os valores padrões.
- **Saturação:** modifica a saturação da imagem em 70, 80, 90, 110 e 120%.
- **Intensidade:** modifica os valores de intensidade da imagem em 80, 90, 110 e 120%.

No total, trabalhamos com uma base de dados que contém 10 conjuntos de 44 imagens parecidas, somando 440 imagens. Na seção seguinte explicamos como extraímos características para representar as imagens de uma maneira mais compacta.

4.2. Criação do vetor de características

Nesta seção descrevemos quais características extraímos das imagens para criar um vetor de características que as represente de uma forma compacta e eficiente. Muitas das características extraídas foram baseadas no artigo ??, e algumas foram escolhidas por julgarmos relevantes para discriminar as imagens e contribuir para a detecção de imagens parecidas. A grande maioria das características foram extraídas utilizando o MATLAB.

Segue abaixo uma lista das características extraídas de cada imagem da nossa base de dados:

- **Número de cores únicas na imagem:** função do programa Image Magick que conta o número de cores únicas que a imagem possui, nos dando uma característica global da imagem.
- **RGB médio:** média dos valores de intensidade para cada canal no sistema de cores RGB, contribuindo com 3 características das imagens.
- **Histograma de cores:** extraímos o histograma de cores das imagens utilizando 64 bins, nos dando 64 características das imagens.
- **Color Moment RGB:** esta característica baseia-se no fato de que os valores de intensidade das imagens possam ser vistos como uma distribuição de probabilidade estatística. Assim, calculamos 3 métricas para cada canal: mean, standard deviation e skewness, o que resultou em 9 características.

- **HSV:** utilizamos o esquema de cores HSV e extraímos o valor médio da intensidade em cada canal, contribuindo com 3 características no vetor.
- **Bordas:** utilizando o detector de bordas “canny”, contabilizamos o número de pixels nas bordas detectadas.
- **Quinas:** utilizamos o algoritmo de “harris” para detectar o número de corners das imagens, nos dando uma característica no vetor.
- **Textura de primeira ordem:** utilizamos também algumas características de textura para diferenciar as imagens e possibilitar a detecção de imagens parecidas. Utilizamos 6 características de textura: média, variância, *skewness*, *kurtosis*, mediana e moda.

A figura 2 mostra um exemplo do vetor de características formado. Cada linha é uma imagem, e cada coluna apresenta o valor de cada uma das características extraídas para cada imagem.

4.3. Mecanismo de detecção

Nosso mecanismo de detecção é baseado em classificação, uma tarefa que vem da área de Mineração de Dados onde o objetivo é classificar objetos em categorias pré-definidas de acordo com as características desse objeto. A tarefa de classificação é constituída de 2 fases. Na primeira fase, chamada de treino, parte da base de dados é passada para o algoritmo, descrevendo qual a categoria que cada objeto pertence e os respectivos vetores de características. O algoritmo então vai “aprender” com os dados, criando um modelo matemático. Na segunda fase, o teste, o restante da base de dados é passada para o algoritmo, sem descrever qual a categoria de cada objeto e passando apenas o vetor de característica. Baseado no modelo gerado no treino, o algoritmo vai então classificar cada objeto em alguma classe.

Utilizamos o algoritmo *Support Vector Machine* (SVM) [7], um algoritmo bastante conhecido e que tem um desempenho competitivo com o estado-da-arte. Utilizamos a implementação *LibSVM* 2.86 [5], pois essa implementação possui algoritmos que fazem uma busca pelos melhores parâmetros para se obter um melhor desempenho com o classificador SVM. O kernel do SVM utilizado foi o *Radial Basis Function* (RBF) [6], que é um kernel não linear.

Para validar nossos experimentos, utilizamos o esquema de *10-fold cross validation*, um procedimento estatístico bastante utilizado. Nesse esquema, a base de dados é embaralhada e dividida em 10 partes de tamanho igual, ou seja, $10/n$, onde n é o número de elementos na base. Então, 9 partes são juntadas e utilizadas como treino, e a parte restante é utilizada para o teste. Repete-se o experimento 10 vezes, com cada vez uma parte sendo o teste.

Utilizamos as métricas *precisão* e *revocação*, padrões na área de Recuperação de Informação. Sendo vp a taxa de verdadeiros positivos, fp a taxa de falsos positivos, e fn a taxa de falsos negativos, a métrica de precisão é dada por $vp/(vp + fp)$ e a revocação é dada por $vp/(vp + fn)$. Em outras palavras, a precisão nos mostra de todos os objetos classificados em cada classe, quantos foram corretamente classificados. Já a revocação nos mostra de todos os objetos de cada classe, quantos foram corretamente classificados.

Criamos duas abordagens para resolver o problema da detecção de imagens parecidas, que serão explicadas na seções 4.3.1 e 4.3.2.

4.3.1. Classificação Multi-classe Nossa primeira abordagem foi fazer uma classificação multi-classe para resolver o problema da detecção das imagens parecidas. Nessa abordagem vamos classificar cada imagem em seu respectivo grupo de imagens parecidas. Como nossa base possui 10 grupos de 44 imagens parecidas, nosso classificador vai classificar as imagens em 10 classes. Para realizar esta tarefa passamos para o classificador o vetor de características de cada imagem e a classe a que a imagem pertence. Na figura 2, que é um exemplo de entrada para o classificador SVM, cada linha representa o vetor de características de cada imagem, e a primeira coluna é a classe que a imagem pertence.

4.3.2. Classificação Binária A nossa segunda abordagem foi comparar um par de imagens e classificá-las em duas classes: parecidas ou diferentes. Para cada par de imagens comparadas calculamos o valor absoluto da diferença dos vetores de características das duas imagens, criando um vetor único de características.

Como nosso mecanismo está demorando muito para ser executado, e nessa abordagem o número de objetos cresce muito, tivemos que criar uma base reduzida. Nessa base, pegamos cada uma das 10 imagens originais e utilizamos apenas 4 das transformações que foram aplicadas: reduzir o tamanho para metade e voltar ao tamanho original, aumentar o contraste, embassar e reduzir a intensidade para 90% da original. No total a base reduzida é composta por 10 conjuntos de 5 imagens parecidas, ou seja, 50 imagens. Além disso, selecionamos mais 4 imagens aleatoriamente dentre as 5,304 da base do VOC 2006 para serem comparadas com cada uma das 50 da base reduzida e formarem os exemplos de imagens diferentes. Para formar os exemplos da classe parecidas comparamos par a par as imagens de cada um dos 10 grupos de imagens parecidas.

Na próxima seção vamos discutir e analisar os resultados de cada uma das duas abordagens utilizadas.

1: 2.2209 2: 2.8432.1039041916 3: 2.8678.2268023952 4: 2.6316.5353053892 5: 1.646.0000000000 6: 1.136.0000000000 7: 9.46.0000000000 8: 1.013.0000000000 9: 1.064.0000000000
6 1: 2.1692 2: 3.0712.2395605469 3: 3.1202.1388802083 4: 3.0148.2991829427 5: 2.38.0000000000 6: 2.406.0000000000 7: 7.073.0000000000 8: 6.660.0000000000 9: 6.726.0000000000
4 1: 1.87401 2: 3.5795.9391093333 3: 3.6668.3832480000 4: 3.2218.2303413333 5: 0.0000000000 6: 0.0000000000 7: 0.0000000000 8: 0.0000000000 9: 0.0000000000 10: 0.0000000000
3 1: 3.06070 2: 2.6691.3554524740 3: 3.2605.6380143229 4: 1.7565.7398470052 5: 0.0000000000 6: 3.0000000000 7: 4.2.0000000000 8: 1.34.0000000000 9: 3.78.0000000000 10: 6.2
8 1: 1.62120 2: 3.4177.4668708709 3: 3.4297.8577237237 4: 3.3979.2649969970 5: 0.0000000000 6: 0.0000000000 7: 0.0000000000 8: 2.0000000000 9: 6.87.0000000000 10: 4.476.0
5 1: 5.7525 2: 2.5942.5075000000 3: 3.25145.3741471354 4: 9.959.7404947917 5: 6.0000000000 6: 8.4.0000000000 7: 4.64.0000000000 8: 1.060.0000000000 9: 1.161.0000000000 10: 1
8 1: 3.1199 2: 2.7341.9225585586 3: 3.27438.2351351351 4: 2.7183.3528288288 5: 3.379.0000000000 6: 2.200.0000000000 7: 3.306.0000000000 8: 1.047.0000000000 9: 7.752.0000000000
7 1: 1.08250 2: 2.2782.2556733871 3: 3.23962.2415403226 4: 2.2055.0965766129 5: 1.179.0000000000 6: 4.08.0000000000 7: 6.49.0000000000 8: 1.430.0000000000 9: 2.868.0000000000
7 1: 1.08734 2: 2.4049.0178830645 3: 3.24907.7281532258 4: 2.3504.0402096774 5: 8.4.0000000000 6: 2.73.0000000000 7: 4.55.0000000000 8: 1.031.0000000000 9: 2.089.0000000000
10 1: 7.2440 2: 3.3013.7355293843 3: 3.3029.7485657649 4: 3.3067.1157921331 5: 1.498.0000000000 6: 2.962.0000000000 7: 4.928.0000000000 8: 4.035.0000000000 9: 4.028.0000000000
1 1: 4.7932 2: 3.2585.7570750751 3: 3.32685.7825885886 4: 3.30036.6698558559 5: 5.758.0000000000 6: 4.437.0000000000 7: 3.143.0000000000 8: 3.139.0000000000 9: 3.767.0000000000
10 1: 7.2440 2: 2.9552.7557719216 3: 3.29729.2730449316 4: 2.8599.0374455846 5: 0.0000000000 6: 8.37.0000000000 7: 5.760.0000000000 8: 8.286.0000000000 9: 5.182.0000000000
6 1: 1.4178 2: 2.5809.8005729167 3: 3.26268.2343619792 4: 2.5413.1270963542 5: 7.7.0000000000 6: 1.104.0000000000 7: 2.022.0000000000 8: 2.061.0000000000 9: 2.140.0000000000
9 1: 5.3081 2: 2.5962.0504850261 3: 3.22945.3272623698 4: 2.0766.3571126302 5: 2.53.0000000000 6: 1.722.0000000000 7: 1.044.0000000000 8: 1.319.0000000000 9: 1.658.0000000000
5 1: 5.5758 2: 3.2139.1231250000 3: 3.31043.0284049479 4: 1.0207.2458398437 5: 3.0000000000 6: 5.51.0000000000 7: 2.00.0000000000 8: 6.88.0000000000 9: 9.85.0000000000 10: 9.5
4 1: 4.3987 2: 3.5795.9249493333 3: 3.6668.4433760000 4: 3.2218.2724960000 5: 0.0000000000 6: 0.0000000000 7: 0.0000000000 8: 0.0000000000 9: 0.0000000000 10: 0.0000000000
1 1: 5.2996 2: 3.6386.5870969089 3: 3.25057.8413784461 4: 3.31320.9560693400 5: 6.531.0000000000 6: 4.606.0000000000 7: 3.868.0000000000 8: 4.160.0000000000 9: 4.404.0000000000
6 1: 2.3749 2: 2.5802.7590071614 3: 3.26244.0184700521 4: 2.5416.3136621094 5: 3.307.0000000000 6: 4.434.0000000000 7: 7.960.0000000000 8: 8.313.0000000000 9: 8.469.0000000000
4 1: 4.3987 2: 3.5630.2632853334 3: 3.6590.0056000000 4: 3.1694.8432320000 5: 0.0000000000 6: 0.0000000000 7: 0.0000000000 8: 0.0000000000 9: 0.0000000000 10: 0.0000000000
5 1: 5.7525 2: 3.3066.0054329427 3: 3.3135.0366210937 4: 3.2695.4811360677 5: 1.294.0000000000 6: 2.816.0000000000 7: 4.243.0000000000 8: 4.843.0000000000 9: 4.938.0000000000
1 1: 1.65356 2: 2.8447.8295435435 3: 3.28728.4751111111 4: 2.6483.3477657658 5: 8.25.0000000000 6: 1.958.0000000000 7: 3.058.0000000000 8: 4.001.0000000000 9: 4.805.0000000000
7 1: 1.08734 2: 2.3203.8066693549 3: 3.24277.1764959677 4: 2.2522.5815362903 5: 1.52.0000000000 6: 3.67.0000000000 7: 5.72.0000000000 8: 1.327.0000000000 9: 2.658.0000000000
3 1: 3.1705 2: 2.6168.3154720052 3: 3.25915.1823144532 4: 1.7872.0722623698 5: 1.0000000000 6: 1.1.0000000000 7: 4.5.0000000000 8: 1.134.0000000000 9: 3.329.0000000000 10: 5.4
5 1: 5.7526 2: 2.4011.5662141023 3: 3.40438.6654437030 4: 1.9875.0797322591 5: 2.0000000000 6: 4.1.0000000000 7: 1.86.0000000000 8: 6.40.0000000000 9: 9.33.0000000000 10: 9.6
10 1: 7.2440 2: 2.9760.7887943097 3: 3.29907.8775419776 4: 2.8965.9997123756 5: 0.0000000000 6: 8.35.0000000000 7: 5.755.0000000000 8: 8.095.0000000000 9: 4.968.0000000000

Figura 2. Exemplo de vetores de características das imagens

5. Resultados

Nesta seção vamos discutir e analisar criticamente os resultados obtidos em cada uma das duas abordagens utilizadas. Na seção 5.1 discutimos os resultados da classificação multi-classe e na seção 5.2 discutimos os resultados da classificação binária.

Em todos os experimentos embaralhamos a base 3 vezes e fizemos a *10-fold cross validation*, totalizando 30 execuções. Também foi calculado a média das execuções para cada métrica e com uma confiança de 95% obtivemos o intervalo de confiança.

5.1. Classificação Multi-classe

Na classificação multi-classe utilizamos a base de dados de 10 conjuntos de 44 imagens parecidas. A tabela 1 apresenta dos valores das métricas de precisão e revocação, com o intervalo de confiança entre parênteses.

Métrica	Valor
Precisão	88,676 (86,609 - 90,743)
Revocação	87,906 (85,502 - 90,310)

Tabela 1. Resultados da Classificação Multi-classe

Podemos observar que os resultados obtidos foram muito bons, com altas taxas de precisão (88,6%) e revocação (87,9%). No entanto devemos ter cuidado ao analisar esses resultados pois fizemos testes em uma base muito pequena. Nosso mecanismo de detecção ainda não está escalando,

então não foi possível realizar testes com uma base maior. No futuro pretendemos resolver esse problema do desempenho para realizar testes com bases maiores e comprovar a eficácia de nosso mecanismo de detecção de imagens parecidas.

	Previsão									
	1	2	3	4	5	6	7	8	9	10
1	112	0	3	2	2	4	3	5	0	1
2	0	121	0	8	0	0	0	2	0	1
3	0	0	119	0	1	0	0	2	10	0
4	5	3	0	115	0	0	1	3	4	1
5	0	0	7	0	114	0	3	5	3	0
6	0	0	1	0	0	120	0	2	3	6
7	3	0	3	0	3	0	112	7	4	0
8	1	0	2	3	0	2	2	116	1	5
9	0	0	11	0	0	3	3	3	112	0
10	0	0	2	0	0	6	0	7	4	113

Tabela 2. Matriz de Confusão da Classificação Multi-classe

A tabela 2 apresenta a matriz de confusão da classificação. Nessa matriz cada linha mostra a classe real de um objeto e cada coluna mostra a classe em que os objetos foram previstos pelo classificador. Quanto mais forte for a diagonal, melhor foi o desempenho do classificador. Como podemos observar, a diagonal ficou bem forte, onde o pior desempenho foi nas classes 1, 7 e 9.

Um fato que chamou a atenção foi um alto número de imagens da classe 3 confundidas com imagens da classe 9 e vice-versa. Exemplos dessas imagens são mostradas na figura 3. Apesar de serem imagens bem diferentes, observamos que elas obtiveram vetores de características bem próximos, principalmente o histograma de cores, e como muitas das nossas características são globais e baseadas em



Figura 3. Exemplo de imagens que foram muito confundidas pelo classificador multi-classe

cores, isso fez com que o classificador se confundisse. Apesar de termos utilizados métricas de textura, que poderiam ser utilizadas para diferenciar as duas classes de imagens, o classificador SVM opta por algumas características que julga mais importantes, e no caso as características de textura devem ter sido descartadas na classificação.

Também verificamos os tipos de transformações que foram mais erradas pelos classificadores. Na ordem, borda, original, embassar, equalizar, intensidade, normalizar e colorizar foram as transformações mais confundidas pelo SVM. Como muito das nossas características extraídas são baseadas em cores, as transformações que modificam as cores foram as mais confundidas. No futuro pretendemos extrair mais características diferentes para tentar melhorar a eficácia do classificador. O uso de características locais também podem ajudar.

5.2. Classificação Binária

Na classificação binária utilizamos a base de dados de 10 conjuntos de 5 imagens parecidas, mais 4 imagens diferentes selecionadas aleatoriamente dentre as 5,304 da base do VOC 2006. O motivo da redução da base é que nosso mecanismo não está escalando, em parte devido à demora do LibSVM para encontrar os melhores parâmetros para o SVM. A tabela 3 apresenta dos valores das métricas de precisão e revocação, com o intervalo de confiança entre parênteses.

Observamos que os resultados foram excelentes, com valores de precisão e revocação altíssimos. No entanto, devemos avaliar esses resultados com certo cuidado pois a base testada foi muito pequena. Devemos realizar mais testes no futuro para ter certeza da eficácia de nossa proposta na detecção de imagens parecidas.

A tabela 4 mostra a matriz de confusão da classificação

Métrica	Binária	Multi-classe Reduzida
Precisão	99,234 (98,427 - 100,000)	75,571 (69,045 - 82,097)
Revocação	99,523 (99,032 - 100,000)	77,627 (70,914 - 84,339)

Tabela 3. Resultados da Classificação Binária

	Previsão	
	Parecidas	Diferentes
Parecidas	299	1
Diferentes	4	596

Tabela 4. Matriz de Confusão da Classificação Binária

binária. Observamos que a diagonal é bastante forte, como esperado devido aos altos valores de precisão e revocação obtidos.

6. Conclusão e Trabalhos Futuros

O problema de detecção de imagens parecidas é amplamente estudado, pois é difícil de se resolver e é essencial para diversas áreas. Por ser um problema muito grande podemos tentar resolvê-lo de diversas formas ou tentar resolver somente algumas de suas partes. A tentativa da resolução de uma parte do problema foi o enfoque do trabalho, somente um tipo de modificação sobre as imagens, as modificações de cena, classificação de acordo com a literatura.

Devido a tentativa de resolução de uma parte do problema verificamos que o mecanismo proposto é viável, devido aos resultados obtidos utilizando as duas abordagens de classificação. Entretanto o mecanismo utilizado peca na

eficiência, pois não escala. O que pode dificultar a utilização do mecanismo em sistemas de detecção de imagens parecidas, no qual a eficiência é essencial devido ao grande volume de dados a ser verificado.

Como proposta para trabalhos futuros podemos avaliar o mecanismo proposto segundo seu desempenho e memória utilizados. Utilizar outra base de dados, explorando outras características de imagens parecidas, assim como utilizar mais características para representar uma imagem e consequentemente avaliar quais características são mais relevantes para o mecanismo de detecção. Propomos também para trabalhos futuros desenvolver uma abordagem mais escalável para o mecanismo, tornando o aplicável para o sistemas de detecção de imagens parecidas que serviram de motivação para o desenvolvimento do presente trabalho.

Referências

- [1] The 2006 pascal visual objects classes challenge (voc 2006). Disponível em: <http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2006/index.html>.
- [2] Base de dados do voc 2006. Disponível em: <http://pascallin.ecs.soton.ac.uk/challenges/VOC/databases.html#VOC2006>.
- [3] Flickr. Disponível em: <http://www.flickr.com>.
- [4] Image magick. Disponível em: <http://www.imagemagick.org>.
- [5] Libsvm. Disponível em: <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.
- [6] Radial basis function. Disponível em: http://en.wikipedia.org/wiki/Radial_basis_function.
- [7] Support vector machine. Disponível em: <http://www.support-vector-machines.org/>.
- [8] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting.
- [9] J. Foo, R. Sinha, and J. Zobel. Discovery of image versions in large collections. pages 433–442, 2006.
- [10] J. J. Foo and R. Sinha. Pruning sift for scalable near-duplicate image matching. In *Proceedings of the Eighteenth Australasian Database Conference (ADC 2007)*, Ballarat, Australia, Jan. 2007.
- [11] J. J. Foo, R. Sinha, and J. Zobel. Clustering near-duplicate images in large collections. In J. Z. Wang, N. Boujemaa, A. D. Bimbo, and J. L. (eds), editors, *Proceedings of the MIR International Workshop on Multimedia Information Retrieval*, pages 21–30, Augsburg, Germany, Sept. 2007.
- [12] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
- [13] B. Herbert, T. Tinne, V. Gool, and L. Surf: Speeded up robust features. In *9th European Conference on Computer Vision*, Graz Austria, May 2006.
- [14] A. Jaimes, S.-F. Chang, and A. C. Loui. Detection of non-identical duplicate consumer photographs. volume 1, pages 16–20 Vol.1, 2003.
- [15] Y. Ke and R. Sukthankar. Pca-sift: a more distinctive representation for local image descriptors. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 506–513, 2004.
- [16] Y. Ke, R. Sukthankar, and L. Huston. An efficient parts-based near-duplicate and sub-image retrieval system. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 869–876, New York, NY, USA, 2004. ACM.
- [17] B. Mehta, S. Nangia, M. Gupta, and W. Nejdl. Detecting image spam using visual features and near duplicate detection. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 497–506, New York, NY, USA, 2008. ACM.
- [18] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, October 2005.
- [19] D.-Q. Zhang and S.-F. Chang. Detecting image near-duplicate by stochastic attributed relational graph matching with learning. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 877–884, New York, NY, USA, 2004. ACM.