Uso da Abordagem Hash como Ferramenta de Análise de Similaridade entre Imagens

Ramon Darwich de Menezes[1], Vânia Cordeiro da Silva[1]

[1]Universidade Estadual de Santa Cruz (UESC)
Rodovia Jorge Amado, Km 16, Bairro Salobrinho – 45662-900 – Ilhéus – BA – Brazil

ramondarwich@gmail.com, vania@uesc.br

Abstract. *Programs capable of comparing similarities between images can be easily found available for download, free or not but with slow comparison algorithms. Using the Hash approach, it was possible to achieve similar results as conventional algorithms, with a visible improvement in performance and execution speed.*

Resumo. *Programas capazes de comparar as semelhanças entre imagens podem ser encontrados rapidamente à disposição para download, gratuitos ou não, mas com algoritmos comparativos lentos. Pelo método Hash, foi possível obter resultados similares aos algoritmos convencionais com uma melhora visível em performance e velocidade de execução.*

1. Introdução

Com a crescente facilidade de se produzir e compartilhar mídias, principalmente fotografias, há uma maior possibilidade de se ter cópias acidentais da mesma imagem ou muito similares, ocupando um espaço na memória do dispositivo que poderia ser utilizado por outros arquivos ou aplicativos. Também é comum termos cópias em baixa resolução de fotos enviadas do Whatsapp[1], ou prévias, miniaturas das fotos geradas.

Dentro da área de Processamento Digital de Imagens, ou PDI[2], para fins de abreviação, algoritmos e programas que são capazes de discernir se uma imagem é similar o suficiente para dizer-se uma cópia de outra, não são raros. Porém, com a mesma facilidade que se encontra tais programas, é igualmente fácil perceber que o processo de se distinguir imagens é um trabalho muitas vezes demorado e computacionalmente intensivo, nos melhores casos.

Nesse artigo, apresentamos um método de análise de imagens que utiliza uma abordagem Hash, método que tem se mostrado muito eficiente para o teste de um grande ou pequeno volume de imagens casuais sobre as soluções popularmente encontradas (Tabela 1).

O texto está organizado da seguinte forma: a sessão 2 apresenta o processo de comparação. Na sessão 3 a abordagem Hash empregada para esse programa e suas otimizações são detalhadas e na sessão 4 os resultados obtidos testando a ferramenta contra outra opção do mercado. O texto encerra-se com as referências bibliográficas.

1 Aplicativo de comunicação instantânea individual ou para grupos. Marca registrada da Facebook Inc.

2 Área da ciência da computação responsável pela manipulação de uma imagem por computador de modo que a entrada e a saída do processo sejam imagens

2. Método

Após todos os campos do programa terem sidos preenchidos e validados, o programa contará quantos arquivos da pasta escolhida são válidos. Esse valor é crucial tanto para definir o fim do processo, quanto para o previsor de consumo de tempo para a referida pasta.

Em seguida, é calculado quantos ciclos serão necessários para o término do processo, onde um ciclo é o trabalho de se converter em Hash a imagem inicial *n*, a próxima imagem *n+1*[3] e a comparação de ambas. No modo *um-contra-todos* o tipo de comparação é simples: o Hash da imagem referencia é salvo, e o Hash das imagens subsequentes são produzidos e sobrescritos na mesma Hash, uma vez que todas imagens só serão comparadas uma única vez em uma única iteração, logo, para esse método, uma pasta levará *n* ciclos para ser concluído o processo de comparação.

Já no modo *todos-contra-todos*, cada imagem *n* será testada contra todas as imagens a frente dela, cada iteração custando 1 ciclo a menos que a anterior, sendo gastos *n-i*[4] ciclos por iteração.

3. Proposta de Ferramenta

A análise de uma imagem é conceitualmente simples, checa-se cada pixel por seus componentes de cor RGB ou RGBA[5] e depois de analisado todos os pixels, o mesmo processo é feito para as próximas *n* imagens restantes até que todas tenham sido testadas. Essa abordagem, apesar de ser a mais precisa, é também uma das mais ineficientes, pois gasta-se muito tempo somente lendo e processando todos os pixels de uma singular imagem, além de que o algoritmo pixel a pixel não é capaz de detectar diferenças sutis nas imagens, como um simples melhoramento de contraste, clareamento ou subdimensionamento.

Pensando nisso, propomos uma solução que manipula a imagem e produz um Hash dela, para depois fazer as análises com outras fotografias, o que reduz drasticamente o tempo necessário.

3 Definimos *n* como a quantidade de imagens restantes de cada iteração *i*
4 Definimos *i* como a quantidade de iterações que faltam para avançar para a imagem *n+1*
5 Padrão de cor geralmente utilizado para fotografias e imagens digitais. Refere-se aos componentes Red (vermelho), Green (verde), Blue (azul) e a depender da imagem, Alpha (transparência)

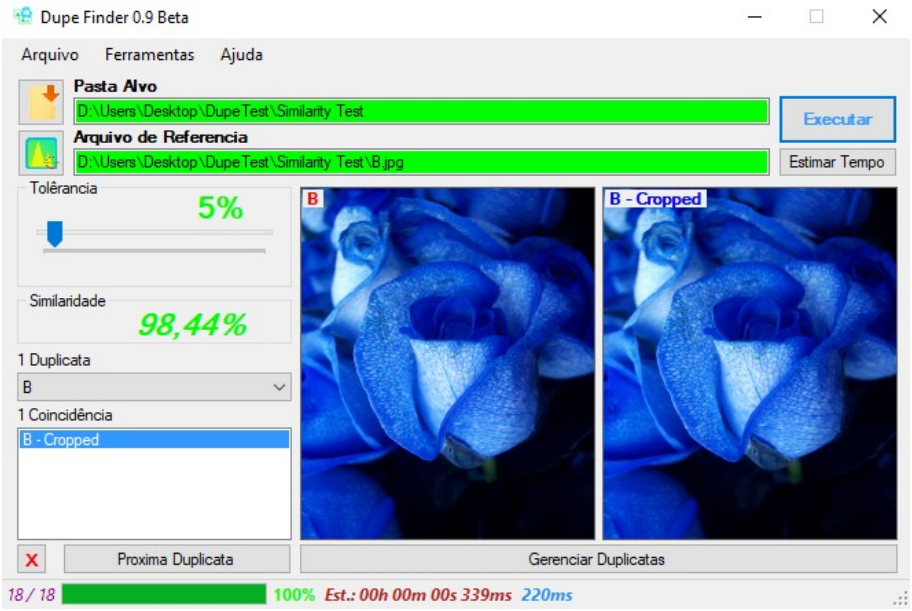

Figura 1. Interface do programa depois de executado no modo *um-contra-todos*

Antes de partir diretamente para o teste, o programa produz um Hash único de cada imagem, que é posteriormente salvo e reutilizado para acelerar testes futuros. O usuário precisa selecionar um dos dois métodos de análise que são suportados pelo programa: o teste *todos-contra-todos*, já selecionado por padrão, onde todas as imagens da pasta alvo que o usuário selecionar são testadas contra si mesmas; ou *um-contra-todos*, onde além da pasta alvo, o usuário selecionará uma imagem de referência, para qual o programa testará ela contra todas as outras da pasta alvo (Figura 1). O método de comparação do programa não influencia o Hash que será gerado para cada imagem, ele apenas define quantas comparações terão de ser feitas.

Figura 2. Imagem de entrada formato PNG, RGBA 1200x1600

Depois de selecionada uma imagem (Figura 2), ela é redimensionada para 16x16 pixels (Figura 3), e uma matriz de cor é aplicada sobre ela, de forma que fique em tons de cinza, onde cada pixel terá apenas brilho, um valor simétrico entre as cores RGB que varia entre 0, preto, e 255, branco. O método de se aplicar uma matriz de cor sobre a fotografia se mostrou muito mais eficiente que o processo convencional de calcular a média das cores de cada pixel.

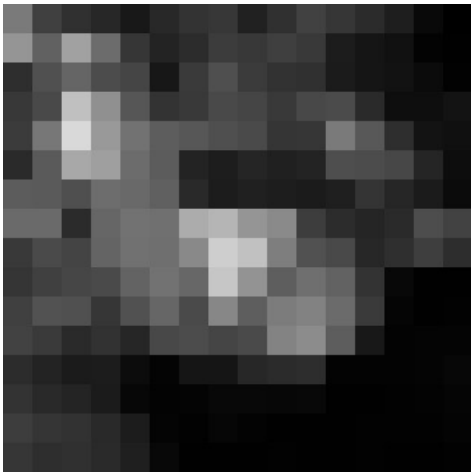

Figura 3. Imagem redimensionada e em escala de cinza

A partir dessa nova imagem, é gerado um vetor de 256 bytes, um byte por pixel, que é ordenado e calculado a sua mediana, valor que será utilizado para aumentar a fidelidade do detector para o Hash. A imagem redimensionada é então novamente analisada e todos os pixels que tiverem um valor menor que a mediana serão categorizados como 1 e todos os outros como 0 numa lista de booleanos, que é o Hash único dessa imagem (Figura 4).

Figura 4. Representação colorida do Hash gerado

Gerado o Hash da imagem *n*, o programa salva esse Hash para utilização nas iterações futuras e produz o Hash da imagem *n+1* e subsequentes fotografias, até que todas as fotos da pasta alvo tenham sido mapeadas.

Depois de terminado todo o processo, o usuário terá a chance de analisar cada coincidência e duplicatas manualmente, podendo conferir o formato, nome, tamanho e dimensões de cada imagem identificada, pelo *gerenciador de duplicatas* (Figura 5).

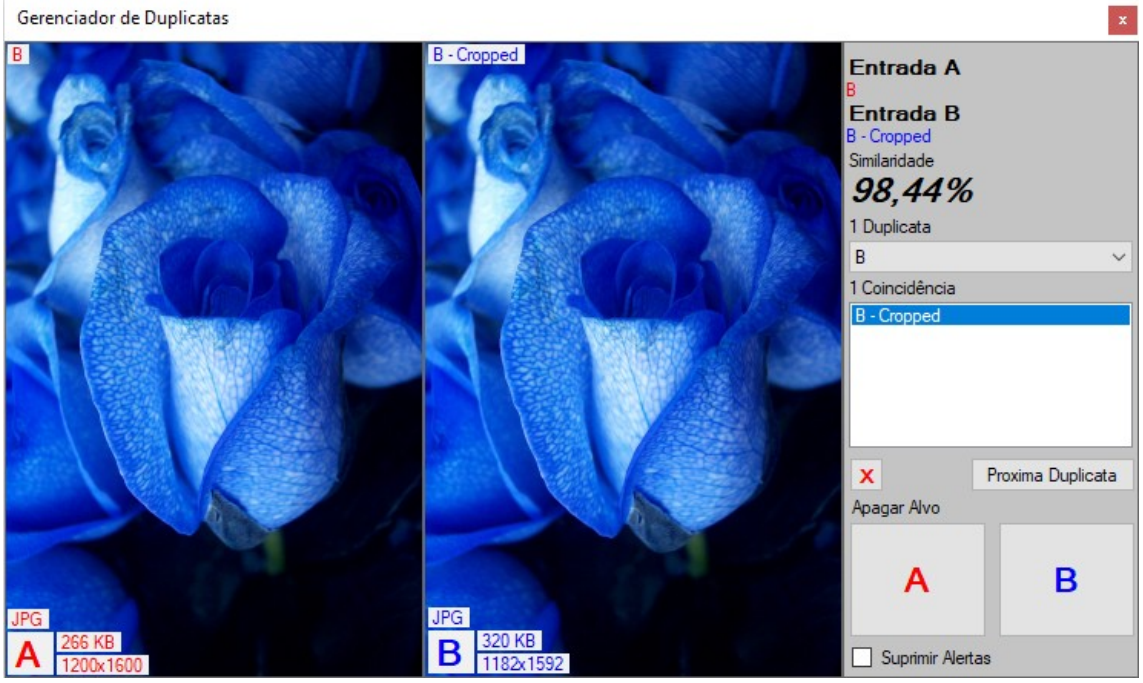

Figura 5. Gerenciador de duplicatas

Essa ferramenta foi programada em linguagem C#, utilizando tão somente bibliotecas e framework padrão da própria linguagem. Por conta disso, todos os códigos para comparação, criação, manutenção e filtragem de Hashes tiveram de ser totalmente implementados do zero, o que implicou em maior tempo de desenvolvimento, mas com a vantagem que o programa não possui dependências além da básica da própria C# e pode ser facilmente ajustado, aperfeiçoado e /ou corrigido futuramente.

4. Resultados

Durante os testes para aferição da performance da ferramenta, foram feitas duas baterias de testes, cada uma delas executada 5 vezes e tirada a média do tempo consumido para a comparação *todos-contra-todos* utilizando tolerância de 5% de diferença entre Hashes.

A primeira bateria foi feita de uma quantidade aleatória de cópias de 6 imagens, totalizando 3.904 arquivos validos para comparação e a segunda, de variações pequenas das mesmas imagens, modificando-se tamanho, dimensões e até mesmo o formato de arquivo, totalizando 18 fotografias.

Para a comparação, foi utilizada a ferramenta Duplicate Photo Finder[6] a qual foi submetida a mesma bateria de testes e com limite de similaridade em 95%, equivalente ao do programa desse artigo. O tempo de ambas foi calculado, mas por limitações no contador de tempo do Duplicate Photo Finder, a margem de erro da sua medição é de $1000 \pm ms$.

6 Ferramenta de comparação de imagens. Marca registrada da WebMinds, Inc.

Tabela 1. Comparação contra ferramenta, Duplicate Photo Finder. Valores aprox.

	Tempo Necessário Ferramenta Proposta	Tempo Necessário Duplicate Photo Finder
Bateria 1 3904 Arquivos	33.186,6 ms (±0.1ms)	62.000 ms (±1s)
Bateria 2 18 Arquivos	242,8 ms (±0.1ms)	1.000 ms (±1s)

5. Conclusão

Apresentamos uma ferramenta intuitiva e com baixo custo computacional, para encontrar imagens consideradas semelhantes, dentro de um percentual escolhido pelo usuário. Objetivamos resolver um problema do mundo moderno, sustentado nas multimídias, de acúmulo de imagens iguais ou consideradas semelhantes, como opção para otimizar espaço de armazenamento requerido por imagens casuais.

Referências

https://docs.microsoft.com/en-us/dotnet/framework/winforms/advanced/how-to-use-a-color-matrix-to-transform-a-single-color

https://www.duplicatephotocleaner.com/