Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - USP

Trabalho Final:
CBIR

Profª: Maria Cristina Ferreira de Oliveira

Giovani Ortolani Barbosa - 8936648
Henrique Anacretto Pereira - 8485799

São Carlos
2016

1. Introdução

O projeto desenvolvido é referente à atividade final da disciplina de Processamento de Imagens e possui como objetivo o desenvolvimento de um pequeno sistema de recuperação de imagens por conteúdo (*CBIR - Content Based Image Retrieval*) que deverá retornar as imagens mais similares à imagem de entrada. Para tal, devemos implementar descritores de imagem, os quais extraem características das imagens de acordo com sua cor, formas etc. O banco de imagens utilizado é o Corel 10K, o qual possui 10.000 imagens em diferentes categorias (fogos de artifício, natureza, animais, carros, paisagens etc).

Instruções de execução:

1) Geração do índice (arquivo principal *indexGenerator.py*)
    a) Criar um diretório para conter os códigos-fonte do projeto, esse diretório deve conter os arquivos "*corel10k.py*", "*indexGenerator.py*" e "*searcher.py*".
    b) Colocar o diretório "descriptors" dentro do novo diretório.
    c) Criar um diretório chamado "*indexes*" dentro do novo diretório.
    d) Utilizar o seguinte comando para gerar o índice:

"python indexGenerator.py --images <diretório com as imagens> --method <fourier, gch ou lch> [--mask <3, 5 ou 7>]"

Obs.: o parâmetro *--mask* é opcional e utilizado apenas para o método *fourier*, seu valor *default* é 7.

e) O índice para o método escolhido será gerado dentro do diretório "*indexes*" e ele terá extensão "*.csv*".

2) Execução da busca (arquivo principal *corel10k.py*)
    a) Utilizar o seguinte comando para realizar a busca:

"python corel10k.py --images <diretório com as imagens> -q <caminho para a query> -n <numero de imagens similares> -m <fourier, gch ou lch> [--mask <3, 5 ou 7>] -d <chi2, manhattan ou euclidean>"

Obs.: o parâmetro *--mask* é opcional e utilizado apenas para o método *fourier*, seu valor *default* é 7.

b) O número das imagens mais similares, em ordem descrescente de similaridade, à imagem de entrada (*query*) de acordo com o limite passado pelo usuário serão exibidas na saída-padrão em modo de texto.

Em caso de dúvida, utilizar o comando:
"python <corel10k.py ou indexGenerator.py> -h"

2. Fundamentação

Durante o desenvolvimento do projeto utilizamos diversos conceitos e técnicas aprendidos em sala de aula, também foi-se necessário buscar informação adicional bem como solução de dúvidas em fóruns e tutoriais na Internet.

Para a extração do vetor de características das imagens, implementamos:

- *Descritor de Fourier - Transformada Discreta de Fourier (DFT)*.
- *Descritor de Cor - Global Color Histogram (GCH).*
- *Descritor de Cor - Local Color Histogram (LCH)*.

As imagens do banco foram convertidas para *HSV* nos *Descritores de Cor* e foram usadas em *tons de cinza* no *Descritor de Fourier*.

O índice para todos os métodos é gerado pela extração de características do banco de imagens e é armazenado em um arquivo *.csv*, cada linha representa um vetor de características de uma imagem, a linha é composta por "<número da imagem>, caract1,caract2,caract3,...,caractN"

A busca das imagens no banco de imagens é feita através da extração de características da imagem de consulta e a respectiva comparação de seu vetor de características com os vetores de características do índice. Essa comparação é feita por uma métrica de distância que pode ser Chi-Quadrado, Euclidiana ou Manhattan. Tal métrica compara a similaridade de dois vetores de características de modo que quanto menor é a distância entre eles, mais similares as imagens são.

3. Descrição do desenvolvimento

3.1. Pré-processamento do banco de dados

Em relação à geração do *índice*, transformamos as imagens para o sistema de cores *HSV* nos *Descritores de Cor* e usamos a imagem em *tons de cinza* para o *Descritor de Fourier*. Como as imagens são todas no mesmo tamanho (400p x 300p ou 300p x 400p), os vetores gerados têm o mesmo tamanho e não nos preocupamos em redimensioná-las.

Do mesmo modo como no *índice*, a realização da *busca* transforma a imagem de consulta para o sistema de cores *HSV* ou *RBG* dependendo do método a ser utilizado para a extração de seu vetor de características.

3.2. Descritores utilizados/implementados

Foram implementados 3 descritores de imagem, 2 Descritores de Cor e 1 Descritor de Fourier.

- Descritor de Fourier - Transformada Discreta de Fourier (*DFT*) (possui invariância à rotação e à escala). Extrai-se características referentes à formas presentes na imagem, representa a imagem de entrada (*domínio espacial*) no *domínio da frequência*.
    - Utilizamos máscaras de tamanho 3, 5 e 7 para o cálculo da DFT, pois o método possui alto custo computacional, porém com máscaras gera resultados similares com custo computacional menor.
- *Descritor de Cor - Global Color Histogram (GCH).* Extrai-se o histograma da imagem toda para cada canal do formato *HSV*. Não é afetado por rotação e escala da imagem.
    - Quantizamos os canais do sistema *HSV (Hue, Saturation, Value)* em 9, 12 e 4 intervalos respectivamente (tais valores foram obtidos através do tentativa e erro até acharmos os mais adequados).
        - Canal *Hue* dividido em 9 intervalos, cada intervalo representa 20º na matiz.
        - Canal *Saturation* dividido em 12 intervalos, cada intervalo representa 21,3 possíveis tons de saturação.
        - Canal *Value* dividido em 4 intervalos, cada intervalo representa 24 possíveis tons de brilho.
        - O vetor de características para cada imagem possui, portanto:
            9 x 12 x 4 = 432 números que representam suas características.

Obs.: A biblioteca OpenCV representa o canal *Hue*, *Saturation* e *Value*, respectivamente, com valores entre [0, 180], [0, 256] e [0, 256].

- *Descritor de Cor - Local Color Histogram (LCH)*. Dividimos a imagem em 5 regiões e extraímos o histograma de cada canal do formato *HSV* para cada região. Pode ser afetado pela rotação da imagem, porém não é afetado pela escala. É um bom descritor para imagens que possuam características específicas que recaiam em uma das 5 regiões definidas.

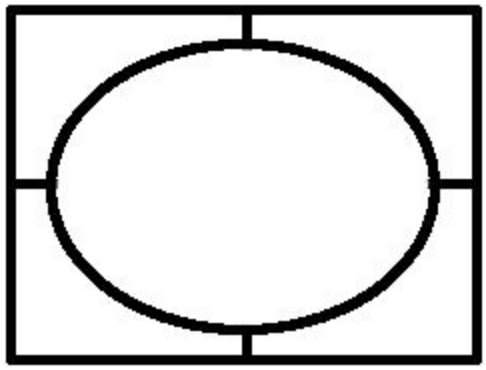


Figura 1 - Divisão da imagem em 5 regiões

  - Quantizamos os canais do sistema *HSV (Hue, Saturation, Value)* em 9, 12 e 3 intervalos respectivamente (tais valores foram obtidos através do tentativa e erro até acharmos os mais adequados)
  - Vide acima o descritor *GCH* para valores de quantização.
    - O vetor de características para cada imagem possui, portanto: 5 (regiões) x 9 x 12 x 3 = 1620 números que representam suas características.

Em complemento aos descritores, implementamos 3 métricas de distância a saber:

Sejam *q* e *d* dois histogramas com *M* cores cada.

- Chi-Quadrado

$$dChi2(q,\ d)\ = \frac{1}{2}\ *\ \sum_{i=0}^{M}((q[i]\ -\ d[i])^2\ /\ (q[i]\ +\ d[i]))$$

- Euclidiana

$$dEuclidean(q,\ d)\ = \sqrt{\sum_{i=0}^{M}(q[i]\ -\ d[i])^2}$$

- Manhattan

$$dManhattan(q, d) = \sum_{i=0}^{M} |q[i] - d[i]|$$

Em todas as métricas, quanto menor é o valor do somátorio, mais similares os vetores de características das imagens são.

3.3. Esquema de normalização

Todos os vetores de características foram normalizados no intervalo 0.0 a 1.0 para representar a porcentagem de uma determinada cor (Descritor de Cor) ou frequência (Descritor de Fourier) nos intervalos quantizados previamente. Desse modo, podemos utilizar uma métrica de distância para avaliar a similaridade de dois vetores de características.

Utilizamos a função "*normalize()*" encontrada na biblioteca OpenCV para a normalização dos vetores.

3.4. Estrutura de dados

As estruturas de dados utilizadas foram todas nativas da linguagem Python como:

- Vetores - utilizados no armazenamento das características extraídas das imagens.
- Dicionários - utilizados para armazenar o nome de uma imagem (chave, pois o nome da imagem é único) e o seu valor de semelhança - calculado pela métrica de distância - em relação à imagem de consulta.

Também foram aproveitadas estruturas do módulo *numpy* como inicialização de matrizes com 0 (*numpy.zeros()*) para o cálculo de máscaras do método *LCH*.

4. Resultados e discussão

4.1. Exemplos ilustrativos

Resultados obtidos com todas as configurações de método de busca para as 5 imagens mais similares:

Obs.: Para simplificar, usaremos apenas a *Chi-Quadrado* como métrica de distância, apesar de termos implementado a *Euclidiana* e a *Manhattan*.

- **Método GCH (Global Color Histogram)**

Com a imagem 5100.jpg como entrada obtivemos o seguinte resultado:

Consulta *1* *2*

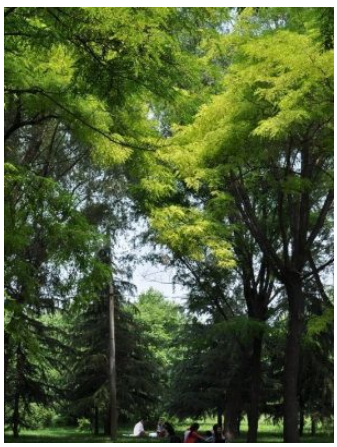
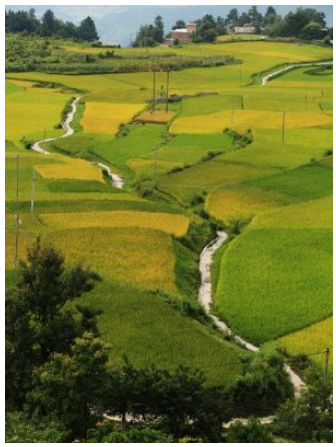
3 *4* *5*

Com a imagem 3021.jpg como entrada obtivemos o seguinte resultado:

Consulta *1* *2*



3 *4* *5*


- **Método LCH (Local Color Histogram)**

Com a imagem 5100.jpg como entrada obtivemos o seguinte resultado:

Consulta *1* *2*



3 *4* *5*



Com a imagem 3021.jpg como entrada obtivemos o seguinte resultado:

Consulta *1* *2*

3 *4* *5*

- **Método Fourier (Transformada Discreta de Fourier)**

Obs.: Foi usada apenas a máscara de 7x7, porém implementamos a de 3x3 e 5x5.

Com a imagem 5100.jpg como entrada obtivemos o seguinte resultado:

Consulta *1* *2*

3 *4* *5*

Com a imagem 3021.jpg como entrada obtivemos o seguinte resultado:

Consulta *1* *2*

3 *4* *5*

4.2. Comparação dos resultados

Tempo (média de 3 execuções) de geração do *índice* para as 10.000 imagens com os diferentes métodos:

Método	Fourier (3x3)	Fourier (5x5)	Fourier (7x7)	GCH	LCH
Tempo (em s)	1252.40	1086.53	863.77	40.87	65.16

Tempo (média de 3 execuções) de *busca* das 5 imagens mais similares à imagem de consulta no banco com as 10.000 imagens:

	Fourier (3x3)	**Fourier (5x5)**	**Fourier (7x7)**	**GCH**	**LCH**
Chi-Quadrado	409.32	151.64	68.32	39.11	150.68
Euclidiana	278.48	94.23	41.60	21.34	85.32
Manhattan	176.49	53.01	23.85	17.90	62.71

Podemos perceber que a utilização de máscaras no *Descritor de Fourier* tanto para a geração do *índice* quanto para a *busca* acelera consideravelmente o

processo. Através de testes que não foram relatados com imagens ilustrativas, percebemos que o uso da máscara 7x7 gerou diferenças insiginificantes em relação ao uso da máscara 3x3, bem como o uso das 3 técnicas de distância também gerou diferenças ínfimas entre elas. Notamos que o uso do *Descritor de Fourier* para imagens que possuem áreas de cor uniformes (céu, fundo preto etc.) não apresenta resultados satisfatórios (vide o exemplo com a imagem 5100.jpg), porém em imagens com intensa variação de frequência o descritor se sai melhor do que todos os outros métodos.

Em contrapartida, o *GCH* apresenta bons resultados para imagens com pouca variação de cor e presença de largas áreas uniformes. Seu resultado pode ser otimizado através da escolha da métrica de distância adequada.

Finalmente, o *LCH* consegue captar detalhes em razão da extração do histograma de regiões especificas da imagem, assim pode ser usado para comparar imagens com maior grau de detalhes na mesma região porém ainda com uma certa uniformidade. Assim como o *GCH* pode ser afetado caso o grau de detalhamento e variação aumente. Seu uso é recomendado em imagens que possuam uma cor concentrada em alguma das regiões da divisão, nesse caso mesmo tempo um tempo de execução maior do que o *GHC* seu uso é justificado.

5. Conclusão

Ao final do desenvolvimento do projeto percebemos que realizar a recuperação de imagem pelo seu conteúdo não é uma tarefa trivial, existem muitas variavéis a se levar em consideração, a primeira dela é qual descritor irá ser utilizado para gerar o vetor de características da imagem, dependendo do descritor escolhido ele saíra melhor com imagens que possuam áreas de cor uniforme (céu, fundo preto etc.), como é o caso do descritor de Fourier.

Outro ponto importante é o desempenho ao realizar uma busca, se haver muitas imagens no banco de imagens ou o vetor de características da imagem for muito grande afetará o desempenho, para contornarmos esse problema utilizamos uma máscara, nela calculamos a média do pixel e seus vizinhos, com isso conseguimos gerar os índices e realizar a busca com mais eficiência e sem perda de qualidade em seu resultado.

6. Referências

- Adrian Rosebrock. The complete guide to building an image search engine with Python and OpenCV: http://www.pyimagesearch.com/2014/12/01/complete-guide-building-image-search-engine-python-opencv/ (último acesso em 27/06/2016)
- Fourier Transform: http://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm (último acesso em 27/06/2016)
- Gonzales and Woods. Processamento Digital de Imagens. 3.ed. Capítulo 11. 2010.