Projeto 3-1: Buscador de imagens
Parte 1

Bruna Mayumi Kimura
Visão computacional

1. Introdução

Este projeto consiste em fazer um programa capaz de, dada uma imagem inicial, buscar as imagens mais semelhantes a ela. Para tanto, o programa utiliza um vocabulário, ou seja, um arquivo com os treinos feitos previamente a partir de um banco de dados de imagem. O programa tem duas opções de método para buscar a imagem que o próprio usuário pode escolher.

2. Modo de uso

Para testar o programa é necessário clonar este repositório. Entre na pasta Projeto3-1.

1. Geração do vocabulário

O primeiro passo é gerar os arquivos de treinamento. Para isso, basta rodar o arquivo 'vocab.py'. Ao executar este programa, ele gerará três arquivos pickle, eles são: vocab.p (arquivo com os vocabulários), descritores.p (arquivo com cada um dos descritores de cada imagem do banco de dados) e hist_banco.p (arquivo que possui os valores do histograma de cada uma das imagens do banco de dados utilizando o vocabulario feito anteriormente).

Este processo é um pouco lento uma vez que ele itera sobre cada uma das imagens do banco de dados (11 pastas com 10 imagens cada).

Todos esses três arquivos já estão contidos na pasta, pois foi anteriormente gerado e estão pronto para uso, sem a necessidade de gerar novamente. Caso queira gerar outros treinos, o arquivo será reescrito.

2. Busca das imagens semelhantes

Para buscar as imagens semelhantes a uma determinada foto basta executar o arquivo banco.py. Ao executar o programa ele irá perguntar a imagem mais o caminho, é necessário inserir o caminho da imagem e seu nome como mostra a **figura 1**. Em seguida ele perguntará qual o método que você gostaria de

utilizar para fazer o cálculo das imagens. São apenas duas opções: qui-quadrado ou uma função de comparação de histograma da biblioteca do OpenCV (cv2.HISTCMP_BHATTACHARYYA). Para selecionar a primeira opção basta digitar ‘1’, para a segunda opção ‘2’.

Após selecionar a método de cálculo, irá aparecer o resultado em uma janela contendo na parte superior a imagem escolhida e logo na parte de baixo a ordem das imagens mais semelhantes (da esquerda para direita, de cima para baixo).

Para finalizar o programa, basta fechar a janela com as imagens.

```
[bruna:~/Documentos … -1/Projeto3-1] master ± python3 banco.py
Nome da imagem + caminho: 101_ObjectCategories/panda/image_0003.jpg
```

figura 1 - como escrever o caminho e o nome da imagem

O terminal irá printar os nomes e a ordem das cinco imagens mais semelhantes como mostra a **figura 2**.

```
1z [bruna:~/Documentos … -1/Projeto3-1] master ± python3 banco.py
Nome da imagem + caminho: 101_ObjectCategories/panda/image_0005.jpg
Qual modo deseja fazer o cálculo? (1:Chi-square, 2:OpenCV): 1
[ INFO:0] Initialize OpenCL runtime...
Imagem buscada
1° imagem semelhante:  101_ObjectCategories/platypus/image_0006.jpg
2° imagem semelhante:  101_ObjectCategories/elephant/image_0009.jpg
3° imagem semelhante:  101_ObjectCategories/platypus/image_0001.jpg
4° imagem semelhante:  101_ObjectCategories/elephant/image_0004.jpg
5° imagem semelhante:  101_ObjectCategories/platypus/image_0004.jpg
```

figura 2 - print do log do terminal

3. Desempenho do Bag of Visual Words (chi-square)

Foram feitos vários testes para analisar o desempenho do programa quando executado com o método 1, chi-square.

A primeira análise foi feita buscando uma imagem que foi utilizada na geração do banco de dados. Dessa forma, espera-se que a imagem mais semelhante seja ela mesma, uma vez que é a que possui menor diferença entre os histogramas. O resultado pode ser observado na **figura 3**.

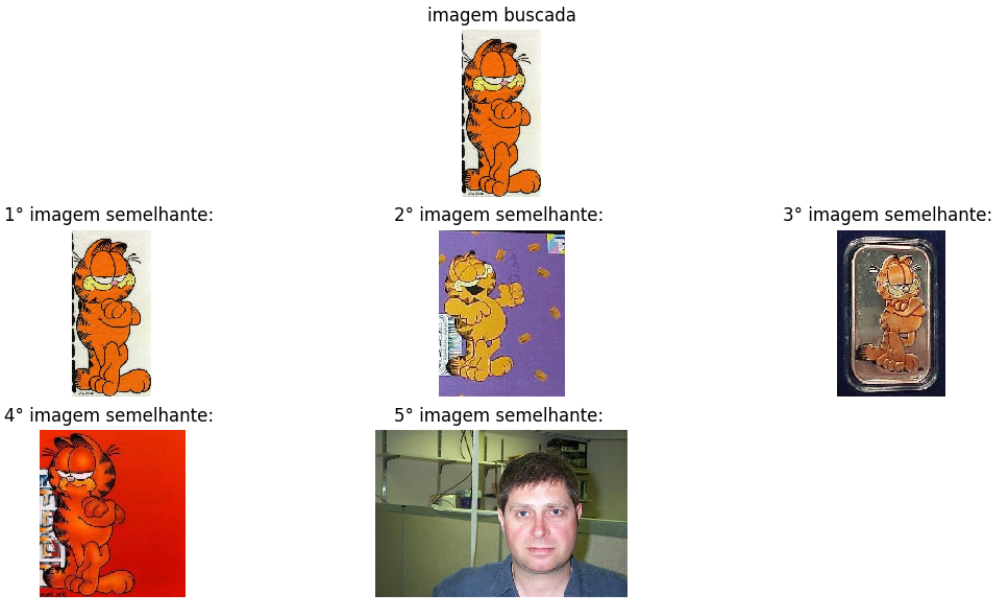

figura 3 - método 1: imagem usada no banco de dados

Como é possível notar, o resultado foi ótimo, já que a imagem mais semelhante é ela mesma, comprovando a veracidade do programa. E a maioria das outras imagens pertencem ao mesmo grupo da imagem buscada.

O segundo teste foi utilizando uma imagem da mesma categoria das usadas para gerar o banco de dados. Como é possível ver na **figura 4**.

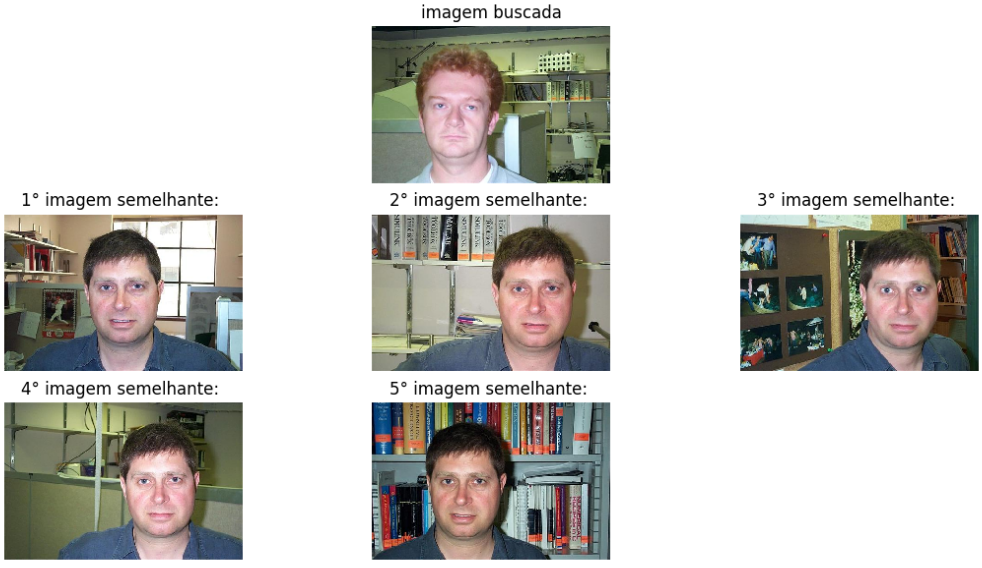

figura 4 - método 1: imagem da mesma categoria que as usadas no banco de dados

O resultado foi como o esperado. A imagens retornadas são todas rostos. Apesar da pessoa não ser a mesma, o programa conseguiu associar a imagem buscada com rosto.

O terceiro teste foi feito utilizando uma imagem que não estava na mesma categoria que as usadas para construir o vocabulário. O resultado obtido está na **figura 5**.

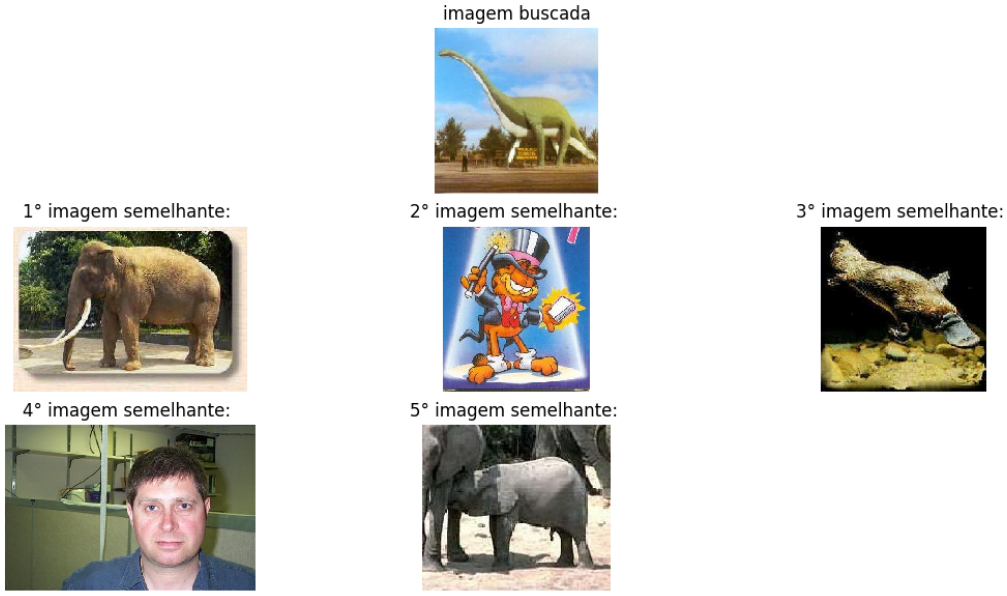

figura 5 - método 1: imagem com nenhuma relação com o banco de dados

Obviamente, as imagens não são brontossauros como a imagem buscada. Porém, é possível ver que neste caso, as imagens retornadas possuem formas ou coloração semelhante a imagem de busca, mostrando a eficiência do programa.

4. Desempenho da função do OpenCV

O segundo método utilizado foi uma função pronta da biblioteca do OpenCV utilizada para comparar histograma (cv2.HISTCMP_BHATTACHARYYA). Para fins de teste foram feitas as mesmas simulações acima, utilizando as mesmas imagens. Os resultados obtidos estão mostradas pelas **figuras de 6 à 8**.

figura 6 - método 2: imagem usada no banco de dados

figura 7 - método 2: imagem da mesma categoria que as usadas no banco de dados

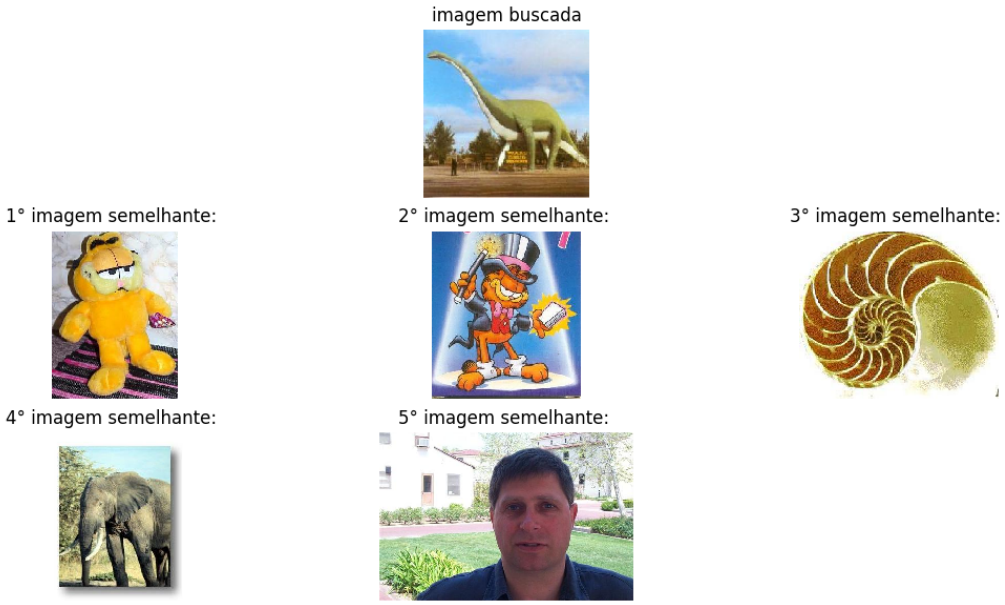

figura 8 - método 1: imagem com nenhuma relação com o banco de dados

Como é possível notar nas três simulações acima, o resultado não é tão exato, uma vez que o retorno nem sempre é da mesma categoria que a imagem. No primeiro caso a imagem mais semelhante realmente foi a mesma imagem, mas as outras imagens retornadas teve uma taxa de erro maior que o algoritmo do método 1. Já a segunda imagem tem um desempenho bem menor se comparado com o primeiro método, já que apenas duas das cinco imagens realmente eram rostos. Por fim, a imagem do brontossauro, assim como no método chi-square, retornou imagens com coloração ou forma semelhante a imagem buscada, não sendo possível verificar se sua eficiência é maior ou menor que o método 1.

5. Conclusões

Foi possível notar que o desempenho do chi-square foi superior à função do OpenCV. Os resultados foram mais satisfatórios no primeiro caso, já que as imagens retornadas eram da mesma categoria que a figura buscada. As velocidades na busca em ambos os métodos são bem parecidas e rápidas, não sendo este um bom critério de desempate.

6. Futuras Melhorias

Como melhoria é possível apontar a interface gráfica, já que no momento é necessário digitar diretamente no terminal, assim não é nada prático para o usuário.

Além de precisar digitar todo o caminho da imagem e seu nome, dando espaço para muito erro de digitação e consequentemente falha na execução do programa.

Outro ponto, seria a melhora na alteração do banco de dados. Caso queira alterar o conteúdo do banco de dados é necessário adicionar diretamente no código do vocab.py o nome da nova pasta. Uma possível melhoria seria facilitar esse processo de retreinar o banco sem a necessidade de digitar código.