Trabalho 4

MC920 João Vitor Araki Gonçalves (176353)

Introdução

O objetivo desse trabalho é analisar a aplicação de operadores morfológicos para identificação de regiões de texto e não texto numa imagem, e posteriormente identificar as palavras indivíduais e calcular o número de palavras numa imagem.

O Programa

O programa foi implementado em python 3.7.3 utilizando as seguintes bibliotecas:

- numpy: Calculos vetorizados
- opencv2

Execução

Primeiramente instalar as dependencias do projeto:

```
pip install -r requirements.txt
```

O programa pode ser executado pela linha de comando:

```
python main.py [input_file1] [input_file2] [distance_ratio]
```

O programa irá utilizar os arquivos `input_file1` e `input_file2` como entradas e utilizará o `distance_ratio` como limiar para determinar as distancias dos descritores.

O programa irá utilizar 4 métodos distintos para determinar os descritores das imagens:

- ORB (Oriented FAST and Rotated BRIEF)
- SURF (Speeded-Up Robust Features)
- SIFT (Scale-Invariant Feature Transform)
- BRIEF (Binary Robust Independent Elementary Features)

Para cada um dos métodos será salvo duas imagens, uma com os matches dos descritores encontrados nas duas imagens com as linhas de correspondência, e uma outra com o resultado da junção das duas imagens com efeito de imagem panorâmica.

Processo

Passo 1: Leitura e conversão para escala de cinza

A imagem é lida pelo método `cv2.imread(filename)` que lê a imagem original rgb, e então convertemos ela para a escala de cinza por meio do método `cv2.cvtColor(image1_, cv2.COLOR_BGR2GRAY)`

Passo 2: Encontrando pontos de interesse

Utilizando as imagens em escala de cinza obtidas no passo anterior vamos aplicar diferentes métodos para encontrar os descritores de cada uma das imagens, para que possamos posteriormente utiliza-los para uni-las. Como dito anteriormente, foram utilizados 4 métodos:

- ORB (Oriented FAST and Rotated BRIEF)
- SURF (Speeded-Up Robust Features)
- SIFT (Scale-Invariant Feature Transform)
- BRIEF (Binary Robust Independent Elementary Features) Com exceção do **ORB**, todos os outros são métodos proprietários que foram removidos da distribuição padrão do opencv, então foi necessário recompilar o opencv junto do pacote opencv-contrib, para reabilitar esses métodos. O **ORB** foi criado justamente como alternativa para esse problema, sendo uma alternativa open source dos outros métodos proprietários.

Uma breve descrição do funcionamento de cada um desses métodos:

1. SIFT: Uma alternativa invariante à escala métodos de identificação de cantos que eram sensíveis a escala, é um método patenteado, portanto não pode ser utilizado para fins comerciais sem a utilização da versão paga do opencv.
2. SURF: O SIFT funciona bem porém é computacionalmente caro, o método SURF foi criado para remediar esse problema, que é basicamente uma versão mais rápida do SIFT. Assim como o SIFT, o método é patenteado.
3. BRIEF: No método SIFT cada feature encontrada pesa um total de 512 bytes, e no SURF no mínimo 256 bytes por feature. Portanto é relativamente caro em questões de memória termos milhares de features encontradas com esses métodos, essa é a vantagem do BRIEF, ele não encontra as features, mas é um método de descrever essas features tomando menos espaço de memória. Para identificar os keypoints nesse método utilizamos o SIFT como identificador de features e o BRIEF para descreve-las.
4. ORB: Método desenvolvido pelo opencv, foi criado como alternativa para os métodos SURF e SIFT, só que aberto e gratuito. Ele funciona basicamente juntando o identificador de features FAST com o descritor BRIEF. Ele tende a identificar cantos nas imagens.

Passo 3: Computando distâncias entre descritores

O método utilizado para encontrar distâncias foi por brute force, utilizando o brute force matcher do opencv `bf = cv2.BFMatcher()`, e os matches foram encontrados comparando os descritores das features de cada imagem. `bf.knnMatch(des1, des2, k=2)`.

Passo 4: Selecionando correspondências

Para selecionar as correspondências, comparamos as distâncias de um descritor para o outro, caso a distância encontrada seja menor que o limiar fornecido pelo usuário, vamos considerar como uma correspondência.

Outro método possível seria comparando as duas melhores correspondencias de um descritor, caso a distancia da melhor seja menor que 0.75 vezes a segunda melhor, consideramos como uma correspondência válida. Esse método elimina 90% das correspondências ruins e 5% das boas, como determinado no paper do D. Lowe sobre o método SIFT.

Passo 5: Utilizando RANSAC para estimar homografia

Foi utilizado a função `cv2.findHomography(src, dst, cv2.RANSAC, 5.0)` para encontrar a matriz de transformação de perspectiva da imagem (matriz de homografia). Só é possível encontrar a homografia se temos pelo menos 4 correspondências válidas. Como podem existir erros nas correspondências, é utilizado o método RANSAC, que faz uma análise estatísca nelas e elimina possíveis outliers. Então mesmo que tenhamos 4 correspondências, pode ser que o programa não consiga gerar a homografia, pois as correspondências podem ser consideradas inválidas pelo RANSAC.

Passos 6 e 7: Alinhando as imagens e gerando uma imagem panorâmica

Para alinhar as imagens, foi utilizado o método `v2.warpPerspective(img1_, H, (img2_.shape[1] + img1_.shape[1], img2_.shape[0]))` do open cv, que com a matriz de homografia realiza a transformação na imagem correspondente para realizar o alinhamento, então com a imagem alinhada, juntamos as duas para gerar a imagem panorâmica.

Passo 8: Desenhando linhas de correspondência

Foi utilizado um método do opencv para desenhar as linhas de correspondência. `cv2.drawMatchesKnn(img1_, kp1, img2_, kp2, matches, None)` , onde passamos os keypoints, as duas imagens e as correspondências, e ele retorna as imagens unidas com as linhas ligando as correspondências encontradas.

Algumas comparações entre os métodos

Primeiramente em relação ao número de descritores encontrados para cada método:

```
Number of sift descriptors: 5342
Number of orb descriptors: 500
Number of surf descriptors: 3150
Number of brief descriptors: 5075
```

O SIFT foi constantemente o método que mais encontrou descritores nas diferentes imagens, seguido do SURF. Como o método BRIEF é apenas um descritor e utilizamos o SIFT para determinar os keypoints, ele possui um número similar de descritores que o SIFT, porém menor, como é esperado pois a vantagem do BRIEF como descritor é salvar espaço de memória nos descritores. Já o método ORB retornou sempre um número constante de 500 descritores, para qualquer imagem, isso é devido à implementação do método, que por padrão retém apenas 500 features.

Provavelmente, devido ao número reduzido de features retornadas pelo ORB, ele é o método mais sensível à limiar das correspondências, com um limiar de 100, apenas ele não encontra correspondências válidas o suficiente para encontrar a matriz de homografia.

A última imagem (5A e 5B) foi a única que não mostrou resultados bons durante o processo. Isso se dá possívelmente pelo fato de que ela tem um padrão muito uniforme, causando com que diversos pontos de interesse tenha distâncias baixas, mesmo não sendo corretos. Ocasionando numa imagem final distorcida.