Trabalho 4 - Introdução ao Processamento de Imagem Digital

Enoque Alves de Castro Neto (230211)[1]

[1]Universidade Estadual de Campinas
Instituto de Computação

enoquealvesufc@gmail.com

1. Introdução

Este relatório foi desenvolvido para detalhar o funcionamento e os resultados do trabalho 4 da disciplina *MO443*.O objetivo deste trabalho é aplicar técnicas de detecção de pontos de interesse para registrar um par de imagens e criar uma imagem panorâmica formado pela ligação entre as imagens após sua correspondência.

Toda a implementação do trabalho foi feita no jupter-notebook e cada função está separada por bloco. Então basta executar os blocos separadamente para usar cada função.

2. Implementação

2.1. Visão Geral

Para a implementação deste trabalho foi utilizado a linguagem Python 3.7.3, com o uso das bibliotecas OpenCV, Numpy, Matplotlib e imageio. De maneira geral, a biblioteca Numpy foi utilizada para a manipulação das imagens através de arrays Numpy. A biblioteca OpenCV foi usada para carregar a imagem e armazenar em um array Numpy e também foi utilizadas várias funções necessárias para cumprir os requisitos deste trabalho, como a função *cv2.findHomography()* que retorna uma máscara que especifica os pontos de valor único e discrepante, por exemplo. Já a biblioteca Matplotlib foi usada para plotar as imagens geradas em cada requisito para este trabalho e o imagemio foi utilizado para salvar as imagens no formato *.jpeg*.

2.2. Carregamento da imagem

Para o carregamento da imagem, foi utilizada a função *cv2.imread("diretório_da_imagem.png", cv2.IMREAD_GRAYSCALE)* do OpenCV que retorna sua matriz de pixel. O primeiro argumento da função indica o diretório da imagem e o segundo indica que estamos interessados apenas nas escalas de cinzas. Para o segundo argumentos também temos as seguintes opções:

1. cv2.IMREAD_COLOR: Carrega uma imagem colorida. Qualquer transparência dessa imagem será negligenciada. Caso nenhum argumento seja colocado, essa opção será a padrão.
2. cv2.IMREAD_UNCHANGED: Carrega uma imagem tal como ela é, incluindo um canal alpha.

2.3. Criar uma imagem panorâmica

O primeiro passo é escolher qual detector de pontos de interesse será usado. Foi utilizado para os testes 5 detectores de pontos de interesses, que são eles:

1. cv2.xfeatures2d.SIFT_create() - **scale-invariant feature transform (SIFT)**
2. cv2.xfeatures2d.SURF_create() - **speeded up robust features (SURF)**
3. cv2.xfeatures2d.BriefDescriptorExtractor_create() - **Robust Independent Elementary Features (BRIEF)**
4. cv2.ORB_create() - **Oriented FAST, Rotated BRIEF (ORB)**

Após escolher o detector de pontos de interesse, basta utilizar sua função de detecção. A função de detecção do **SIFT** é **sift.detectAndCompute(img, None)**, por exemplo.

Para executar o passo 3 do trabalho, foi utilizada a função **cv2.FlannBasedMatcher**. *Fast Library for Approximate Nearest Neighbors***(FLANN)** é uma biblioteca para realizar pesquisas aproximadas de vizinhos mais próximos em espaços dimensionais elevados. Ela contém uma coleção de algoritmos que encontramos para funcionar melhor na pesquisa de vizinhos mais próximos e um sistema para escolher automaticamente o melhor algoritmo e os parâmetros ideais, dependendo do conjunto de dados. Após inicializar um **FLANN**, basta usar a função **FLANN.knnMatch()** passando como parâmetro o resultado obtido no passo anterior.

Após realizar o passo 3, é preciso escolher um limiar para selecionar as melhores correspondência para os descritores. O limiar escolhido para realizar os testes computacionais deste trabalho foi de *0.4*. Para realizar o passo 5, como a descrição do trabalho sugere, foi utilizado a função **cv2.findHomography**. Essa função recebe os bons pontos da primeira imagem e os bons pontos da segunda. O retorno dessa função é a matriz de homografia e uma máscara que especifica os pontos de outlines.

Para a aplicar a projeção de perspectiva também foi utilizada a função sugerida pela a descrição do trabalho. A função **cv2.warpPerspective** como parâmetro a primeira imagem, a matriz de homografia, a largura da primeira imagem somado com a da segunda e, por ultimo, a altura da primeira imagem. Essa função retorna a primeira imagem com a intersecção da segunda. Para criar a imagem panorâmica final, basta juntar com a segunda imagem.

2.4. Desenhar retas entre pontos correspondentes no par de imagens

Para este passo final, foi criado uma função chamada **visualize_homografia**. Essa função apenas ajusta alguns parâmetros como cor das retas e chama a função **cv2.drawMatches**, que é a responsável por desenhar as retas na imagem. Essa função recebe 7 argumentos como parâmetros, que são eles: Imagem1, Imagem2, KeyPoints1, KeyPoints2, Macthes, Homografia e Mask. O retorno é a imagem com as retas desenhadas.

3. Resultados

Para os testes computacionais, foi escolhido as imagens *foto1A.jpg* e *foto1B.jpg*, disponibilizada nas especificações do trabalho. Foi utilizado 4 tipos diferentes de detectores de pontos de interesse, sendo eles: **SIFT**, **SURF**, **BRIEF** e **ORB**. Primeiramente, as duas imagens que geraram os resultados apresentado nos próximos capítulos pode ser encontrada na figura 1.

Figura 1. Imagens recebidas na entrada

3.1. Imagens panorâmicas

3.1.1. SURF

O resultado para o detector de pontos de interesses **SURF** pode ser visto a seguir:

Figura 2. Imagem panorâmica usando SURF

A junção da imagem ficou praticamente perfeito usando **SURF**. O que difere de uma imagem realmente perfeita é apenas uma leve mudança de tons de cinza que ocorre mais ou menos no centro da imagem. O motivo desse fato ocorrer é que a iluminação das imagens de entrada são diferentes. Caso as imagens tivessem a mesma iluminação, mudando apenas a perspectiva, ficaria praticamente perfeito o resultado. A matriz de homografia resultante é a seguinte:

$$H = \begin{bmatrix} 7.67009536e-01 & 4.09952690e-02 & 4.46316114e+02 \\ -1.35480532e-01 & 9.16000416e-01 & 7.53882873e+01 \\ -2.10694458e-04 & -2.78507085e-05 & 1.00000000e+00 \end{bmatrix}.$$

3.1.2. SIFT

Para a detector **SIFT**, os resultados obtidos são os seguintes:

Figura 3. Imagem panorâmica usando SIFT

A diferença do resultado obtido por **SURF** para o **SIFT** é apenas uma pequena ”esticada”na foto resultante. A matriz de homografia resultante é a seguinte:

$$H = \begin{bmatrix} 7.59387670e-01 & 3.20371160e-02 & 4.48625890e+02 \\ -1.36675710e-01 & 9.06463015e-01 & 7.73930002e+01 \\ -2.13362770e-04 & -3.77624603e-05 & 1.00000000e+00 \end{bmatrix}.$$

3.1.3. BRIEF

Para a detector **BRIEF**, os resultados obtidos são os seguintes:

Figura 4. Imagem panorâmica usando BRIEF

Se olhar de maneira bem atenta na imagem resultante, a junção das duas imagens está um pouco distorcida. Porém isso passa quase despercebido e o resultado final é bem positivo. A matriz de homografia resultante é a seguinte:

$$H = \begin{bmatrix} 7.58519584e-01 & 8.54691495e-02 & 4.37959105e+02 \\ -1.55776095e-01 & 9.53332172e-01 & 7.21015438e+01 \\ -2.45436825e-04 & 2.74935641e-05 & 1.00000000e+00 \end{bmatrix}.$$

3.1.4. ORB

Para a detector **ORB**, os resultados obtidos são os seguintes:

Figura 5. Imagem panorâmica usando ORB

O resultado obtido com o detector **ORB** é aparentemente o melhor de todos. É a imagem que é mais bem integrada e mais nítida. A matriz de homografia resultante é a seguinte:

$$H = \begin{bmatrix} 7.67665827e-01 & 5.32997842e-02 & 4.41198428e+02 \\ -1.40929579e-01 & 9.22041770e-01 & 7.57455890e+01 \\ -2.18924837e-04 & -1.62979452e-05 & 1.00000000e+00 \end{bmatrix}.$$

3.2. Retas entre pontos correspondentes

Para todos os testes computacionais, o limiar usado foi *0.4*.

3.2.1. SIFT

As retas entre os pontos correspondentes obtidos a partir do **SIFT** é apresentado a seguir:

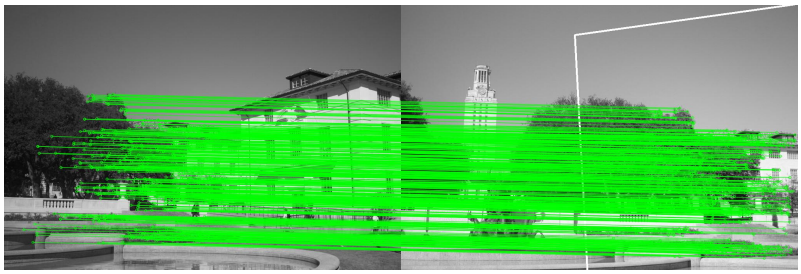

Figura 6. Retas entre pontos correspondentes usando SIFT

A quantidade de retas obtidas com o limiar *0.4* foi bem grande. Provavelmente cobrindo todos os pontos em comum. Um pequeno ruido foi criado durante o processo de criação das retas.

3.2.2. SURF

As retas entre os pontos correspondentes obtidos a partir do **SURF** é apresentado a seguir:

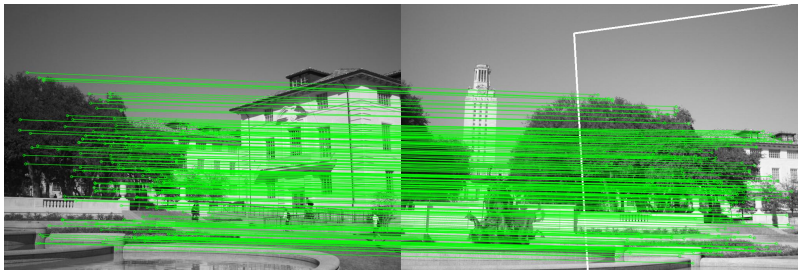

Figura 7. Retas entre pontos correspondentes usando SURF

A quantidade de retas obtidas com o limiar *0.4* foi bem grande, porem é bem menor que o obtido com **SIFT**. O mesmo ruído obtido por **SIFT** também persistiu apareceu em **SURF**.

3.2.3. BRIEF

As retas entre os pontos correspondentes obtidos a partir do **BRIEF** é apresentado a seguir:

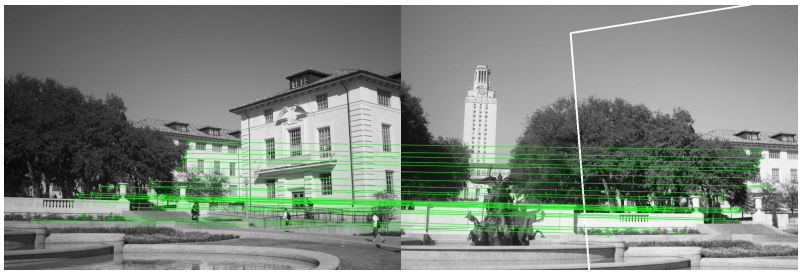

Figura 8. Retas entre pontos correspondentes usando BRIEF

A quantidade de retas obtidas com o limiar *0.4* foi bem pequena. Passando pouco do limite de 4 pontos de interesse, apresentado nos requerimentos do trabalho. O ruído da imagem também aparece nesse método. Mesmo com poucos pontos de interesses encontrados, a imagem panorâmica resultante desse método foi de alta qualidade.

3.2.4. ORB

As retas entre os pontos correspondentes obtidos a partir do **ORB** é apresentado a seguir:

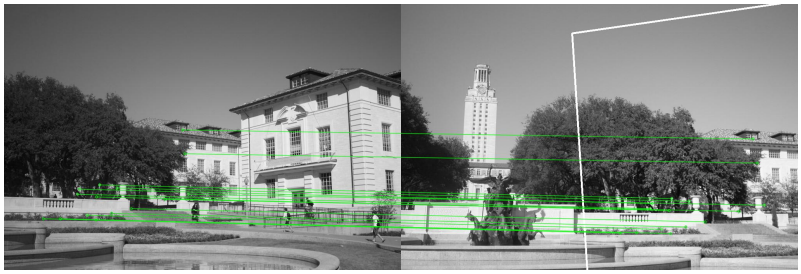

Figura 9. Retas entre pontos correspondentes usando ORB

A quantidade de retas obtidas com o limiar *0.4* foi bem pequena, ainda menor que apresentada anteriormente. Chegando assim bem próximo do limite de 4 pontos de interesse, apresentado nos requerimentos do trabalho. Infelizmente o ruído da imagem também aparece nesse método. Mesmo com poucos pontos de interesses encontrados, a imagem panorâmica resultante desse método foi, na minha opinião, a de melhor qualidade.

4. Conclusão

Neste trabalho foi utilizadas várias técnicas de detecção de pontos de interesses para registrar um par de imagens e criar uma imagem panorâmica formado pela ligação entre as imagens. Novamente, neste trabalho foi utilizado várias funções do OpenCV, se mostrando assim bem poderoso. Os resultados obtidos foram satisfatórios e cumpriram bem os requisitos do trabalho.

Referências

Pedrini, H. and Schwartz, W. R. (2008). *Análise de imagens digitais: princípios, algoritmos e aplicações*. Thomson Learning.