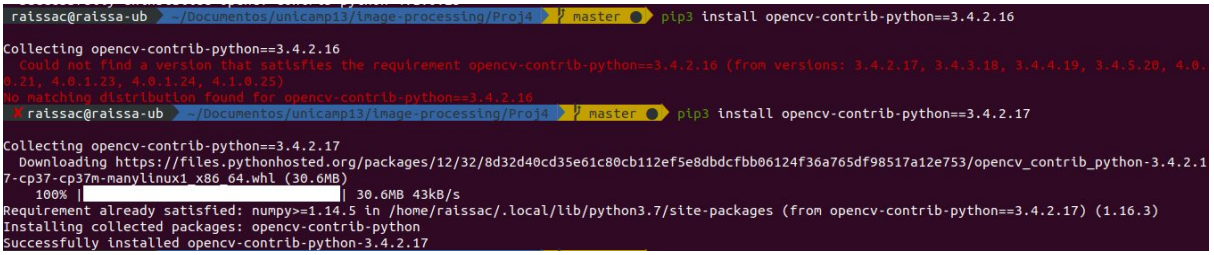
Relatório do Trabalho 4

Raissa Cavalcante Correia - RA: 150619
Disciplina: MC920 - 1s2019 - Turma A
10 de Junho de 2019

Condições do Ambiente de Execução

O último trabalho foi feito usando a versão 4.1 do CV porém para este foi necessário desinstalar usando o comando "`pip3 uninstall opencv-python`" assim ficando com a versão 3.4.2, juntamente com o pacote do pip3 "`opencv-contrib-python==3.4.2.17`" que foi a versão mais antiga que foi possível instalar, como podemos ver na captura de tela abaixo. Foi descoberto o problema das versões e uma inspiração para a solução no link do stack [2].



Sobre o código

Leitura da Imagem

```
print("type a number between 1 to 5")
num_pair = input()
image_got_1 =  'foto'+num_pair+'A.jpg'
image_got_2 =  'foto'+num_pair+'B.jpg'
img1 = cv2.imread(image_got_1)
img2 = cv2.imread(image_got_2)
```

A ideia é que ao digitar um número, apenas o número de 1 a 5, você escolhe entre os pares de imagens de 1 a 5, e assim fazemos a leitura

Escala cinza

```
grayA = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
grayB = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
```

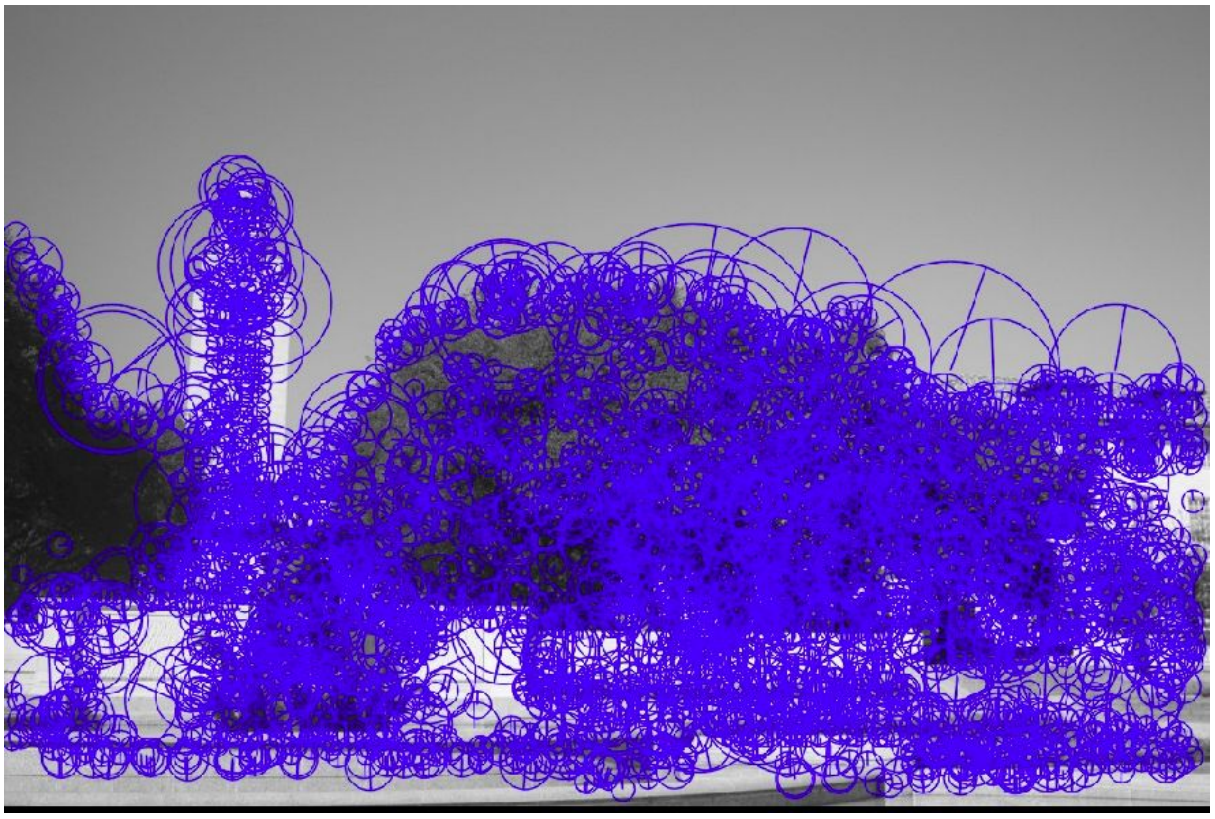
Essa foi a forma mais prática encontrada para tornar a imagem para a escala cinza conforme visto na referência[1], e é posta na pasta raiz pois será usado com frequência nas próximas etapas. Um resultado de exemplo:

SIFT

Graças ao links da referência [3] foi possível fazer os próximos itens com exceções de modificações necessárias para a versão 3 do CV. Da mesma forma que os algoritmos abaixo, o SIFT gera os keypoints e o descritor, que são salvos para cada uma das duas imagens, para cada método. A explicação para o SIFT pode ser encontrada em [4]. E o resultado pode ser visto na pasta SIFT, pois também desenhamos os keypoints e escrevemos a imagem no arquivo de saída.

SURF

A partir da referência [3] temos um link para [5] que contém toda a explicação do algoritmo em si. Na primeira linha temos que o SURF é criado com o Hessian threshold igual a 400, foi mantido assim dada a explicação fornecida resumidamente em [6] sobre quais as implicações em um threshold maior ou menor, visto que 400 é um valor intermediário para este contexto. Assim temos um código conforme a referência [7]. No caso deste código o DrawKeyPoints veio com a flag=4 e com a cor azul para as marcações, para diferenciar do SIFT deixei assim. Resultados na pasta SURF, e um exemplo abaixo:



BRIEF

O BRIEF não tem uma imagem resultante, por conta de seus keypoints serem usados em conjunto com outro algoritmo como o ORB. O que temos é apenas o cálculo do descritor e dos keypoints para cada uma das 2 imagens. Feito conforme a referência [8] e modificado para que seja executável com o CV3 como no link [9].

ORB

Conforme a referência [10] o ORB segue o mesmo padrão que os códigos acima. Gerando os keypoints e o descritor apenas com o DrawKeyPoints com flag 0 e com as marcações em verde, neste caso. O ORB não precisou de uma adaptação usando o xfeatures2d como os 3 anteriores, e o create tem um parâmetro que é o número máximo de features a se detectar que colocamos arbitrariamente como ORB_FAST_SCORE, que achei apropriado pelo fato do ORB ser apoiado no Fast e o fast determinará o número máximo de features[11]. O ORB não seguirá em frente nos próximos itens como o SIFT e o SURF pois por conta de uma

issue no github[12], pode-se perceber que não é possível aplicar o resultado do ORB diretamente sobre o FLANN pois:

You got the error because you tried hamming type over euclid type:
"Unsupported format or combination of formats (type=0)", where type=0 is CV_8U (as hamming descriptors outputs like BRIEF, ORB, FREAK, AKAZE etc).

Distâncias/Similaridade com FLANN

Conforme em [13] é explicado que

“For FLANN based matcher, we need to pass two dictionaries which specifies the algorithm to be used, its related parameters etc. First one is IndexParams. For various algorithms, the information to be passed is explained in FLANN docs. As a summary, for algorithms like SIFT, SURF etc. you can pass following:”

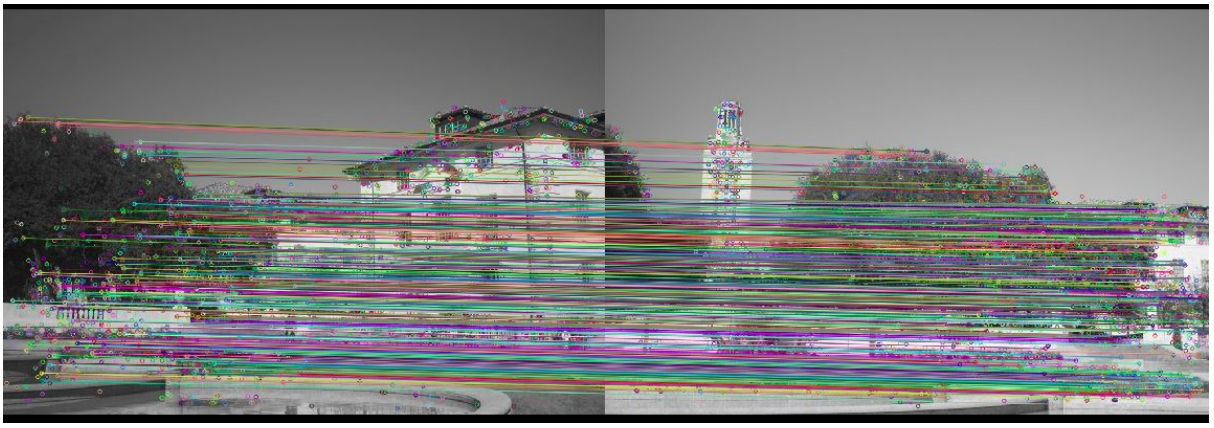
Por conta disso não corri tanto atrás em qual seria o melhor valor para estes 2 parâmetros, segui o que é mais comumente usado e pesquisei seu significado em [14]:

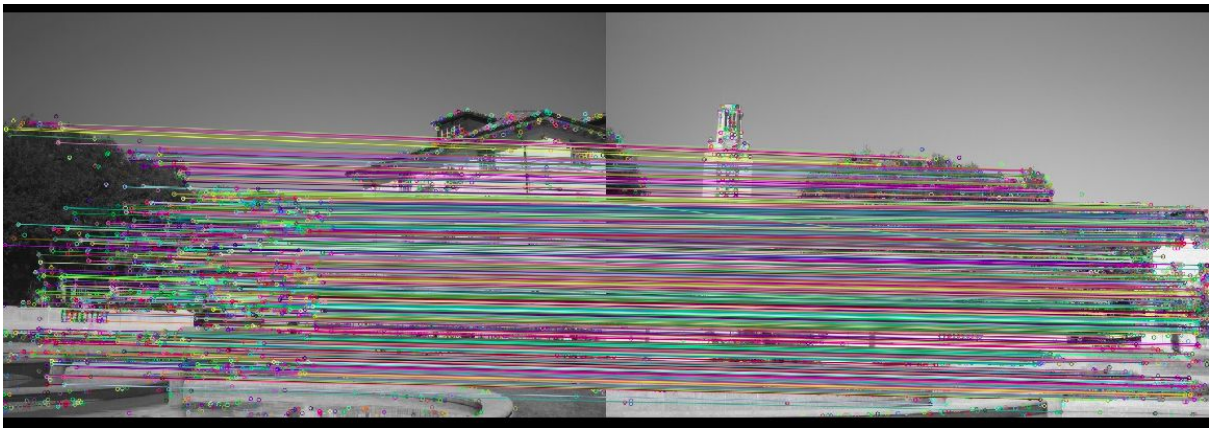
 # number of times the trees in the index should be recursively traversed
 # Higher values gives better precision, but also takes more time

E além disso é baseado na busca “K-nearest neighbor” onde k=2.

Selecionando as melhores relações

Aplicando para o resultado do SURF e do SIFT, podemos ver em suas respectivas pastas, a escolha dos pontos onde no caso escolhemos 0.6 como razão máxima da distância entre os pontos comparados pelo FLANN nos descritores da imagem A e da imagem B.
Conforme foi dado de exemplo em [15]. Um exemplo do resultado dos itens 3 e 4 é visível com o SURF e com o SIFT respectivamente:





Homografia, Warp e Panorâmica

Os itens 5, 6 e 7 foram feitos todos como parte de uma coisa só, foi totalmente baseado em [15], [16] e [17] uma para cada item do enunciado. O objetivo inicial é encontrar a matriz que usaremos no warp no caso “M”. Calculando através de todos os pontos considerados bons dado a razão do item 4, e seus key points, usando o RANSAC conforme pedido. Conforme proposto originalmente realizamos tudo isso apenas se houver 10 pontos no mínimo.

Após isso é realizado um trabalho de transformação de perspectiva levando em conta a matriz Homography, e traçando as linhas entre as versões P&B das imagens originais.

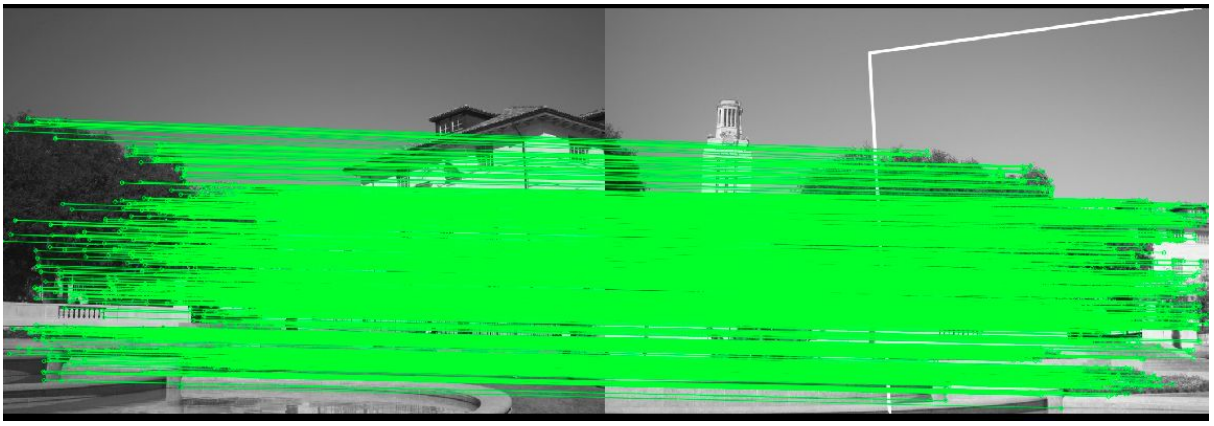
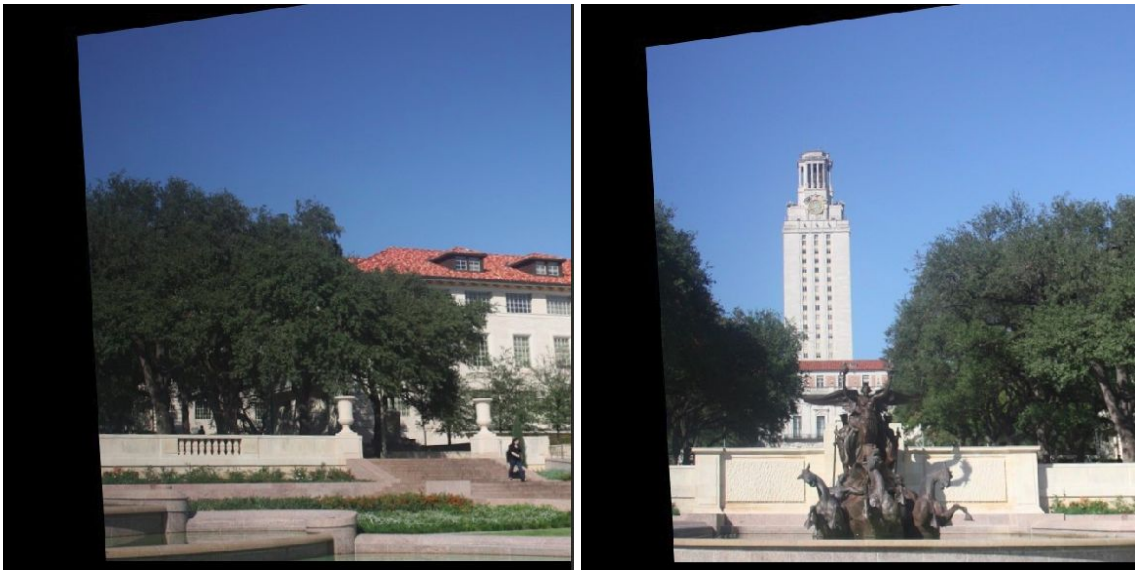


Imagem após o item 5 com os keypoints calculados pelo SIFT.

Na sequência aplicamos a warpPerspective, aplicando tudo que temos até agora, que é a matriz, os parâmetros de largura e altura da imagem, e a imagem original tanto na imagem A quanto na imagem B, obtendo algo como este par:



E por fim criamos a panorâmica como no final da referência [17], obtendo a panorâmica:

Conclusões

Devido a ampla pesquisa por referências, os resultados foram de acordo com o esperado, uma vez que tais algoritmos são amplamente conhecidos, usados e debatidos em diversas variações. Seja variações da versão do CV quanto dos parâmetros que podemos utilizar em cada função dessas bibliotecas.

Referências

[1] https://docs.opencv.org/3.1.0/de/d25/imgproc_color_conversions.html

[2] https://stackoverflow.com/questions/37039224/attributeerror-module-object-has-no-attribute-xfeatures2d-python-opencv-2

[3]https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_table_of_contents_feature2d/py_table_of_contents_feature2d.html

[4] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html#sift-intro

[5] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html#surf

[6] https://stackoverflow.com/questions/18744051/opencv-surf-hessian-minimum-threshold

[7]https://docs.opencv.org/3.4.0/df/dd2/tutorial_py_surf_intro.html

[8]https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_brief/py_brief.html#brief

[9] https://docs.opencv.org/3.4.3/dc/d7d/tutorial_py_brief.html

[10]https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_orb/py_orb.html#orb

[11]https://medium.com/software-incubator/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf

[12]https://github.com/opencv/opencv/issues/5937

[13]https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html

[14] https://www.programcreek.com/python/example/89440/cv2.FlannBasedMatcher

[15]https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_feature_homography/py_feature_homography.html#feature-homography

[16]https://docs.opencv.org/3.1.0/da/d6e/tutorial_py_geometric_transformations.html

[17]https://medium.com/pylessons/image-stitching-with-opencv-and-python-1ebd9e0a6d78