

Produção de panoramas e detecção de pontos de interesse

Nathália Harumi Kuromiya
Engenharia da Computação - Graduação
E-mail: n175188@dac.unicamp.br
RA 175188

10 de junho de 2019

1 Introdução

Esse trabalho tem o objetivo de mostrar como utilizar a detecção de pontos de interesse para produzir panoramas de imagens. Além disso, vamos comparar diferentes detectores e descritores de pontos para encontrar o melhor método para cada caso. Para a elaboração desse trabalho, foi necessário um estudo maior sobre registros, incluindo detectores e descritores. Para esse fim, utilizou-se o livro [1] e os slides de registros [2] e [3], disponibilizados pelo professor *Hélio Pedrini*. Além dessas fontes, foi utilizado tutoriais do OpenCV ([4], [5], [6], [7], [8] e [9]) para a aplicação de métodos utilizando essa biblioteca.

2 Trabalho Proposto [10]

O trabalho proposto se resume em: dada duas imagens de entrada no formato JPEG (*Joint Photographic Experts Group*), o algoritmo reconhece os pontos de interesse entre as duas imagens e produz duas imagens de saída para cada método testado: uma panorama com a junção das imagens, outra com os pontos de interesses destacados e conectados. Essas duas imagens também estão no formato JPEG. O programa também retorna o número total de pontos de interesses encontrado nas duas imagens em cada método.

3 Programas e bibliotecas utilizadas

A versão de Python[11] utilizada para implementação foi 3.5.4 e as bibliotecas de auxílio foram NumPy[12] 1.11.2 e OpenCV[13] 4.1.0-dev, juntamente com a OpenCV-contrib, dado que várias funções utilizadas são patenteadas e estão disponíveis nessa extensão.

4 Entradas e Saídas

A imagem de entrada e as de saída possuem o formato JPEG (*Joint Photographic Experts Group*). O programa também retorna dados no *stdout*.

5 Implementação e Resultados

5.1 Imagem-referência

Para comparação das técnicas utilizadas, usaremos as figuras 1, 2 e 3 como base para todos os experimentos.



Figura 1: Figuras originais 1 que servirão de base para comparações.

5.2 Metodologia

Para começar a produzir o panorama, as imagens precisam ser convertidas para níveis de cinza para que seja possível encontrar os pontos de interesse. Seguimos, então, para a utilização de cada detector e descritor, que será detalhado nas sub-subseções seguintes.



Figura 2: Figuras originais 2 que servirão de base para comparações.

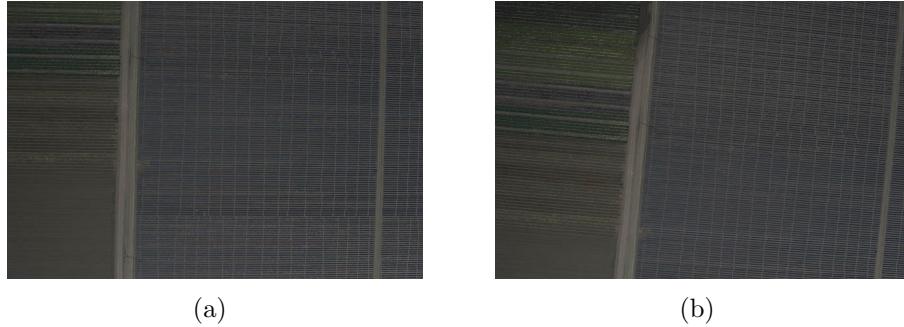


Figura 3: Figuras originais 3 que servirão de base para comparações.

A partir dos pontos de interesses encontrados, foi aplicado duas filtragens das combinações de pontos diferentes para a produção do panorama: uma filtragem se baseia em pegar as 20 primeiras combinações de pontos que possui a menor diferenças entre eles; já o segundo método aplica um limiar de 70% de proximidade entre as combinações e pega todas que passam desse limiar. Esses dois métodos foram escolhidos, uma vez que 20 combinações pode ser muito pouco para produzir o panorama, ao mesmo tempo que um limiar de 70% precisa ser flexibilizado em algumas situações, seja pra mais ou menos correspondência. Os resultados desses dois métodos será apresentado conforme a explicação dos descritores utilizados.

Por fim, para a junção do panorama, foi calculada a matriz de homografia com *RANSAC* pela função *cv2.findHomography()*. Nesse ponto, é importante

ressaltar que é possível encontrar duas matrizes, dependendo da ordem que é assumida para as imagens. Por restrições do método de ajuste de perspectiva (`cv2.warpPerspective()`), a matriz escolhida sempre foi a que altera a imagem da direita/de baixo, dado que se a alteração fosse feita na imagem da esquerda/de cima, haveria uma perda de informação na imagem final que prejudica totalmente o resultado, de forma a parecer não ter feito um panorama, ou seja, a imagem final fica igual a imagem original.

5.2.1 SIFT

O **SIFT** (*Scale-Invariant Feature Transform*) é responsável por encontrar pontos de interesses quando há diferença de escala entre as imagens.

Para a criação e aplicação desse detector e descritor, foi utilizado a função `cv2.xfeatures2d.SIFT_create()`. Ele foi capaz de encontrar 4239 combinações de pontos para a imagem 1, 1280 para a imagem 2 e 54 para a imagem 3.

Após a filtragem das combinações pelos dois métodos citados anteriormente, encontramos as combinações apresentadas nas figuras 4, 5, 6, 7, 8, 9.

5.2.2 SURF

O **SURF** (*Speeded-Up Robust Features*) é um método baseado em SIFT, porém mais rápido. Ele é responsável por adaptar vários conceitos utilizados em SIFT, como a Laplaciana da Gaussiana, de forma a agilizar o encontro dos pontos.

Para a criação e aplicação desse detector e descritor, foi utilizado a função `cv2.xfeatures2d.SURF_create()`. Ele foi capaz de encontrar 5440 combinações de pontos para a imagem 1, 1551 para a imagem 2 e 2822 para a imagem 3.

Após a filtragem das combinações pelos dois métodos citados anteriormente, encontramos as combinações apresentadas nas figuras 10, 11, 12, 13, 14, 15.

5.2.3 BRIEF

O **BRIEF** (*Binary Robust Independent Elementary Features*) é um descritor que usa um método para encontrar strings binárias sem encontrar de fato descritores. Isso vem para resolver o problema de uso de memória encontrada nos métodos anteriores.

Dado que o *BRIEF* não oferece nenhum tipo de detector de pontos, utilizamos um detector *FAST*, oferecido pela biblioteca. Assim, temos como detector a função `cv2.xfeatures2d.StarDetector_create()` e como descritor `cv2.xfeatures2d.BriefDescriptorExtractor_create()`. Ele foi capaz de encontrar 918 combinações de pontos para a imagem [1](#) e 149 para a imagem [2](#). Porém, ele não foi capaz de encontrar pontos e descritores para a imagem [3](#), não sendo possível fazer uma panorama dessa imagem com esse método.

Após a filtragem das combinações pelos dois métodos citados anteriormente, encontramos as combinações apresentadas nas figuras [16](#), [17](#), [18](#), [19](#).

5.2.4 ORB

O **ORB** (*Oriented FAST and Rotated BRIEF*) é uma alternativa ao SIFT e ao SURF em relação a custo computacional, além de não ser patenteado. Ele é uma fusão do detector FAST com o descritor BRIEF (como fizemos anteriormente), porém com uma série de modificações para melhorar performance, principalmente em relação a rotação, em que o BRIEF não funciona muito bem.

Para a criação e aplicação desse detector e descritor, foi utilizado a função `cv2.ORB_create()`. Ele encontra um número máximo fixo de pontos, por isso, foi capaz de encontrar 500 combinações de pontos para a imagem [1](#) e [2](#). Para a imagem [3](#), encontrou 174 combinações.

Após a filtragem das combinações pelos dois métodos citados anteriormente, encontramos as combinações apresentadas nas figuras [20](#), [21](#), [22](#), [23](#), [24](#), [25](#).

5.3 Imagens das combinações

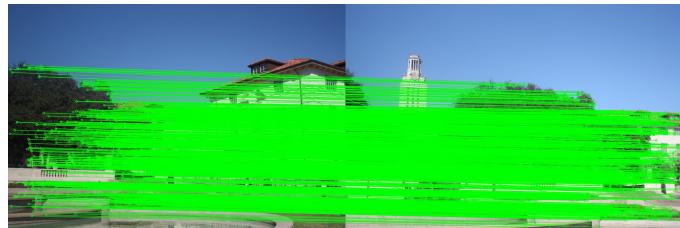


Figura 4: Figura 1 com os pontos de interesse utilizando SIFT e com filtro de 75% de proximidade entre os pontos combinados



Figura 5: Figura 1 com os pontos de interesse utilizando SIFT e com filtro de 20 combinações com maior proximidade

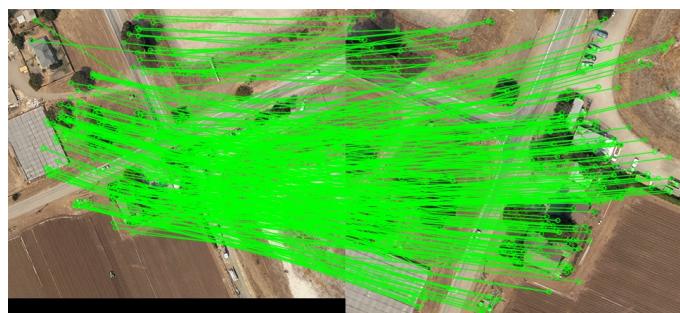


Figura 6: Figura 2 com os pontos de interesse utilizando SIFT e com filtro de 75% de proximidade entre os pontos combinados



Figura 7: Figura 2 com os pontos de interesse utilizando SIFT e com filtro de 20 combinações com maior proximidade



Figura 8: Figura 3 com os pontos de interesse utilizando SIFT e com filtro de 75% de proximidade entre os pontos combinados



Figura 9: Figura 3 com os pontos de interesse utilizando SIFT e com filtro de 20 combinações com maior proximidade

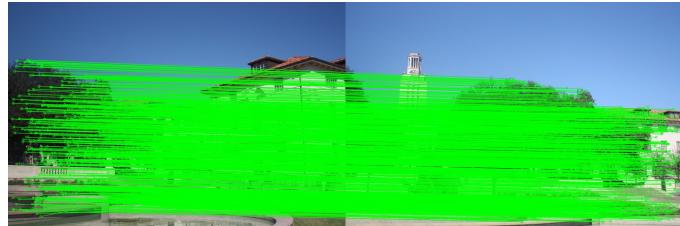


Figura 10: Figura 1 com os pontos de interesse utilizando SURF e com filtro de 75% de proximidade entre os pontos combinados



Figura 11: Figura 1 com os pontos de interesse utilizando SURF e com filtro de 20 combinações com maior proximidade



Figura 12: Figura 2 com os pontos de interesse utilizando SURF e com filtro de 75% de proximidade entre os pontos combinados



Figura 13: Figura 2 com os pontos de interesse utilizando SURF e com filtro de 20 combinações com maior proximidade

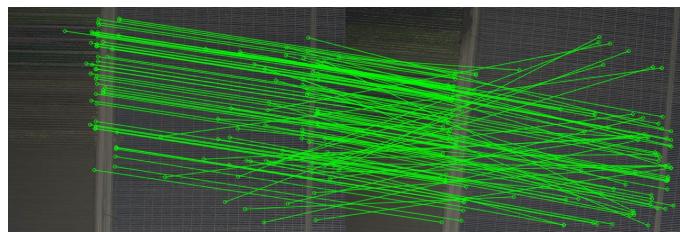


Figura 14: Figura 3 com os pontos de interesse utilizando SURF e com filtro de 75% de proximidade entre os pontos combinados

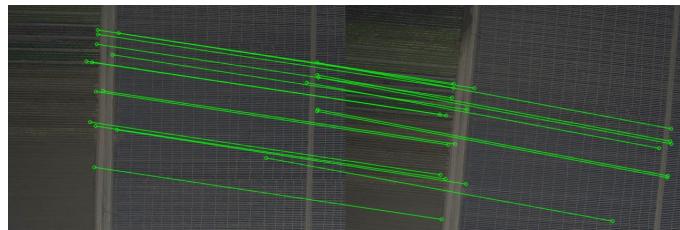


Figura 15: Figura 3 com os pontos de interesse utilizando SURF e com filtro de 20 combinações com maior proximidade



Figura 16: Figura 1 com os pontos de interesse utilizando BRIEF e com filtro de 75% de proximidade entre os pontos combinados

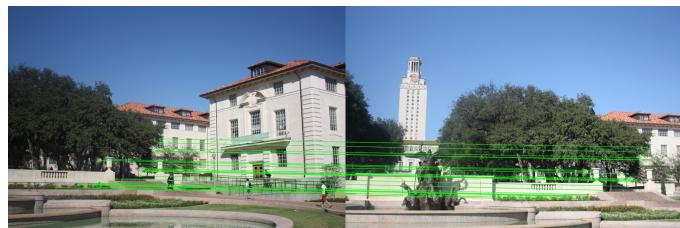


Figura 17: Figura 1 com os pontos de interesse utilizando BRIEF e com filtro de 20 combinações com maior proximidade



Figura 18: Figura 2 com os pontos de interesse utilizando BRIEF e com filtro de 75% de proximidade entre os pontos combinados

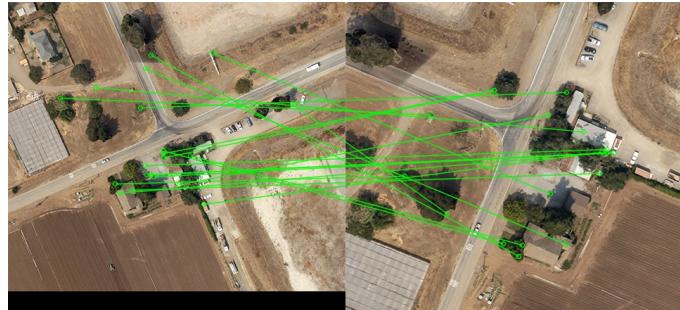


Figura 19: Figura 2 com os pontos de interesse utilizando BRIEF e com filtro de 20 combinações com maior proximidade

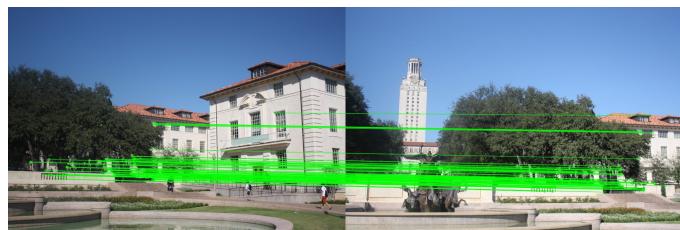


Figura 20: Figura 1 com os pontos de interesse utilizando ORB e com filtro de 75% de proximidade entre os pontos combinados



Figura 21: Figura 1 com os pontos de interesse utilizando ORB e com filtro de 20 combinações com maior proximidade



Figura 22: Figura 2 com os pontos de interesse utilizando ORB e com filtro de 75% de proximidade entre os pontos combinados



Figura 23: Figura 2 com os pontos de interesse utilizando ORB e com filtro de 20 combinações com maior proximidade



Figura 24: Figura 3 com os pontos de interesse utilizando ORB e com filtro de 75% de proximidade entre os pontos combinados

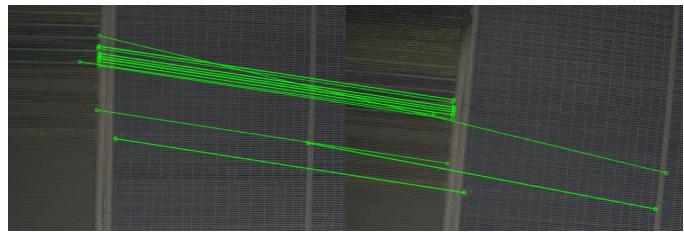


Figura 25: Figura 3 com os pontos de interesse utilizando ORB e com filtro de 20 combinações com maior proximidade

5.4 Panoramas

Os panoramas se encontram a seguir, juntos para melhor comparação:



(a) SIFT - 75% de proximidade



(b) SIFT - 20 combinações mais próximas



(c) SURF - 75% de proximidade



(d) SURF - 20 combinações mais próximas

Figura 26: Panoramas SIFT e SURF resultantes da figura 1



(a) BRIEF - 75% de proximidade



(b) BRIEF - 20 combinações mais próximas



(c) ORB - 75% de proximidade



(d) ORB - 20 combinações mais próximas

Figura 27: Panoramas BRIEF e ORB resultantes da figura 1



(a) SIFT - 75% de proximidade



(b) SIFT - 20 combinações mais próximas



(c) SURF - 75% de proximidade

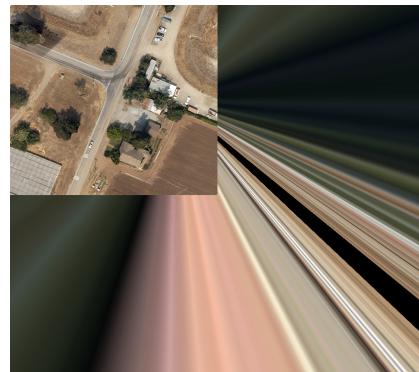


(d) SURF - 20 combinações mais próximas

Figura 28: Panoramas SIFT e SURF resultantes da figura 2



(a) BRIEF - 75% de proximidade



(b) BRIEF - 20 combinações mais próximas

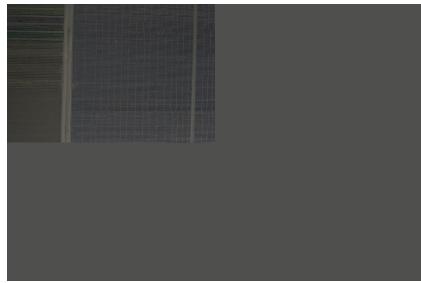


(c) ORB - 75% de proximidade

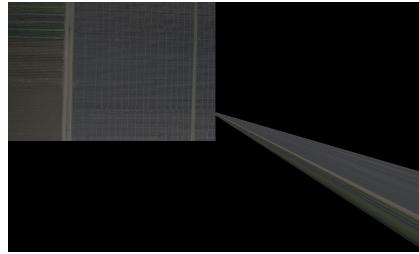


(d) ORB - 20 combinações mais próximas

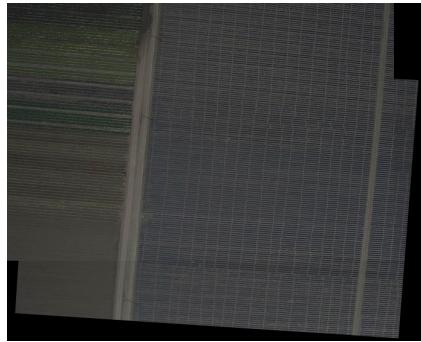
Figura 29: Panoramas BRIEF e ORB resultantes da figura 2



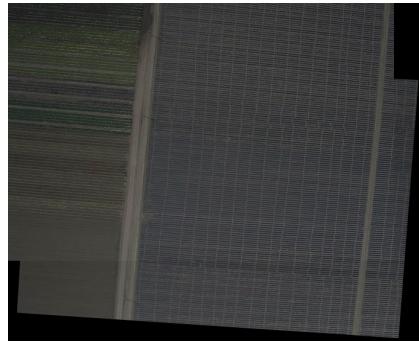
(a) SIFT - 75% de proximidade



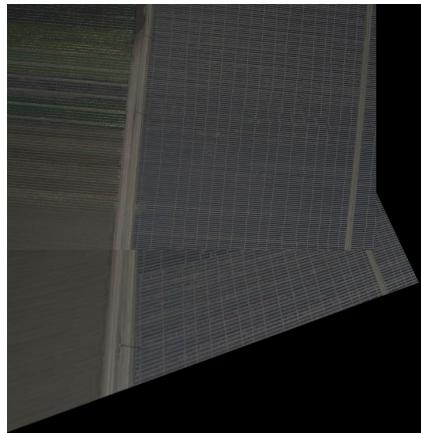
(b) SIFT - 20 combinações mais próximas



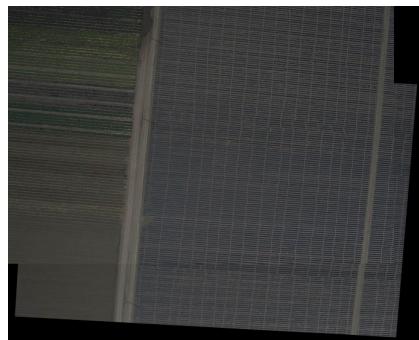
(c) SURF - 75% de proximidade



(d) SURF - 20 combinações mais próximas



(e) ORB - 75% de proximidade



(f) ORB - 20 combinações mais próximas

Figura 30: Panoramas SIFT, SURF e ORB resultantes da figura 3

6 Conclusão

A implementação dos métodos para criar o panorama se mostrou eficiente, dado que conseguimos pelo menos 1 panorama de cada imagem como resultado dos 4 métodos.

No que tange a imagem 1, todos os panoramas apresentaram um bom resultado. Isso porque as duas imagens a serem conectadas estão na mesma perspectiva, sem problemas de rotação ou de escala. Isso faz com que os métodos tenham mais facilidade para encontrar a correspondência entre as imagens.

Já para a imagem 2, que possui tanto uma variação de escala quanto uma rotação significativa, conseguimos ver mais problemas nos métodos. Devido ainda a grande correspondência das imagens, os métodos de SIFT e SURF mostraram panoramas satisfatórios. Em contrapartida, o BRIEF, que é um método que não possui um bom tratamento para rotações, não conseguiu entregar um panorama da imagem. O ORB conseguiu um panorama bom com o método de filtragem de 20 melhores combinações, porém o de 75% de correspondência resultou em um panorama levemente desfigurado.

A imagem 3 mostra os resultados mais significativos. É uma imagem com uma rotação leve, porém com uma textura muito parecida ao longo da imagem toda. Isso faz com que os pontos de correspondência encontrados nem sempre estejam corretos, ainda que a correspondência seja alta. O resultado disso são panoramas irregulares. Com o método de SIFT, não foi possível encontrar uma panorama regular. As duas panoramas encontradas fazem correspondências errôneas e, portanto, mostram um resultado não satisfatório. O método de SURF encontrou bons panoramas com os dois métodos de filtragem. O BRIEF não encontrou correspondências entre as duas imagens, sendo impossível a produção de um panorama, ainda que errôneo. Por fim, o ORB mostrou um panorama satisfatório com o método de 20 correspondências mais próximas, porém um desfigurado com o método de 75% de proximidade das correspondências.

Com toda essa análise, conseguimos dizer que o trabalho conseguiu mostrar tanto a utilização dos métodos quanto seus resultados esperados com as diferentes imagens e suas configurações (em relação a escala, rotação e textura).



Referências

- [1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. Pearson, 2017. 1
- [2] H. Pedrini, “Aula: Registros de imagens.” [Online]. Available: http://www.ic.unicamp.br/~helio/disciplinas/MC920/aula_registro.pdf 1
- [3] ——, “Aula: Registros - complemento.” [Online]. Available: http://www.ic.unicamp.br/~helio/disciplinas/MC920/aula_complemento.pdf 1
- [4] “Opencv: Introduction to sift (scale-invariant feature transform).” [Online]. Available: https://docs.opencv.org/4.1.0/da/df5/tutorial_py_sift_intro.html 1
- [5] “Opencv: Introduction to surf (speeded-up robust features).” [Online]. Available: https://docs.opencv.org/4.1.0/df/dd2/tutorial_py_surf_intro.html 1
- [6] “Opencv: Brief (binary robust independent elementary features).” [Online]. Available: https://docs.opencv.org/4.1.0/dc/d7d/tutorial_py_brief.html 1
- [7] “Opencv: Orb (oriented fast and rotated brief).” [Online]. Available: https://docs.opencv.org/4.1.0/d1/d89/tutorial_py_orb.html 1
- [8] “Opencv: Feature matching.” [Online]. Available: https://docs.opencv.org/4.1.0/dc/dc3/tutorial_py_matcher.html 1
- [9] “Opencv: Feature matching + homography to find objects.” [Online]. Available: https://docs.opencv.org/4.1.0/d1/de0/tutorial_py_feature_homography.html 1
- [10] H. Pedrini, *trabalho 4*. [Online]. Available: <http://www.ic.unicamp.br/~helio/disciplinas/MC920/trabalho4.pdf> 1
- [11] “Python 3.” [Online]. Available: <https://www.python.org/> 2
- [12] “Numpy.” [Online]. Available: <https://www.numpy.org/> 2
- [13] “Opencv.” [Online]. Available: <https://opencv.org/> 2