

Universidade Estadual de Campinas

Instituto de Computação

Introdução ao Processamento Digital de Imagem (MC920 / MO443)

Trabalho 4

Leodécio Braz - 230219

9 de junho de 2019

1 Especificação do Problema

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.

2 Script

O script foi desenvolvido em Python na versão 3.0 e foi utilizado o framework Jupyter para documentação do mesmo. As bibliotecas utilizadas foram **OpenCV** [3], **Numpy** [2] e **matplotlib** [1].

2.1 Entrada

As imagens de entrada estão no formato JPEG (*Joint Photographic Experts Group*) e foram disponibilizadas pelo professor no endereço que se encontra em sua página¹. Além destas, também foram colocadas fotografias retiradas pelo autor.

As imagens estão localizadas no diretório `./imagens-registro/`, e já se encontram declaradas no código.

2.2 Saída

Após a execução, o script irá gerar tanto resultados intermediários visíveis na tela, quanto as imagens resultante dos processos que serão salvas com o formato JPEG (*Joint Photographic Experts Group*) no diretório `./output/` e seus nomes seguem o padrão como na Tabela 1..

¹<http://www.ic.unicamp.br/~helio/imagens-registro/>

Tabela 1: Nome dos arquivos de saída

Nome do arquivo da imagem	Descrição
[imagem_entrada]_[metodo].resultado	O resultado final do processo com a imagem de entrada e o método.
[imagem_entrada]_[metodo].matches	As retas desenhadas entre os keypoints obtidos utilizando o método

2.3 Leitura das imagens e declaração dos kernels

As imagens de entrada foram lidas através da função **imread** da biblioteca do OpenCV, que gera um *Array* do Numpy de dimensões MxN que corresponde ao tamanho da imagem.

3 Resolução

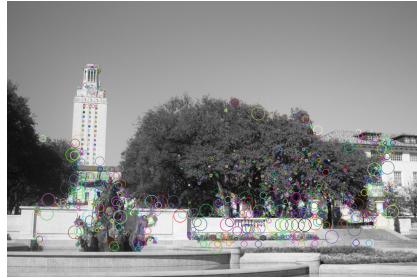
Após a conversão das imagens coloridas de entrada, em imagens de níveis de cinza. Encontramos os pontos de interesse (*KeyPoints*) e descritores invariantes locais para o par de imagens. Para isso, utilizamos os descritores SIFT (*Scale Invariant Feature Transform*), SURF (*Speed Up Robust Feature*), BRIEF (*Robust Independent Elementary Features*) e ORB (*Oriented FAST, Rotated BRIEF*) através das bibliotecas do OpenCV. A Figura 1 mostra os desenhos de pontos de interesses encontrados pelo descritor BRIEF para um par de imagens.

Figura 1: Par de imagens e seus pontos de interesse

(a) Imagem A



(b) Imagem B



Após obter os pontos de interesse e descritores para as imagens de entrada, computamos a distância de similaridade entre estes descritores. Para isso, utilizamos a função **DescriptorMatcher_create** do OpenCV com o parâmetro “BruteForce” que faz com que utilize a distância L2. Em seguida utilizamos a função **knnMatch** que devolve as k melhores correspondências.

Após computar as distâncias entre os descritores, baseado em um valor de limiar (*threshold*), selecionamos apenas as melhores correspondências entre os descritores da imagens. Sendo assim, se a distância de um descritor for menor

que a distância do outro vezes o valor do limiar que foi passado, assinalamos essa correspondência como ‘boa’.

Se pelo menos 4 correspondências forem obtidas após a etapa anterior, através da função **findHomography** do OpenCV utilizando a técnica RANSAC (*RANdom SAmple Consensus*) podemos estimar a matriz de homografia. Para as imagens da Figura 1 a matriz de homografia H gerada foi:

$$H = \begin{bmatrix} 7.57676209e-01 & 4.22670533e-02 & 4.46828726e+02 \\ -1.41974964e-01 & 9.15875760e-01 & 7.68268810e+01 \\ -2.21657315e-04 & -2.35689040e-05 & 1.00000000e+00 \end{bmatrix}$$

Além disso, também desenhamos retas entre pontos correspondentes no par de imagens. Primeiramente, colocamos as imagens uma ao lado da outra. Em seguida, percorremos o vetor de correspondências e se os pontos de uma imagem e da outra tiverem tido um “match” desenhamos uma reta entre estes pontos. A Figura 2 mostra as retas desenhadas entre uma boa parte dos pontos que são mostrados na Figura 1.

Figura 2: Correspondência entre as imagens



Após estas etapas, utilizamos a função **warpPerspective** do OpenCV para aplicar uma projeção de perspectiva e alinhar as imagens. Esta função recebe um imagem e, baseado na matriz de homografia, aplica uma transformação na mesma. Após isso, unimos as duas imagens gerando apenas uma, com o aspecto de imagem panorâmica. A Figura 3 ilustra a imagem obtida após o final destes processos.

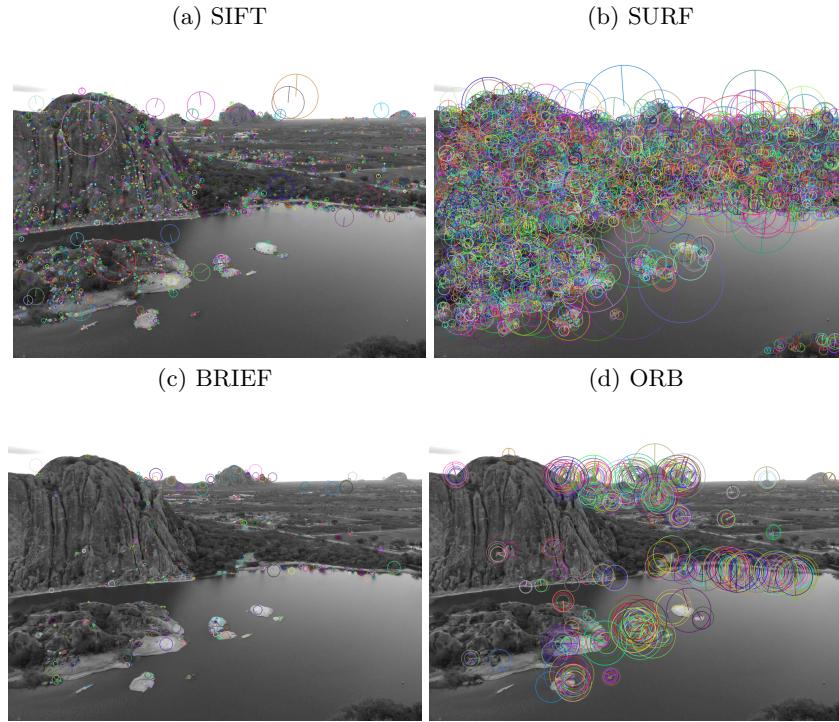
Figura 3: Imagem panorâmica obtida



4 Resultados

Para cada um dos detectores e descritores utilizados, diferentes pontos de interesse e vetores de características são obtidos. Observamos que para as imagens de exemplos analisadas, em geral, utilizando os métodos SIFT e SURF assinalam uma quantidade maior de *Keypoints* em comparação com os dois outros métodos BRIEF e ORB como é possível visualizar na Figura 4.

Figura 4: *Keypoints* detectados de uma imagem de entrada utilizando os métodos

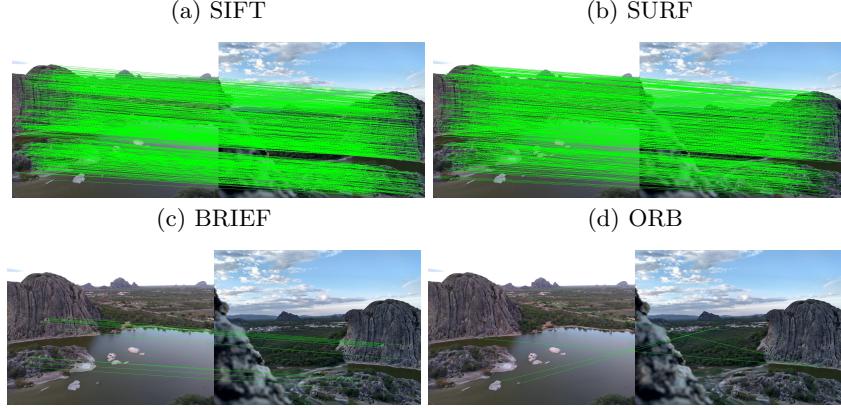


Estas maiores quantidade de *keypoints* encontrados está diretamente ligado a quantidade de correspondências assinaladas na imagem. Para a Figura 4 mostrada e o seu par, a Figura 5 mostra as retas entre os *keypoints* correspondentes que foram encontrados utilizando os métodos apresentados.

É possível notar a grande quantidade de correspondências entre as imagens quando utilizamos os descritores SIFT e SURF. É importante ressaltar que, para o exemplo que está sendo mostrado, o quanto bem essas correspondências são assinaladas por ambos os métodos. Da mesma forma, as correspondências traçadas pelo descriptor BRIEF são visualmente satisfatórias. Já o descriptor ORB visualmente não obteve boas correspondências.

A quantidade de correspondências encontradas não é o fator determinante da qualidade de um método e podemos perceber isso através da imagem que

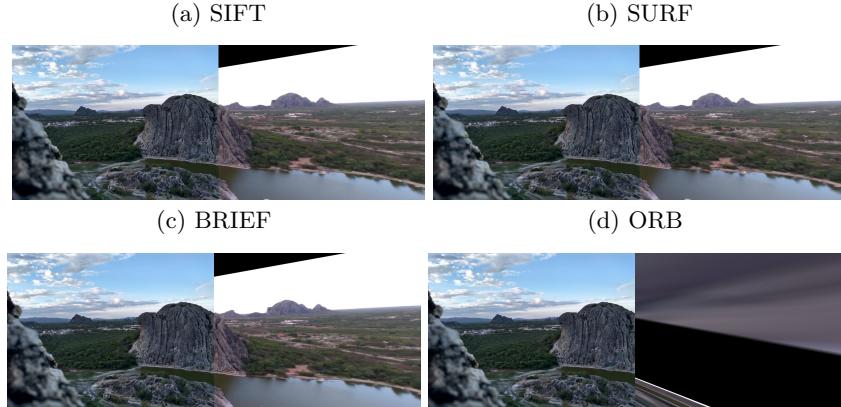
Figura 5: Correspondência entre os *keypoints* de um par de imagens



estamos utilizando como exemplo, onde, tanto o SIFT e o SURF quanto o BRIEF, obtiveram um resultado satisfatório ao final do processo. Por outro lado, o descriptor ORB que não foi capaz de encontrar boas correspondências, não obteve resultado satisfatório. Podemos ver o resultado ao final dos processos através da Figura 6.

Nos experimentos realizados, os métodos SIFT, SURF e BRIEF foram os que mais apresentaram resultados satisfatórios para uma boa parte das imagens. O método ORB no entanto, apesar de ser robusto, não convergiu tão bem para grande parte das imagens que foram analisadas.

Figura 6: Correspondência entre os *keypoints* de um par de imagens



Referências

- [1] Matplotlib user guide. Accesso em: 05/05/2019.
- [2] Numpy user guide. Accesso em: 05/05/2019.
- [3] Welcome to opencv documentation! Accesso em: 05/05/2019.