MC920 - Trabalho 4: Registro

Naomi Takemoto
RA 184849
n184849@dac.unicamp.br

I. Introdução

Registro é o processo de combinar imagens de um mesmo cenário, porém capturadas em diferentes condições, como por exemplo iluminação ou perspectiva. Um exemplo de aplicação desse processo é a construção de mosaicos que combinam imagens que contém diferentes fragmentos de uma área. O resultado seria assim capaz de representar uma região maior. Isso tem utilidade por exemplo em sensoriamento remoto e análise de imagens de satélites, domínios nos quais a captura de dados pode acontecer em momentos diferentes e contar com distintas transformações geométricas. Ao se utilizar técnicas corretas também é possível fazer o reconhecimento de objetos, que é uma ferramenta útil em vários áreas, como veículos autônomos e sistemas de segurança.

Assim, métodos robustas para reconhecer pontos de interesse e criar de descritores invariantes a escala, rotação, e demais transformações geométricas são essenciais para o sucesso nas tarefas descritas anteriormente.

II. Resumo

Neste trabalho estudamos o algumas técnicas conhecidas e implementadas no *framework OpenCV*. Estudamos os seguintes detectores de pontos de interesse e descritores: SIFT (*Scale Invariant Feature Transform*), SURF (*Speed Up Robust Feature*), BRIEF (*Robust IndependentElementary Features*) e ORB (*OrientedFAST,RotatedBRIEF*).

III. Teoria

Esta seção foi escrita com base nos tutoriais sobre os algoritmos mencionados acima, disponíveis na documentação do *OpenCV* [1] [2] [3] [4]. A seguir uma breve descrição sobre os princípios de cada Algoritmo utilizado.

A. SIFT (Scale-Invariant Feature Transform)

Uma das maiores dificuldades das técnicas de registro é encontrar descritores que sejam invariantes às principais transformações geométricas, como rotação, escala, perspectiva, etc. Nesse sentido, o SIFT, tenta contornar o problema da escala, usando 4 passos principais:

1) Busca por descritores em diversas escalas. Isso se justifica pelo fato de que, alguns tipos de cantos por exemplo, não são precisamente detectados quando se usa técnicas cujos filtros têm tamanho constante. Já que quando há uma mudança de escala se torna inadequado.
2) Localização de pontos de interesse. Para cada localidade candidata da imagem, um modelo é determinado para determinar a localização e escala dessa região de interesse.
3) Atribuição de orientação: para cada ponto de interesse, uma ou mais orientações são atribuídas de acordo com o gradiente local na imagem. As operações futuras são realizadas considerando-se essas características de orientação, escala e localização para cada *feature*. Provendo assim invariância a essas transformações.
4) Descrição de ponto de interesse: para uma dada escala na região no entorno de cada ponto de interesse os gradientes locais são calculados. É feita então uma representação significantemente robusta a distorção local de forma e mudanças de iluminação.

A ideia do algoritmo é calcular o Laplaciano do Gaussiano para vários valores de σ. Essa matriz serve como detector de *blobs* em várias escalas distintas, por conta das mudanças em σ, variável que atua como parâmetro de escala.

B. SURF

SURF é uma versão mais rápida do SIFT, esse algoritmo conta com otimizações no cálculo do Laplaciano do Gaussiano (LoG) por um filtro caixa, que permite paralelização para diferentes escalas. Além disso o SURF se vale do determinante da matriz Hessiana para escala e localização dos descritores.

Para determinação de orientação o algoritmo faz uso de outras técnicas como *Wavelets*, que não serão tratadas neste relatório.

C. BRIEF

Este algoritmo surgiu para atacar o problema do excesso de memória que os algoritmos anteriores demandam para representar os descritores. Como é ressaltada na documentação da implementação no *OpenCV*, BRIEF é um descritor de *features*, não fornecendo portanto um método para encontrá-las. Em resumo essa técnica é permite que *matching* entre descritores de duas imagens ocorra mais rápido. Um ponto negativo é que ele não é robusto à rotação, tendo um desempenho ruim para pares imagens com essas características.

D. ORB

É uma alternativa eficiente e (não patenteada) para o SIFT e o SURF. Conforme definido na documentação da implementação, ORB é uma fusão de métodos de detecção rápida de pontos de interesse e do descritor BRIEF (modificado). Podendo ser utilizado em dispositivos com relativamente baixo poder de processamento, como celulares por exemplo para funcionalidades como criação de fotos panorâmicas.

IV. Metodologia

Para a realização dos experimentos foram utilizadas as funções já implementadas no *framework OpenCV - opencv-python* e *opencv-contrib-python* ambos em versão 3.4.2.

De maneira geral a aplicação dos algoritmos seguem os passos a seguir:

1) Computar pontos de interesse e descritores.
2) Calcular a distância entre os descritores das imagens que se deseja combinar.
3) Selecionar os melhores pontos de acordo com algum critério definido.
4) Calcular a matriz de homografia (técnicas como RANSAC - *Random Sample Consensus*, podem ser utilizadas).
5) Aplicar a transformada em uma das imagens e combinar com a outra para formar a imagem panorâmica.

V. Experimentos

A. Aplicação do SIFT

Nesta subseção será serão mostradas a execução dos passos descritos na metodologia para o SIFT de modo a exemplificar o processo que foi repetido para os outros algoritmos em experimentos subsequentes.

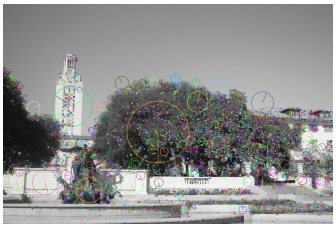
Primeiramente foi calculado os pontos de interesse e descritores para cada imagem (convertida para escala de cinza) que se desejava combinar (1a e 1b):

(a) Entrada para o SIFT, imagem A.

(b) Entrada para o SIFT, imagem B.

Figura 1

(a)



(b)

Figura 2: Pontos de interesse e descritores desenhados na imagem de entrada convertida para escala de cinza.

Posteriormente, com o uso da funcionalidade de *feature matching* do *framework*, calculou-se as distâncias entre os descritores, para o caso do SIFT, a métrica utilizada foi a norma $L2$.

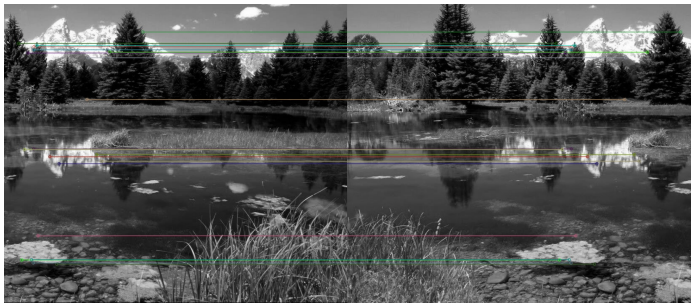


Figura 3: *Matching* entre os 20 descritores mais próximos (em termo da norma L2).

Usando é métrica descrita acima, foi possível selecionar os N melhores pontos de interesse, isto é aqueles cujas distâncias são mínimas. Depois foi calculada a matriz de homografia que (no caso 2D) é definida como a transformação projetiva planar que faz o mapeamento entre pontos de um plano para outro. Sendo necessários para tando a existência de pelo menos 4 pares de pontos correspondentes.

Para o par de imagens em questão a matriz de homografia encontrada foi:

$H_{3\times3}$ =

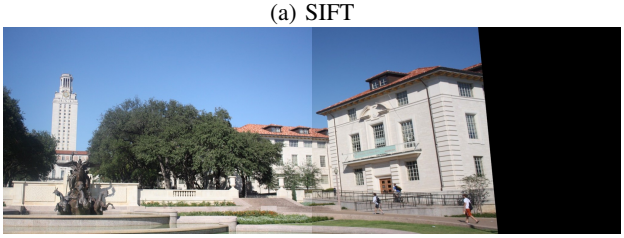
1.00009504e+00	1.90077844e-07	4.62993378e+02
4.01452659e-05	1.00003101e+00	-6.88286265e-03
2.64601459e-08	4.47316316e-08	1.00000000e+00

Vale notar que a escolha desses dos pontos para o cômputo da matriz de homografia é crucial para a obtenção da transformação correta, nesse sentido, para dar maior robustez ao procedimento foi utilizada a técnica RANSAC (*Random Sample Consensus*). Este método é iterativo e pode ser sumarizado como se segue:

1) Selecionar aleatoriamente os pontos necessários, dentre os de interesse.
2) Calcular a matriz de homografia H.
3) Calcular os *inliners*, ou seja os pontos que concordam com a matriz H.
4) Selecionar o maior conjunto de *inliners* encontrados.
5) Se o procedimento acima encontrar um subconjunto de cardinalidade maior ou igual que um limiar t. Utilizar esse subconjunto para estimar novamente o valor de H.
6) Caso contrário repete-se o procedimento desde o passo 1.
7) Se no fim de um número pré-determinado de tentativas, a condição do item 5 não for atendida o programa pode terminar com falha ou então ser solucionado para o maior conjunto de consenso encontrado.

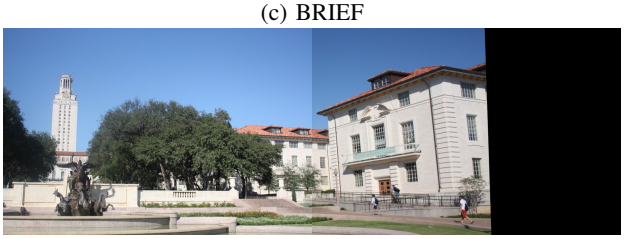
Depois de determinada a matriz de homografia, utilizando RANSAC, para aumentar a robustez do processe contra *outliers*, foi obtido o resultado mostrado na figura 4 (que conta com o resultado da aplicação dos demais algoritmos) fazendo o uso da matriz de homografia na imagem colorida.

(a) SIFT



(b) SURF

(c) BRIEF



(d) ORB

Figura 4: Resultados da combinação das imagens capturadas em diferentes condições de luminosidade.

Observando a figura 4, percebe-se que os 4 algoritmos analisados levaram a resultados semelhantes. Nota-se no uma característica particular para este par de imagens: as imagens contam com pouca diferença de rotação. Sendo possível verificar pela observação das imagens originais.

B. Imagens com alto grau de rotação relativa

Com o intuito verificar se existem pares de imagens para os quais a performance de uma técnica é melhor ou pior que as demais, utilizou se o par de imagens a seguir: Enquanto SIFT, SURF e ORB tiveram um desempenho visual parecido (na figura 6 (foram mostrados apenas os resultados do ORB como representante deste trio), o desempenho do BRIEF não foi satisfatório. No entanto esta limitação era prevista, já que este último, conforme descrito anteriormente, não apresenta invariância à rotação. Foi evidenciado também que o ORB de fato pode ser uma alternativa para SIFT e SURF.

Para uma análise mais quantitativa, comparamos as matrizes de homografias:

(a)

(b)

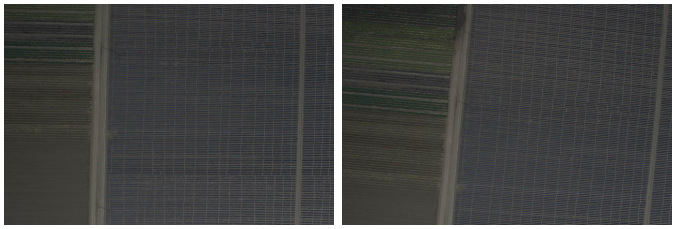
Figura 5: Imagens aéreas com maior rotação

$$H_{BRIEF} = \begin{bmatrix} -2.31323622e{+}00 & 4.00696614e{-}01 & 2.43469753e{+}02 \\ -3.32771103e{+}00 & 4.18530109e{-}01 & 3.77978834e{+}02 \\ -9.65601443e{-}03 & 1.70082281e{-}03 & 1.00000000e{+}00 \end{bmatrix}$$

$$H_{SIFT} = \begin{bmatrix} 1.15457795e{+}00 & 1.09149484e{+}00 & -2.19098569e{+}02 \\ -1.09342386e{+}00 & 1.15214489e{+}00 & 2.13192863e{+}02 \\ 1.09371737e{-}06 & -5.31778058e{-}06 & 1.00000000e{+}00 \end{bmatrix}$$

$$H_{SURF} = \begin{bmatrix} 1.15707937e{+}00 & 1.09329371e{+}00 & -2.19664047e{+}02 \\ -1.09509869e{+}00 & 1.15463546e{+}00 & 2.13060782e{+}02 \\ 5.72456926e{-}06 & -7.01077873e{-}06 & 1.00000000e{+}00 \end{bmatrix}$$

$$H_{ORB} = \begin{bmatrix} 1.22044468e{+}00 & 1.12228738e{+}00 & -2.33196258e{+}02 \\ -1.10425713e{+}00 & 1.19621756e{+}00 & 2.10556840e{+}02 \\ 9.95996405e{-}05 & 7.83459315e{-}06 & 1.00000000e{+}00 \end{bmatrix}$$



(a) (b)

Figura 7

Observa-se que as homografias resultantes do SIFT e do SURF são bastante semelhantes entre si, com todas as entradas correspondentes da matriz na mesma ordem de grandeza. Isso é válido também para a ORB - SIFT/SURF, com exceção da posição $H_{3,1}$, que para o ORB é uma ordem de grandeza maior. Para o BRIEF, diversas entradas diferem em várias ordens de suas correspondentes dos outros algoritmos.

(a) ORB

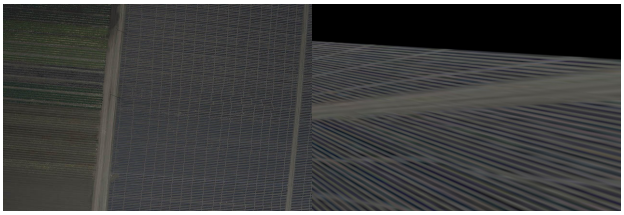
(b) BRIEF

Figura 6: Comparação entre os diferentes algoritmos quando aplicadas em um par de imagens com alto grau de rotação relativa.

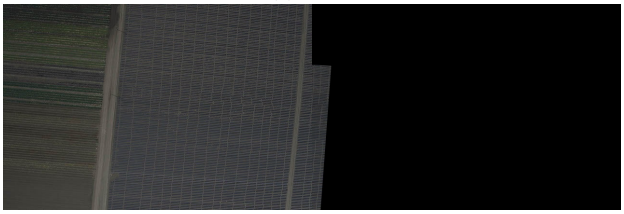
C. Imagens com múltiplas bordas e cantos

Finalmente, usou-se um par de imagens com vários possíveis pontos de interesse, figura 7, (tais como diversas bordas e cruzamentos) para avaliar as respostas dos algoritmos.

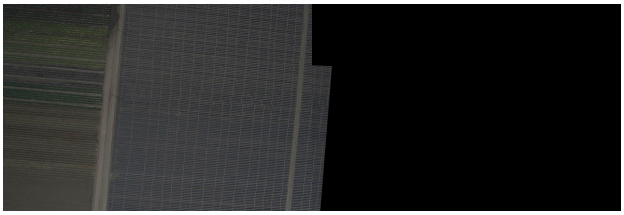
Ao executar os algoritmos, ocorreu um problema com o BRIEF, que não conseguiu encontrar descritores com *matching* entre as duas imagens. Isso impossibilitou o cálculo da matriz de homografia. Analisando as panorâmicas das outras imagens, percebe-se que o SIFT, falhou em combinar corretamente as imagens. Apenas ORB e SURF foram capazes de gerar um a resposta satisfatória.



(a) SIFT



(b) SURF



(c) ORB

Figura 8: Resposta dos algoritmos para o par de imagens com múltiplas bordas e cruzamentos.

VI. Conclusão

Aplicando os algoritmos acima, percebe-se que para o ORB apresenta a melhor relação custo benefício, podendo ser executado rapidamente até em dispositivos com poucos recursos computacionais. Este algoritmo gerou resultados comparáveis (ou melhores) ao SIFT e ao SURF, contatando com a vantagem de não estar patenteado.

Referências

[1] Introduction to SIFT (Scale-Invariant Feature Transform). Disponível em https://docs.opencv.org/trunk/da/df5/tutorial_py_sift_intro.html. Acesso em 2 de junho de 2019.

[2] Introduction to SURF (Speeded-Up Robust Features). Disponível em https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html. Acesso em 10 de junho de 2019.

[3] BRIEF (Binary Robust Independent Elementary Features).https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_brief/py_brief.html. Acesso em 10 de junho de 2019.

[4] ORB (Oriented FAST and Rotated BRIEF). https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_orb/py_orb.html. Acesso em junho de 2019.