Lista 2 – Visão Computacional

Aluno: Rennan de Lucena Gaio DRE: 119122454

Todo o código do trabalho pode ser acessado pelo link do github:

https://github.com/RennanGaio/visao_computacional/tree/master/lista2

1 Detector de Harris

1.1 Implementação

Diante de todas as imagens testadas, os procedimentos base de implementação foram os mesmos. Os procedimentos do exercício 1 são todos encontrados no arquivo "detector_de_harris.py" no diretório indicado.

Para calcular as derivadas d_x e d_y foi utilizado o operador de Sobel de dimensão $W_d = 3$. Para o cálculo da matriz de covariância C utilizou-se uma janela de tamanho $W_c = 5W_d$ assim como o sugerido no enunciado. Porém, em um dos testes, foi utilizado $W_c = 2$, como o sugerido pela biblioteca do OpenCV, e pode se analisar que o resultado encontrado foi interessante em um ponto de vista diferente. Uma vez que utilizando este novo valor de janela, ao invés de serem criadas regiões grandes dadas pela resposta de Harris, foi observado alguns pontos separados sem precisar de um novo limiar.

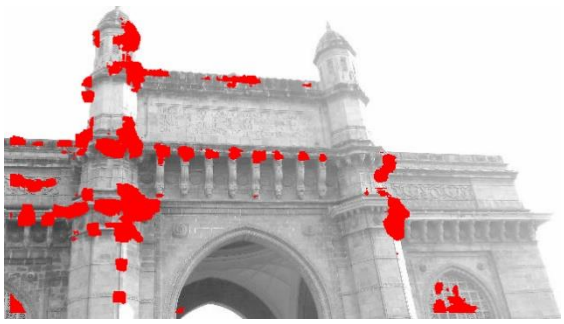
A implementação do Detector de Harris foi totalmente feita do zero, mas o seu tempo de resposta estava um pouco elevado para fazer vários experimentos mesmo utilizado o determinante e o traço da matriz C. Para calcular R de forma eficiente, não foi utilizado o cálculo do auto valor da matriz C, mas sim o seu traço e determinante. Então além da implementação própria, foi utilizada a implementação também do OpenCV do Python para realizar os experimentos mais rapidamente. Porém, foi observado que ambos os resultados eram iguais. O valor de $k = 0.04$ foi escolhido conforme o enunciado.

Para selecionar os pontos característicos, foi utilizado diferentes testes por imagem como é descrito no próximo sub-capítulo de resultados. Logo os valores T_r e W_h foram escolhidos de forma diferente para cada imagem e teste realizado.

1.2 Resultados

Para cada imagem será descrito os resultados abaixo do detector de Harris. Primeiramente, pode ser visto um comparativo entre o algoritmo implementado e o do OpenCV com os mesmos parâmetros de entrada $W_h = 50$, $T_R = 0.03 \max(R)$:

Implementação própria

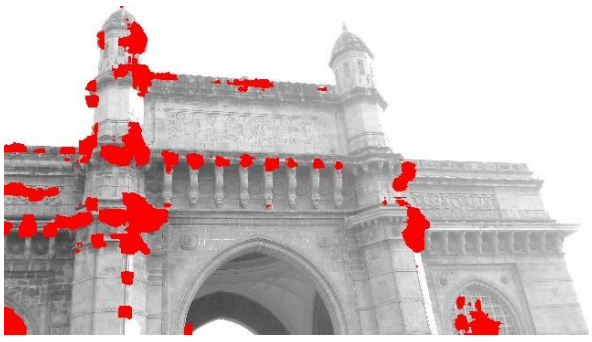


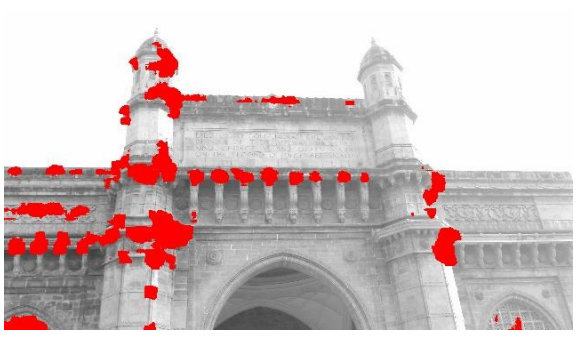
Implementação OpenCV

1.2.1 Imagens goi

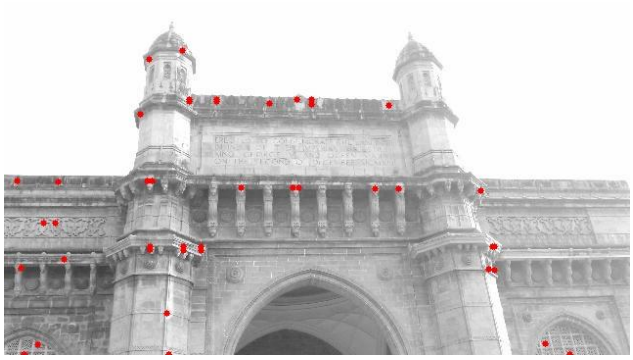
Imagens originais (1 e 2 respectivamente).

Imagem representando os pontos cuja resposta de Harris foi maior que o limiar $T_R = 0.03 \max(R)$ antes de utilizar a redução de grupos pela janela W_H.





Como foi observado que a quantidade de pontos era muito grande em “clusters” foram testadas diferentes janelas para esta configuração de limiar. Dentre elas janelas de tamanho 20, 50 e 100. Os resultados para janelas de tamanho 20 e 100 foram piores do que para as janelas de tamanho 50. A primeira por possuir muitos pontos e a segunda por ter pouquíssimos pontos. Então neste relatório só será apresentado os resultados com janelas de 50 pixels, porém os demais resultados podem ser vistos no repositório git. Para a formação das janelas, a imagem foi dividida em grids de tamanho 50 x 50. Abaixo estão os resultados para as 2 imagens.

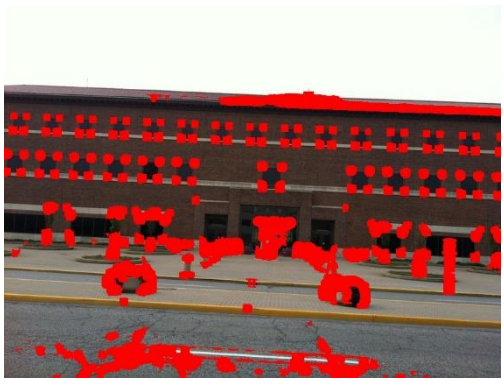


Observando os resultados foi possível observar uma correspondência relevante entre pontos nas suas localizações. Logo este resultado foi mantido com as configurações já citadas.

1.2.2 Imagens building

Imagens originais (1 e 2 respectivamente).

Imagem representando os pontos cuja resposta de Harris foi maior que o limiar $T_R = 0.01 \max(R)$ antes de utilizar a redução de grupos pela janela W_H. Foi utilizado um limiar diferente do da primeira imagem pois se observou que o limiar de 0.03 não capturava todas as quinas das janelas.



Como foi observado que a quantidade de pontos era muito grande em “clusters” foram testadas diferentes janelas para esta configuração de limiar. Dentre elas janelas de tamanho 20, 25, 50 e 100. Os resultados para janelas de tamanho 20, 50 e 100 foram piores do que para as janelas de tamanho 25. Então neste relatório só será apresentado os resultados com janelas de 25 pixels, porém os demais resultados podem ser vistos no repositório git. Para a formação das janelas, a imagem foi dividida em grids de tamanho 25 x 25. Abaixo estão os resultados para as 2 imagens.

Pode-se perceber que ainda existem muitos pontos que não são quinas, mas neste resultado foi possível capturar a maior parte das quinas presentes nas janelas.

2 Casamento de Características

2.1 Implementação

Todo o código referente a segunda parte pode ser encontrado de forma separada no arquivo “casamento_caracteristicas.py” no repositório git. A partir dos pontos encontrados pelo detector de Harris, foi elaborada uma função de casamento de pontos a partir de sua vizinhança, como mencionado no enunciado da lista. Para comparar os tons de cinza presentes nas janelas dos pontos selecionados de ambas as imagens foi utilizado o SSD com o tamanho da janela W_{SSD} diferente para cada teste realizado presente no próximo subcapítulo.

Para eliminar falsas correspondências foram definidas 2 variáveis T_{SSD} e $T_{razaoSSD}$. A primeira delas cria um limitador superior para o erro SSD, e a segunda elimina casos em que o ponto pode ser facilmente confundido entre a primeira e segundo melhor opção.

Abaixo serão apresentadas as imagens compostas com suas respectivas correspondências, além de todos os parâmetros utilizados para cada uma delas.

2.2 Resultados

2.2.1 Imagens goi

Para o casamento das imagens goi foram utilizados os seguintes parâmetros para o melhor resultado:

W_H = 20

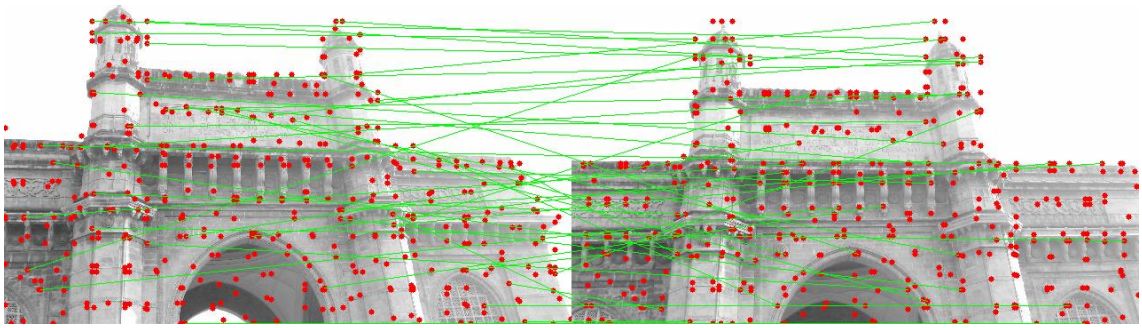
T_R = 0.02

W_{SSD} = 31

T_{SSD} = 3500

$T_{razaoSSD}$ = 0.9

Nota-se que os valores podem variar com os primeiros resultados, pois o casamento da configuração anterior não teve bons resultados. Abaixo observa-se a imagem composta pelas 2 imagens com os pontos característicos marcados, e ligados por uma reta verde os pontos classificados como correspondentes pelo algoritmo.



Nota-se que o resultado ficou pior do que o inicialmente esperado. Houveram muitos falsos negativos, porém ao restringir mais os limiares, pouquíssimos pontos eram selecionados. Logo foi necessário a utilização de uma janela menor, para mais pontos serem selecionados. Fazer a calibragem de parâmetros à mão se provou muito ineficaz, e isso ficou muito perceptível nos resultados encontrados. Os demais resultados podem ser encontrados no repositório do git na pasta de “resultados/casamento”. Para cada arquivo retornado, foi utilizado a seguinte construção:

nome do arquivo = 'imagem_concatenada_goi_Wh_TR_Wssd_Tssd_Trazao.png'

2.2.1 Imagens building

Para o casamento das imagens building foram utilizados os seguintes parâmetros para o melhor resultado:

W_H = 25

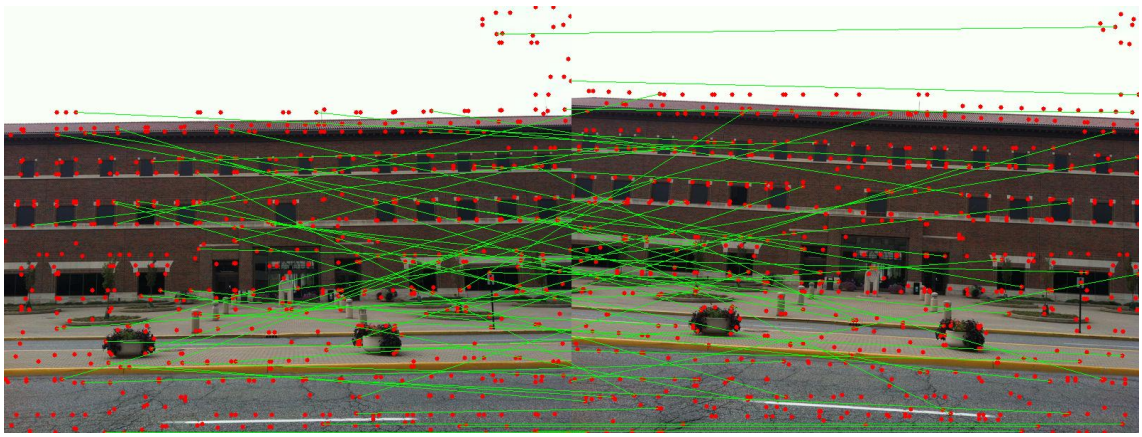
T_R = 0.01

W_{SSD} = 31

T_{SSD} = 3500

$T_{razaoSSD}$ = 0.9

Nota-se que os valores podem variar com os primeiros resultados, pois o casamento da configuração anterior não teve bons resultados. Abaixo observa-se a imagem composta pelas 2 imagens com os pontos característicos marcados, e ligados por uma reta verde os pontos classificados como correspondentes pelo algoritmo.



Assim como os resultados relacionados a primeira imagem, o casamento não foi tão bom quanto o esperado. Uma das prováveis razões é o fato de ter muitas janelas similares, e isso faz com que a razão entre os menores SSD's fique alta, gerando muitos correspondentes possíveis para 1 só ponto. A quantidade de pontos que não são quinas em alta quantidade também pode ter afetado muito os resultados, gerando ainda mais falsas associações. Os demais resultados podem ser encontrados no repositório do git na pasta de "resultados/casamento". Para cada arquivo retornado, foi utilizado a seguinte construção:

nome do arquivo = 'imagem_concatenada_building_Wh_TR_Wssd_Tssd_Trazao.png'