Visão computacional

# Lista 2 - Detector de Harris e casamento de características

Isabelle Rodrigues Vaz de Melo

13 de agosto de 2023

## 1 Detector de Harris

Neste exercício construimos a implementação de um detector de Harris, um algoritmo usado no processamento de imagens e visão computacional para identificar pontos de interesse em uma imagem. Ele calcula a variação da intensidade dos pixels em várias direções e determina se uma região é plana, de borda ou de canto, com base na mudança de intensidade ao redor dela.

Para tal, utilizamos o canal preto e branco de uma imagem, pois apenas a luminância importa e a variação dos pixels vizinhos.

Tendo a imagem P&B, calculamos as derivadas em $x$ e $y$ para cada pixel na imagem utilizando o filtro de Sobel de dimensão $W_d$. O intuito é construir a matriz de Harris $H$ e a partir daí achar sua resposta $R$. A matriz de Harris é dada por:

$$H = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \tag{1}$$

Em que $I_x$ e $I_y$ são as derivadas definidas em $x$ e $y$, e o somatório é sobre uma janela deslizante de tamanho $W \times W$ ($W$ ímpar) com o pixel referente no centro da janela. Este termo ficou livre para a escolha do usuário. Apesar de na lista ser indicado que $W = 5W_d$, fui variando os valores para verificar qual a melhor configuração que faça uma varredura mais limpa de pontos.

Encontrados os termos da matriz, podemos definir a resposta de Harris $R$ como :

$$R = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \tag{2}$$

Para $k = 0.04$. Uma maneira mais eficiente de calcular $R$ é reescrevendo a equação (2) como:

$$R = det(H) - kTr(H)^2 \tag{3}$$

Pois assim não precisamos fazer o cálculo de autovetores e autovalores.

Para cada pixel na imagem, teremos um valor de $R$. Isso ocorre pois usaremos estes valores - em comparação com um limiar - para definir quais são os keypoints que melhor detectam os cantos. Antes de definir este limitar, normalizamos os valores $R$ para que fiquem entre 0-255, valores possíveis para os pixels na imagem.

Feito isso, definimos que keypoints serão definidos para a condição em que os valores de $R$ ultrapassem um limiar $T_R$ relativo escolhido pelo usuário (ou seja, os pixels em que $R$ possuem valor maior que o limiar multiplicado pelo valor de máxima resposta). A variação de $T_R$ nos permite capturar mais ou menos pontos que o detector considera como cantos.

Por fim, para as imagens dos prédios, os parâmetros escolhidos foram $W_d = 5$, $W = 5$, e um limiar $T_R = 0.035$. A figura 1 mostra a imagem resultante.

Podemos ver que para esses parâmetros, os cantos são, no geral, bem definidos. Apesar disso, ainda conseguimos ver certo "ruído"como nos arbustos e na faixa de pedestres. Aumentar mais o limiar de fato filtrava estes pontos indesejados, mas também fazia sumir alguns cantos das quinas e alguns das janelas, como você poderá ver na figura 2.

Já para as figuras do castelo, utilizei $W_d = 3$ e $T_R = 0.05$. Neste caso, podemos perceber que como há transições mais suaves entre pixels na imagem, o detector não performa tão bem

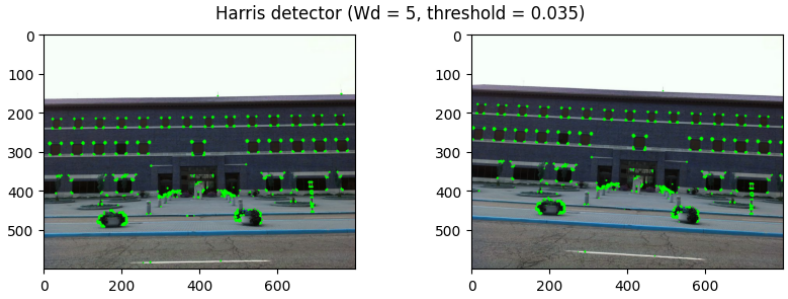


Figura 1: Detector de Harris para figuras dos prédios.

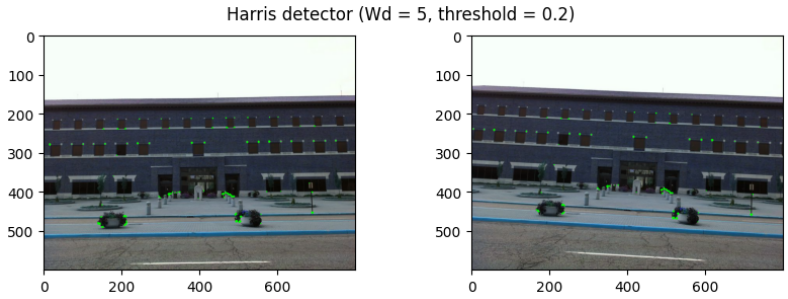


Figura 2: Detector de Harris para figuras dos prédios.

assim, capturando muitos pontos que não são cantos. A figura 3 mostra a primeira configuração. Verifique na figura 4 que se eu reduzir o limiar, a imagem fica muito poluída com pontos que foram

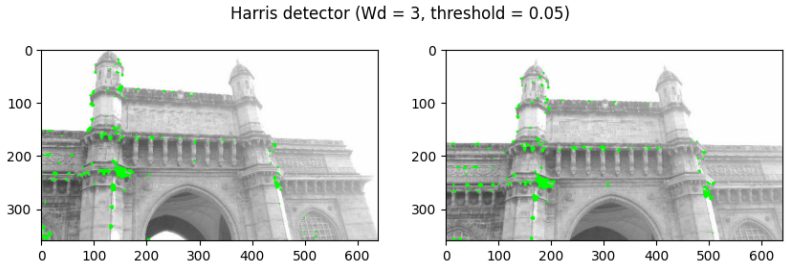


Figura 3: Detector de Harris para figuras dos castelos.

classificados incorretamente, e na figura 5, aumentando $T_R$, a imagem fica com pontos quase que indefinidos.

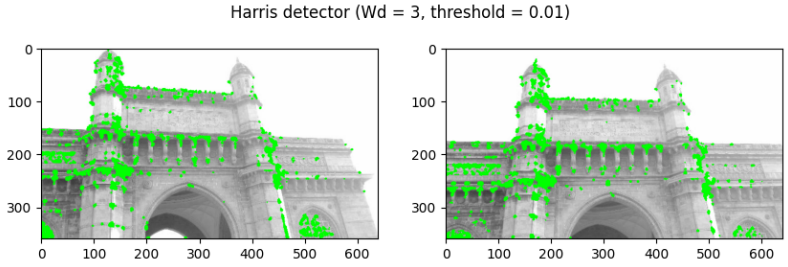


Figura 4: Detector de Harris para figuras dos castelos.

Desta forma, podemos concluir que o detector de Harris implementado esta funcionando de maneira correta, e este não é muito robusto para imagens com transições mais suaves entre pixels - sendo necessário o uso de algoritmos mais avançados para detecção de keypoints.

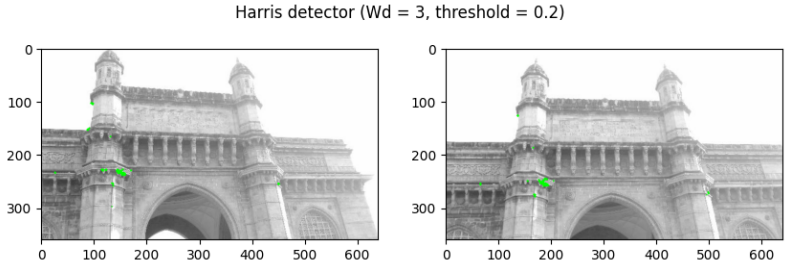


Figura 5: Detector de Harris para figuras dos castelos.

## 2 Casamento de características

Para este exercício buscamos utilizar os keypoints do exercício 1 em um procedimento de casamento de características.

O casamento de características é uma maneira de ligar pontos chaves em imagens semelhantes - por exemplo, com certa diferença de perspectiva - a fim de tornar possível procedimentos como panoramização, detecção, busca em banco de dados, etc.

O teste aqui é extensivo: comparamos um keypoint de uma imagem com todos os outros da outra figura e utilizamos uma métrica para definir qual deles têm o melhor match. A métrica utilizada é a SSD (sum of squared differences), dada por:

$$SSD = \sum_{(x,y) \in WSSD} |f_1(x,y) - f_2(x,y)|^2 \tag{4}$$

onde WSSD é o tamanho da janela ao redor do ponto característico e $f_1$ e $f_2$ são os valores dos pixels nas coordenadas $x$ e $y$ para as janelas extraídas das imagens 1 e 2, respectivamente. Para isto, defini um tamanho de janela WSSD ímpar para que o keypoint sempre fosse o ponto central da janela e pudéssemos comparar de maneira efetiva os valores de SSD.

É feita uma iteração para todos os pares de pontos e calculada a matriz com todos os valores de SSD possíveis, nos levando à etapa de seleção de pontos que casam, afinal 1 dentre esses vários valores calculados, para cada keypoint na figura 1, será o "melhor"valor.

Defini um limiar $TSSD$, ou seja, pontos mínimos da SSD são característicos se possuem um valor abaixo deste limiar. Além disso, se busca a eliminação de falsas correspondências utilizando:

$$\frac{minimoSSD}{segundominimoSSD} < TrSSD \tag{5}$$

Comparando o primeiro e o segundo mínimos com este segundo limiar também definido pelo usuário.

O procedimento experimental é variar os parâmetros para os dois pares de imagens que vieram na lista, e tentar encontrar uma configuração em que sejam vistas mais boas correspondências. A figura 6 e 7 mostram os resultados com seus respectivos parâmetros tunados.

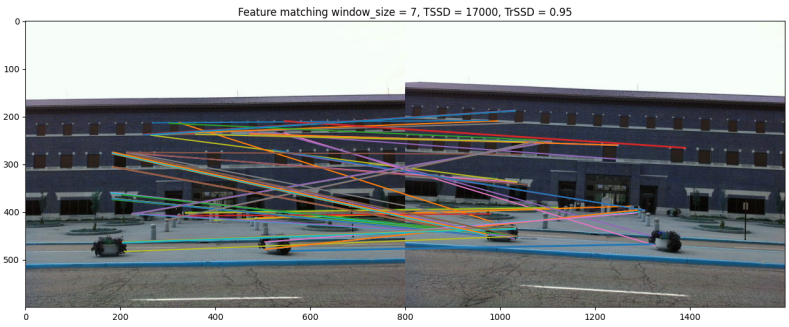


Figura 6: Casamento de características para figuras dos prédios.

Por mais que a visualização esteja confusa, podemos claramente ver que há muitas correspondências erradas, mas ainda sim algumas que conferem ou são próximas.

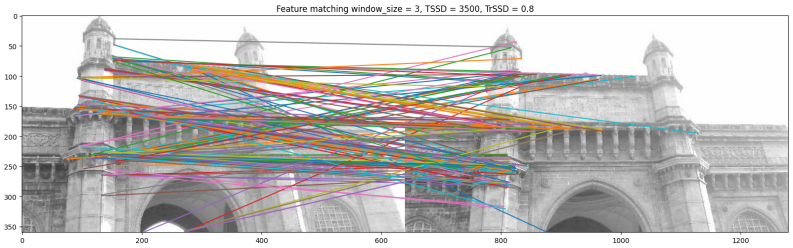


Figura 7: Casamento de características para figuras dos castelos.

Era de se esperar que os resultados não ficassem tão bons pois estamos usando esta única métrica para avaliar se dois pontos casam ou não, e isso pode levar à muitas correspondências erradas, como ocorre experimentalmente. Algoritmos mais complexos como SIFT podem levar a resultados que façam melhores casamentos.

Por exemplo, nos prédios, temos alguns elementos que podem levar o algoritmo ao erro. Veja as janelas, que apesar de terem cantos detectados, possuem todas as mesmas bolinhas brancas ao redor, podendo levar a uma possível confusão na hora do cálculo do SSD. Consigo ver correspondências corretas no arbusto (em azul claro) e na janela (azul escuro).

Já para o castelo, a figura fica mais confusa ainda pois como vimos na questão anterior, ficam muitos pontos acumulados em localidades que não são exatamente bordas. Ainda conseguimos enxergar alguns matches nas pilastras esquerdas.

Por fim, o algoritmo é uma demonstração interessante de como podemos tentar encontrar estas correlações, mas ainda não se situa como um bom algoritmo global para ser aplicado em geral no casamento de características.

# 3 Informações adicionais

O código fonte para o algoritmo de Harris é o **HarrisDetector.py**, e sua implementação - para melhor visualização por meio de jupyter notebook - encontra-se em **testharris.ipynb**. Já para o casamento de características, o código encontra-se em **testFM.ipynb**