UNIVERSIDADE D
COIMBRA

# Feature Detection and Matching

Relatório do segundo
trabalho da disciplina de
Visão Por Computador

João Vítor Sgotti Veiga (2017170653)
Ulisses Barreira Reverendo (2017243116)

2020/2021

# 1- Introdução

O trabalho proposto tem como objetivo desenvolver algoritmos de detecção e correspondência de regiões de imagens que fosse robusto a variações (iluminação, rotação, translação e escala).

A estratégia adotada foi inicialmente detectar os cantos. Aos cantos detectados, aplicar a supressão não máxima para reduzir a quantidade de pontos a serem calculados, em seguida selecionar os melhores pontos para se tornarem os “pontos-chave”, com características que melhor definiam um patch em determinada escala. Esses pontos eram utilizados para selecionar regiões da imagem para serem utilizadas para criar um Descritor de cada patch.

Por fim, o processo foi repetido às imagens disponíveis para comparar, e fazíamos *match* entre os Descritores de cada uma para avaliar o quão semelhantes são as regiões de cada patch. Cada etapa deste processo será detalhadamente descrita no decorrer do relatório.

# 2- Desenvolvimento

## 2.1- Detecção dos cantos de Harris

Este algoritmo consiste na aplicação de filtros na imagem que eliminam as componentes de altas frequências e selecionam as regiões com grandes variações de tons de cinza. Esses valores são utilizados para composição de uma matriz de derivadas, e dessa matriz serão obtidos os valores próprios para cada região. Com base nesses valores, pode-se determinar se a região em questão se trata de um corner, e, para tal, os valores próprios devem ser de alta magnitude e possuir valores próximos um do outro.

$$M = \sum_{x,y} \omega(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \qquad R = \det(M) - k(trace(M))^2$$


*Figura 1 - Corners detectados pelo algoritmo de Harris*

## 2.2– Supressão não máxima

Esta função reduz drasticamente o número de pontos a serem processados e evitar que um mesmo corner na imagem fosse representado por mais do que um ponto. Para tal efeito,

foi utilizado a função NMS, que devolve apenas o ponto com maior significância dada a dimensão desejada, caso exista uma aglomeração de pontos em uma determinada região, foi utilizada uma região de análise de 21x21 pixels.

## 2.3 – Detecção dos pontos-chave

Este estágio é utilizado para detectar qual a escala associada a cada canto detectado, e consequentemente determinar qual região da imagem está associada a determinada característica. Este objetivo foi alcançado aplicando um filtro de Laplaciano da Gaussiana em várias escalas distintas, e então selecionando as escalas em que se obtinham valores máximos.

Em nossa implementação, consideramos escalas que eram determinadas pela formula, $scal = 2{,}1 \cdot 1.3^{j}$ , com j assumindo os valores de 2 a 7.

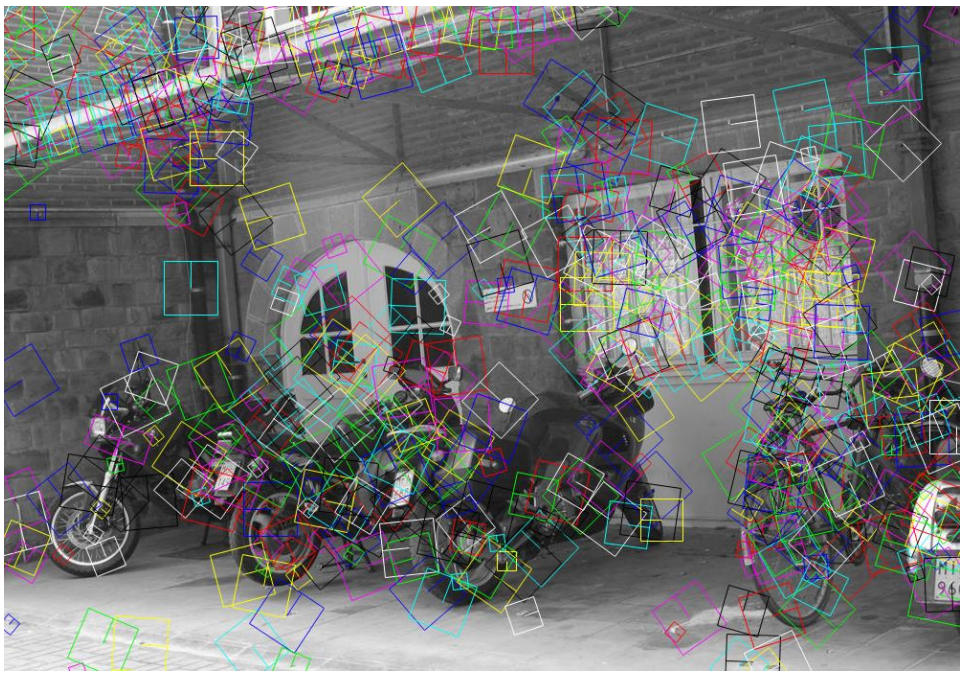

*Figura 2: Resultado do algoritmo Keypoints Detection*

## 2.4- Descritor

Esta função cria um tipo de “impressão digital” para cada imagem, isto é, uma estrutura que armazena os patches e as características associadas a cada uma delas.

Neste trabalho foram implementados **dois** descritores, um mais simples que apenas cria patches com uma quantidade fixa de pontos da imagem, e o MOPS que também coleta o “gray level” dos patches que contém os keypoints, levando em consideração os fatores de escala e rotação.

Já o MOPS funciona da seguinte forma: é selecionada a região associada à respectiva escala, então é feita uma rotação do patch, levando em conta o ângulo que recebido no input, e depois a informação é condensada em uma matriz de 8x8, e finalmente é feita a

normalização dos dados para que eles assumam média 0 e variância unitária. Trazendo robustez à variação da iluminação das imagens.

### 2.5 – Correspondência

Esse processo tem como objetivo indicar quais patches de um descritor estão relacionados com os patches de outro descritor.

Para tal, foram implementadas duas métricas de comparação: a de que calcula a distância entre dois pontos nos patches e seleciona os que possuem menor distância, e a de RATIO que também faz a SSD, mas o seu resultado é o rácio entre o melhor valor do SSD e o segundo melhor.

A implementação do segundo método, em detrimento do primeiro, melhorou muito o número de pontos certos que o nosso algoritmo detectava.

## 3 – Conclusões

**Não foi possível realizar a última parte do trabalho, portanto não podemos expressar numericamente a eficiência do nosso algoritmo,** por isto para ajudar-nos a percebermos melhor o funcionamento do nosso programa, desenvolvemos funções que apresentam de forma visual a correspondência dos *keypoints* e dos *matches*: *markKeyPts.m* e *markMatch.m*.

No geral, as imagens apresentaram bons resultados, principalmente quando as situações de luminosidade eram semelhantes e a rotação relativa entre elas era paralela ao plano da imagem. A imagem que apresentou melhor resultado foi a das motocicletas, embora o algoritmo tenha se mostrado ineficiente se o desfoque fosse muito elevado.

A imagem que apresentou pior resultado foi do grafite, que precisava de valores de threshold muito altos para apresentar qualquer ponto.

A imagem de Leuven, o algoritmo funcionou bem, mas quando havia muita variação no componente de luminosidade, começavam a surgir muitos erros no match. Quanto mais escura a imagem, menos pontos eram detectados.

A imagem do muro apresentou bons resultados, apesar de as imagens serem de tamanhos diferentes, e terem muitos padrões semelhantes. Entretanto, o algoritmo começou a falhar quando houve rotações em planos não paralelos ao plano da imagem original.