

Departamento de Engenharia Eletrotécnica e de Computadores

Visão por Computador

2017/2018

Trabalho Prático Nº3

Deteção e Cantos e Deteção de retas e circunferências utilizando transformada de Hough

Eduardo Filipe da Fonseca Rodrigues

2007105369

**Parte 1: *Deteção de cantos***

No trabalho prático nº2 foi calculado o gradiente de tons de cinza numa imagem de modo a detetar arestas, sendo estas indicadas por uma variação rápida dos tons de cinza, ou seja, uma derivada acentuada.

Utilizando o mesmo conceito, podemos também detetar cantos. Sabendo que uma aresta é caracterizada por um gradiente acentuado numa certa orientação (perpendicular à orientação da aresta), um canto pode ser caracterizado por uma área de uma imagem onde o gradiente tem duas orientações distintas.

Para calcular então os cantos numa imagem, teremos então de calcular o gradiente desta. Para tal foi utilizado um filtro de *Sobel* utilizando a função *imgradientxy(Imagem, ‘sobel’).*

Tendo o gradiente calculado, foi definida uma vizinhança para cada pixel de NxN pixéis, sendo ‘N’ um valor ímpar. Utilizando os valores de gradiente da vizinhança é calculada a matriz definida por

onde e são os valores do gradiente calculado nos pontos pertencentes à vizinhança NxN definida anteriormente.

Sendo a matriz simétrica, pode então calcular-se os valores próprios, utilizando a função *eig(C)*, obtendo uma matriz diagonal

Utilizando os valores próprios e pode-se então estudar o comportamento do gradiente da imagem na vizinhança, sendo assim possível detetar cantos.

Para tal, é escolhido o menor de ambos os valores próprios e estudado o seu valor. Para melhor esclarecimento, é definido como . Se , então estamos perante uma zona onde não há variação de tons de cinza () ou uma zona onde está presente uma aresta (). Se , estamos então numa zona onde o gradiente tem dois sentidos, ou seja, um possível canto.

Como a imagem utilizada “chess\_2.png” não é sintética, aproximadamente todos os pixéis foram detetados como sendo cantos, sendo então necessário a utilização de um valor limiar para reduzir a deteção a cantos reais e não produzidos por ruído na imagem.

**2)**

Para definir um bom valor para o limiar de deteção, foi originalmente utilizado o valor 0, no entanto, no histograma gerado apenas se conseguia observar o valor máximo de aproximadamente 500000. Utilizando o valor médio dos valores medidos (aprox. 1000), foi então selecionado o valor 2000 como limiar, sendo obtido o seguinte histograma

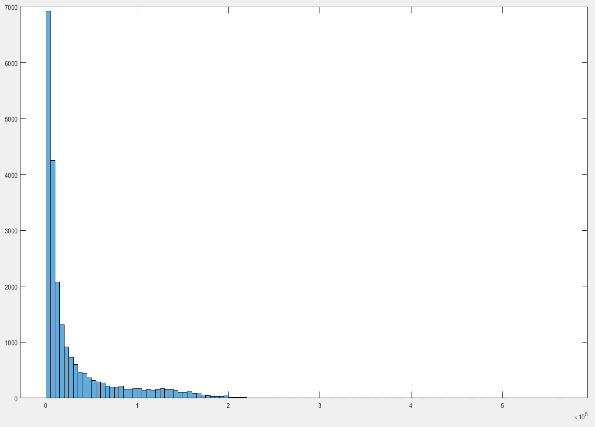
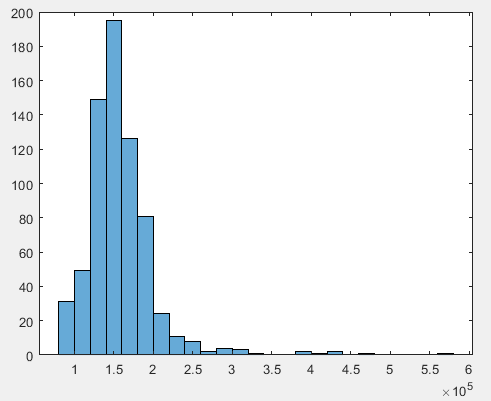


Figura 1 – Histograma com valor limiar 100000

Figura 2 – Histograma com valor limiar 2000

Após observação do histograma, foi então decidido utilizar como limiar o valor 100000. Sendo assim possível detetar os cantos com maior precisão.

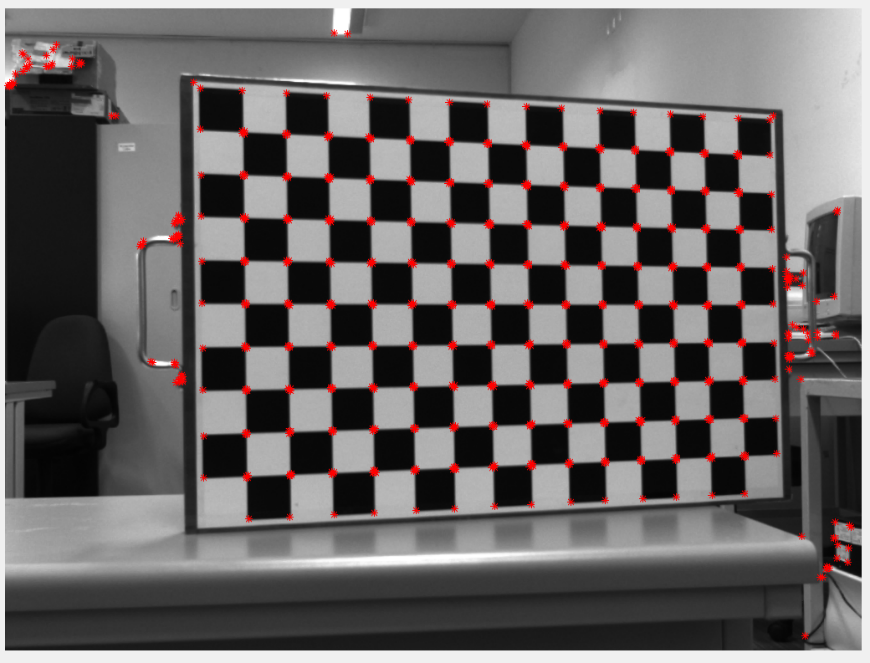


Figura 3 – Cantos detetados

**3)**

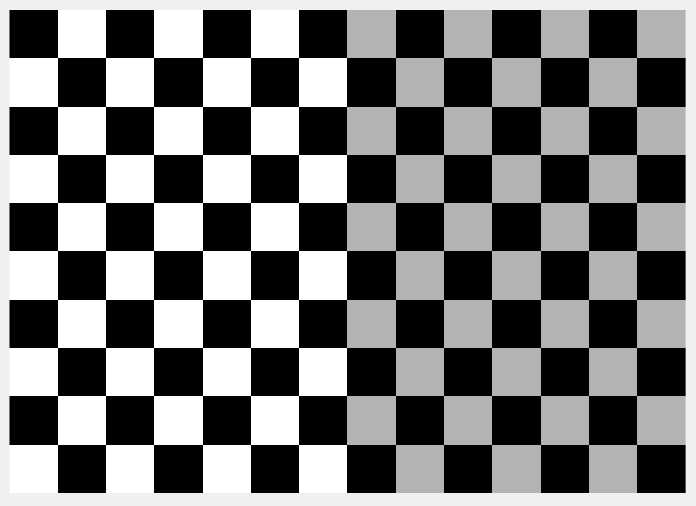
 Para estudar o efeito do tamanho da janela utilizada pelo algoritmo, foi gerada uma imagem com um padrão xadrez com quadrados de 10x10 pixéis

Figura 4 - imagem sintética utilizada

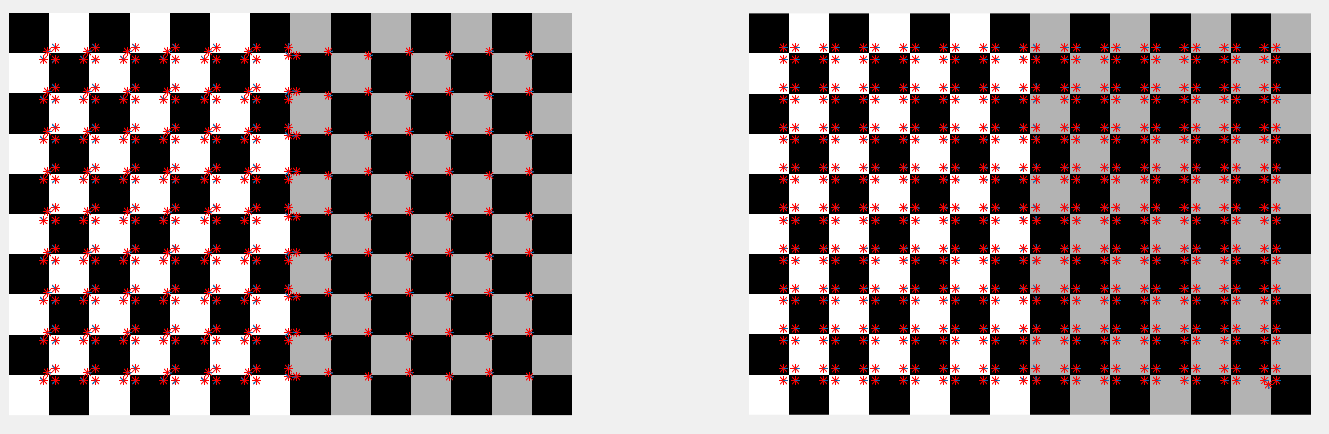
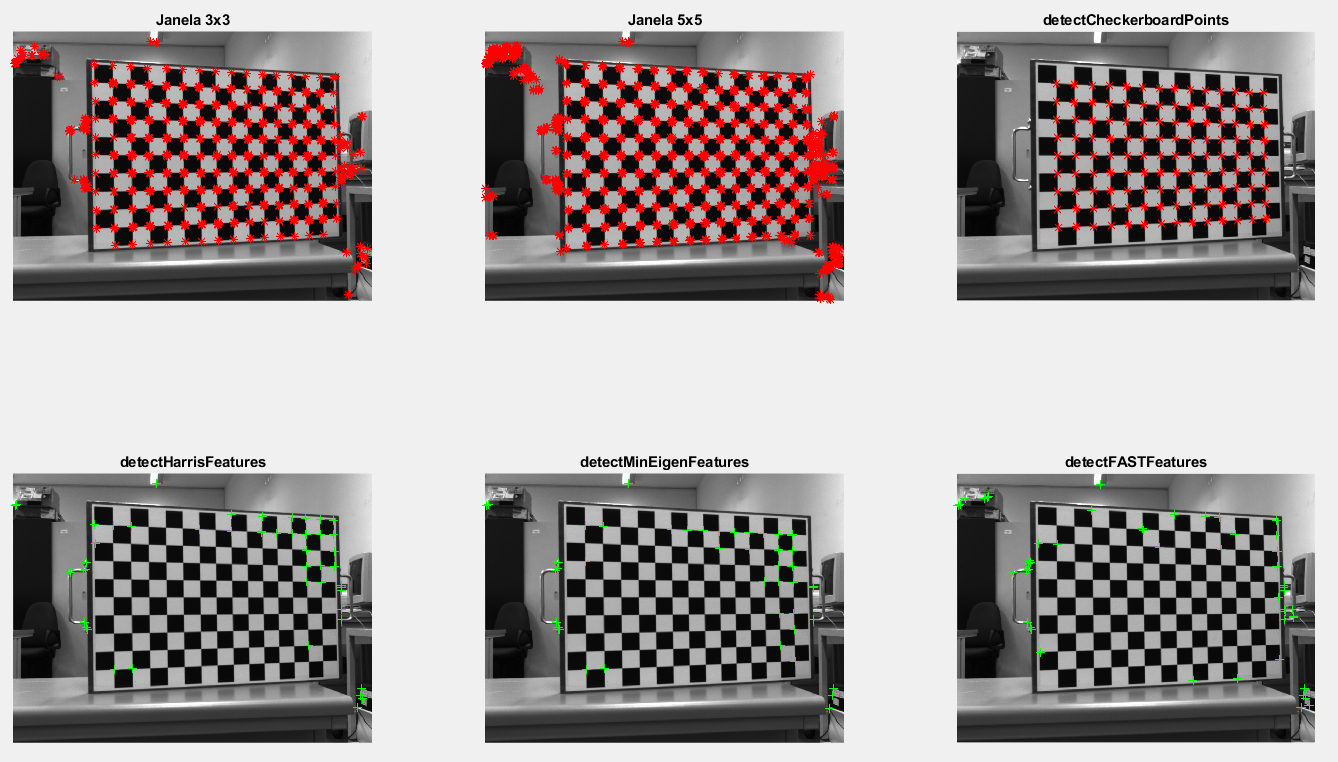
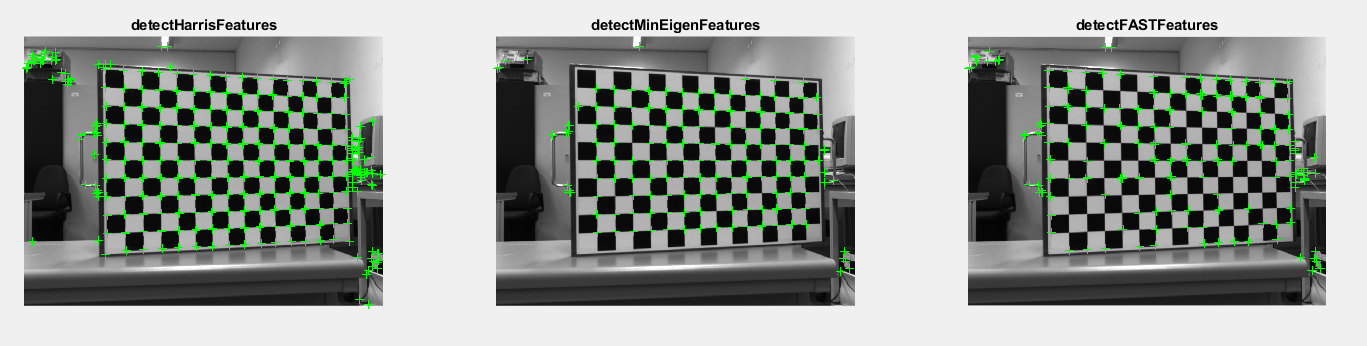
 Foi então aplicado a esta imagem o algoritmo com o valor limiar 20, sendo uma imagem sintética, não possui ruído que possa provocar erros na deteção dos cantos, e uma janela 3x3 e de novo com uma janela 5x5.

Figura 5 - Comparação de algoritmo de deteção de cantos com janelas 3x3 (esquerda) e 5x5 (direita)

Como se pode comprovar, o algoritmo deteta todos os cantos. No entanto, com uma janela 3x3, o algoritmo tem maior dificuldade a detetar cantos quando o fundo é cinza. Tal pode ser explicado com o funcionamento do algoritmo. Como é utilizado o gradiente da imagem para determinar a posição dos cantos, quando os valores de cinza dos pixéis são aproximados, o gradiente terá um menor módulo, o que por sua vez irá reduzir os valores calculado na matriz , reduzindo os seus valores próprios, podendo inclusivamente descer abaixo do valor definido como limiar e deixando de ser detetados como canto.

Este efeito deixa de se verificar aumentando o tamanho da janela para 5x5.

**4)**

Utilizando de novo a imagem “chess\_2.png”, foram testadas as funções *detectCheckerboardPoints, detectHarrisFeatures, detectMinEigenFeatures, detectFASTFeatures* e o algoritmo escrito com janelas 3x3 e 5x5.

**Parte 2: *Deteção de retas e circunferências***

Uma forma geométrica simples pode ser definida através da sua equação matemática. No caso de uma reta, todos os pontos pertencentes obedecem à equação , sendo o valor do declive da reta o valor da abcissa quando e e variáveis. No entanto, para verificar a existência de uma reta, teria de ser verificado a existência de todos os pontos numa vizinhança para cada valor de e , para cada possível reta. Outra possibilidade é então reescrever a equação da reta segundo a forma , sendo desta forma possível representar uma reta através da sua distância à origem () e do ângulo que esta faz com o eixo ().

Utilizando então uma representação RhoxTheta, uma reta é representada como o ponto .

Para detetar retas utilizando a transformada de Hough numa imagem previamente tratada com um filtro *Canny*, foi então aplicado o seguinte algoritmo:

1. Criar uma matriz de acumulação com dimensões x
2. Para cada pixel detetado na imagem:
   1. Definir um valor
   2. Calcular equivalente
   3. Incrementar na posição
3. Calcular os máximos locais de
4. Calcular valores de e correspondentes aos valores e dos máximos locais através da equações e
5. Devolver valores calculados e matriz de acumulação

Deste modo, para cada ponto pertencente a uma aresta, teremos calculado todas as possíveis retas que poderiam passar nesse ponto e guardado os seus valores na matriz .

Ao aplicar este processo a um segundo pixel pertencente a uma reta, um dos dupletos calculado, será idêntico ao calculado noutro ponto, tornando o valor nessa coordenada de superior aos restantes.

Para testar o algoritmo foi utilizada a imagem “lines\_circles\_2.jpg”, onde foi aplicado um filtro *Canny* como método de binarização da imagem.

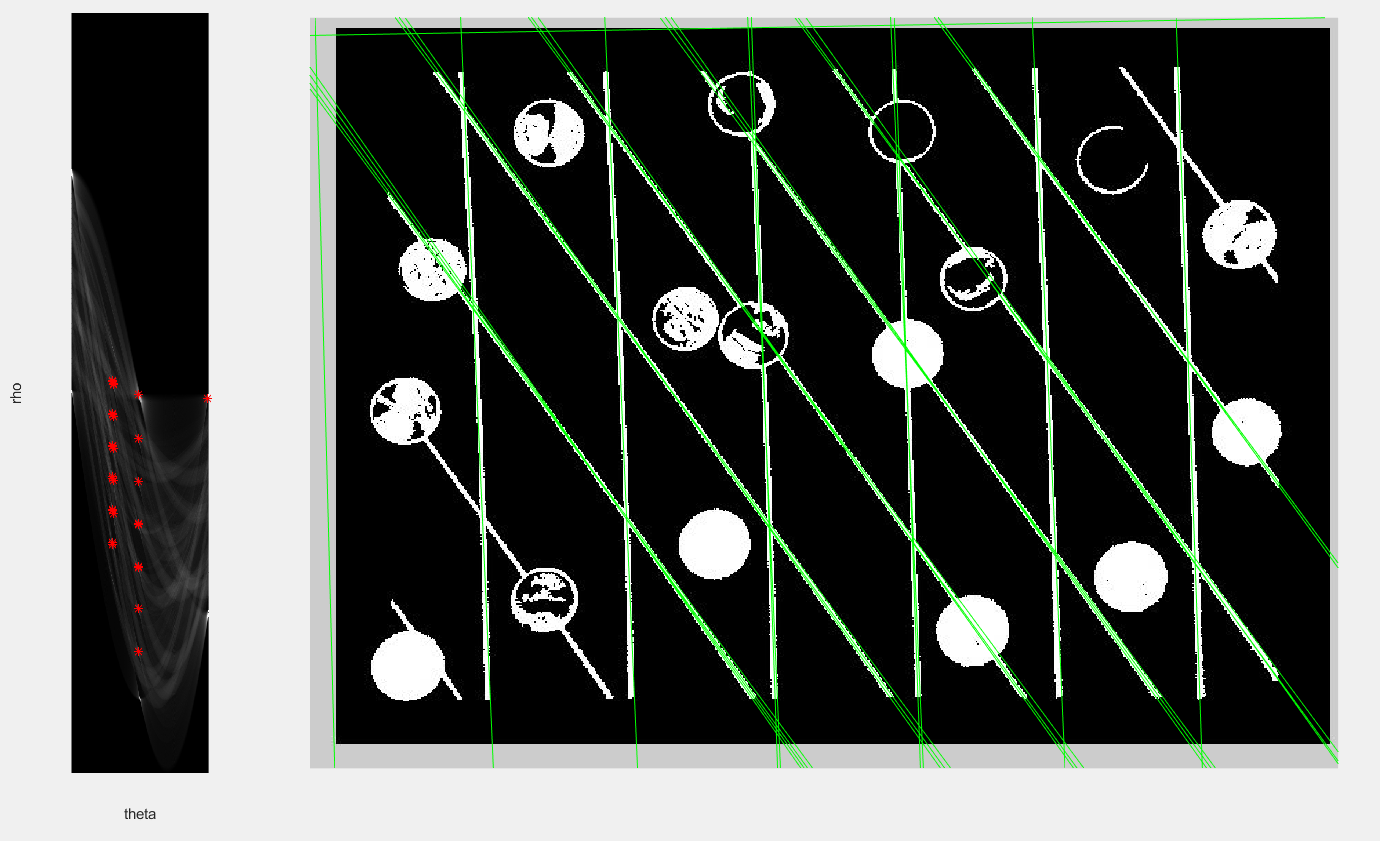
 Após ser aplicado o algoritmo, foram encontradas as seguintes retas

Figura 6 - Matriz de acumulação e retas detetadas

Do mesmo modo, pode então procurar por circunferências na imagem.

Utilizando como equação de parametrização

poderemos então definir o seguinte algoritmo para deteção de circunferências:

1. Ordenar as coordenadas de todos os pontos detetados pelo filtro *Canny* um vetor coluna
2. Criar uma matriz “x” tridimensional repetindo esse vector, obtendo uma matriz de dimensões
3. Criar uma nova matriz “x\_radius” com o mesmo tamanho da anterior, calculando para cada elemento as suas coordenadas através das equações de parametrização definidas anteriormente
4. Subtrair à matriz “x” a matriz “x\_radius”, obtendo assim uma matriz “possibleCentersA” de possíveis centros. Onde em cada elemento da matriz está representada a posição do centro de uma circunferência que teria representado naquela posição da imagem um ponto a ela pertencente
5. Criar uma matriz de acumulação de dimenções onde e serão as dimensões da imagem e a gama de raios para a qual estamos a procurar circunferências
6. Retirar da matriz “possibleCentersA” os valores exteriores à imagem. (Serão apenas consideradas as circunferências cujo centro possa ser representado na imagem)
7. Para cada elemento de “possibleCentersA”, incrementar o valor da matriz de acumulação nas coordenadas , onde e são as coordenadas no centro da circunferência e o seu raio
8. Calcular os máximos locais de
9. Devolver as coordenadas dos máximos locais

Deste modo, de forma análoga ou algoritmo para deteção de retas, pontos pertencentes a uma circunferência, terão as coordenadas do seu centro e o raio em comum entre si, provocando o valor superior nessa coordenada da matriz .

Utilizando o algoritmo na mesma imagem utilizada para a deteção de retas, foi obtida a seguinte matriz de acumulação e as seguintes circunferências

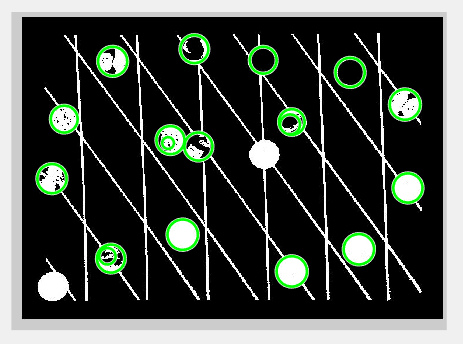
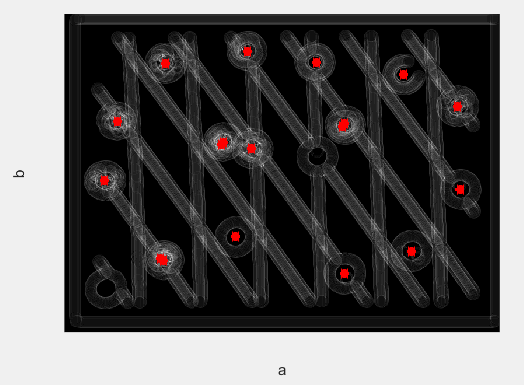


Figura 7 - matriz de acumulação

Figura 8 - circunferências detetadas