Visão por Computador
04 - Edge Detection

Diogo Corte, Diogo Silva
DETI, Universidade de Aveiro
Aveiro, Portugal
{diogo.corte, dbtds}@ua.pt

Resumo –
Este relatório foi realizado para a unidade curricular de Visão por Computadores com o objetivo de demonstrar a perceção da aula (VC Exercises 04). Sendo estes, Sobel, Canny, Laplacian, Harris, Hough Transforms. Todos os temas referidos foram testados utilizado a biblioteca OpenCV.

I. OPERATORS

A biblioteca OpenCV fornece 3 tipos de filtros de gradiente ou filtros passa alto, sendo eles Sobel, Scharr e Laplacian.

A. Sobel

Sobel é um operador que consiste em calcular aproximações de derivadas, tendo em conta que para calcular uma aresta idealmente seria derivar. Sobel aplica operações matriciais de forma a obter uma aproximação das mesmas.
Sendo que faz operações sobre o eixo do X e o eixo do Y, de forma a detetar as arestas verticais e as horizontais.

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Se fizermos um varrimento pelos pixels de uma imagem com a matrix acima referida, vão ser detetadas arestas verticais, sendo que para detetar arestas horizontais basta fazer o mesmo estilo de matrix mas com a linha do meio nula, idêntico a matrix de cima.

$$Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

O problema surge quando se pretende calcular arestas que não são verticais, nem horizontais, mas sim um conjunto de ambas. O que sobel faz é combinar o resultado de ambas, fazendo:

$$G = \sqrt{Gx^2 + Gy^2}$$

Obtendo assim uma detecção de arestas bastante eficaz.

Na imagem 1 é possível verificar as arestas da imagem perfeitamente definidas usando um kernel de tamanho 3, que é exactamente o kernel acima referido.

Figura 1: Transformação de imagem obtida por camara com operador de Sobel com kernel 3 e escala 2

B. Scharr

O operador de Scharr é apenas uma variação do Sobel em que o kernel varia, sendo que o kernel em vez de conter as linhas habituais em que o elemento central é o dobro do que o dos cantos, no de Scharr o elemento central contém um pouco mais que o tripo dos cantos, sendo a matrix constituida sempre por [3 10 3] em vez de [1 2 1] como se teria no Sobel.

Figura 2: Transformação de imagem obtida por camara com operador de Scharr com escala 1

O resultado é bastante similar ao anterior, não passando de uma optimização feita ao operador Sobel.

C. Laplacian

O operador laplaciano é uma aproximação a segunda derivada, sendo que tal como os anteriores também serve para fazer deteção de zonas onde a intensidade se altere, ou seja, de arestas.

Figura 3: Transformação de imagem obtida por camara com operador Laplacian com kernel 3 e escala 2

II. CANNY - EDGE DETECTION

O algoritmo de Canny não é um operador como os referidos anteriormente, mas sim um conjunto de operações. Sendo estas operações as seguintes:

1. Filtro de ruído, tendo em conta que os detetores de arestas são facilmente suscetíveis ruído, o algoritmo aplica um filtro gaussiano de kernel 5x5 de forma a minimizar o erro
2. Calculo do gradiente aplicando um operador (e.g. Sobel) e a respetiva direção
3. Após calcular o gradiente e a direção do mesmo, faz-se um varimento total a imagem novamente de forma a verificar onde estão as edges, sendo que se o ponto intermédio na direção do gradiente tiver um gradiente mais forte que os vizinhos, significa que estamos perante uma aresta.
4. Por ultimo aplica-se um threshold por baixo e por cima, filtrando gradientes abaixo de um determinado valor e acima de um determinado valor.

Note-se o seguinte resultado:

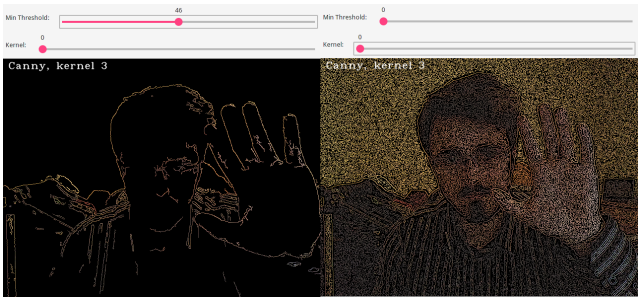

Figura 4: Canny com threshold 40 a 120 (esquerda) e com um threshold de 0 (direita)

Pode-se verificar que é possível ver as arestas nas mãos e na cara, sendo assim possível verificar se determinadas formas existem numa imagem.

III. HARRIS - CORNER DETECTION

Cantos são regiões nas imagens com grandes variações de intensidade em todas as direções o que faz deles pontos de interesse com aplicação em por exemplo deteção de movimento ou reconhecimento de objectos.

O algoritmo de Harris consiste em verificar as variações de gradiente tal como o Sobel faz, mas em vez de verificar apenas em duas direcções confirma mudanças de intensidades em todas as direções para ter a certeza que se trata de um canto. Sendo que a única variação que não existe é a zona de onde o canto origina.

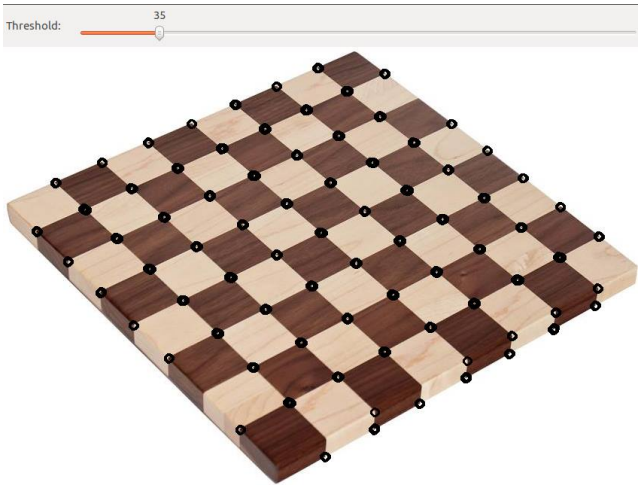

Figura 5: Pontos de interesse demarcados com circulo preto, função de Harris com tamanho de vizinhança 2, kernel de operador de Sobel 3 e parametro livre 0.008

Pode-se verificar na imagem 5 seguinte que o algoritmo de Harris consegue classificar cantos com uma taxa de sucesso bastante alta.

IV. LINES AND CIRCLES DETECTION

A. FindContours and filtering objects

O algoritmo findContours tem resultados bastante positivos ao utilizar o drawContours, no entanto isto representa todos os contornos, o que não é pretendido. No nosso caso era o objectivo extrair circulos e linhas.

No entanto o resultado obtido foi o seguinte:

A detecção de círculos e linhas no vector definido pelo findContours é uma operação bastante complexa.
O findContours retorna um vector de contornos, em que cada contorno é um vector de pontos (x e y) que definem os limites desse contorno.

O problema surge ao fazer a análise desses pontos, para tentar verificar se um contorno é uma recta tentou-se aplicar declives das rectas entre pontos. Ou seja, entre a recta com

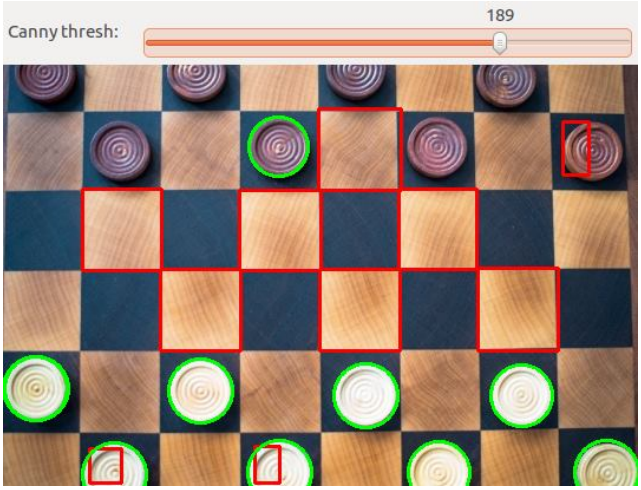

Figura 6: Damas após extrapolação de circulos e retangulos dos contornos devolvidos por findContours

os pontos i e i-1 e a recta com os pontos i e i+1, mas ao comparar estes pontos por todo o contorno verificou-se que como os pontos estão demasiado próximos, os ângulos também vão variar muito o que torna impossível de comparar os declives com pontos próximos, sendo que a próxima aproximação foi considerar a distância euclidiana entre pontos.

O que se pode concluir é que este tipo de trabalhos são bastante complexos de efectuar manualmente, sendo que convêm verificar a existência de ferramentas. Neste caso o Hough Transform faria o trabalho de encontrar linhas e círculos.

B. Hough Transform

A biblioteca OpenCV permite o uso da transformada de Hough para encontrar linhas e círculos.

O algoritmo de Hough (para deteção de linhas) consiste em traçar várias rectas em diferentes ângulos a volta de cada ponto da imagem como mostra a imagem seguinte.

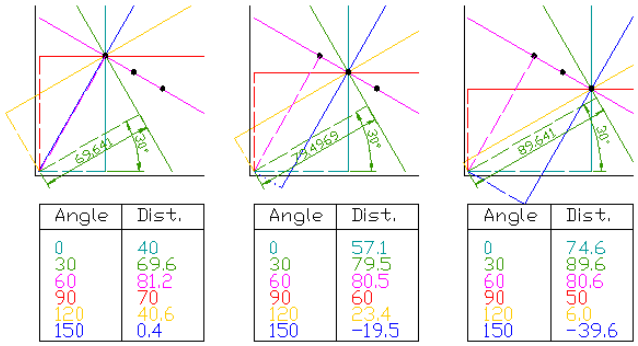

Figura 7: Rectas traçadas de diferentes ângulos (Hough)

Depois de traçar as rectas é calculada a distância a origem com uma recta perpendicular e intersectando a origem. Um gráfico de distância por ângulo é depois representado, sendo assim possível verificar qual o angulo de uma recta e a distância da mesma a origem, obtendo assim a localização da mesma.

Note-se a imagem seguinte depois de aplicar uma transformação de Hough para linhas e circulos:

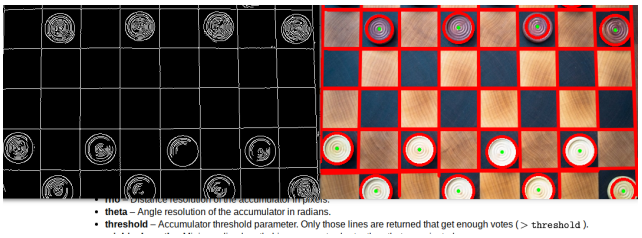

Figura 8: Damas após uma transformação de Hough para Linhas e para Circulos

Apesar de a deteção ser perfeita, não foi simples, a transformada de Hough é bastante sensível aos parametros, sendo que se os parametros estiverem errados pode detectar bastantes falsos positivos. Para além de falsos positivos também se reparou que existe uma situação bastante dificil de cobrir, que é vários circulos de diferentes tamanhos, e.g. Smith Chart. Para fazer a detenção dos circulos todos é preciso aplicar a transformada de Hough para os circulos várias vezes.

REFERÊNCIAS

[1] Robert Collins, Penn State*Harris Corner Detector - Lecture 06*