Projeto Final
Detecção Automática da Pupila

Marina P. Garcia
Introdução ao Processamento de Imagens
Engenharia Mecatrônica
Universidade de Brasília - UnB
Brasília, DF

***Abstract*—Este é o Projeto Final da disciplina *Introdução ao Processamento de Imagens*, ministrada pelo professor Bruno Macchiavello, foi baseado no artigo *Detecção Automática da Pupila usando ajuste de Elipse* [2] e consiste na implementação de um programa em Python 3, utilizando as bibliotecas *Numpy* e *OpenCV*, que realiza a deteção de pupilas a partir do ajuste de elipse.**

1. Introdução

A detecção automática de pupilas é muito utilizada atualmente, principalmente no rastreamento ocular, que tem aplicações como nas interações homem máquina, interfaces para usuários com deficiências, estudos do comportamento humano, avaliação de motoristas, pesquisas nas neurociências e outras [1].

Esse projeto tem como objetivo realizar a detecção automática de pupilas em imagens do *database* MMU-Iris [3], e foi desenvolvido com base no artigo [2], apresentado no Congresso Brasileiro de Engenharia Biomédica de 2016.

Foi desenvolvido em Python 3 utilizando as bibliotecas Numpy e OpenCV, e consiste no pré processamento de uma imagem utilizando operações morforlógicas, e então no ajuste de uma elipse na imagem resultante.

Nas próximas seções, detalharemos os Materiais (2) e Métodos (3) utilizadas para a implementação do programa e então será apresentado os resultados finais (4).

Novembro 30, 2020

2. Materiais

Os dados utilizados neste trabalho são provenientes da base de dados de imagens oculares MMU-Iris [3] que contêm 46 pastas (cada uma de um indivíduo diferente) totalizando 460 imagens, em formato bmp. Este *database* contém imagens não ideais de olhos que incorporam elementos como interferência dos cílios, oclusão dos olhos e variações abruptas de iluminação. No artigo base para este projeto foi utilizada a base de dados CASIA V2, que possui 1200, similares as disponíveis no MMU-Iris.

3. Métodos

Os métodos utilizados neste projeto foram divididos em duas etapas. A primeira consiste no pré-processamento da imagem de entrada, para que então, na segunda seja realizado o ajude de uma elipse na pupila.

Na Figura 1 é mostrado o diagrama de blocos do algoritmo de detecção da pupila implementado.

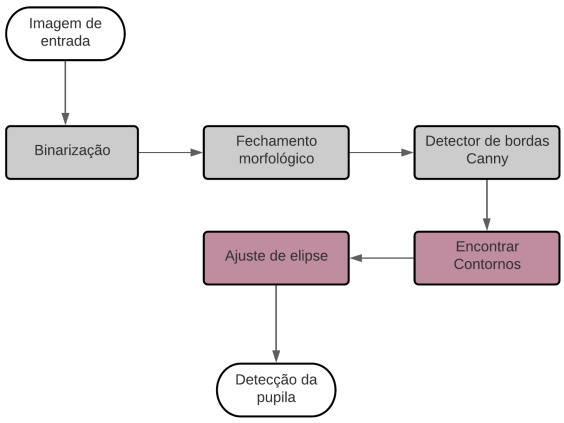

Figure 1. Diagrama do Algorítimo

No diagrama, temos em cinza os processamentos a serem realizados na primeira etapa, e em rosa, os processamentos realizados na segunda.

3.1. Pré-Processamento da imagem

3.1.1. Binarização.

A primeira operação realizada na imagem foi a binarização, utilizando a função *threshold()* da biblioteca OpenCV, os parâmetros da função foram escolhidos de forma empírica de forma que funcionasse bem para todas as imagens do *database* utilizado, de forma que na imagem resultante restasse praticamente só a pupila.

3.1.2. Fechamento.

Foi realizada então uma operação morfológica de fechamento, com um *kernel* quadrado de tamanho 2x2, também escolhido de forma empírica, para reduzir os efeitos de ruído causados pelos cílios e os outros artefatos indesejáveis presentes na imagem binária obtida no ítem anterior.

3.1.3. Bordas.

Com a pupila isolada pelos processamentos anteriores, utilizando a função *Canny()* da biblioteca OpenCV, foi gerada uma imagem com apenas as bordas dos elementos da imagem (em branco) em um fundo preto. Essa é a última imagem a ser obtida nesta etapa de pré-processamento, e será ela a utilizada na segunda etapa do programa.

No artigo, após obtermos as bordas aplicando o algorítimo Canny, ainda foi realizada a união dos segmentos de bordas, implementando uma análise de componentes conectadas em uma vizinhança de 8. Porém como a segunda etapa foi realizada de uma maneira diferente à apresentada em [2], não foi necessário realizar a análise de componentes conectadas.

3.2. Ajuste da Elipse

Nesta segunda etapa, temos como objetivo encontrar uma elipse que se encaixe no formato ta pupila presente na imagem de entrada do programa.

3.2.1. Contornos.

Na imagem com as bordas obtida na etapa anterior, foi utilizada a função *findContours()* da OpenCV, que encontra os contornos presentes na imagem e retorna o valor de seus pontos.

Com os contornos encontrados, foram utilizadas as funções *arcLength()* da OpenCV e *argmax()* da Numpy, para encontrar o maior dos contornos presentes na imagem, que seria o contorno da pupila, os outros contornos menores encontrados ocorrem por conta das interferências de pálpebra, cílios e iluminação.

3.2.2. Elipse.

Nesta última parte, foi utilizada a função *fitEllipse()* da biblioteca OpenCV para enccontrar a elipse que melhor se encaixa no contorno da pupila encontrado anteriormente. Utlilizando então as funções *ellipse()* e *circle()*, ambas da OpenCv, foi desenhada na imagem original a elipse encontrada e seu centro.

4. Resultados

4.1. Pré-Processamento da imagem

O pré processamento da imagem foi realizado em três partes, cujos resultados estão ilustrados abaixo. Usando a Imagem 2 de entrada foi realizada a binarização (3), então o fechamento morfológico (4) e, por fim, a detecção de bordas utilizando o algorítimo Canny (5).

Figure 2. Imagem original

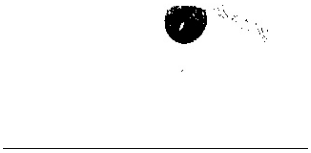

Figure 3. Imagem binária

Figure 4. Fechamento morfológico

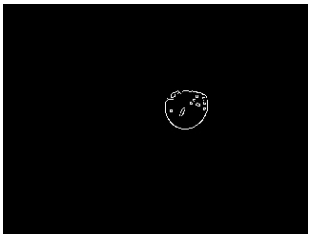

Figure 5. Detecção de bordas de Canny

No artigo utilizado de base para este projeto [2] ainda foi realizada uma análise de componentes conectadas na imagem com as bordas, porém como dito em 3.1.3, utilizado o *database* MMU-Iris não foi necessária esta etapa para se obter um resultado satisfatório.

4.2. Ajuste da Elipse

Esta etapa do processamento, o ajuste da elipse à pupila, foi realizada de forma bastante distinta da utilizada pelos autores do artigo [2], que foi baseado no trabalho do Fitzgibbon, Pilu e Fisher [4]

Já neste projeto, o primeiro passo para o ajuste da elipse foi encontrar o maior contorno presente na imagem de bordas (5). Desenhando o contorno encontrado (apenas para melhor visualização), obtemos a imagem mostrada na Figura 6

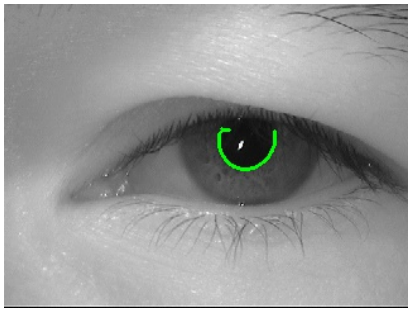

Figure 6. Imagem com a maior borda encontrada desenhada

Utilizando o contorno encontrado, usamos a função mencionada na Seção 3.2.2, que recebe o contorno como entrada e retorna a elipse que melhor se encaixa nele, desenhamos essa elipse e seu centro na imagem origina. Então podemos observar o resultado da detecção da da pupila da imagem de entrada 2 na Figura 7.

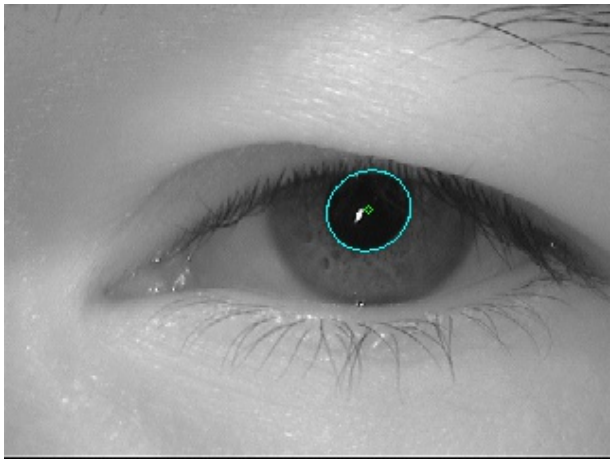

Figure 7. Imagem de entrada com a elipse sjustada à pupila

Alguns resultados dessa estapas estão ilustrados nas Figuras 8, 9, 10, 11, 12 e 12.

Essas imagens mostradas nesses resultados foram escolhidas aleatoriamente do *database*, e como podemos observar, todas possuem resultado bastante satisfatório.

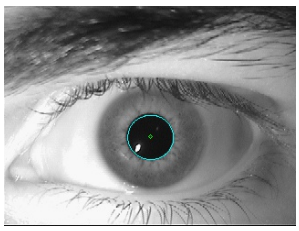

Figure 8.

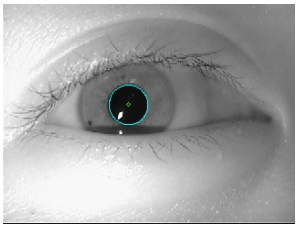

Figure 9.

.

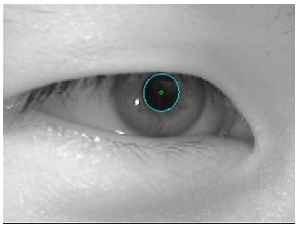

Figure 10.

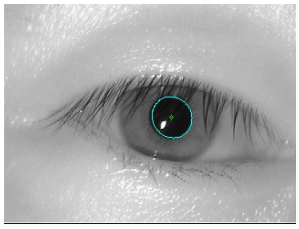

Figure 11.

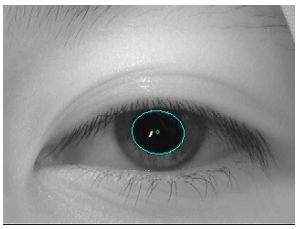

Figure 12.

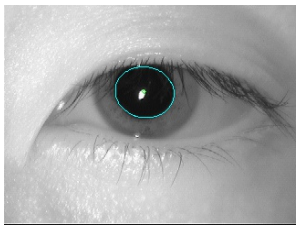

Figure 13.

Foram escolhidas então, à mão, as 10 imagens do *database* MMU-Iris que possuiam mais interferência de cílios, pálpebra e iluminação, e destas as 3 que possuiram o pior resultado são as ilustradas abaixo nas Figuras 14, 15 e 16.

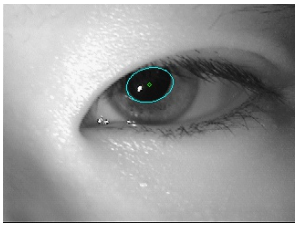

Figure 14.

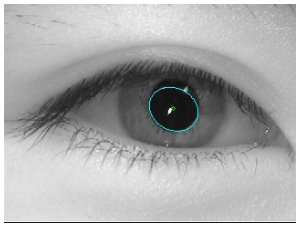

Figure 15.

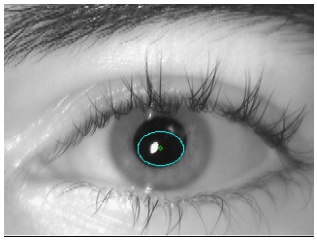

Figure 16.

Como podemos observar nessas três imagens, das 10 imagens selecionadas como as ”mais difíceis” de realizar a detecção, apenas essas obtiveram resultado insatisfatório.

5. Conclusão

Utilizando conhecimentos de processamento de imagens adquiridos ao decorrer da disciplina e utilizando o artigo [2] como base, os resultados obtidos neste projeto foram bastante satisfatórios em em seu objetivo de detectar a pupila.

Apesar de não terem sido testadas todas as imagens do database neste projeto, das selecionadas como as que possuem mais interferências, foram muito poucas que não obtiveram o resultado esperado, e em todas as imagens escolhidas aleatoriamente, foi possível realizar a detecção da pupila sem problemas.

References

[1] Rafael C. Gonzalez e Richard E. Woods, *Processamento Digital de Imagens*, 3rd ed, 2009.

[2] L. V. Romaguera, F.P. Romero, C. R. V. Seisdedos e M. G. F. Costa, *Detecção Automática da Pupila usando ajuste de Elipse*, CBEB 2016.

[3] MMU-Iris Database disponível em: (acessado em 25/11/2020) https://www.kaggle.com/naureenmohammad/mmu-iris-dataset

[4] Fitzgibbon A, Pilu M, Fisher R. *Direct Least Square Fitting of Ellipses*. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 5. DOI: 0162-8828/99; 1999.

[5] Gwon S Y et al. *Robust Eye and Pupil Detection Method for Gaze Tracking*. In: International Journal of Advanced Robotic System.Int J Adv Robotic Sy, Vol. 10, 98:2013DOI 10.5772/55520; 2013.