RECONHECIMENTO DE IMAGEM PARA O DIAGNÓSTICO PRECOCE DO RETINOBLASTOMA

Stella Fráguas (IC) e Luciano Silva (Orientador)

Apoio:PIBITI Mackenzie

RESUMO

O Retinoblastoma, ou câncer de retina, é uma doença pediátrica e com sintomas discretos em seus estágios iniciais. Apesar de ser tratado com facilidade e possuir boas perspectivas de cura, o diagnostico costuma ser retardado principalmente pela falta de informação dos pais e das crianças para identificar os sintomas, levando a consequências mais grave como cegueira ou morte. A presença da enfermidade também pode ser percebida através de reflexos esbranquiçados nos olhos das crianças em fotografias não-clinicas, cujo a ocorrência é atribuída ao desvio da retina provocado pelo tumor. Através da computação visual, esse projeto visa fazer uso dos recursos tecnológicos para a percepção deste padrão característico da doença, além da notificação de seus usuários quanto a necessidade de buscar um oncologista, se for o caso. Com objetivo de ser uma aplicação de fácil acesso, este projeto apresenta sugestões de metodologias para a implementação deste recurso de forma simples e acessível, usando a biblioteca de uso livre OpenCV e alguns dentre os diversos recursos de computação visual e aprendizado de máquina oferecidos por ela. A disponibilidade de recursos como esse é bem pequena e a abrangência deste tipo de produto no mercado pode salvar a vida de muitas crianças, trazendo conscientização do diagnóstico precoce de forma simples.

Palavras-chave: Computação visual. Diagnóstico precoce. Retinoblastoma

ABSTRACT

Retinoblastoma, or retinal cancer, is a pediatric disease with discrete symptoms in its early stages. Although usually treated with ease and have good healing prospects, the diagnosis is usually delayed mainly by the lack of information from parents and children to identify the symptoms, leading to more serious consequences such as blindness or death. The presence of the disease can also be perceived through whitish reflexes in the eyes of children in non-clinical photographs, whose occurrence is attributed to the deviation of the retina caused by the tumor. Through visual computing, this project aims to make use of technological resources for the perception of this characteristic pattern of the disease, as well as the notification of its users about the need to seek an oncologist, if appropriate. In order to be an easy-to-access application, this project presents suggestions of methodologies for the implementation of this resource in a simple and accessible way, using the OpenCV open source library and some of

the various visual computing and machine learning features offered by it. The availability of resources like this is very small and the scope of this type of product on the market can save the lives of many children, bringing awareness of the early diagnosis in a simple way.

Keywords: Computer Vision. Early diagnosis. Retinoblastoma.

1. INTRODUÇÃO

O retinoblastoma (GALINDO, 2016) é o tipo de câncer mais comum na infância e tem grandes chances de cura quando diagnosticado precocemente. Quanto antes feito o diagnóstico, maiores são as chances de cura e menores são as chances de perda da visão.

Um grande desafio para o diagnóstico precoce em países em desenvolvimento, como o Brasil, é a falta de conhecimento sobre os sinais que podem indicar a presença do tumor ocular, como por exemplo a leucocoria - reflexo branco da retina cancerígena em fotografias não-clínicas com flash, também conhecido como “olho de gato”.

Por ser uma enfermidade pediátrica, um diagnóstico precoce pode ser dificultado, devido à falta de entendimento da criança dos seus sintomas e, para os pais, o primeiro sintoma aparente costuma ser a aparição da leucocoria nas fotografias

Sabendo-se disso, este projeto aborda o desenvolvimento de um algoritmo para identificar a possibilidade de um retinoblastoma baseado em fotografias não-clinicas. Buscando os padrões característicos da leucocoria, é proposto um algoritmo de acesso livre capaz de identificar a ocorrência do sintoma e de notificar o usuário sobre sua possibilidade.

2. REFERENCIAL TEÓRICO

2.1. O Retinoblastoma

O retinoblastoma é o tipo de câncer mais comum na infância, podendo ser hereditário, esporádico ou por deleção cromossômica. Sendo raro em crianças maiores do que 6 anos, tem a média de idade do diagnóstico de 2 anos. Quando não diagnosticado e tratado corretamente, ele é potencialmente fatal, mesmo com uma taxa de cura de aproximadamente 100% quando diagnosticado precocemente e iniciado o tratamento em seguida (GALINDO, 2016).

Ele se manifesta através da leucocoria (“olho de gato”), protrusão ocular, estrabismo, cor diferente dos olhos e hiperemia conjuntival. A leucocoria é identificada através do Teste do Reflexo Vermelho, ou TRV, que se trata de um teste simples que analisa o reflexo da luz no olho da criança (SINGH, 2014).

Ao realizar o TRV, o examinador se posiciona próximo à criança e utiliza um oftalmoscópio direto para examinar ambas as pupilas, separadamente e simultaneamente, para assim comparar o reflexo entre elas. Caso o reflexo em ambos os olhos seja equivalente em cor, intensidade e clareza e não existam opacidades, pontos brancos ou desvios, pode ser considerado normal. Já o inverso, como mostrado na figura 1, pode caracterizar leucocoria, recomendando-se o exame de fundo de olho para diagnóstico definitivo.

No Brasil, assim como em outros países em desenvolvimento, o TRV não é realizado com rotina e, combinado com a falta de informação sobre a doença, fazem com que o diagnóstico do retinoblastoma seja muito mais tardio, aumentando o risco de perda de visão e morte (BERTOLDI et al., 2012).

Como o câncer é uma doença que vai desenvolvendo com o passar do tempo e quanto mais avançado, menores são as chances de cura, o diagnóstico precoce é de extrema importância.

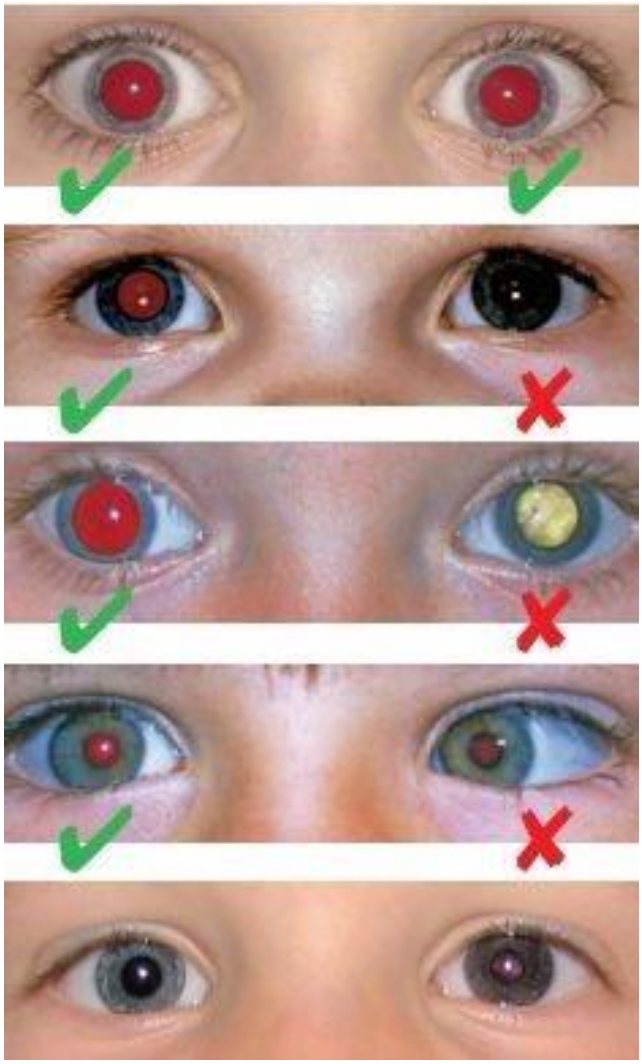

Figura 1: Reflexo do olho considerado normal em exame de TRV.

Fonte: Instituto de Oftalmologia de Assis.

No caso do retinoblastoma, o diagnóstico precoce acompanhado do tratamento pode evitar a metástase e a perda da visão, melhorando a qualidade de vida do paciente, além de reduzir o risco de morte. Em países desenvolvidos e com melhores condições socioeconômicas, o diagnóstico precoce ocorre com mais frequência, facilitando o tratamento e diminuindo a mortalidade (CELEBI, 2015).

Por não ser um exame considerado de rotina por muitos médicos pediátricos no Brasil, somado a falta de informação da população sobre a doença, o diagnóstico do retinoblastoma acaba sendo retardado, evoluindo para estágios mais avançados antes de sua detecção.

Além do TRV, o reflexo branco da pupila, provocado pelo tumor, pode ser percebido por fotografias com flash, como mostra a imagem. Ao notar esse reflexo anormal na pupila da criança, conforme exemplo mostrado na Figura 2 (FRANCIS e ABRAMSON, 2015), o

recomendável é levá-la a um médico especialista imediatamente para que seja feito o diagnóstico.

Figura 2: Criança com retinoblastoma no olho esquerdo.

Fonte: https://www.rnib.org.uk/

O problema é que, devido à falta de informação, muitos pais acabam negligenciando ou às vezes nem percebendo o reflexo branco e, quando notado, muitas vezes o tumor já se encontra em estágios mais avançados.

2.2. Reconhecimento de imagens

O Reconhecimento de imagens tem como objetivo distinguir objetos em categorias, conforme seus padrões. Para tal várias etapas estão envolvidas, sendo elas de sensoriamento, de segmentação, de extração de características e de pós processamento (BURGER, 2011).

O sensoriamento implica na conversão do que será analisado em um conjunto de dados compostos por objetos e plano de fundo, geralmente possuindo algum tipo de ruído.

Então, os objetos são segmentados e são extraídas as características relevantes para a classificação. No pós-processamento, são ponderadas as saídas de diferentes classificadores e então é tomada a decisão final (Figura 3).

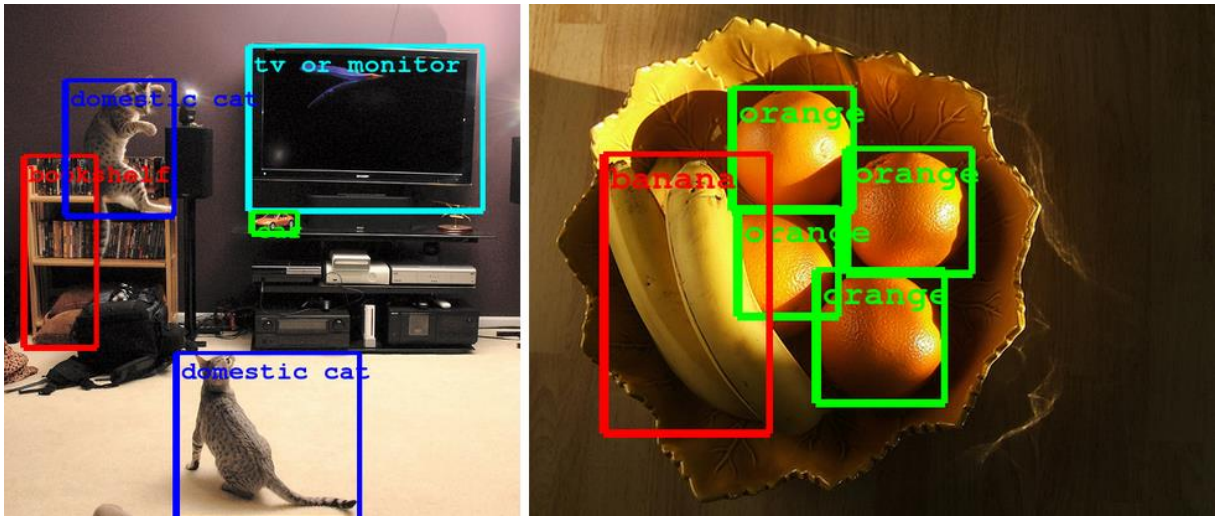

Figura 3: Reconhecimento de objetos por algoritmo de reconhecimento de imagens.

Fonte: https://ai.googleblog.com/

Uma boa definição dos padrões de cada objeto é essencial para sua identificação. Com os dados dos pixels das imagens convertidos em dados computacionais, ou seja, que podem ser processados por um computador, usamos esse padrão para identificar os objetos naquela imagem.

No caso do reconhecimento facial, os padrões buscados são as características faciais, como a largura da boca, espaço entre os olhos ou tamanho do nariz, como é demonstrado na Figura 4. Existem diversas formas diferentes de detectar esses pontos de controle (BURGER, 2014).

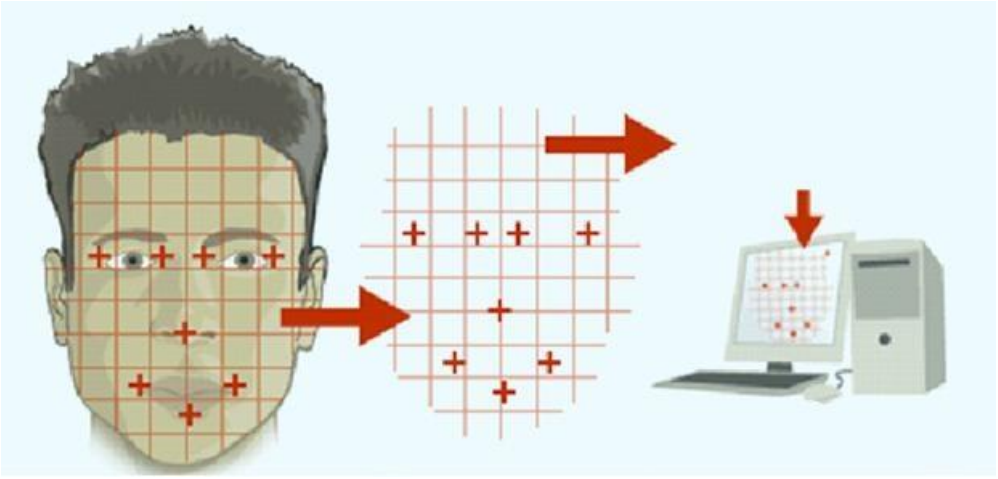

Figura 4: Reconhecimento de características faciais e tradução em dados.

Fonte: http://www.facerecognitionsolution.com/

2.3. A biblioteca OpenCV

O OpenCV (Open Source Computer Vision Library) é uma biblioteca de visão computacional e de aprendizagem de máquina de uso livre desenvolvida pela Intel nos anos 2000. Ela suporta as principais plataformas, como Windows e Linux, além de interfaces para as principais linguagens de programação, como C++, Java e Python.

A biblioteca possui mais de 2.500 algoritmos otimizados para computação visual e aprendizagem de máquina e citaremos alguns a seguir:

2.3.1 Haar Cascade

A abordagem baseada em aprendizado de máquina é um método eficiente (VIOLA, P., 2001) proposto para o reconhecimento de imagens em preto e branco. Tendo como base os classificadores Haar (Figura 5), os padrões de cada objeto são catalogados e identificados.

Inicialmente o algoritmo necessita treinar os classificadores através de imagens daquele objeto, chamadas de positivas, e imagens negativas, o oposto. São buscadas ocorrências relevantes dos classificadores Haar ao longo da imagem, figura 6, e catalogados na base de dados. Essa base de dados então é usada para a identificação futura deste objeto.

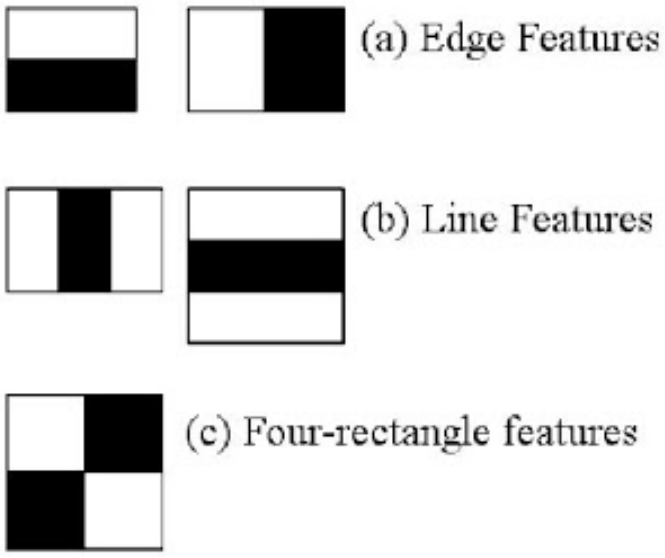

Figura 5: Classificadores utilizados pelo algoritmo Haar Cascade.

Fonte: https://docs.opencv.org/master/index.html

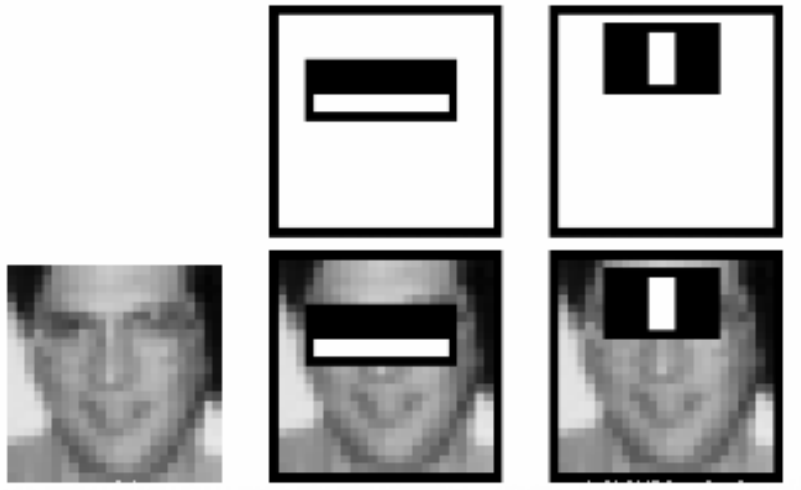

Figura 6: Busca pelos classificadores Haar relevantes da imagem.

Fonte: https://docs.opencv.org/master/index.html

Quando recebe uma imagem para identificar, o algoritmo busca pelos classificadores e, com os conhecimentos da base de dados, vai determinando as chances deste conjunto de pixels serem parte do objeto. O que possuir menor taxa de erro é então considerado como sendo o objeto.

2.3.2 Análise de Histogramas

Nem sempre os objetos podem ser identificados apenas pela observação do seu formato. Existem vários casos de objetos que possuem formato semelhante, mas com cores que os distinguem, como por exemplo as frutas laranja e limão, que possuem mesmo formato, mas podem ser diferenciados pelas cores de suas cascas ou gomos. A análise do histograma da imagem é um grande aliado nesses casos.

Um histograma analisa a concentração de pixels da imagem para cada tom daquela cor, seja a imagem cinza ou em RGB, exemplificado na Figura 7. Conhecendo o histograma do objeto, podemos compará-lo ao histograma da imagem que deseja identificar e obter o resultado esperado.

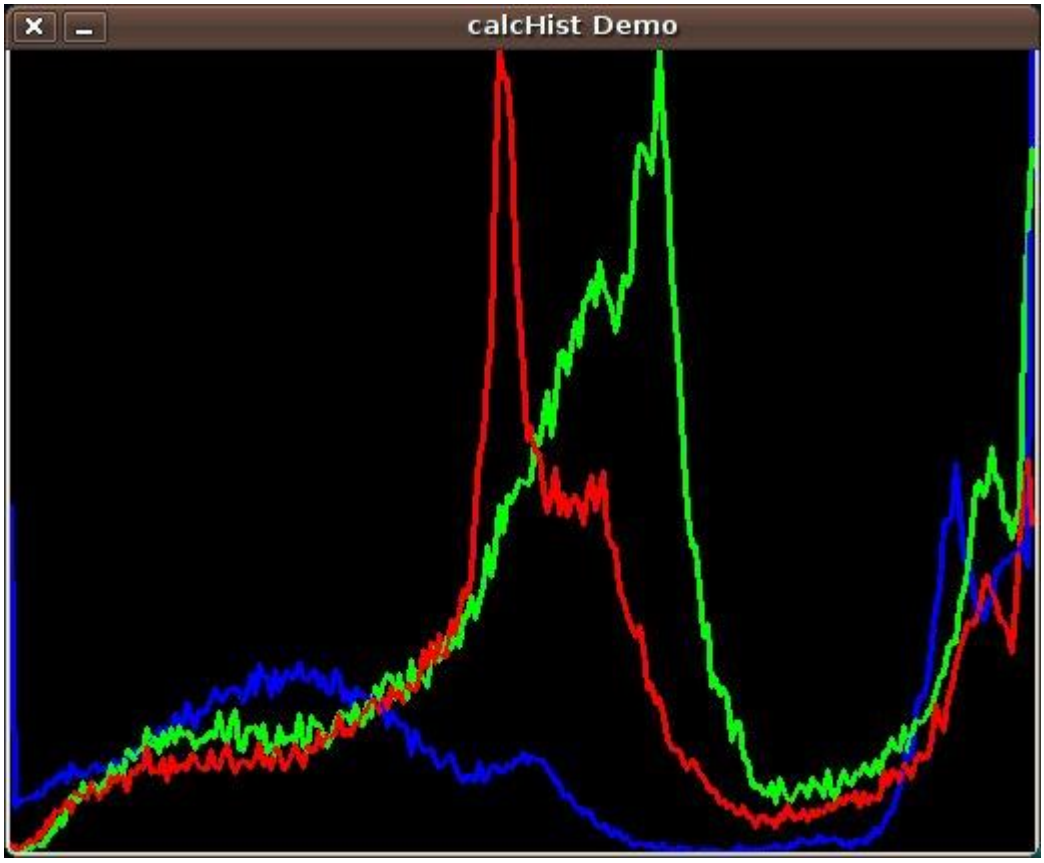

Figura 7: Exemplo de Histograma de imagem em RGB

Fonte: https://docs.opencv.org/master/index.html

Ao comparar o histograma do limão com o da laranja, por exemplo, podemos observar uma concentração maior de pixels nas tonalidades de verde, enquanto na laranja temos uma certa concentração de pixels em tons verdes, assim como em tons vermelhos. A partir dessa informação é possível então distinguir o objeto entre laranja ou limão no meio computacional.

3 METODOLOGIA

O início deste projeto foi acompanhado de um estudo sobre o retinoblastoma com foco na leucocoria, sintoma que pode ser percebido em fotografias não-clínicas, para definição do método mais eficiente para a identificação do reflexo característico da doença.

Em seguida, foi realizada a configuração da biblioteca OpenCV em um ambiente de desenvolvimento através do canal Anaconda, que disponibiliza de forma simples a inclusão da biblioteca no ambiente de execução desta aplicação. A aplicação web Jupyter Notebook foi escolhida por fornecer um ambiente de desenvolvimento mais interativo e a linguagem usada para implementação foi o Python(CHITYALA, R., 2016).

3.3 Definição da Região de Interesse

Com o objetivo de otimizar o processamento, é recomendada a prática de determinar uma região de interesse e então é buscado o objeto em si apenas nessa região. Essa prática faz com que, ao tentar classificar a imagem, menos pixels tenham que ser processados e analisados pelo computador, reduzindo o tempo de execução da aplicação.

A biblioteca OpenCV disponibiliza de alguns arquivos XMLs que contém uma base de dados já treinada para identificação de alguns objetos usando o algoritmo Haar Cascade. Entre eles, faces e olhos já possuem uma base de dados treinada e por isso esse método foi escolhido para a redução da área de interesse da imagem inteira para a face e, posteriormente, o olho.

Estes arquivos podem ser encontrados em opencv/data/haarcascades/ e foram utilizados para face e olhos, respectivamente, os arquivos nomeados por “haarcascade_frontalface_default.xml” e “haarcascade_eye.xml”.

Para tal, é necessário que a aplicação acesse a base de dados (arquivos XMLs) e possua uma versão da imagem em escalas de cinza. Então é utilizado os classificadores de face e a partir de cada face encontrada será utilizado o classificador para os olhos, como ilustrado na figura 8.

```
In [1]: import numpy as np
        import cv2

In [2]: face_cascade = cv2.CascadeClassifier('haarcascade/haarcascade_frontalface_default.xml')
        eye_cascade = cv2.CascadeClassifier('haarcascade/haarcascade_eye.xml')

In [3]: img = cv2.imread('img/retinoblastoma.jpg')
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

In [*]: faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        for (fx,fy,fw,fh) in faces:
            cv2.rectangle(img,(fx,fy),(fx+fw,fy+fh),(255,0,0),2)
            roi_gray = gray[fy:fy+fh, fx:fx+fw]
            roi_color = img[fy:fy+fh, fx:fx+fw]
            eyes = eye_cascade.detectMultiScale(roi_gray)
            for (ex,ey,ew,eh) in eyes:
                cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
                roi_gray = roi_gray[ey:ey+eh, ex:ex+ew]
                roi_color = roi_color[ey:ey+eh, ex:ex+ew]
                roi_bgr = cv2.cvtColor(roi_gray, cv2.COLOR_GRAY2BGR)

In [*]: cv2.imshow('img',img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
```

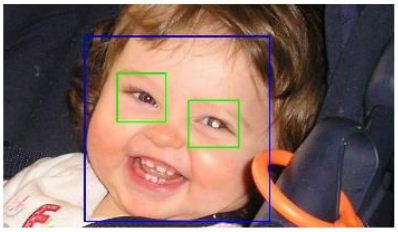

Figura 8: Identificação de face e olhos usando Haar Cascade

A identificação da íris possibilita reduzir ainda mais a área de interesse e para esta finalidade a seguinte abordagem foi tomada:

3.3.1 Threshold

Para preparar a imagem para alguns os processamentos a seguir, é necessário obter uma versão binária e invertida desta. Para essa finalidade, foi utilizado o método de Threshold (Figura 9), que possui os seguintes parâmetros:

- Uma imagem em escala de cinza;
- O valor de Threshold, que determina que qualquer valor abaixo dele é branco e acima é preto;
- O valor máximo que dos pixels;
- Tipo de Threshold escolhido. Neste parâmetro dizemos ao OpenCV que queremos a imagem binária e invertida.

```
In [1]: import numpy as np

In [2]: import cv2

In [3]: face_cascade = cv2.CascadeClassifier('haarcascade/haarcascade_frontalface_default.xml')
        eye_cascade = cv2.CascadeClassifier('haarcascade/haarcascade_eye.xml')

In [*]: img = cv2.imread('img/retinoblastoma.jpg')
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

In [*]: faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        for (fx,fy,fw,fh) in faces:
            cv2.rectangle(img,(fx,fy),(fx+fw,fy+fh),(255,0,0),2)
            roi_gray = gray[fy:fy+fh, fx:fx+fw]
            roi_color = img[fy:fy+fh, fx:fx+fw]
            eyes = eye_cascade.detectMultiScale(roi_gray)
            for (ex,ey,ew,eh) in eyes:
                cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
                roi_gray = roi_gray[ey:ey+eh, ex:ex+ew]
                roi_color = roi_color[ey:ey+eh, ex:ex+ew]
                roi_bgr = cv2.cvtColor(roi_gray, cv2.COLOR_GRAY2BGR)
                if roi_bgr is not None:
                    _, thresh = cv2.threshold(roi_gray, 92, 255, cv2.THRESH_BINARY_INV)
                    cv2.imshow('thresh',thresh)
                    cv2.waitKey(0)

In [*]: cv2.imshow('img',img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
```

Figura 9: Processamento Threshold e a imagem binária e invertida resultante

O valor de threshold se mostrou um problema quando exposto à diferentes imagens, pois a diferença de iluminação e contraste entre as elas fez com que o mesmo valor não se aplicasse para todas elas. Ao decorrer do projeto foram usados valores obtidos manualmente para cada objeto.

3.3.2 Morfologia Matemática

As operações de fechamento e abertura morfológica tem como objetivo extrair componentes úteis para o reconhecimento de imagens, onde a primeira reduz buracos e conecta componentes e a segunda elimina pequenos componentes e suaviza o contorno. É requisito deste processamento que a imagem esteja binaria e este se mostrou eficiente para obter contornos mais precisos da região de interesse (Figura 10).

```
In [1]: import numpy as np

In [2]: import cv2

In [3]: face_cascade = cv2.CascadeClassifier('haarcascade/haarcascade_frontalface_default.xml')
        eye_cascade = cv2.CascadeClassifier('haarcascade/haarcascade_eye.xml')

In [*]: img = cv2.imread('img/retinoblastoma.jpg')
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

In [*]: faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        for (fx,fy,fw,fh) in faces:
            cv2.rectangle(img,(fx,fy),(fx+fw,fy+fh),(255,0,0),2)
            roi_gray = gray[fy:fy+fh, fx:fx+fw]
            roi_color = img[fy:fy+fh, fx:fx+fw]
            eyes = eye_cascade.detectMultiScale(roi_gray)
            for (ex,ey,ew,eh) in eyes:
                cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
                roi_gray = roi_gray[ey:ey+eh, ex:ex+ew]
                roi_color = roi_color[ey:ey+eh, ex:ex+ew]
                roi_bgr = cv2.cvtColor(roi_gray, cv2.COLOR_GRAY2BGR)
                if roi_bgr is not None:
                    _, thresh = cv2.threshold(roi_gray, 92, 255, cv2.THRESH_BINARY_INV)
                    kernel=np.ones((3, 3), np.uint8)
                    morph_close = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)
                    cv2.imshow('morph_close',morph_close)
                    cv2.waitKey(0)
                    morph_open = cv2.morphologyEx(morph_close, cv2.MORPH_OPEN, kernel)
                    cv2.imshow('morph_open',morph_open)
                    cv2.waitKey(0)
```

Figura 10: Fechamento e abertura morfológica da região de interesse

3.3.3 Definição de contornos para segmentação

A biblioteca OpenCV possui um método de identificação de contornos em imagens através da ligação de pontos contínuos de mesma cor e intensidade encontrados nas bordas. Para tal é recomendado o uso de uma imagem binária, onde o fundo tenha cor preta e o objeto a ser encontrado seja branco.

Como a região de interesse atual são os olhos e então teríamos um fundo branco com objeto em preto, ou seja, o oposto do requisito para este processamento, a imagem é previamente invertida pelo Threshold.

Após obtenção dos contornos, foi escolhido usar uma função capaz de segmentar a menor circunferência que pode englobar o contorno inteiro para definir a nova região de interesse, que seria a íris.

Os resultados desta segmentação podem ser observados na Figura 11.

```
In [1]: import numpy as np
        import cv2

In [2]: face_cascade = cv2.CascadeClassifier('haarcascade/haarcascade_frontalface_default.xml')
        eye_cascade = cv2.CascadeClassifier('haarcascade/haarcascade_eye.xml')

In [3]: img = cv2.imread('img/retinoblastoma.jpg')
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

In [4]: faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        for (fx,fy,fw,fh) in faces:
            cv2.rectangle(img,(fx,fy),(fx+fw,fy+fh),(255,0,0),2)
            roi_gray = gray[fy:fy+fh, fx:fx+fw]
            roi_color = img[fy:fy+fh, fx:fx+fw]
            eyes = eye_cascade.detectMultiScale(roi_gray)
            for (ex,ey,ew,eh) in eyes:
                cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
                roi_gray = roi_gray[ey:ey+eh, ex:ex+ew]
                roi_color = roi_color[ey:ey+eh, ex:ex+ew]
                roi_bgr = cv2.cvtColor(roi_gray, cv2.COLOR_GRAY2BGR)
                if roi_bgr is not None:
                    _, thresh = cv2.threshold(roi_gray, 92, 255, cv2.THRESH_BINARY_INV)
                    kernel=np.ones((3, 3), np.uint8)
                    morph_close = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)
                    morph_open = cv2.morphologyEx(morph_close, cv2.MORPH_OPEN, kernel)
                    _, contours, hierarchy = cv2.findContours(morph_open, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
                    temp1 = roi_bgr.copy()
                    cv2.drawContours(temp1, contours, -1, (0, 0, 255))
                    if contours :
                        (x, y), radius = cv2.minEnclosingCircle(contours[0])
                        center = (int(x), int(y))
                        radius = int(radius)
                        cv2.circle(roi_color, center, radius, (0, 0, 255), 2)

In [5]: cv2.imshow('img',img)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
```

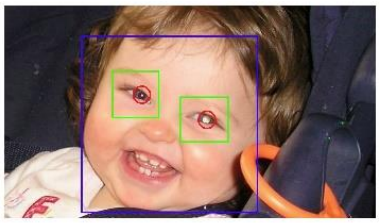

Figura 11: Segmentação da Região de interesse

3.4 Identificação da leucocoria

No início deste projeto, foi idealizado a identificação da leucocoria através do algoritmo de Haar Cascade, mas com o desenvolver do projeto a alternativa foi reavaliada e se mostrou não ser a melhor alternativa.

Então foi decidido por abordar a identificação da leucocoria através da análise do histograma da região de interesse. Isso seria possível, já que na presença do tumor, a retina adquire um reflexo esbranquiçado, enquanto na saudável a coloração é mais próxima do preto ou do vermelho.

Foi montada uma base de dados com fotografias não clínicas de crianças com o sintoma da leucocoria visível para coleta de dados dos histogramas, a ser usada para futura identificação.

Devido à problemas com a seleção da região de interesse, esta parte da aplicação continua sem implementação, pois possui grande dependência dessa segmentação, já que

sem a mesma, a análise do histograma não trará resultados satisfatória para a identificação da leucocoria.

4 RESULTADO E DISCUSSÃO

A implementação deste algoritmo se mostrou problemática em alguns pontos, a serem tratados a seguir, que impossibilitou a finalização da aplicação de identificação da leucocoria.

Primeiramente o algoritmo Haar Cascade, considerado no início do projeto para identificação da leucocoria se mostrou não satisfatório para esta questão. Isso ocorre principalmente devido à ausência de uma base de imagens não-clínicas do sintoma para o treinamento de máquina eficiente.

Também foram encontradas dificuldades com a conversão da imagem da região do olho para binária. Isso porque conforme a iluminação e contraste de cada imagem, o valor do threshold deve variar entre as imagens.

O OpenCV fornece ferramentas, como a *Otsu's binarization*, que realiza o cálculo do valor de Threshold de forma automática, mas este não se mostrou satisfatório diante do problema proposto.

Assim foi proposta a obtenção deste valor pelo cálculo da mediana do histograma da imagem, como mostrado na figura. Porém, algumas imagens possuem mais de um pico em seu histograma, o que faz com que esse método também apresenta um grande risco de falha.

Para prosseguir com o projeto, então, foi optado pela manipulação manual do valor de Threshold para cada imagem que, apesar de não ser o ideal, se mostrou suficiente para o objetivo de estudo.

5 CONSIDERAÇÕES FINAIS

A necessidade de uma aplicação de conscientização ao diagnóstico precoce do retinoblastoma tem potencial para provocar impacto positivo na saúde pública e por isso medidas como essa são de extrema importância e necessidade. Apesar dos problemas encontrados ao longo de sua implementação, o algoritmo não é muito complexo, mas para que possa atingir ao público, é necessário a adequação do valor do Threshold, para que todas as imagens submetidas na plataforma possam ser devidamente processadas e catalogadas.

6 REFERÊNCIAS

BERTOLDI, A.R; GONÇALVES, B.; SANTOS, T. Importância da Inclusão do Teste do Reflexo Vermelho no Protocolo de Exames da Infância para Diagnóstico Precoce do Retinoblastoma. In: **Revista Ciência em Saúde**, 3(2), 2012, pp. 50-65.

BURGER, W. **Principles of Digital Image Processing**. New York: Springer, 2011.

CELEBI, M.E. **Dermoscopy Image Analysis**. New York: CRC Press, 2015.

CHITYALA, R. **Image Processing and Acquision using Python**. New York: CRC Press, 2016.

FRANCIS, J.; ABRAMSON, D.H. **Recent Advances in Retinoblastoma Treatment**. New York: Springer, 2015.

GALINDO, C.; WILSON, M.W. **Retinoblastome**. New York: Springer, 2016.

SINGH, A.D. **Clinical Ophtalmic Oncology**. New York: Springer, 2014.

VIOLA, P.; JONES, M. **Rapid object detection using a boosted cascade of simple features**. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. IEEE, 2001.

Contatos: stella.fraguas@gmail.com e luciano.silva@mackenzie.br