https://eventos.utfpr.edu.br//sicite/sicite2020

Desenvolvimento de Ferramentas para Instrumentação Científica em Física - Reconhecimento de imagens como forma de Instrumentação

Development of Tools for Scientific Instrumentation in Physics – Image Recognition as a form of Instrumentation

Vitor Stefano Lopes Messias
vitormessias@alunos.utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Cornélio Procópio, Paraná, Brasil

Cássio Henrique dos Santos Amador
cassioamador@utfpr.edu.br
Universidade Tecnológica Federal do Paraná, Cornélio Procópio, Paraná, Brasil

RESUMO

Neste projeto, mostramos a primeira etapa na construção de ferramentas para reconhecimento e análise de velocidade de objetos, cujo movimento é capturado por uma câmera. A análise é feita sobre um vídeo, utilizando o pacote OpenCV. Nesta primeira etapa, concentrada no aprendizado de ferramentas computacionais, o estudante aprendeu a linguagem Python, sobre o pacote OpenCV, e também a programar num ambiente Linux. Como prova de conceito, foi utilizado o pacote de reconhecimento de rostos do OpenCV, no qual um rosto é rastreado e a velocidade de movimento é calculada. Ao final, foi possível fazer com que o software reconheça o rosto de uma pessoa e mostrar que é possível medir a velocidade em pixels por segundo, de acordo com o movimento que a face executa.

PALAVRAS-CHAVE: Processamento de imagens. Reconhecimento de rostos.

ABSTRACT

In this project, we show the first step in the construction of tools for the recognition and analysis of the speed of objects, whose movement is captured by a camera. The analysis is done on a video, using the OpenCV package. In this first step, focused on learning computational tools, the student learned the Python language, about the OpenCV package, and also to program in a Linux environment. As a proof of concept, the OpenCV face recognition package was used, in which a face is tracked and the speed of movement is calculated. In the end, it was possible to make the software recognize a person's face and show that it is possible to measure the speed in pixels per second, according to the movement that the face performs.

KEYWORDS: Image processing. Face recognition.

Recebido: 19 ago. 2020.
Aprovado: 01 out. 2020.

INTRODUÇÃO

O reconhecimento de coisas ou pessoas ao nosso redor faz parte da vida do ser humano desde seu nascimento. Fazemos isso diariamente ao olhar para nossos pais, animais de estimação, um amigo, etc. Com o avanço das tecnologias, bem como o avanço da programação, tornou-se possível criar softwares para identificação de criminosos, carros com sistemas de reconhecimento de espaço que podem estacionar sozinhos, drones que reconhecem pessoas em desastres naturais. Este avanço tornou possível ultrapassar barreiras que há um século jamais seria possível imaginar.

Neste trabalho, procuramos explorar algumas das muitas possibilidades fornecidas pelos avanços citados, utilizando a linguagem de programação Python e o pacote OpenCV que, como descrito em sua documentação, é uma biblioteca de software para aprendizado de máquina e reconhecimento de imagens, construída como uma infraestrutura comum para aplicações relacionadas a estes tópicos. Visamos fazer que com o software reconheça o rosto de uma pessoa e, além disso, mostrar que é possível medir a velocidade em pixels por segundo, de acordo com o movimento que a face executa.

METODOLOGIA

A metodologia empregada neste trabalho teve como finalidade auxiliar na concretização do objetivo principal de nossa investigação: encontrar rostos de pessoas em qualquer tipo de vídeo. Para melhor explicitá-la, dividiremos nossa apresentação em etapas de acordo com o código desenvolvido.

Decidimos inserir capturas de tela do código para melhor exemplificar o passo a passo seguido durante o aprendizado e a pesquisa, além de explicitar a forma de indentação e sintaxe para pessoas que estejam iniciando estudos na linguagem Python e utilização do pacote OpenCV.

A. Importar a biblioteca de funções OpenCV para o código

Figura 1

```
import cv2
import numpy as np
```

Fonte: código desenvolvido durante a pesquisa

Neste trecho, a função "*import*" importa a biblioteca do OpenCV para o código, desta forma podemos utilizar todas as suas funções.

B. Importar o vídeo para o código

Figura 2

```
cap = cv2.VideoCapture('videoic3.mp4')
fps = cap.get(cv2.CAP_PROP_FPS)
```

Fonte: código desenvolvido durante a pesquisa

Após a inserção do vídeo no código, o atribuímos à variável "*cap*" para facilitar sua utilização durante o programa.

C. *Selecionar o classificador de detecção de faces*

Figura 3

```
car_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
x_old = 0
y_old = 0
```

Fonte: código desenvolvido durante a pesquisa

Selecionado o classificador de detecção de faces, o atribuímos à uma variável denominada *"car_cascade"*. Desta forma, podemos trabalhar com o classificador durante o desenvolvimento do código apenas chamando a variável criada.

D. *Selecionar o vídeo gravado para que ele seja convertido em uma escala de cinza*

Figura 4

```
while True:
    ret, frame = cap.read()
    if ret:
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Fonte: código desenvolvido durante a pesquisa

Após a seleção do vídeo, o mesmo deve ser convertido em preto e branco. Caso contrário, o algoritmo não funciona.

E. *Detectar faces de diferentes tamanhos no vídeo*

Figura 5

```
while True:
    ret, frame = cap.read()
    if ret:
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        objects = car_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)
```

Fonte: código desenvolvido durante a pesquisa

F. *Desenhar círculos em cada face do vídeo*

Figura 6

```
for (x,y,w,h) in objects:
    #cv2.rectangle(frame,(x,y),(x+w,y+h),(0,0,255),2)
    cv2.circle(frame,(x+int(w/2),y+int(h/2)),int(w/8),(0,0,255),2)
```

Fonte: código desenvolvido durante a pesquisa

G. *Obter o retorno das coordenadas do eixo X e Y da face em movimento e, em seguida, desenhar uma flecha no centro do rosto indicando a direção do movimento e sua velocidade, exibindo as coordenadas no terminal.*

Figura 7

```
if len(objects) != 0:
    xc = x+int(w/2)
    yc = y+int(h/2)
    escala = 4
    x_delta = xc - x_old
    y_delta = yc - y_old
    modulo = np.sqrt(x_delta ** 2 + y_delta ** 2)
    if modulo > 10:
        thickness = 10
    else:
        thickness = 4
    cv2.arrowedLine(frame, (xc,yc), (xc + x_delta * escala
        ,yc + y_delta * escala), (255,0,0),thickness)
    print(modulo)
    x_old = xc
    y_old = yc
```

Fonte: código desenvolvido durante a pesquisa

H. *Exibir o vídeo frame by frame em uma janela separada e possibilitar o encerramento da exibição pressionando a tecla Esc.*

Figura 8

```
    cv2.imshow('video2', frame)

    if cv2.waitKey(33) == 27:
        break
else:
    break
```

Fonte: código desenvolvido durante a pesquisa

I. *Liberar o espaço da memória secundária utilizado pelo programa*

Figura 9

```
cv2.destroyAllWindows()
```

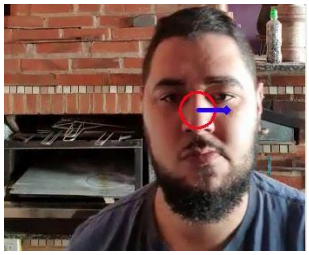
Fonte: código desenvolvido durante a pesquisa

A função *destroyAllWindows* é utilizada para liberar o espaço de memória que foi alocado especificamente para a execução do programa. É de extrema importância que ela seja utilizada ao final do código, pois desta forma a memória secundária não fica sobrecarregada ao término do processamento.

RESULTADOS

Apesar de alguns empecilhos, como a identificação de faces em objetos que não eram faces (falsos positivos), conseguimos fazer com que o algoritmo detectasse apenas os rostos presentes nos vídeos. A seguir, podemos ver alguns *prints* do algoritmo em funcionamento. As fotos a seguir são do autor (orientando) do projeto.

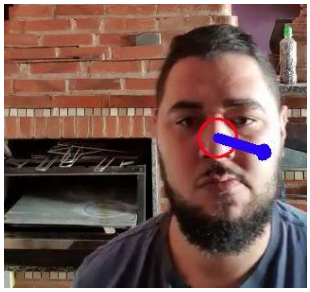
Figura 10



Fonte: vídeo utilizado durante a pesquisa

Neste *frame* (Figura 10), podemos ver o círculo localizado exatamente no centro da face detectada. A flecha desenhada aponta para a direção do movimento que o rosto está executando. No caso da foto, o movimento é para a direita.

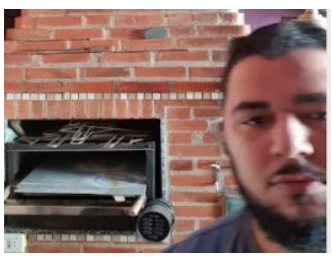
Figura 11



Fonte: vídeo utilizado durante a pesquisa

O *frame* seguinte (Figura 11) mostra a mudança de tamanho da flecha, indicando a variação da velocidade (quanto maior a flecha, maior a velocidade e vice versa). O movimento do rosto continua para o lado direito, com uma inclinação para baixo, o que também interfere em sua velocidade.

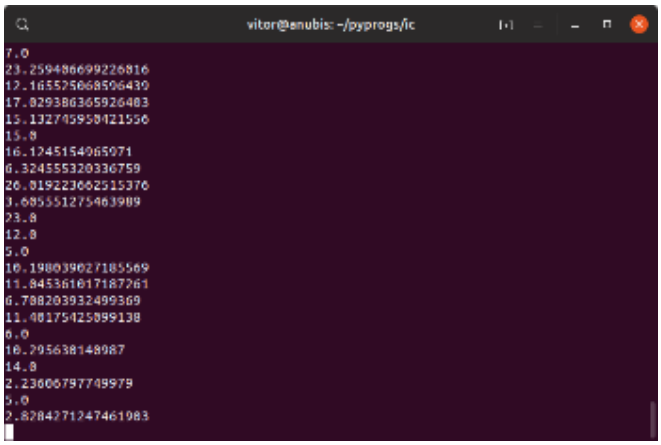
Figura 12



Fonte: vídeo utilizado durante a pesquisa

Na Figura 12, podemos notar que o algoritmo deixa de trabalhar quando algum parâmetro do rosto (olhos, nariz, orelhas e boca) não estão presentes ou estão incompletos no vídeo. Imediatamente, o círculo central e a flecha somem, fazendo com que o retorno das coordenadas X e Y no terminal também pare.

Figura 13



Fonte: código desenvolvido durante a pesquisa

Na captura de tela da Figura 13, podemos ver as coordenadas do eixo X (antes do ponto final) e eixo Y (números após o ponto final). Este retorno é automático e é listado de acordo com o movimento da face detectada no vídeo.

CONCLUSÃO

Nossa pesquisa apresenta o desenvolvimento de um código para reconhecimento facial em vídeos, desde o aprendizado da linguagem de programação Python até a utilização da biblioteca OpenCV, sendo esta imprescindível para a realização do trabalho. Observamos a importância da iluminação do ambiente em que o vídeo é gravado. Sendo assim, quanto maior a claridade, mais precisa se torna a identificação do rosto. Nesse sentido, para aprofundamentos futuros pensamos que essa seja uma dificuldade a ser resolvida. Além disso, este trabalho servirá como base para futuras ferramentas de reconhecimento de velocidade de objetos.

REFERÊNCIAS

ANTONELLO, Ricardo. **Introdução a Visão Computacional em Python e OpenCV**. v. 08, p. 45-55, p. 63-67. Disponível em: https://cv.antonello.com.br/wp-content/uploads/2017/02/Livro-Introdu%C3%A7%C3%A3o-a-Vis%C3%A3o-Computacional-com-Python-e-OpenCV-3.pdf. Acesso em: 03 ago. 2020.

BORGES, Luiz Eduardo. **Python para Desenvolvedores**, Rio de Janeiro, Edição do Autor, 2010. Disponível em: https://ark4n.files.wordpress.com/2010/01/python_para_desenvolvedores_2ed.pdf. Acesso em: 03 ago. 2020.

LOPES MENDES, Marcos André. **Python Brasil: Lista de Exercícios**. Disponível em: https://wiki.python.org.br/ListaDeExercicios. Acesso em: 03 ago. 2020.

OpenCV Documentation: <https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html>. Acesso em: 04 ago. 2020.

OpenCV Documentation: < https://opencv.org/about>. Acesso em: 08 out. 2020.