Alexandre Bruno dos Santos GRR20194226
Eduardo Ribeiro Mayer GRR20196018
Marcio Antonio GRR20199926

Avaliação de Calibre de Laranja por Análise de Imagem em Python

Curitiba, Paraná, Brasil
2019

Alexandre Bruno dos Santos GRR20194226
Eduardo Ribeiro Mayer GRR20196018
Marcio Antonio GRR20199926

Avaliação de Calibre de Laranja por Análise de Imagem em Python

Relatório exigido como nota parcial na disciplina Fundamentos de Programação de Computadores

Universidade Federal do Paraná

Setor de Ciências Agrárias

Agronomia

Curitiba, Paraná, Brasil

2019

Resumo

Com o passar do tempo a produção de laranjas teve um crescimento bastante expansivo e precisa-se de um modo pratico e facil para estimar a tamanha produção. Com isso foi elaborado um programa no qual ele calcula o volume da laranja apartir da imagem da mesma.

Palavras-chaves: Calibre de Laranjas. Fruticultura. Visão Computacional. Python. Pós-Colheita.

Sumário

1 Introdução

A fruta mais produzida no Brasil é a Laranja, seguida de outros cítricos. E uma produção tão expressiva leva a uma produção mais variada. Existem laranjas de várias variedades, tamanhos, cores e sabores. Como é comum o policultivo de laranjas numa mesma propriedade, normalmente laranjas de categorias diferentes precisam ser separadas para receber destino adequado. Por muito tempo, esse trabalho foi feito manualmente seguindo critérios arbitrários. Entretanto, com a chegada da Revolução Digital, hoje é possível encarregar uma máquina da seleção das frutas. Isso acarreta menos custos com mão-de-obra e uma seleção mais precisa embasada em termos quantitativos, como o calibre da laranja.

O presente trabalho se propõe a desenvolver um programa escrito na linguagem de programação Python, usando as bibliotecas Numpy e OpenCV, capaz de identificar laranjas em imagens e estimar o seu calibre a partir das mesmas. Com isso, seria possível implementar em um sistema industrial um mecanismo automatizado para seleção de laranjas por calibre. O que traria mais eficiência à empresa envolvida e cortaria custos.

2 Fundamentação Teórica

Laranja é a fruta mais produzida no pais responde por mais de 28,8% do total produzido no mundo, isso corresponde 18 toneladas anualmente, visando o problema no qual o produtor precisa ter uma estimativa de sua produção foi desenvolvido um progroma em phyton no qual ele calcula o volume esferico de uma laranja determinando seu raio e logo com equações matematicas solucionando o calculo.

3 Metodologia

3.1 Bibliotecas

Neste trabalho, foram usadas três bibliotecas: math, numpy e opencv.

A biblioteca math contém funções e constantes matemáticas fundamentais, por exemplo, math.pow(), math.sqrt(), math.pi. Ela é muito utilizada por ser uma biblioteca nativa do Python.

Por outro lado, a biblioteca NumPy permite trabalhar de forma eficiente com arrays n-dimensionais e operações matemáticas mais complexas. Ela é muito utilizada em trabalhos científicos e, como é usada em funções da biblioteca OpenCV, aparece também trabalhos de visão computacional.

Já a biblioteca OpenCV2 traz ferramentas para tratar imagens e vídeos. Suas funções permitem leitura e registro de imagens e vídeos, tratamento, alterações e criação de imagens, reconhecimento de linhas, vértices, círculos, rostos, etc. em imagens e vídeos, entre outros. Ela é amplamente utilizada por ser de código aberto e trazer métodos bastante eficientes para as áreas de Inteligência Artificial e Robótica.

3.2 Código

Começamos importando as bibliotecas já citadas e inicializando as principais variáveis que serão usadas no código. A partir disso, é definido um laço "infinito"que executará a análise de uma imagem a cada *loop*. Esse laço pedirá ao usuário o endereço da imagem no computador (as especificações são esclarecidas ao usuário) e acabará quando o usuário digitar Enter (retorna string vazia ou). A partir desse endereço, a imagem é guardada em uma variável sob a forma de um nparray. O programa exibe uma mensagem de erro caso uma imagem não seja encontrada no endereço informadado ou caso ocorra problema similar.

Diversos tratamentos são realizados sobre a imagem para maximizar a eficiência da função cv2.HoughCircles(), que procurará por formas circulares na imagem e as guardará numa lista.

Em seguida, o programa verifica se pelo menos dois círculos foram detectados: a moeda e uma laranja. Se não mostra mensagem de erro. E então procura-se o círculo que deve corresponder à moeda de referência. Ele pode ser facilmente encontrado supondo-se que o raio da moeda é menor que o raio de qualquer laranja comercial e, portanto, a moeda

Figura 1: Linhas 1-39

```python
import math
import cv2
import numpy as np

#Inicializando as variáveis principais
fator = 0 #constante de conversão pixel -> mm (muda em cada imagem, pois depende da moeda de 0.25: referencial constante)
raio_medio = 0 #Soma dos raios das n laranjas medidos em mm
volume_medio = 0 #Soma dos volumes das n laranjas medidos em mL
n=0 #Total de laranjas analisadas

#Pedindo ao usuário o endereço da primeira imagem
address = input("Por favor, digite o endereco da imagem. Nao se esqueca da extensao nem de que o arquivo deve estar na mesma pasta que este programa: ")

#Verificar imagens até o usuário apertar Enter
while(address!=""):
    print() #Pula linha
    img = cv2.imread(address,0) #Guardando a imagem numa variável

    #Verificando se img guardou uma imagem com sucesso
    if isinstance(img,np.ndarray):

        #Redimensionar img de modo que a maior dimensão tenha 700 pixel. Assim, evita-se o uso de imagens muito grandes ou muito pequenas em
        #HoughCircles(), o que poderia causar erros.

        height, width = img.shape[:2]
        escala=700/max(height,width)
        img = cv2.resize(img,(round(escala*width), round(escala*height)), interpolation = cv2.INTER_AREA)

        #Tratamento de img para uso em funções futuras. A saber, HoughCircles(), circle() e imshow()
        img = cv2.medianBlur(img,5)
        cimg = cv2.cvtColor(img,cv2.COLOR_GRAY2BGR)

        #HoughCircles() procura por círculos em img e os guarda em circles. Parâmetros com Valores Padrão
        circles = cv2.HoughCircles(img,cv2.HOUGH_GRADIENT,1,100,
                                   param1=50,param2=60,minRadius=0,maxRadius=0)

        #Verificando se circles não está vazio, ou seja, se algum círculo foi encontrado
        if isinstance(circles,np.ndarray):
            #Verificando se alguma laranja foi encontrada além da moeda
```

corresponderá ao círculo de menor raio. Conhecendo-se o raio da moeda na imagem (em pixels) e o raio real (12.5 mm para a moeda marrom de 25 centavos), calcula-se um fator de conversão pixel -> mm que será usado para encontrar o raio real das outras laranjas a partir do valor (em pixels) encontrado por cv2.HoughCircles().

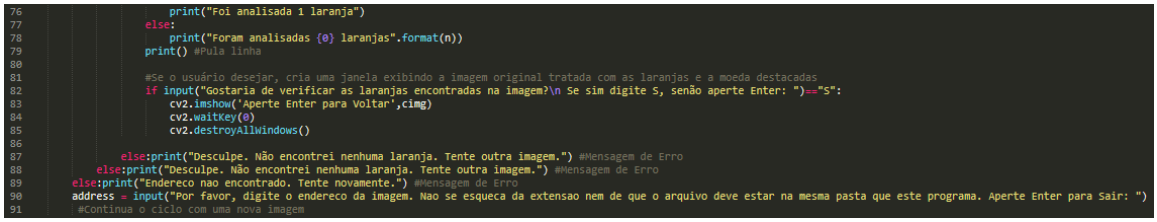
Figura 2: Linhas 39-76

```python
            #Verificando se alguma laranja foi encontrada além da moeda
            if len(circles[0])>1:

                #Procura-se pelo círculo de menor raio. Considera-se esse círculo como uma moeda de referência de 25 centavos e 25 mm de diâmetro.
                raio_ref = circles[0,0][0]
                for i in circles[0,:]:
                    if i[2]<raio_ref:
                        raio_ref=i[2]
                fator = 12.5/raio_ref #Uma vez que o raio da moeda é conhecido. Ela é usada para definir um fator de conversão pixel -> mm

                #Aqui avaliam-se os círculos. Mostra-se para o usuário os raios e volumes estimados de cada laranja
                #individualmente e soma-se esses valores aos totais para uso futuro.

                #Também são desenhados em cimg os contornos e centros das circunferências das laranjas e da moeda.

                for i in circles[0,:]:
                    if i[2]==raio_ref:
                        cv2.circle(cimg,(i[0],i[1]),i[2],(255,0,0),2) #Desenha a circunferência da moeda
                        print("Moeda")
                        print('raio: {0:.2f} mm'.format(i[2]*fator))
                        print()
                    else:
                        cv2.circle(cimg,(i[0],i[1]),i[2],(0,255,0),2) #Desenha a circunferência da laranja
                        n += 1
                        raio_medio += i[2]*fator
                        volume_medio += 4*math.pi*((i[2]*fator)**3)/3000
                        print("Laranja {0}".format(n))
                        print('raio: {0:.2f} mm'.format(i[2]*fator))
                        print('volume: {0:.2f} mL'.format(4*math.pi*((i[2]*fator)**3)/3000))
                        print()
                    cv2.circle(cimg,(i[0],i[1]),2,(0,0,255),3) #Desenha o centro do círculo

                #Informações resumidas de todas as n laranjas analisadas até o momento
                print("Raio Médio: {0:.2f} mm".format(raio_medio/n))
                print("Volume Médio: {0:.2f} mL".format(volume_medio/n))
                #Apenas para preservar a Gramática:
                if n==1:
                    print("Foi analisada 1 laranja")
```

Sabendo qual círculo é a moeda, é possível separá-lo das contas relacionadas às laranjas, como cálculo de volume e nas variáveis que somam o raio total, o volume total e a quantidade total de laranjas. O programa também calcula e exibe as médias das laranjas analisadas desde que foi aberto o arquivo .py. As circunferências correspondentes à moeda e às laranjas e seus centros são destacados numa versão tratada da imagem para serem

exibidos posteriormente se for da vontade do usuário (se for digitado 'S' e não qualquer outra coisa). A imagem original então pode ser reexibida em escala de cinza e com todos os círculos encontrados com suas circunfências e centros destacados (a moeda em azul, as laranjas em verde e os centros em vermelho). Ela some se qualquer tecla for pressionada. E o laço continua pedindo o caminho para mais uma imagem.

Figura 3: Linhas 76-91



Conclusão

Conclui-se com o presente trabalho que podemos desenvolver e solucionar qualquer problema com a elaboração com um progrma simples em phyton.