```
# -*- coding: utf-8 -*-
#Load the opencv library and drawing tools
#------------------------------------------------------
import cv2
import matplotlib.pyplot
import numpy as np
#------------------------------------------------------

#Read the path of the image
#------------------------------------------------------
# read image, support bmp,jpg,png,tiff format
#Read the path of the image in the computer
img = cv2.imread("C:/Users/cheng/Desktop/opencv/opencv/3.jpg")#Change the image path
to your own
#Set the RGB value in the image to a 64-bit floating point
img=np.array(img,dtype='float64')
#Declare the three 0 matrices, and put the R,G, and B values of the picture into the
three matrices.
b = np.zeros((img.shape[0],img.shape[1]), dtype=img.dtype)
g = np.zeros((img.shape[0],img.shape[1]), dtype=img.dtype)
r = np.zeros((img.shape[0],img.shape[1]), dtype=img.dtype)
r[:,:] = img[:,:,0]
g[:,:] = img[:,:,1]
b[:,:] = img[:,:,2]
#A formula for generating grayscale images
new=2*g-r-b
w=new.min()
new=new-w
e=new.max()
new=255*new/e
new=np.around(new)
new=np.array(new,dtype='uint8')
#------------------------------------------------------------------------

# Otsu filtering method and fill the hole
#------------------------------------------------------------------------
# A Otsu filtering method is used for filtering
ret2, th2 = cv2.threshold(new, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
#Check the threshold of Otsu filtering method
print("threshold of Otsu filtering:",ret2)
# Otsu filtering's results are replicated to hole
hole=th2.copy()
#Find the hole and fill it
cv2.floodFill(hole,None,(0,0),255)
hole=cv2.bitwise_not(hole)
filledEdgesOut=cv2.bitwise_or(th2,hole)
#------------------------------------------------------------------------

#The image of corrosion
#------------------------------------------------------------------------------
#A circle of diameter 5 is used as the corrosion structure
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(11,11))
#image of corrosion
#eroded = cv2.erode(filledEdgesOut,kernel)
eroded = cv2.morphologyEx(filledEdgesOut, cv2.MORPH_OPEN, kernel)
print("number of pixels in the plant:",len(eroded.nonzero()[0]))
#------------------------------------------------------------------------------
#Eliminate connected region
#------------------------------------------------------------------------------
def baweraopen(image,size):
    output=image.copy()
    nlabels, labels, stats, centroids = cv2.connectedComponentsWithStats(image)
    for i in range(1,nlabels-1):
```

```
        regions_size=stats[i,4]
        if regions_size<size:
            x0=stats[i,0]
            y0=stats[i,1]
            x1=stats[i,0]+stats[i,2]
            y1=stats[i,1]+stats[i,3]
            for row in range(y0,y1):
                for col in range(x0,x1):
                    if labels[row,col]==i:
                        output[row,col]=0
    return output
im2=baweraopen(eroded,180)#200 is the size of the connected region to be eliminated.
#-------------------------------------------------------------------------------

# Count the number of pixels in the plant
#-------------------------------------------------------------------------------
print("number of pixels in the plant:",len(im2.nonzero()[0]))
#distance is 50
distance_top=50
Area=(pow((0.000122*(distance_top-0.304)/0.304),2)*len(im2.nonzero()[0]))
print("leaf area:",round(Area, 2))
#-------------------------------------------------------------------------------

#show the new image.
#-------------------------------------------------------------------------------
img[:,:,2]=im2*r
img[:,:,1]=im2*g
img[:,:,0]=im2*b
matplotlib.pyplot.imshow((img * 255).astype(np.uint8))
#-------------------------------------------------------------------------------
```