**File S1** The python code of auxiliary analysis (Fig. 3).

import numpy as np  
import matplotlib.pyplot as plt  
import cv2 as cv  
  
  
def BGR2LAB(path="./Auv.jpg", show=True):  
 """  
   
 **:param** path:  
 **:return**:  
 """  
 img\_path = path  
 img\_BGR = cv.imread(img\_path)  
 img\_LAB = cv.cvtColor(img\_BGR, cv.COLOR\_BGR2LAB)  
 img\_RGB = cv.cvtColor(img\_BGR, cv.COLOR\_BGR2RGB)  
 img\_L = img\_LAB[..., 0]  
 img\_A = img\_LAB[..., 1]  
 img\_B = img\_LAB[..., 2]  
 if show:  
 fig = plt.gcf()  
 fig.set\_size\_inches(10, 15)  
 plt.subplot(221)  
 plt.imshow(img\_LAB)  
 plt.axis('off')  
 plt.title('Image\_LAB')  
 plt.subplot(222)  
 plt.imshow(img\_L, cmap='gray')  
 plt.axis('off')  
 plt.title('L channel image')  
 plt.subplot(223)  
 plt.imshow(img\_A, cmap='gray')  
 plt.axis('off')  
 plt.title('A channel image')  
 plt.subplot(224)  
 plt.imshow(img\_B, cmap='gray')  
 plt.axis('off')  
 plt.title('B channel image')  
 plt.show()  
 return img\_RGB, img\_L, img\_A, img\_B  
  
  
def ErosionDilationTophat(img, show=True):  
 """  
   
 **:param** img: img\_L  
 **:param** show:  
 **:return**:  
 """  
 kernel\_hat = np.ones((9, 19), np.uint8)  
 img\_Tophat = cv.morphologyEx(img, op=cv.MORPH\_TOPHAT, kernel=kernel\_hat)  
 img\_ero = cv.erode(img\_Tophat, (3, 3), iterations=1)  
 if show:  
 plt.subplot(111)  
 plt.imshow(img\_ero, cmap='gray')  
 plt.axis('off')  
 plt.title('Image\_ero')  
 plt.show()  
 return img\_ero  
  
  
def mask(img\_org, img\_ero, show=True):  
 """  
  
 **:param** img:  
 **:param** show:  
 **:return**:  
 """  
 lower\_white = (10)  
 uper\_white = (30)  
 mask = cv.inRange(img\_ero, lower\_white, uper\_white)  
 res = cv.bitwise\_and(img\_ero, img\_ero, mask=mask)  
 if show:  
 plt.figure(dpi=300, figsize=(8, 24))  
 plt.subplot(131)  
 plt.imshow(img\_org, cmap='gray')  
 plt.axis('off')  
 plt.title('Image\_org')  
 plt.subplot(222)  
 plt.imshow(img\_ero, cmap='gray')  
 plt.axis('off')  
 plt.title('Image\_ero')  
 plt.subplot(132)  
 plt.imshow(mask, cmap='gray')  
 plt.axis('off')  
 plt.title('Mask')  
 plt.subplot(133)  
 plt.imshow(res, cmap='gray')  
 plt.axis('off')  
 plt.title('Bright areas')  
 plt.show()  
 # np.set\_printoptions(threshold=np.inf)  
 # f = open('img.txt',"w")  
 # print(res,file=f)  
 # f.close  
  
 return mask, res  
  
  
def Canny(mask, show=True):  
 """  
  
 **:param** mask:  
 **:param** show:  
 **:return**:  
 """  
 lowThreshold = 40  
 max\_lowThreshold = 50  
 edge = cv.Canny(mask, lowThreshold, max\_lowThreshold)  
 if show:  
 plt.figure(dpi=300, figsize=(10, 8))  
 # plt.subplot(1, 2, 1)  
 # plt.imshow(mask, cmap="gray")  
 # plt.axis('off')  
 # plt.title('Mask')  
 # plt.subplot(1, 2, 2)  
 plt.imshow(edge, cmap="gray")  
 plt.title("Canny")  
 plt.axis('off')  
 plt.show()  
 return edge  
  
  
def DrawContours(img\_org, edge, show=True):  
 """  
  
 **:param** edge:  
 **:param** show:  
 **:return**:  
 """  
 out, counters, hierarchy = cv.findContours(edge, cv.RETR\_EXTERNAL, cv.CHAIN\_APPROX\_SIMPLE)  
 img = cv.cvtColor(img\_org, cv.COLOR\_GRAY2RGB)  
 for i in range(len(counters)):  
 rect = cv.minAreaRect(counters[i])  
 cx, cy = rect[0]  
 box = cv.boxPoints(rect)  
 box = np.int0(box)  
 cv.drawContours(img, [box], 0, (255, 0, 0), 1)  
 if show:  
 plt.figure(dpi=300, figsize=(10, 8))  
 # plt.subplot(1, 2, 1)  
 # plt.imshow(img\_org)  
 # plt.title("Original Image")  
 # plt.subplot(1, 2, 2)  
 plt.imshow(img)  
 plt.title("Contours-Drawed Image")  
 plt.axis('off')  
 plt.show()  
 return counters, img  
  
  
def ReadTheNumber(image=None, image\_channel=1):  
 def opencv\_mouse(event, x, y, flags, param):  
 # 左键点击  
 if event == cv.EVENT\_LBUTTONDBLCLK:  
 if image\_channel == 1:  
 print(f"({x},{y}) luminance value is {image[y, x]}")  
 if image\_channel == 3:  
 print(f"({x},{y}) luminance value is {image[y, x, 1]}")  
 print()  
  
 def init\_log():  
 events = [i for i in dir(cv) if 'EVENT' in i]  
 print(events)  
  
 init\_log()  
 if image is None:  
 print(f'The image do not existed.')  
 import sys  
 sys.exit()  
 print("The shape of the image is ", image.shape)  
 if len(image.shape) == 3:  
 height, width, image\_channel = image.shape  
 else:  
 height, width = image.shape  
 cv.namedWindow('L channel image', cv.WINDOW\_NORMAL)  
 cv.setMouseCallback('L channel image', opencv\_mouse)  
 cv.imshow('L channel image', image)  
 cv.waitKey(0)  
 cv.destroyWindow()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 img\_org, img\_L, img\_A, img\_B = BGR2LAB(show=False)  
 img\_ero = ErosionDilationTophat(img\_L, show=False)  
 mask, res = mask(img\_L, img\_ero, show=False)  
 edge = Canny(mask, show=False)  
 counters, img\_drawed = DrawContours(img\_L, edge, show=False)  
 ReadTheNumber(img\_L)