*# OLD version of image color cropping, use crop\_image\_from\_gray instead*

*# The above code work only for 1-channel. Here is my simple extension for 3-channels image*

def crop\_image(img,tol=7):

if img.ndim ==2:

mask = img>tol

return img[np.ix\_(mask.any(1),mask.any(0))]

elif img.ndim==3:

h,w,\_=img.shape

*# print(h,w)*

img1=cv2.resize(crop\_image1(img[:,:,0]),(w,h))

img2=cv2.resize(crop\_image1(img[:,:,1]),(w,h))

img3=cv2.resize(crop\_image1(img[:,:,2]),(w,h))

*# print(img1.shape,img2.shape,img3.shape)*

img[:,:,0]=img1

img[:,:,1]=img2

img[:,:,2]=img3

return img

*'''all of these do not work'''*

def crop\_image2(image,threshold=5):

if len(image.shape) == 3:

flatImage = np.max(image, 2)

else:

flatImage = image

assert len(flatImage.shape) == 2

rows = np.where(np.max(flatImage, 0) > threshold)[0]

if rows.size:

cols = np.where(np.max(flatImage, 1) > threshold)[0]

image = image[cols[0]: cols[-1] + 1, rows[0]: rows[-1] + 1]

else:

image = image[:1, :1]

return image

def crop\_image3(image):

mask = image > 0

*# Coordinates of non-black pixels.*

coords = np.argwhere(mask)

*# Bounding box of non-black pixels.*

x0, y0 = coords.min(axis=0)

x1, y1 = coords.max(axis=0) + 1 *# slices are exclusive at the top*

*# Get the contents of the bounding box.*

cropped = image[x0:x1, y0:y1]

return cropped

def crop\_image4(image):

\_,thresh = cv2.threshold(image,1,255,cv2.THRESH\_BINARY)

contours,hierarchy = cv2.findContours(thresh,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_SIMPLE)

cnt = contours[0]

x,y,w,h = cv2.boundingRect(cnt)

crop = image[y:y+h,x:x+w]

return crop

def circle\_crop(img, sigmaX=10):

*"""*

*Create circular crop around image centre*

*"""*

img = cv2.imread(img)

img = crop\_image\_from\_gray(img)

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

height, width, depth = img.shape

x = int(width/2)

y = int(height/2)

r = np.amin((x,y))

circle\_img = np.zeros((height, width), np.uint8)

cv2.circle(circle\_img, (x,y), int(r), 1, thickness=-1)

img = cv2.bitwise\_and(img, img, mask=circle\_img)

img = crop\_image\_from\_gray(img)

img=cv2.addWeighted ( img,4, cv2.GaussianBlur( img , (0,0) , sigmaX) ,-4 ,128)

return img