VIETNAMNATIONALUNIVERSITY, HANOI
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Bùi Danh Hưng
K64-CACLC1

IMAGE PROCESSING

“DETECT THE DIFFERENCES BETWEEN 2 GIVEN IMAGE”

Source Code:
https://github.com/hung117/DetectDifferences.git

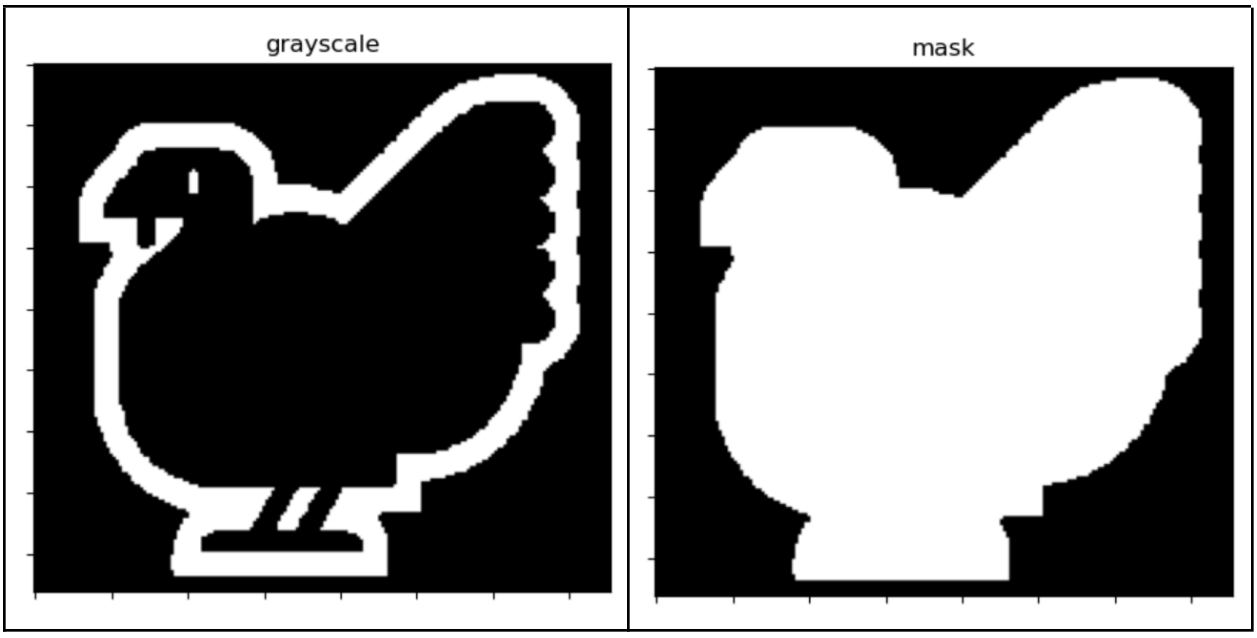
Approaches

Image Generation

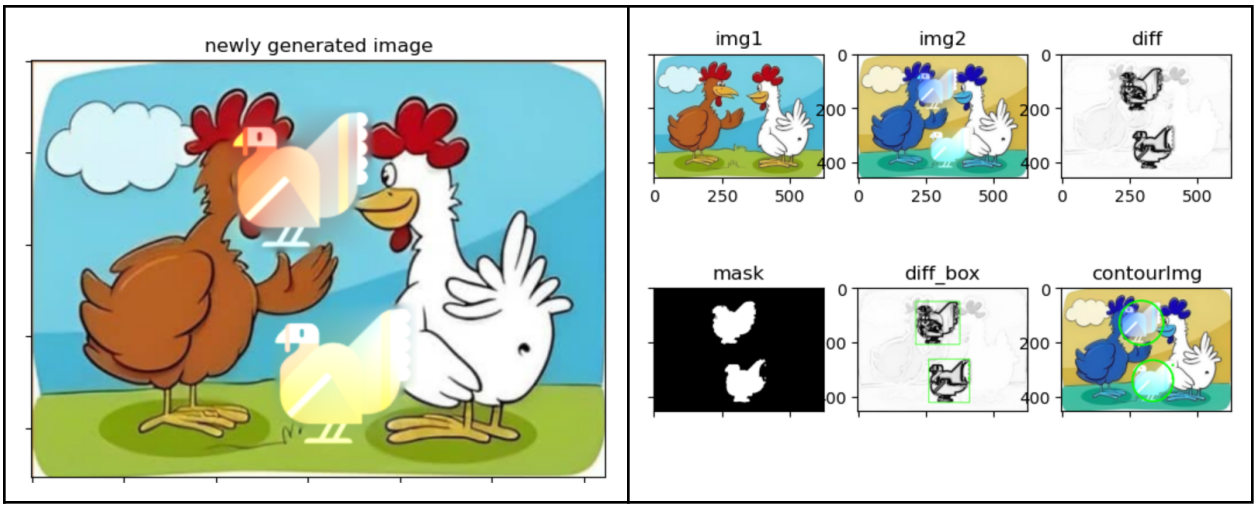
Using cv2.seamlessClone() to add new image into the original image, hence creating new one. In this project, I add them randomly.

```
number_of_turkeys = 2
for i in range(number_of_turkeys):
    x = int(random.randint(40, width-40))
    y = int(random.randint(40, height-50))
    image1 = cv2.seamlessClone(smol_turkey, image1, binMask, [x,y], cv2.NORMAL_CLONE)
```

In order for seamlessClone() to performs, a bin mask is required, using grayscale and threshold, I achieved this following mask for the turkey:



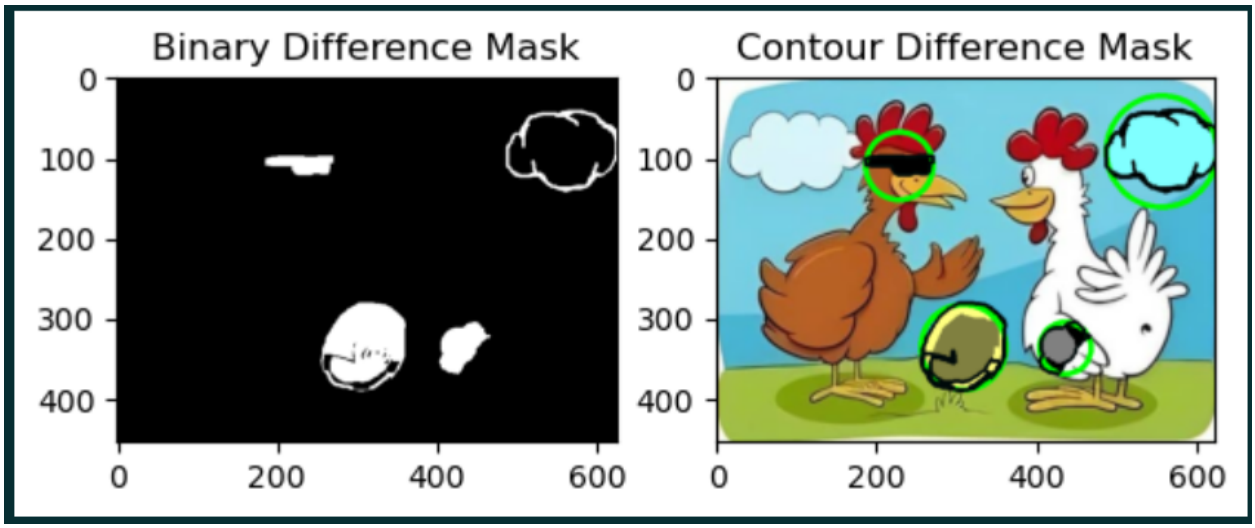
Generated Image and Different detection still works properly:



Different Detection

Approaching this particular problem, I have 2 method:

- Using the differences in value (range 0-255) between 2 image, and detected edges from Canny edge to get grayscale mask and improve it with threshold. This mask then used to get Contours and draw the output as well as gather game data accordingly.



To reduce noise and take every differences into account, I set threshold value to 0, any change in pixel value in the second image shall be put into the binMask.

```
# get Bit Mask

bin_threshold = 0

binMask = Conv_hsv_Gray > bin_threshold #forming the binary image

binary_img = binMask
binary_img.dtype='uint8' # convert binary_img in type of true false to decimal 0-1 array
binary_img = np.array(binary_img*255,dtype=np.uint8) # convert binary_img in type of true false to decimal 0-255 a
mask = binMask
```

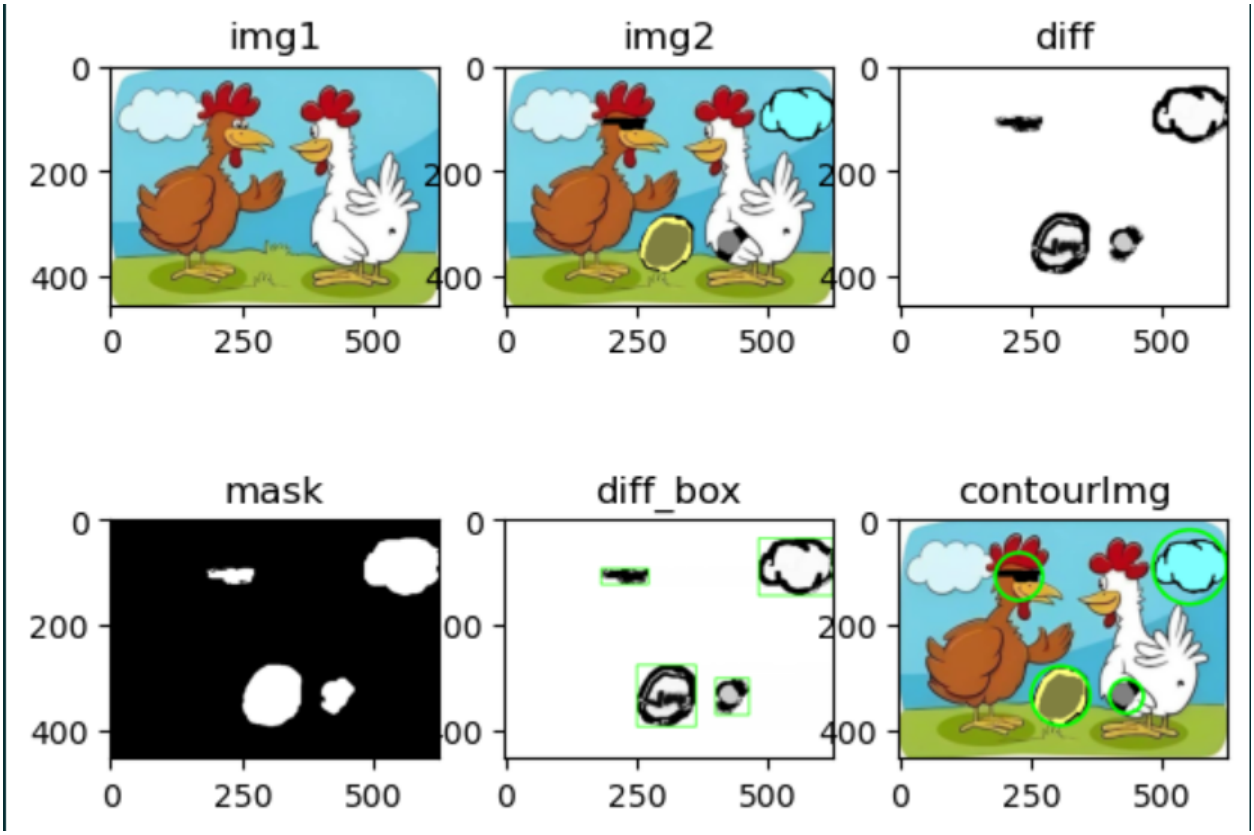
Select and draw contours:

```python
for contour in contours:
    area = cv2.contourArea(contour)
    if area > 20:   # get rid of contours that too small (if exist)
        x,y,w,h = cv2.boundingRect(contour)
        # cv2.rectangle(contourImg,(x,y),(x+w,y+h),(0,255,0),5)
        cx = x+int(w/2)
        cy =y+ int(h/2)
        center_coordinates = (cx,cy)
        print(cx,cy)
        radius = int(w/2)
        cv2.circle(contourImg, center_coordinates, radius, (0,255,0), 5)
cv2.drawContours(contourImg, contours, -1,  (0, 10, 10), 3) # -1 signifies drawing all contours
```

- Using SIMM, this method calculates the similarity between 2 image, in this simple case, it show similar result.

```python
# convert to grayscale:
img1_gray = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
img2_gray = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

# Compute SSIM between the 2 images
(score, diff) = SIMM(img1_gray, img2_gray, full=True)
diff = (diff * 255).astype("uint8") # diff is True-False rn, cvrt it to int (range 0-255)
diff_box = cv2.merge([diff, diff, diff]) # allow it to have 3 channel for coloring purpose
```



Trial And Errors

Before implementing those above methods, I tried to get the differences based solely on edge detection:

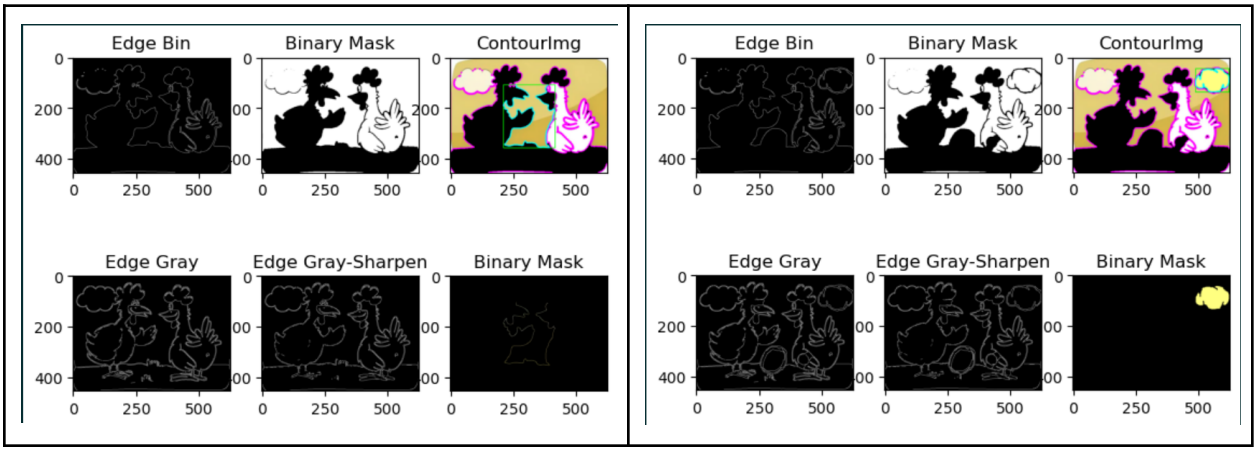
```
# Denoise
deNoise = cv2.bilateralFilter(gray,9,75,75) # blur but keeping edge
deNoise = cv2.medianBlur(deNoise,5)
deNoise = cv2.bilateralFilter(gray,9,75,75)

# Get Bin Mask
binary_img = deNoise > bin_threshold #forming the binary image
binary_img.dtype='uint8' # convert binary_img in type of true false to decimal array
binMask = binary_img

binary_img.dtype='uint8' # convert binary_img in type of true false to decimal 0-1 array
binary_img = np.array(binary_img*255,dtype=np.uint8) # convert binary_img in type of true false to decimal 0-2

# Get Edge
## from bin
# edged = cv2.Canny(binary_img, 30, 200)
blurBin = cv2.GaussianBlur(binary_img,(3,3), 0, 0)
edged = cv2.Canny(blurBin, 100, 200)
## from gray
edge_gray = cv2.Canny(gray, 200, 255)
## sharpen the grayscale even more and get edge via canny-gray
sharpen_filter=np.array([[-1,-1,-1],
                         [-1,9,-1],
                         [-1,-1,-1]])
### applying kernels to the input image to get the sharpened image
sharp_image=cv2.filter2D(deNoise,-1,sharpen_filter)
edge_gray_sharp = cv2.Canny(sharp_image, 200, 255)
```

The result is not desirable, in the attempt of reducing noise, it appears there are even more noise in the sharpen image:



Problems

These methods still have it’s drawback, such as the 2 image have to be in the exact dimension. With 2 different dimension images, reshape() does not do the trick, what we get instead would be this unusable mess of a binMask: