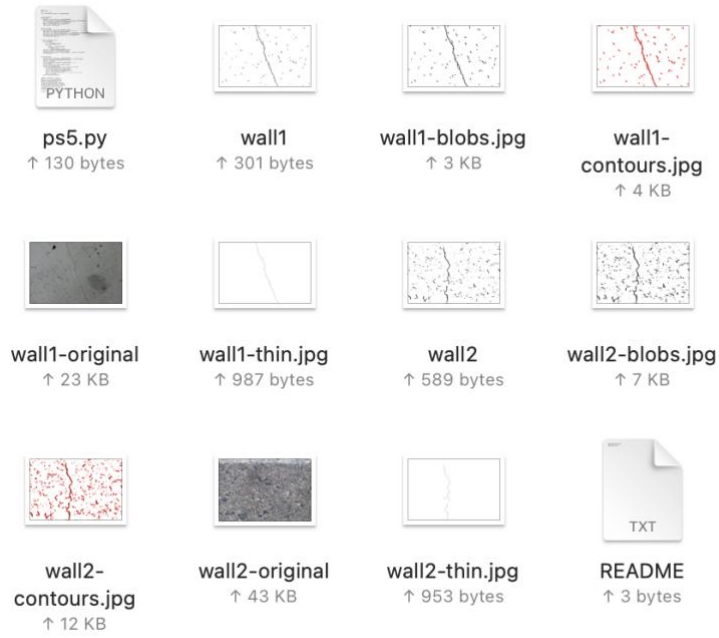
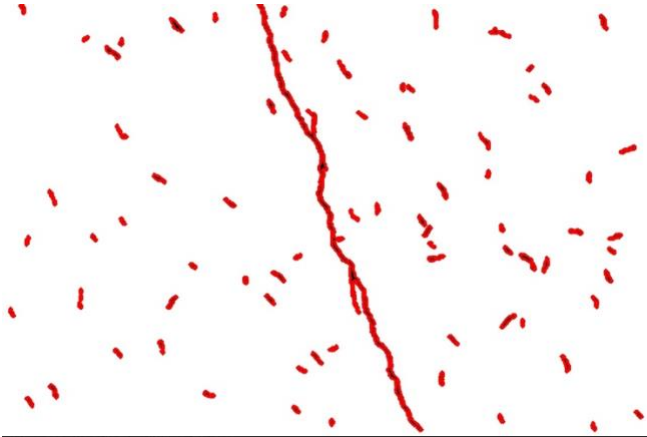
PS5 Report



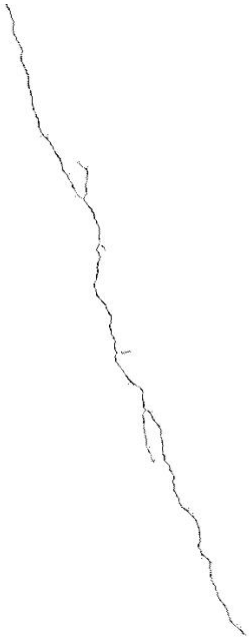
```
# erode and dilate
def blobs(img):
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    thresh, img_binary = cv2.threshold(img_gray, 128, 255, cv2.THRESH_BINARY)
    k_e = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
    img_b = cv2.erode(img_binary, k_e, iterations=2)
    img_b = cv2.dilate(img_b, k_e, iterations=1)
    return img_b  # binary image
```

For the wall I used cv2.erode and cv2.dilate to close the black pixels and strengthen the crack. I eroded it 2 times and dilate it one time to get best performance.



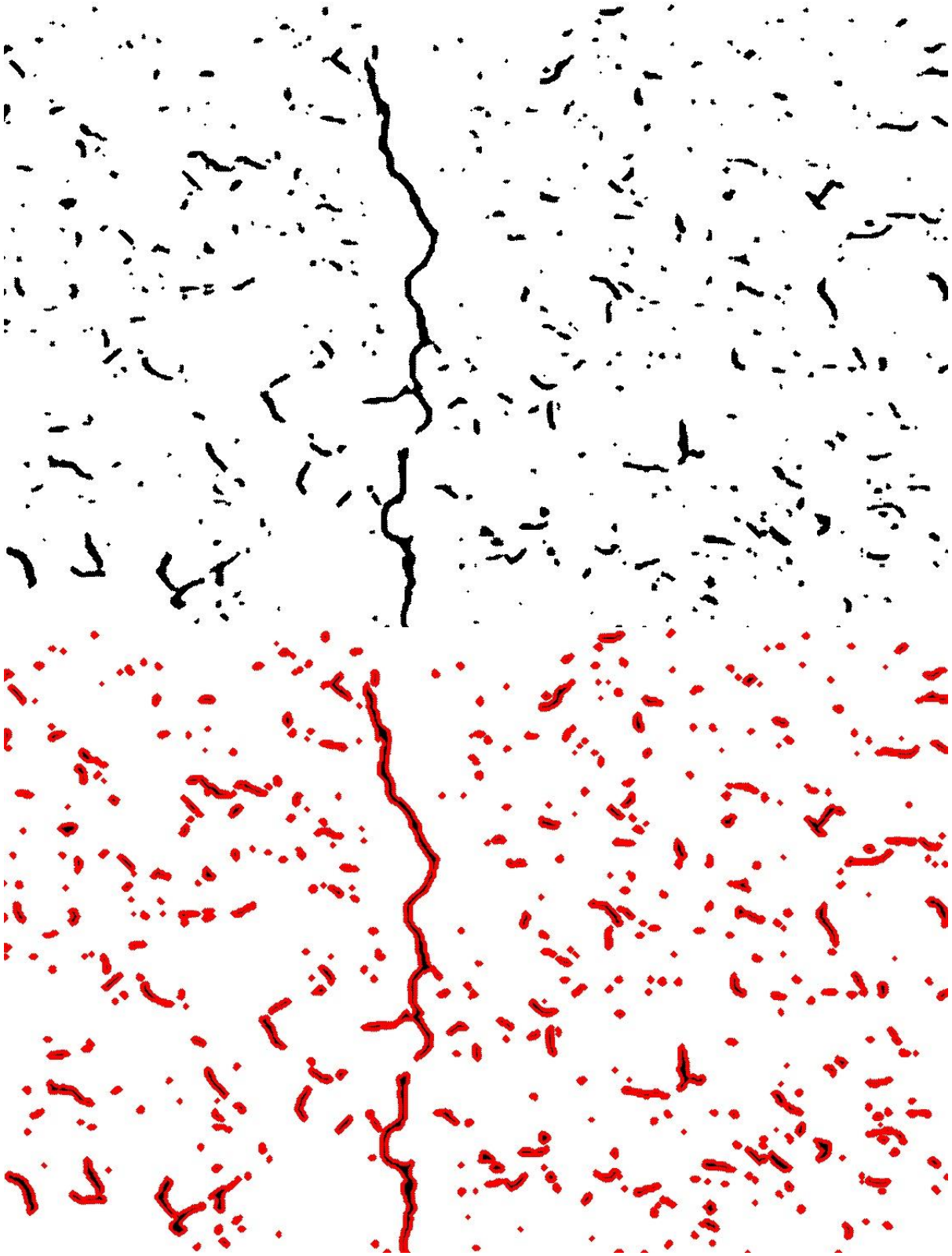
```
# contour tracing
def con_trace(img):  # will take eroded blobs image
    img_contour = img.copy()  # create an image for contour use
    img_contour_inv = cv2.bitwise_not(img_contour)
    cont, hier = cv2.findContours(img_contour_inv, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    img_contour_rgb = cv2.cvtColor(img_contour, cv2.COLOR_GRAY2BGR)
    cv2.drawContours(img_contour_rgb, cont, -1, (0, 0, 255), 3)
    return img_contour_rgb, cont
```

First, we need to change the image into binary format. Then we use cv2.findcontours and cv2.drawcontours command to draw the contour of every possible crack.



```python
def detect_crack(img, contour):
    contour_candidate = []
    for i in contour:
        p = cv2.arcLength(i, True)
        area = cv2.contourArea(i)
        if p > 0:
            roundness = 4*np.pi*area/p**2
            if roundness < 0.1:
                contour_candidate.append(i)
    img_white = np.zeros(img.shape, dtype=np.uint8)
    img_white.fill(255)  # create RGB white image
    cv2.drawContours(img_white, contour_candidate, -1, (0,0,0), cv2.FILLED)
    return img_white
def thinning(img):
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    thresh, img_bw = cv2.threshold(img_gray, 127, 255, cv2.THRESH_BINARY)
    img_inv = cv2.bitwise_not(img_bw)# take binary
    # Kernel: 4 neighbor
    k_e = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
    # Target image
    thin = np.zeros(img_inv.shape, dtype=np.uint8)
    # repeat until no white area
    while cv2.countNonZero(img_inv) != 0:  # loop until no noise
        er = cv2.erode(img_inv, k_e)# shrink noise
        # OPEN: erosion then dilation (remove noise)
        op = cv2.morphologyEx(er, cv2.MORPH_OPEN, k_e)
        subset = er-op  # only noise left
        thin = cv2.bitwise_or(subset, thin)  # map noise to black image
        img_inv = er.copy()
    return cv2.bitwise_not(thin)
```

In the last step, I try to detect the crack by measuring the roundness of the contour by looping them one by one. I choose the roundness as my threshold and I set it as low as possible to 0.1. The crack will have lowest roundness thus it is easily to identify. Finally, we also need to thin the crack to output the image that clearly locate the crack.



Process the second wall image with same procedures mentioned above.