Report

Task1

For task1, I use two lists to record threshold and epoch, the main algorithm has explained in Assignment Specification and some comments below.

```
def task1(img):
    # choose epsilon = 0.001, 0.01 and 0.001 won't lead to different results for my algorithm
    e = 0.001
    # choose a random number
    t = random.randint(0, 255)
    # initial number of test is 0
    time = 0
    # create an empty list for threshold
    t_l = []
    # create an empty list for test number
    time_l = []
    while True:
        # update t_l and time_l
        t_l.append(t)
        time_l.append(time)
        # compute u0 and u1
        bi = img >= t
        bi_inv = img < t
        fore_pix = np.sum(bi)
        back_pix = np.sum(bi_inv)
        if fore_pix == 0:
            break
        if back_pix == 0:
            continue
        w0 = fore_pix / img.size
        u0 = np.sum(img * bi) / fore_pix
        w1 = back_pix / img.size
        u1 = np.sum(img * bi_inv) / back_pix
        # if the different betweem threshold and the mean of the means less than epsilon, then jump out,else update the mean of the means
        if abs(t - (u0 + u1) / 2) <= e:
            break
        else:
            t = (u0 + u1) / 2
        time = time + 1
    # traversal image if pixel > threshold, pixel = 0(rice kernel) , else pixel = 255(background)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i][j] >= t:
```

```
                img[i][j] = 0
            else:
                img[i][j] = 255
    return img, t, time_l, t_l
```

Task2

For task2, the first step is to do median filter, then first pass and second pass. But my coding is not strictly obey the order of two-pass algorithm, it slightly change the order. I used two dictionaries, the first one, key is label, value is minimal key among its 8 neighbors and itself. The second dictionary exchange key and value, it will reversely traversal(it is very necessary to reversely traversal, or some labels should be included in minimal label will not belong to minimal which mean more rice will be created) the dictionary to record final minimal label and all other labels should become the minimal label.

```
def task2(img):
    # 5*5 MedianBlur: 5*5 filter have better performance than 3*3 filter
    # Create a new empty (img.shape[0]+4 * img.shape[1]+4) image and fill it with white
    new_img = []
    for i in range(img.shape[0] + 4):
        new_img.append([])
        for j in range(img.shape[1] + 4):
            new_img[i].append(255)
    new_img = np.array(new_img, dtype=np.uint8)
    # leave new_img[:2,:], new_img[-2:,:], new_img[:,:2], new_img[:,-2:] stay unchanged, the rest replace with input image.
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            new_img[i + 2][j + 2] = img[i][j]
    # median filter
    c = copy.copy(new_img)
    for i in range(2, new_img.shape[0] - 2):
        for j in range(2, new_img.shape[1] - 2):
            a = sorted([c[i - 2][j - 2], c[i - 2][j - 1], c[i - 2][j], c[i - 2][j + 1], c[i - 2][j + 2],
                        c[i - 1][j - 2], c[i - 1][j - 1], c[i - 1][j], c[i - 1][j + 1], c[i - 1][j + 2],
                        c[i][j - 2], c[i][j - 1], c[i][j], c[i][j + 1], c[i][j + 2],
                        c[i + 1][j - 2], c[i + 1][j - 1], c[i + 1][j], c[i + 1][j + 1], c[i + 1][j + 2],
                        c[i + 2][j - 2], c[i + 2][j - 1], c[i + 2][j], c[i + 2][j + 1], c[i + 2][j + 2]])
            new_img[i][j] = a[12]
    # First pass
    # Copy the filtered image to record all label
    img_c = copy.copy(new_img)
    t = 1
    # dictionary record label(key = value = label)
    d = {}
```

```
    d[t] = t
    # if pixel is not background, then if its neighbor is not all background, then the pixel is minimal of its neighbor, else the pixel = current label and update label and label dictionary
    for i in range(1, img_c.shape[0] - 1):
        for j in range(1, img_c.shape[1] - 1):
            if img_c[i][j] < 255:
                if img_c[i - 1][j - 1] < 255 or img_c[i - 1][j] < 255 or img_c[i - 1][j + 1] < 255 or img_c[i][j - 1] < 255:
                    a = [img_c[i - 1][j - 1], img_c[i - 1][j], img_c[i - 1][j + 1], img_c[i][j - 1]]
                    a = [k for k in a if k < 255]
                    img_c[i][j] = min(a)
                else:
                    img_c[i][j] = t
                    t = t + 1
                    d[t] = t
    # Second pass
    # Go through the pixel's 8 neighbors, key is label, value is the minimal label for all its 8 neighbors' label and its label
    for i in range(1, img_c.shape[0] - 1):
        for j in range(1, img_c.shape[1] - 1):
            if img_c[i][j] < 255:
                a = [img_c[i - 1][j - 1], img_c[i - 1][j], img_c[i - 1][j + 1],
                     img_c[i][j - 1], img_c[i][j], img_c[i][j + 1],
                     img_c[i + 1][j - 1], img_c[i + 1][j], img_c[i + 1][j + 1], d[img_c[i][j]]]
                d[img_c[i][j]] = min([k for k in a if k < 255])
    # Create a new dictionary, use previous dictionary's value to be key, key to be value
    new_d = dict()
    for keys, values in d.items():
        new_d[values] = []
    for keys, values in d.items():
        new_d[values].append(keys)
    key = sorted(new_d.keys())

    # From tail to head, traversal the new dictionary, if key[i] is included in key[i-1]'s value, then add key[i] 's value to key[i-1]'s value and delete key[i]. Then the new dictionary's keys is all minimal label, value is the minimal label 's 8 neighbors' label.
    key.reverse()
    for i in key:
        for j in new_d[i]:
            if j != i and j in new_d.keys():
                new_d[i].extend(new_d.pop(j))
    new_d.pop(sorted(new_d.keys())[-1])
    nums = len(new_d)
    return new_img, nums, new_d, img_c
```

Task 3

For task3, new dictionary record minimal label and its 8 neighbors' label list, for all minimal label, sum the number of pixel in label recorded image equal to label in its label list, and use a dictionary named dd to record all these minimal keys and its number of pixel, if the number of pixel > min_area, then good rice, else damaged rice.

```
def task3(new_d,img_c,min_area):
    dd = dict()
    l = len(new_d)
    for key,value in new_d.items():
        dd[key] = 0
        for i in value:
            dd[key] += np.sum(img_c==i)
    s = []
    ll = []
    for key,value in dd.items():
        if dd[key] >= min_area:
            s.append(key)
    for i in s:
        ll.extend(new_d[i])
    for i in range(img_c.shape[0]):
        for j in range(img_c.shape[1]):
            if img_c[i][j] in ll:
                img_c[i][j] = 0
            else:
                img_c[i][j] = 255
    re = round((l-len(s))/l,4)
    return re,img_c
```

Finally, my code run about 5 min in my computer, please wait for a few minutes.

Thanks!