HOMEWORK 1B
ECE/CS 8690 2302 Computer Vision

Experiments with Color Edge Detection

Xuanbo Miao

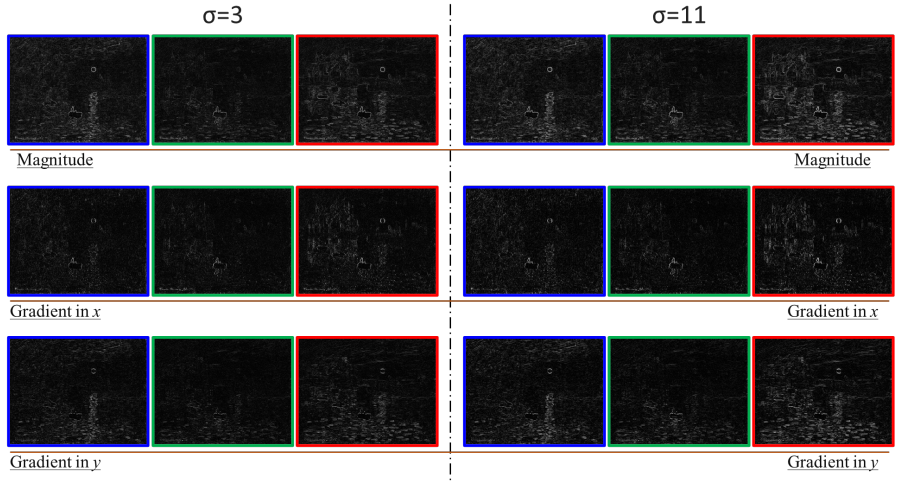
14422044

xmiao@mail.missouri.edu

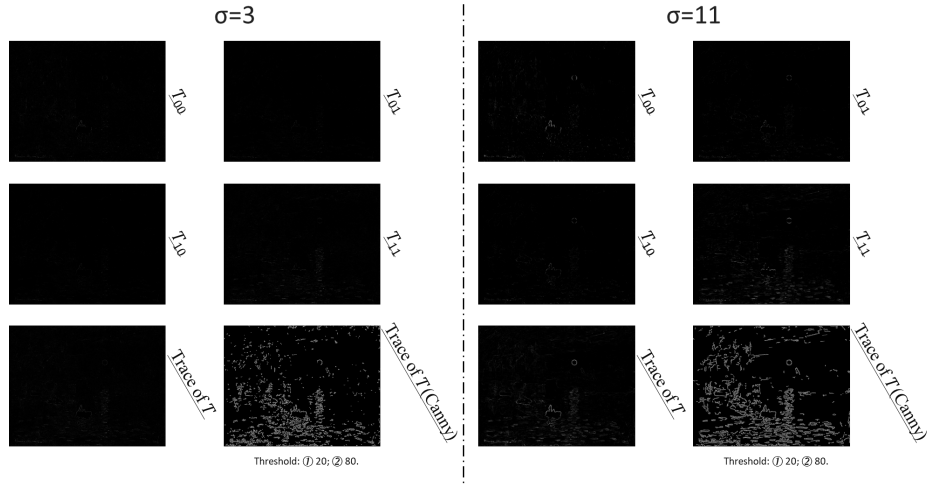
Abstract

The main goal of this assignment is to perform edge detection on color images using 2D color structure tensor and to compare the results with traditional grayscale edge detection. The assignment requires a deep understanding of image processing concepts and techniques such as image gradients, convolution, and Gaussian filtering. The results of the analysis will help us understand the limitations of traditional grayscale edge detection and the advantages of using 2D color structure tensor for edge detection in color images.

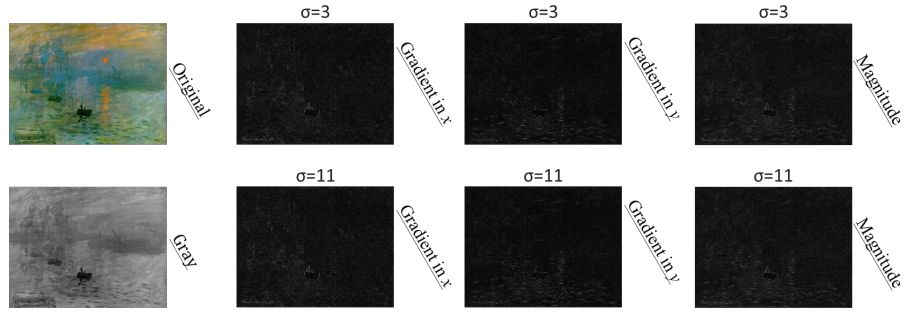
Task 1+2 ①



Task 3+4 ②



Task 5 ③



Introduction & My Code

1. Load the color image using OpenCV library in Python.
2.Compute I_x, the image gradient in x for the R, G, B channels, by convolving each channel with the Sobel filter in x direction.
3. Compute I_y, the image gradient in y for the R, G, B channels, by convolving each channel with the Sobel filter in y direction.
4. Compute 2D color structure tensor by taking the outer product of I_x and I_y, i.e. then calculate $Jc(x, y)$.
5. Compute trace of 2D color structure tensor by summing the diagonal elements.
6. Convert the color image to grayscale using the cv2Color function in OpenCV.
7.Display the computed I_x, I_y, and the gradient magnitude.
8.Repeat the above processes using another σ.

CVSpring2023_Miao_Xuanbo_Assignment2_code.py

CVSpring2023_Miao_Xuanbo_Assignment2_function.py > ...

```python
import cv2
import numpy as np
kernel_scale=11

def gradient_calc(img_data):
    sobelx = cv2.Sobel(img_data, cv2.CV_64F, 1, 0, ksize=kernel_scale)
    sobely = cv2.Sobel(img_data, cv2.CV_64F, 0, 1, ksize=kernel_scale)
    img_grad = np.sqrt(sobelx**2 + sobely**2)
    sobelx= np.uint8(255*abs(sobelx)/np.max(np.abs(sobelx)))
    sobely= np.uint8(255*abs(sobely)/np.max(np.abs(sobely)))
    img_grad= np.uint8(255*img_grad/np.max(img_grad))

    return img_grad, sobelx, sobely



def compute_color_structure_tensor(img):
    rows, cols, channels = img.shape

    # Convert the image to a float type
    img = img.astype(np.float32)

    # Compute the gradient in x and y directions
    sobel_x = cv2.Sobel(img, cv2.CV_32F, 1, 0, ksize=kernel_scale)
    sobel_y = cv2.Sobel(img, cv2.CV_32F, 0, 1, ksize=kernel_scale)

    # Initialize the 2D Color Structure Tensor
    color_structure_tensor_colored = np.zeros((rows, cols, 3, 2, 2))

    # Loop over the color channels (R, G, B)
    for c in range(channels):
        # Compute the elements of the 2D Color Structure Tensor for this color channel
        color_structure_tensor_colored[..., c, 0, 0] = sobel_x[..., c] ** 2
        color_structure_tensor_colored[..., c, 0, 1] = sobel_x[..., c] * sobel_y[..., c]
        color_structure_tensor_colored[..., c, 1, 0] = color_structure_tensor_colored[..., c, 0, 1]
        color_structure_tensor_colored[..., c, 1, 1] = sobel_y[..., c] ** 2
    color_structure_tensor=np.sum(color_structure_tensor_colored,axis=2)

    return color_structure_tensor

def compute_and_display_color_structure_tensor(img):
    color_structure_tensor = compute_color_structure_tensor(img)

    # Compute the eigenvalues and eigenvectors of the 2D Color Structure Tensor
    eigenvalues, eigenvectors = np.linalg.eig(color_structure_tensor[..., 0, :, :])

    # Normalize the eigenvectors
    eigenvectors = eigenvectors / np.linalg.norm(eigenvectors, axis=1, keepdims=True)

    # Plot the eigenvectors on top of the input image
    img_with_eigenvectors = np.uint8(img.copy())
    for y in range(img.shape[0]):
        for x in range(img.shape[1]):
            x2 = int(x + eigenvectors[y, x, 0] * 100)
            y2 = int(y + eigenvectors[y, x, 1] * 100)
            cv2.line(img_with_eigenvectors, (x, y), (x2, y2), (0, 0, 255), 1)

    cv2.imshow("2D Color Structure Tensor", img_with_eigenvectors)
    cv2.waitKey(0)
```

CVSpring2023_Miao_Xuanbo_Assignment2_function.py

CVSpring2023_Miao_Xuanbo_Assignment2_code.py > ...

```python
import numpy as np
from datetime import datetime
import time
import matplotlib.pyplot as plt
import cv2
from CVSpring2023_Miao_Xuanbo_Assignment2_function import *
current_time = datetime.now().strftime("%H:%M:%S")

print("Current Time =", current_time)


img_orig = cv2.imread('./Claude_Monet,_Impression,_soleil_levant,_1872.jpg')

img_raw_b, img_raw_g, img_raw_r  = cv2.split(img_orig)

img_grad_b, img_grad_b_x, img_grad_b_y=gradient_calc(img_raw_b)
img_grad_g, img_grad_g_x, img_grad_g_y=gradient_calc(img_raw_g)
img_grad_r, img_grad_r_x, img_grad_r_y=gradient_calc(img_raw_r)

color_structure_tensor=compute_color_structure_tensor(img_orig).astype(np.float32)
color_structure_tensor_trace=np.zeros(color_structure_tensor.shape[0:2])
color_structure_tensor_trace=np.trace(color_structure_tensor,axis1=2,axis2=3)

color_structure_tensor_maxvalue=np.max(np.abs(color_structure_tensor))

color_structure_tensor_00=np.uint8(np.abs(color_structure_tensor[:,:,0,0]/color_structure_tensor_maxvalue*255))
color_structure_tensor_01=np.uint8(np.abs(color_structure_tensor[:,:,0,1]/color_structure_tensor_maxvalue*255))
color_structure_tensor_10=np.uint8(np.abs(color_structure_tensor[:,:,1,0]/color_structure_tensor_maxvalue*255))
color_structure_tensor_11=np.uint8(np.abs(color_structure_tensor[:,:,1,1]/color_structure_tensor_maxvalue*255))
color_structure_tensor_trace=np.uint8(255*color_structure_tensor_trace/np.max(color_structure_tensor_trace))

color_structure_tensor_trace_cannied=cv2.Canny(color_structure_tensor_trace,20,80)

img_gray = cv2.cvtColor(img_orig, cv2.COLOR_BGR2GRAY)
img_gray_grad,img_gray_grad_x,img_gray_grad_y=gradient_calc(img_gray)

image_datalist=[ img_orig , img_grad_b , img_grad_b_x , img_grad_b_y , img_grad_g , img_grad_g_x , img_grad_g_y , img_grad_r , img_grad_r_x , img_grad_r_y , color_structure_tensor_00 , color_structure_tensor_01 , color_structure_tensor_10 , color_structure_tensor_11 , color_structure_tensor_trace , color_structure_tensor_trace_cannied , img_gray , img_gray_grad , img_gray_grad_x , img_gray_grad_y ]
image_namelist=
['img_orig','img_grad_b','img_grad_b_x','img_grad_b_y','img_grad_g','img_grad_g_x','img_grad_g_y','img_grad_r','img_grad_r_x','img_grad_r_y','color_structure_tensor_00','color_structure_tensor_01','color_structure_tensor_10','color_structure_tensor_11','color_structure_tensor_trace',
'color_structure_tensor_trace_cannied','img_gray','img_gray_grad','img_gray_grad_x','img_gray_grad_y']

if 1:
    for image_id in range(len(image_datalist)):
        cv2.imshow(image_namelist[image_id], image_datalist[image_id]);
        #cv2.resizeWindow(image_namelist[image_id], 808, 1051);
        cv2.imwrite('./img/'+image_namelist[image_id]+'.jpg',image_datalist[image_id])

cv2.waitKey(0)
```

Task 1+2. Compute and display I_x and I_y, image gradient in x for R, G, B channels

σ=3

σ=11

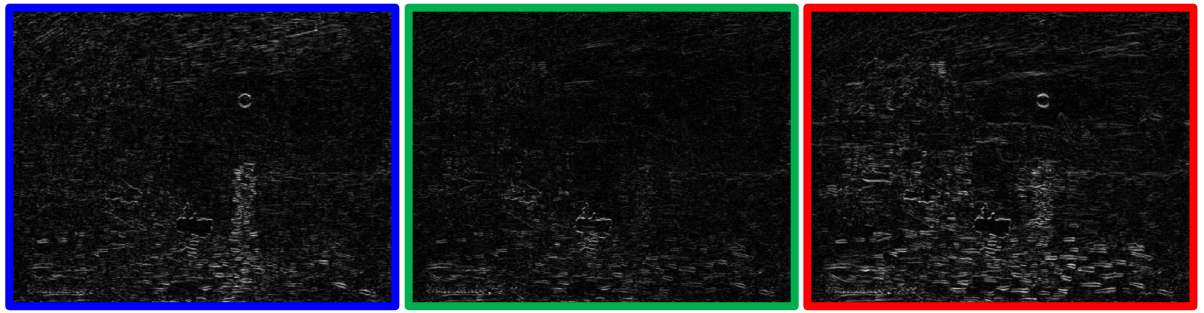
Magnitude

Magnitude

Gradient in x

Gradient in x

Gradient in y



Gradient in y

Task 3+4. Compute and display the elements 2D color structure tensor and its trace.

σ=3

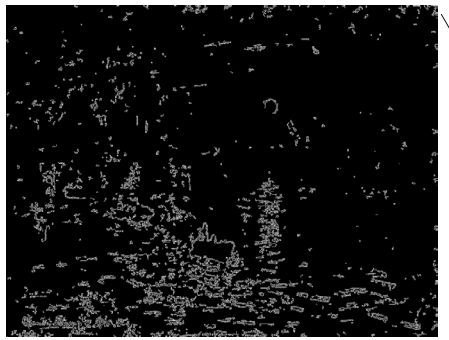
T_{00}

T_{01}

T_{10}

T_{11}

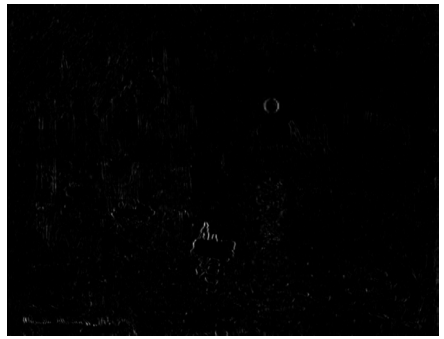
Trace of T



Trace of T (Canny)

Threshold: ① 20; ② 80.

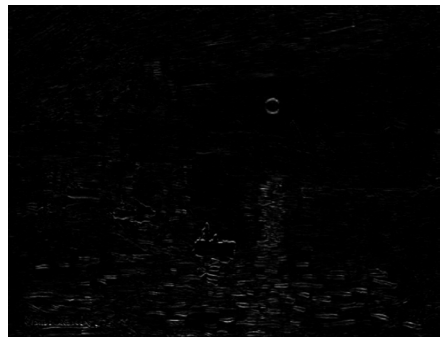
σ=11



T_{00}

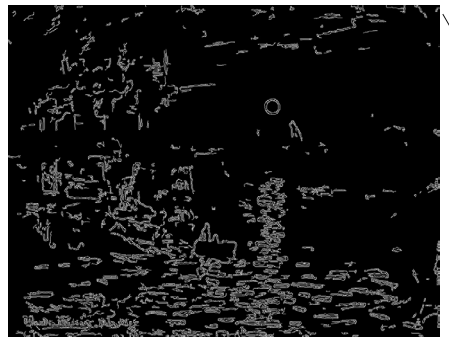
T_{01}

T_{10}



T_{11}

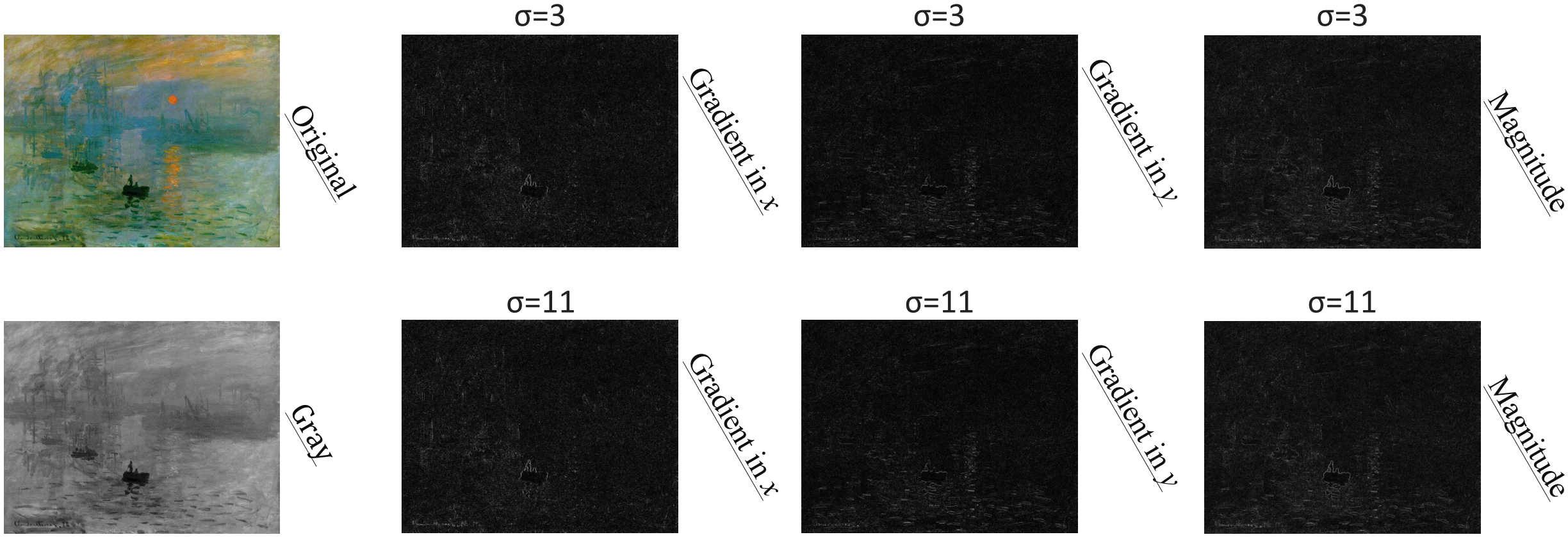
Trace of T



Trace of T (Canny)

Threshold: ① 20; ② 80.

Task 5. Convert original color image to grayscale. Compute I_x, I_y for the grayscale image. Display gradient magnitude.



Interpretation & Discussion

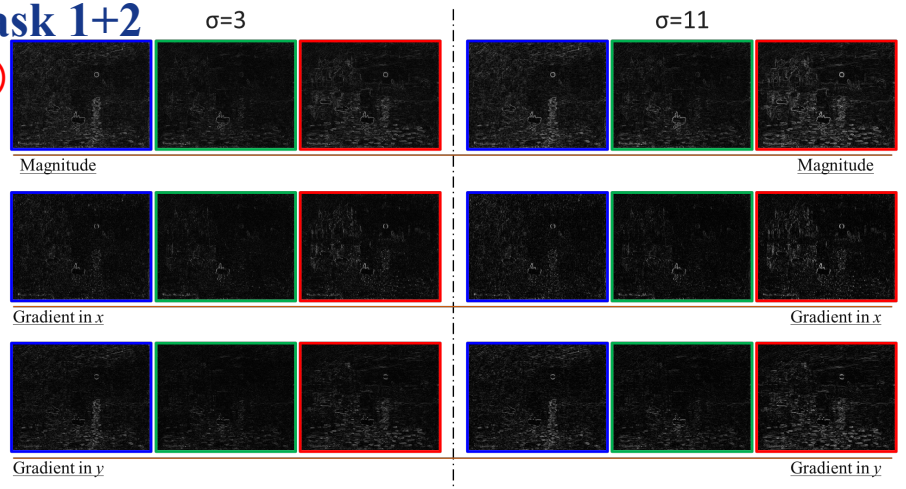
① In *Task 1+2*, I calculated the gradient in x and y direction, and the gradient magnitude. We noticed that there may be differences in the results (x, y, or magnitude). The reason is the gradient magnitude in each direction depends on **the edge orientation in that direction**. For example, if the edge is oriented vertically, the gradient magnitude in the x direction would be low compared to the y direction. This difference in gradient magnitude in different directions helps in accurately detecting edges and preserving edge orientation information. For gradient magnitude, it is likely to be the combination of x and y gradient, which is more generally to be used.

② In *Task 3+4*, I calculated the 2D color structure tensor, its trace and I detected the image edge using the trace. It worth to note that, if we calculated the image gradient using different Gaussian kernel sizes, there will be differences in the results because the size of the Gaussian kernel **affects the scale of edges that are detected**. A smaller Gaussian kernel size focuses on detecting **fine details** in the image and is more sensitive to small edges. A larger Gaussian kernel size, on the other hand, focuses on detecting **prominent structures** in the image and is more sensitive to larger edges. By changing the size of the Gaussian kernel, we can control the scale of edges that are detected in the image and thus preserve different levels of information. The save difference **also happens on *Task 1+2***.

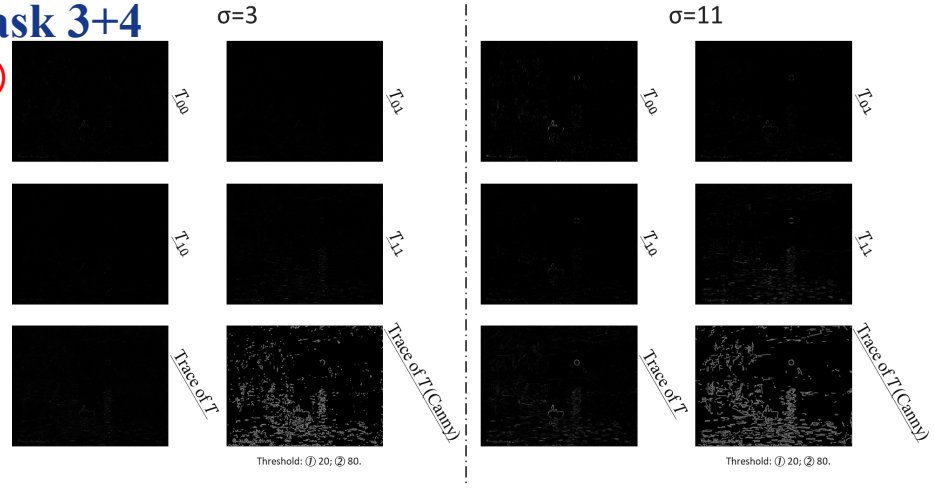
③ In *Task 3+4*, I calculated corresponding matters on gray image. The conversion of a color image to grayscale is a common approach for edge detection in computer vision. However, this approach has limitations when dealing with images that contain noticeable edges between **iso-luminant colors**, or colors that have the same luminance. These edges **are not detected** by grayscale edge operators because the grayscale image **only provides information about the luminance**, or brightness, of the image and does not preserve the color information. To overcome this limitation, one approach is to combine the outputs of grayscale detectors run on each color band separately. However, this approach can result in the **cancellation of the signed gradients** if they are summed up, which leads to an inaccurate representation of the edges in the image.

In conclusion, while the conversion of a color image to grayscale is a simple and effective approach for edge detection, it may not always provide accurate results when dealing with images that contain noticeable edges between iso-luminant colors. Other approaches, such as combining the outputs of grayscale detectors or detecting edges independently in each color band, are necessary to accurately preserve and detect these edges.

Task 1+2 ①



Task 3+4 ②



Task 5 ③