DIP homework 2

4109061012 陳柏翔

Problem 8：

```
# 4109061012 B.S.Chen
import cv2
import numpy as np
import matplotlib.pyplot as plt

IMAGE_PATH = "homework_2/src/Fig0308(a)(fractured_spine).tif"
TARGET_PATH = "homework_2/result"
I_MAX = 2**8

def histogram(img: np.ndarray, value_range: tuple, normalize: bool) -> np.ndarray:
    """ Problem 8(a): Compute histogram of input image """
    img = img.ravel()
    hist = np.zeros(value_range[1] - value_range[0])
    for i in range(img.shape[0]):
        hist[img[i]] += 1
```

⟸ Problem 8(a)

```
    if normalize:
        hist = hist / img.shape[0]
    return hist

def hist_equalize(img: np.ndarray) -> tuple[np.ndarray, np.ndarray]:
    """ Histogram equalization """
    # Cumulated PDF of original image intensities
    hist = histogram(img, (0, I_MAX), normalize=True)
    cumsum_hist = np.cumsum(hist)
```

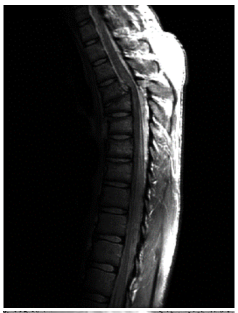
⟸ Problem 8(b)

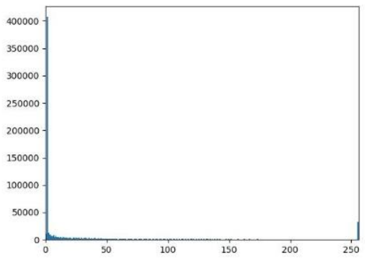
```
    # Transform
    new_img = np.empty_like(img)
    trans_func = np.empty(I_MAX)
    for i in range(I_MAX):
        intensity = round((I_MAX - 1) * cumsum_hist[i])
        new_img[img == i] = intensity
        trans_func[i] = intensity
    return new_img, trans_func
```

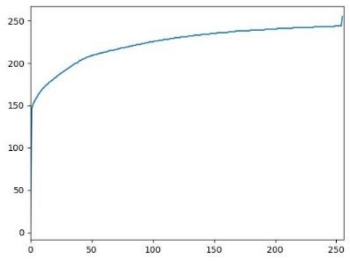
⟸ Problem 8(c)

```
if __name__ == "__main__":
    orig_img = np.asarray(cv2.imread(IMAGE_PATH, cv2.IMREAD_GRAYSCALE), dtype=np.uint8)
    new_img , trans_func= hist_equalize(orig_img)

    cv2.imwrite(TARGET_PATH + "/prob8_result_img.jpg", new_img)
    plt.hist(orig_img.ravel(), I_MAX, (0, I_MAX)), plt.xlim(left=0, right=I_MAX)
    plt.savefig(TARGET_PATH + "/prob8_orig_hist.jpg"), plt.clf()
    plt.hist(new_img.ravel(), I_MAX, (0, I_MAX)), plt.xlim(left=0, right=I_MAX)
    plt.savefig(TARGET_PATH + "/prob8_result_hist.jpg"), plt.clf()
    plt.plot(trans_func), plt.xlim(left=0, right=I_MAX)
    plt.savefig(TARGET_PATH + "/prob8_transform_func.jpg")
```

(Source)

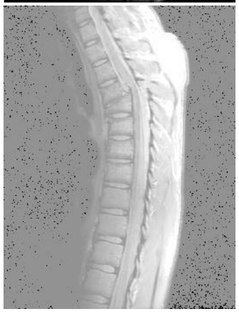


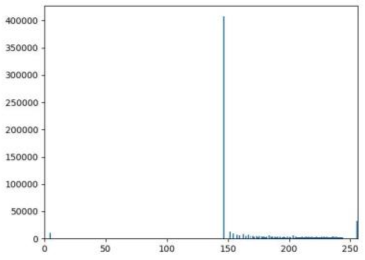




(Transformation function)

(Result)





Problem 9：

```
# 4109061012 B.S.Chen
import cv2
import numpy as np


IMAGE_PATH = "homework_2/src/Fig0338(a)(blurry_moon).tif"
TARGET_PATH = "homework_2/result"

SIZE = 3


def setting_filter() -> np.ndarray:
    i = 0
    filter = np.empty((SIZE, SIZE), dtype=np.int32)
    print(">> Please input values of the filter (use ',' to separate):")
    while i < SIZE**2:
        data = input().split(',')
        for value in data:
            filter[i // SIZE, i % SIZE] = int(value.replace(' ', ''))
            i += 1
    return filter
```

Problem 9(a)

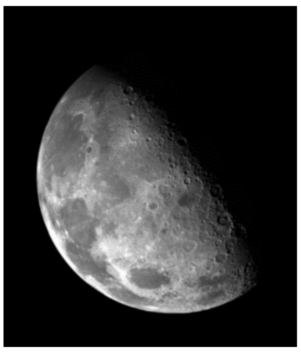
```
def filtering(img: np.ndarray, kernel: np.ndarray, coeff = 9.0) -> np.ndarray:
    """ Filtering an image with symmetric padding """
    s, pad = kernel.shape[0], kernel.shape[0] // 2
    new_img = np.empty_like(img)
    img = np.pad(img, ((pad, pad), (pad, pad)), "symmetric")

    for r in range(new_img.shape[0]):
        for c in range(new_img.shape[1]):
            new_img[r, c] = round(coeff * np.average(kernel * img[r : r+s, c : c+s]))
    return new_img

def Laplacian_filtering(img: np.ndarray, kernel: np.ndarray) -> np.ndarray:
    """ Enhancement using laplacian with user inputted filter """
    img = img.astype(dtype=np.int32)

    # Enhancing
    mask = filtering(img, kernel)
```

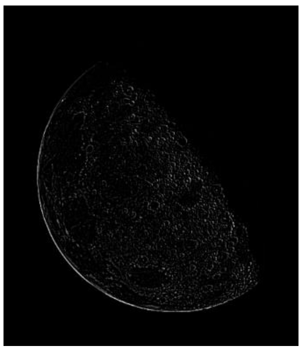
Problem 9(b)

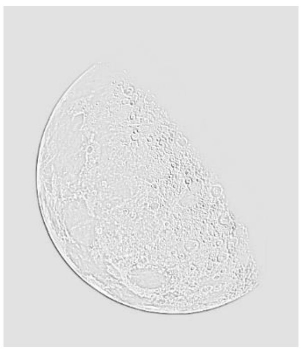
```
    scale_mask = (mask - mask.min()) / mask.max() * 255  # Eqs. (2.6-10), (2.6-11)
    img = img + mask

    img[img < 0], img[img > 255] = 0, 255
    mask[mask < 0], mask[mask > 255] = 0, 255
    scale_mask[scale_mask < 0], scale_mask[scale_mask > 255] = 0, 255
    return img.astype(np.uint8), mask.astype(np.uint8), scale_mask.astype(np.uint8)
```

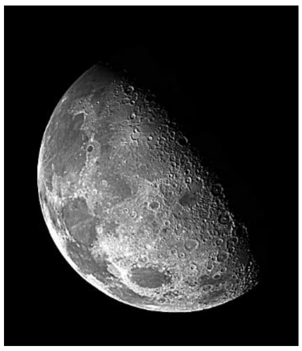
Problem 9(c)

```
if __name__ == "__main__":
    orig_img = np.asarray(cv2.imread(IMAGE_PATH, cv2.IMREAD_GRAYSCALE), dtype=np.uint8)
    kernel = setting_filter()

    print(">> start enhancing..")
    new_img, mask, scale_mask = Laplacian_filtering(orig_img, kernel)
    cv2.imwrite(TARGET_PATH + "/prob9_result_img.jpg", new_img)
    cv2.imwrite(TARGET_PATH + "/prob9_mask.jpg", mask)
    cv2.imwrite(TARGET_PATH + "/prob9_scale_mask.jpg", scale_mask)
    print(">> done")
```


(Source)


(mask)


(scaled mask)


(Result)

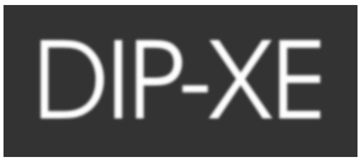
Problem 10：

```
# 4109061012 B.S.Chen
import cv2
import numpy as np


IMAGE_PATH = "homework_2/src/Fig0340(a)(dipxe_text).tif"
TARGET_PATH = "homework_2/result"
COEFF = 4.5  # highboost ( > 1)


def filtering(img: np.ndarray, kernel: np.ndarray, coeff = 1.0) -> np.ndarray:
    """ Filtering an image with symmetric padding """
    s, pad = kernel.shape[0], kernel.shape[0] // 2
    new_img = np.empty_like(img)
    img = np.pad(img, ((pad, pad), (pad, pad)), "symmetric")

    for r in range(new_img.shape[0]):
        for c in range(new_img.shape[1]):
            new_img[r, c] = round(coeff * np.average(kernel * img[r : r+s, c : c+s]))
    return new_img
```

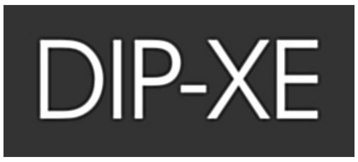
Problem 9(a)

```
def Unsharp_masking(img: np.ndarray, coeff: float) -> np.ndarray:
    """ Apply unsharp masking on input image. """
    img = img.astype(dtype=np.int32)

    # Filter setting
    ave_filter = np.ones((3, 3), dtype=np.int32)

    # Enhancing
    mask = img - filtering(img, ave_filter)
    img = img + coeff * mask
    img[img < 0], img[img > 255] = 0, 255
    return img.astype(dtype=np.uint8)


if __name__ == "__main__":
    orig_img = np.asarray(cv2.imread(IMAGE_PATH, cv2.IMREAD_GRAYSCALE), dtype=np.uint8)

    print(">> start masking..")
    new_img = Unsharp_masking(orig_img, COEFF)
    cv2.imwrite(TARGET_PATH + "/prob10_result_img.jpg", new_img)
    print(">> done")
```

Problem 9(b)



(Source)



(Result)