استخراج صفحات بیتی یک تصویر

تمرین شماره ۲

تهیمنه توکلی

۹۹۱۲۷۶۲۲۶۷

موعد تحویل: ۰۲/۱۲/۱۹

تاریخ تحویل: ۰۲/۱۲/۱۹

شرح تکنیکی تمرین

الف- هر یک از صفحات باینری ۸ گانه را بصورت یک تصویر در خروجی نمایش دهید.

در این قسمت برای نتایج هر صفحه؛ بیت مربوط به آن صفحه نگه داشته شده و ۷ بیت دیگر به صفر تبدیل شده است. (تابع seperate_pages())

نتایج در مسیر ./partA قرار دارد.

تصویر page_i.png مربوط به صفحه iام از تصویر اصلی است. یعنی بیت iام از سمت راست حفظ شده است و ۷ بیت دیگر تبدیل به صفر شده.

ب- هر بار یکی از صفحات بیتی را تماما صفر کرده و مابقی را بدون تغییر رها کنید و سپس تصویر اصلی را با ترکیب ۸ صفحه بازسازی کنید. در نهایت ۸ تصویر خواهید داشت. تصاویر را نمایش دهید.

نتایج در مسیر ./partB قرار دارد.

تصویر out_i.png مربوط به صفر کردن صفحه iام از تصویر اصلی است. یعنی بیت iام از سمت راست تبدیل به صفر شده است و ۷ بیت دیگر حفظ شده.

نتایج

الف) تفاوت بین ۸ تصویر به دست آمده با تنظیم یک بیت روی ۱ و بقیه روی ۰ (برای هر موقعیت بیت) و تصویر اصلی در نحوه نمایش مقادیر شدت پیکسل ها است.

وقتی یک بیت را روی ۱ و بقیه را روی ۰ تنظیم می کنید، به طور مؤثر مقادیر پیکسل آن بیت خاص را جدا می کنید. در هر یک از ۸ تصویر به دست آمده، تنها پیکسل هایی که بیت متناظر آنها ۱ است، مقدار شدت غیر صفر دارند، در حالی که تمام پیکسل های دیگر مقدار شدت ۰ خواهند داشت. هر کدام از ۸ تصویر جنبه های مختلف تصویر اصلی را بر اساس اهمیت رنگ بیتی که روی ۱ تنظیم شده است برجسته می کند.

در تصویری که کمترین بیت مهم (LSB) روی ۱ تنظیم شده است، فقط پیکسل هایی با مقدار شدت فرد (۱=LSB در باینری) دارای مقدار غیر صفر خواهند بود. تمام پیکسل های دیگر (آنهایی که مقدار شدت زوج دارند) ۰ خواهند بود.

در تصاویری که بیت مهمتر حفظ شده است؛ کلیات تصویر اصلی قابل تشخیص است. اما جزییات تصویر از بین رفته است. زیرا بیت‌های کم ارزش که حاوی جزییات تصویر هستند صفر شده است. (page_۶, page_۷)

در تصاویری که بیت کم ارزش حفظ شده است؛ کلیات تصویر اصلی از بین رفته است و جزییات قابل مشاهده است. (page_۱, page_۲)

ب) اگر همه بیت ها را دست نخورده نگه دارید اما هر بار یک بیت را برای هر پیکسل صفر قرار دهید، منجر به ۸ تصویر می شود که در هر تصویر یک بیت روی ۰ تنظیم شده است. هر تصویر مشابه تصویر اصلی در مقیاس خاکستری خواهد بود، اما با تفاوت های جزئی. به طور کلی، این ۸ تصویر، تغییراتی از تصویر اصلی را نشان می‌دهند که در آن یک بیت از مقادیر شدت ۰ باشد، که منجر به تغییرات ظریف در روشنایی و کنتراست در بین تصاویر می‌شود.

در هر تصویر، یک بیت از مقادیر شدت برای هر پیکسل روی ۰ تنظیم می شود. این بدان معنی است که مقادیر شدت برخی از پیکسل ها نسبت به تصویر اصلی کمی کمتر می شود و در نتیجه ظاهر آن قسمت ها تیره تر می شود.

مناطقی که بیت خاص روی ۰ تنظیم شده است در مقایسه با تصویر اصلی تیره تر به نظر می رسد، در حالی که بقیه تصویر بدون تغییر باقی می ماند.

اگر کمترین بیت مهم (LSB) را برای هر پیکسل روی ۰ تنظیم کنید، شدت همه پیکسل ها را ۱ واحد کاهش می دهید. این باعث می شود که یک نسخه کمی تیره تر از تصویر اصلی ایجاد شود. (out_۰.png)

به طور مشابه، تنظیم سایر بیت ها روی ۰ باعث کاهش مقادیر شدت پیکسل های خاص می شود که منجر به تیره شدن موضعی در مناطق خاصی از تصویر می شود.

پیوست کدهای پیاده سازی شده

```
import cv2
import numpy as np


def load_image(image_path):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    return image


def bcd_to_bin(image):
    binary_array = np.array(
        [[np.binary_repr(num, width=8) for num in row] for row in image]
    )
    return binary_array


def seperate_binary_pages(binary_arr):
    row = binary_arr.shape[0]
    col = binary_arr.shape[1]
    binary_pages = np.empty((8, row, col))
    for r in range(row):
        for c in range(col):
            for x in range(8):
                binary_pages[x][r][c] = binary_arr[r][c][x]
    # binary_pages = np.split(binary_arr, 8, axis=2)
    return binary_pages
```

```
def seperate_pages(image):
    row = image.shape[0]
    col = image.shape[1]
    pages = np.empty((8, row, col))
    for x in range(8):
        for r in range(row):
            for c in range(col):
                value = image[r][c]
                for j in reversed(range(8)):
                    if j > x:
                        if value >= 2**j:
                            value -= 2**j
                limit = 2**x
                pages[x][r][c] = 0 if value < limit else limit
    return pages


def display_image(image):
    cv2.imshow("Image", image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
    return
```

```
def zero_out(image, page_num):
    row = image.shape[0]
    col = image.shape[1]
    for r in range(row):
        for c in range(col):
            value = image[r][c]
            for j in reversed(range(8)):
                if j > page_num:
                    if value >= 2**j:
                        value -= 2**j
            limit = 2**page_num
            if value >= limit:
                image[r][c] -= limit
    return image


if __name__ == "__main__":
    image_path = "./image.jpg"
    image = load_image(image_path)
    binary_array = bcd_to_bin(image)
    binary_pages = seperate_binary_pages(binary_array)
    # PART A
    pages = seperate_pages(image)
    for page in pages:
        display_image(page)
    # PART B
    for i in range(8):
        out = zero_out(image, i)
        display_image(out)
```