レポート提出票

科目名: 情報工学実験3

実験課題名: 課題4 画像変換

実施日: 2021年 4月 29日

学籍番号: 4619055

氏名: 辰川力駆

共同実験者:

1　実験の要旨

畳み込み演算を用いた画像変換を実装して高速化し、グラフを用いて速度を比較する。

2　実験の目的

画像変換の処理を題材に、画像処理のプログラミングと評価を通じて、画像処理の基本的な考え方を理解することを目的とする。

第 2 回目の実験では、畳み込み演算を用いた画像変換を実装することで空間フィルタリングについて理解することを目的とする。

3　課題 1

3.1　実験方法

himeji_noise.png を用いて平均化フィルタを実装する。

図 1: himeji_noise.png

1. OpenCV、scikit-image 等のライブラリを使用せずに、python と numpy のみを用いて平均化フィルタ myaverage_naive(image, size) を実装する。

2. scikit-image を用いて実装した平均化フィルタと結果が一致することを確認する。(平均二乗誤差 MSE が 1 未満であることを確認する。)

3. 積分画像 (integral image) を用いて平均化フィルタを高速化した myaverage_integral(image, filter_size) を実装し、2 と同じように scikit-image による平均化フィルタの処理結果と一致することを確認する。

4. 画像サイズを変化させながら、2 つのフィルタ関数 myaverage_naive()、myaverage_integral() の処理時間を計測し、グラフを描いて比較するとともに、計算量の観点から考察する。

5. フィルタサイズを変化させながら、2 つのフィルタ関数 myaverage_naive()、myaverage_integral() の処理時間を計測し、グラフを描いて比較するとともに、計算量の観点から考察する。

3.2 実験結果

作成したプログラムは付録のソースコード 1 に載せた。このプログラムを実行して、scikit-image を用いて実装した平均化フィルタとの MSE は、

$$\begin{aligned} \text{MSE(naive)} &\approx 3.178 \times 10^{-28} \\ \text{MSE(integral)} &\approx 9.316 \times 10^{-21} \end{aligned}$$

となった。つまり、MSE が 1 未満なので、scikit-image を用いて実装した平均化フィルタと結果が一致することがわかる。

また myaverage_naive()、 myaverage_integral() において画像サイズを変更したグラフを図 2 に示した。さらに myaverage_naive()、 myaverage_integral() においてフィルタサイズを変更したグラフを図 3 に示した。

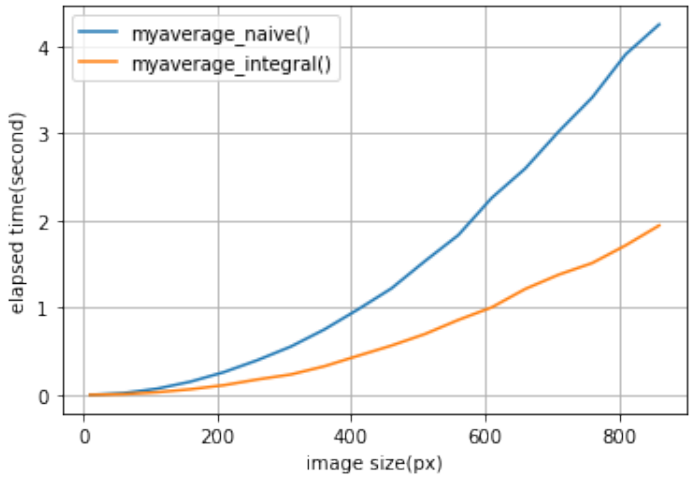

図 2: 画像サイズを変化した結果

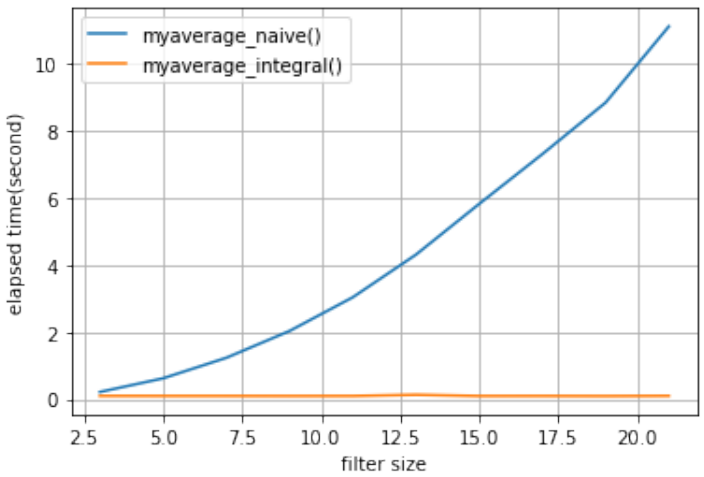

図 3: フィルタサイズを変化した結果

3.3　検討・考察

実験結果の図 2,3 より、myaverage_integral() は、myaverage_naive() より処理速度が速い。これを計算量の観点から考える。

myaverage_naive() は、画像の高さ (px)× 幅 (px)× フィルタサイズ × フィルタサイズであるから、O(n^4) である。それに対し、myaverage_integral() は、画像の高さ (px)× 幅 (px) のみなので、O(n^2) である。これは挿入ソートやバブルソートなどの遅めのソートに匹敵する。

よって、積分画像の平均化フィルタの方が高速化できていることが計算量の観点からも分かる。図 3 において、myaverage_integral() がフィルタサイズの影響をあまり受けてない理由も同様である。

4　まとめ

本実験では、畳み込み演算を用いて画像変換を実装し、空間フィルタリングについて理解することができた。また、課題では畳み込み演算を行う時とそうでない時とでどれくらいの速度が違うのかをグラフで考察し、畳み込み演算の有用性について学んだ。

A 付録

ソースコード 1: kadai1

```
#4619055
import time
import numpy as np
from skimage.io import imread, imsave
from skimage.color import rgb2gray, gray2rgb
from skimage.transform import resize
from scipy.ndimage.filters import convolve, correlate
import skimage.transform
import matplotlib.pyplot as plt
%matplotlib inline

def mse(y1, y2):
    return ((y1 - y2)**2).mean()

def myaverage_naive(im, filter_size = 5):
    ''' im : 入力画像, filter_size : 平均化フィルタのサイズ(奇数) '''
    iheight, iwidth = im.shape[:2]
    imout = np.zeros((iheight, iwidth))

    # ここにコードを追加
    filter = np.ones((filter_size, filter_size)) / (filter_size ** 2)
    im_pad = np.pad(im, (filter_size//2, filter_size//2), mode="constant")

    for x in range(filter_size):
        for y in range(filter_size):
            for height in range(iheight):
                for width in range(iwidth):
                    imout[height, width] += filter[y, x]*im_pad[y+height, x+width]

    return imout

def myaverage_integral(im, filter_size = 5):
    ''' im : 入力画像, filter_size : 平均化フィルタのサイズ(奇数) '''

    def integral_image(im,pad):
        ''' 積分画像の作成自分で実装 '''
        s = np.zeros_like(im)
        iheight, iwidth = im.shape[:2]
        # 自分で積分画像を求める関数を実装する場合はここにコードを追加
        for i in range(pad+1,iwidth):
            for j in range(pad+1,iheight):
                s[j, i] = s[j, i - 1] + s[j - 1, i] - s[j - 1, i - 1] + im[j, i]
        return s

    iheight, iwidth = im.shape[:2]
    imout = np.zeros((iheight, iwidth))

    # ここにコードを追加
```

```
49          pad = filter_size//2
50          im_pad = np.pad(im, ([pad+1, pad], [pad+1, pad]), mode="constant")
51          im_integral = integral_image(im_pad,pad)
52
53          for i in range(pad+1, iwidth+pad+1):
54              for j in range(pad+1, iheight+pad+1):
55                  imout[j-pad-1, i-pad-1] = (im_integral[j+pad, i+pad] - im_integral[j-pad
                        -1, i+pad] - im_integral[j+pad, i-pad-1] + im_integral[j-pad-1, i-
                        pad-1]) / filter_size ** 2
56          return imout
57
58      def myaverage_separable(im, filter_size):
59          ''' im : 入力画像, filter_size : 平均化フィルタのサイズ(奇数) '''
60          iheight, iwidth = im.shape[:2]
61          imout = np.zeros((iheight, iwidth))
62
63          # ここにコードを追加
64
65          return imout
66
67      im = 255 * rgb2gray(imread("data/himeji_noise.png"))
68      filter_size = 3
69
70      kernel = np.ones((filter_size, filter_size)) / (filter_size ** 2)
71      im2a = myaverage_naive(im, filter_size)
72      im2b = myaverage_integral(im, filter_size)
73      im2c = myaverage_separable(im, filter_size) # オプション
74      im2_gt = correlate(im, kernel, mode="constant")
75
76      print("mse_naive=", mse(im2_gt, im2a)) # 1未満であればOK
77      print("mse_integral=", mse(im2_gt, im2b)) # 1未満であればOK
78      print("mse_separable=", mse(im2_gt, im2c)) # オプション
79
80      #グラフのため
81      time_naive_image = []
82      time_integral_image = []
83      time_naive_filter = []
84      time_integral_filter = []
85
86      filter_size = 3
87      for i in range(10, 900, 50):
88          im = 255 * rgb2gray(imread("data/himeji_noise.png"))
89          im = skimage.transform.resize(im, (i, i))
90
91          elapsed_time = time.time()
92          im2a = myaverage_naive(im, filter_size)
93          time_naive_image.append(time.time() - elapsed_time)
94
95          elapsed_time = time.time()
96          im2b = myaverage_integral(im, filter_size)
97          time_integral_image.append(time.time() - elapsed_time)
98
```

```
    for i in range (3,22,2):
        filter_size = i
        im = 255 * rgb2gray(imread("data/himeji_noise.png"))

        elapsed_time = time.time()
        im2a = myaverage_naive(im, filter_size)
        time_naive_filter.append(time.time() - elapsed_time)

        elapsed_time = time.time()
        im2b = myaverage_integral(im, filter_size)
        time_integral_filter.append(time.time() - elapsed_time)

    plt.plot(range(10,900,50),time_naive_image,label="myaverage_naive()")
    plt.plot(range(10,900,50),time_integral_image,label="myaverage_integral()")

    plt.xlabel("image␣size(px)")
    plt.ylabel("elapsed␣time(second)")
    plt.grid()
    plt.legend()
    plt.show()

    plt.plot(range(3, 22, 2), time_naive_filter, label="myaverage_naive()")
    plt.plot(range(3, 22, 2), time_integral_filter, label="myaverage_integral()")

    plt.xlabel("filter␣size")
    plt.ylabel("elapsed␣time(second)")
    plt.grid()
    plt.legend()
    plt.show()
```