数字图像处理 · 第一次实验

Project Number: 3-2, 3-5, 3-6, 2-1

Course: 数字图像处理

Student's Name: 罗柯

Date Due: 2017-03-28

Date Handed-in: 2017-03-27

Abstract

本次实验的内容主要是进行空域上的图像增强，包括 Project 03-02 中的直方图均衡化，Project 03-05 中基于二阶微分拉普拉斯算子的锐化以及 Project 03-06 中的反锐化掩模及高提升滤波的实现与使用。

1. Technical Discussion

1.1. Project 03-02

1.1.1. 数字图像直方图

数字图像直方图是对数字图像灰度值的统计图，它可以用离散函数$h(r_k)$进行表示。

$h(r_k) = n_k$，，其中r_k是灰度级别，n_k是对应的像素个数

如果对函数进行归一化，则得到归一化的直方图。

$$p(r_k) = \frac{n_k}{n}$$

1.1.2. 直方图均衡化

一般情况下，如果图像的直方图成均匀态势分布，则图像具有较高的对比度和比较丰富的灰度色调，因此基于图像直方图，能够应用一种变换，使得变换后的图像直方图分布尽可能均衡，即所谓的直方图均衡化，从而使得图像具有较好的效果。

将灰度级别归一化，变换$s = T(r)$应当满足如下条件：

（1） $T(r)$在$[0,1]$上单值且单调递增；

（2） $T(r)$在$[0,1]$上满足$0 \le T \le 1$。

满足条件（1）保证其逆变换存在并且变换后灰度大小关系得以保持，满足条件（2）保证变换之后灰度级别不会超出范围。

在满足条件（1）（2）的情况下，由概率论的知识，有

$$p_s(s) = p_r(r)|\frac{dr}{ds}|$$

考虑直方图函数在$[0, \mathrm{r}]$上的累积分布函数

$$F(r) = \int_0^r p_r(w)dw$$

当$p_r(w)$连续时，$F(r)$在$[0,1]$上满足条件（1）（2），将累积分布函数作为变换函数，通过上述两式能够得到

$$s = F(r) = \int_0^r p_r(w)dw$$

$$p_s(s) = 1, 0 \le s \le 1$$

$p_s(s)$处处为 1，即变换得到的直方图是处处均衡的。离散情况可以视作是连续情况下的近似，因此能够得到变换公式。

$$s_k = \sum_{j=0}^{k} p_r(r_j)$$

1.2. Project 03-05

1.2.1. 基于二阶微分的图像增强

二阶微分对图像中突变的位置有比较强的响应，即能够提取到图像的细节，通过将二阶微分得到的结果叠加至原始图像，能够在保持图像原始背景的同时增强细节。

1.2.2. 拉普拉斯算子

最简单且各向同性的二阶微分算子是拉普拉斯算子，其中一种拉普拉斯滤波器如下。

−1	−1	−1
−1	8	−1
−1	−1	−1

1.3. Project 03-06

1.3.1. 反锐化掩模

反锐化掩模是指使用原图减去经过模糊的图片，从而得到相对清晰的部分。

$$f_s(x,y) = f(x,y) - \overline{f}(x,y)$$

1.3.2. 高提升滤波

高提升滤波在反锐化掩模的基础上增加了参数 A，目的是对原图的亮度进行提升。

$$f_{hb}(x,y) = Af(x,y) - \overline{f}(x,y)$$

2. Discussion of Results

2.1. Project 03-02

如图 1 所示，将直方图均衡化的方法直接应用到 Fig. 3.8(a) 上虽然使其提升了细节，但整体的图像效果却不好，这说明并非所有情况下灰度值分布均衡都能说明图像具有较好的显示效果。

直方图均衡化适用于原图像本身的灰度级别相对连续、均衡，但覆盖范围较小的情境，如图 2 所示，仅通过直方图均衡化就能使曝光过度的原图获得非常好的显示效果；而对于灰度值出现断层或某些灰度级别显著多于其他灰度级别的原图像，应当使用直方图匹配等其他方法以获得更好的显示效果。

2.2. Project 03-05

如图 3 所示，经过拉普拉斯算子进行增强的图像（Result）明显要比原图锐利许多，细节更加突出。

2.3. Project 03-06

图 4 至图 6 为在参数 A 为 1.0、1.4、1.7 下使用高提升滤波分别得到的结果；当参数 A 为 1.0 时，实验操作实质上是拉普拉斯增强，而依次增大参数 A，图像的整体亮度也不断增加。

2.4. Project 02-01

图 7 至图 11 为使用 Halftoning 方法表示图像灰度级别的实验结果，注意到图 7 和图 8 是同一幅图像在不同 dpi 下的输出结果，dpi 为 1200 时效果明显好于 600 时的结果；但事实上，图像的大小仅为 768*768，在均为高 dpi 的情况下，理论上不应该有差别；截至提交实验报告时还未能找出具体原因，猜测可能是图像输出的压缩方法或插值方法所导致的。

3.Results

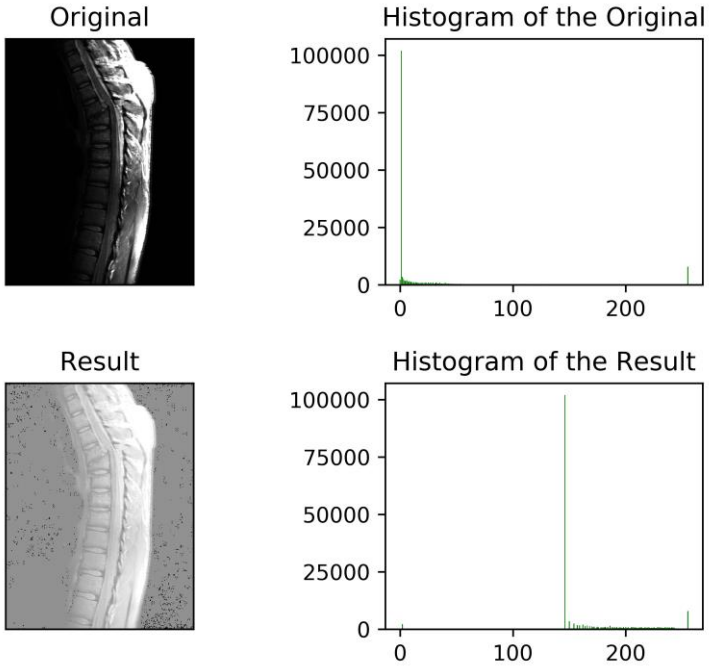

图 1.Project 03-02 直方图均衡化 1

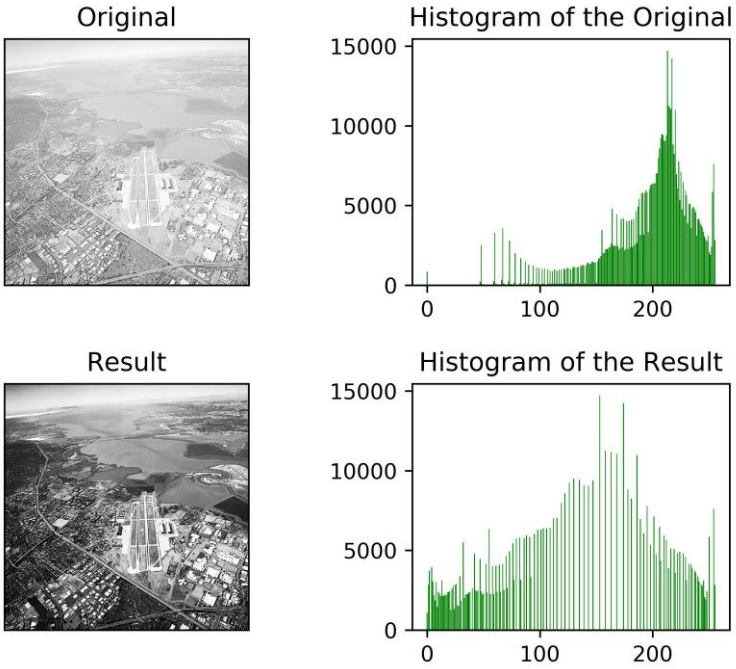

图 2.Project 03-02 直方图均衡化 2

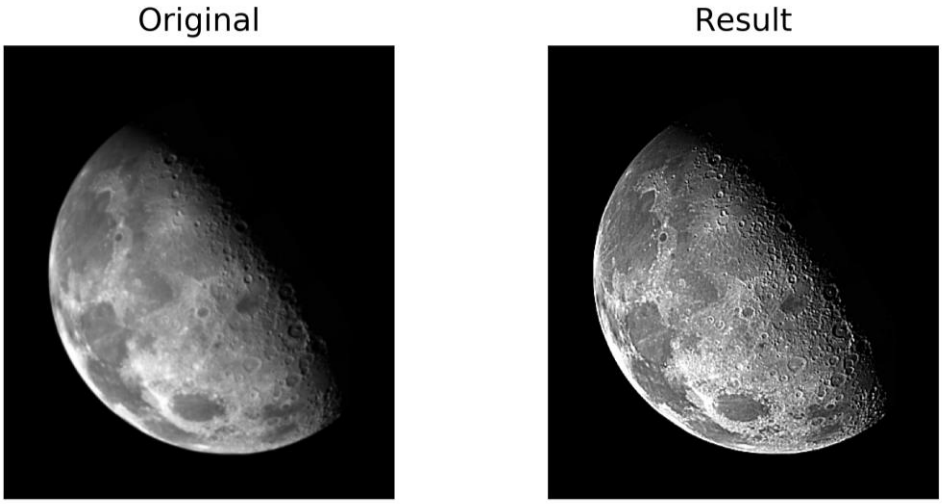

图 3.Project 03-05 基于拉普拉斯算子的锐化

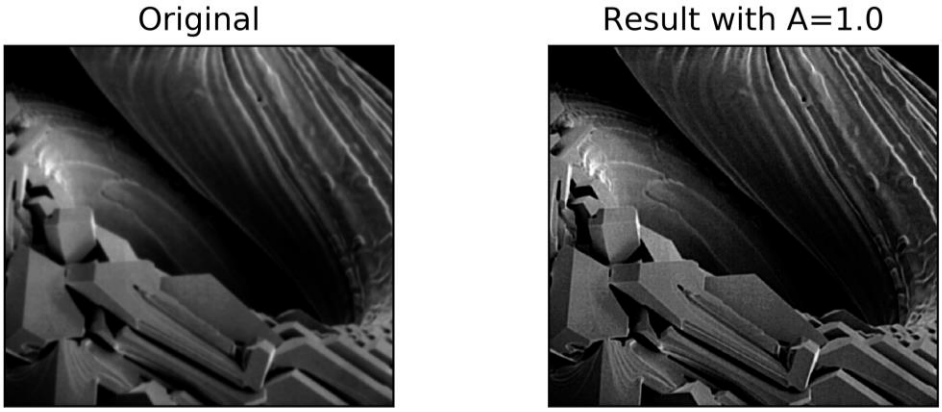

图 4.Project 03-06 高提升滤波 A=1.0

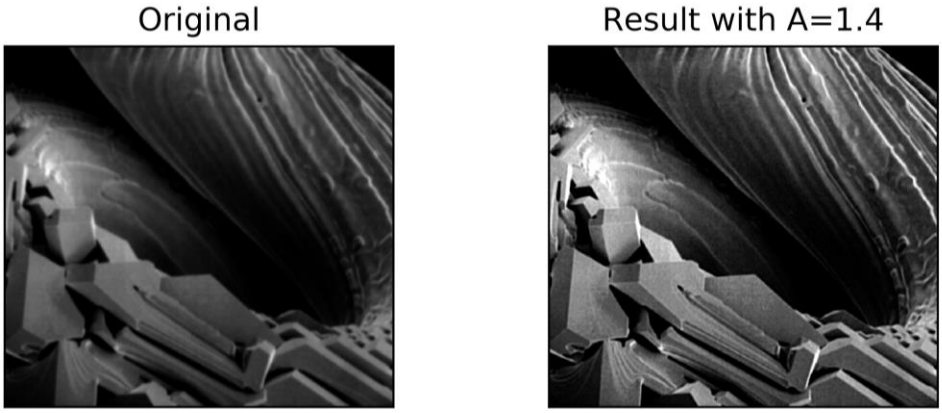

图 5.Project 03-06 高提升滤波 A=1.4

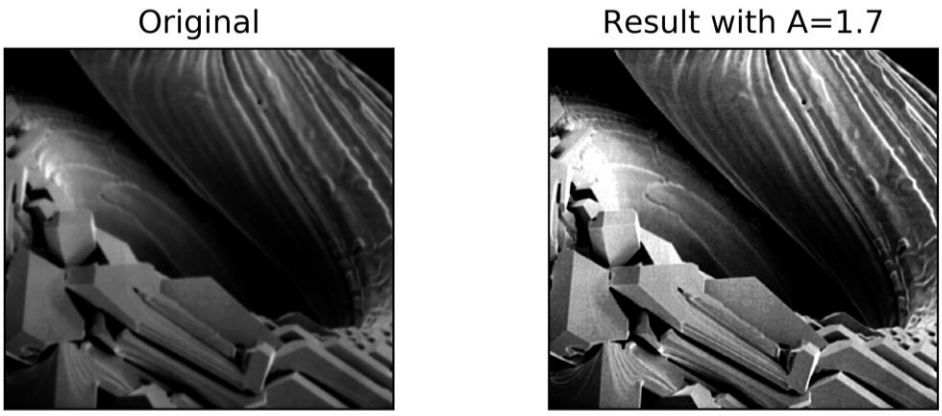

图 6.Project 03-06 高提升滤波 A=1.7

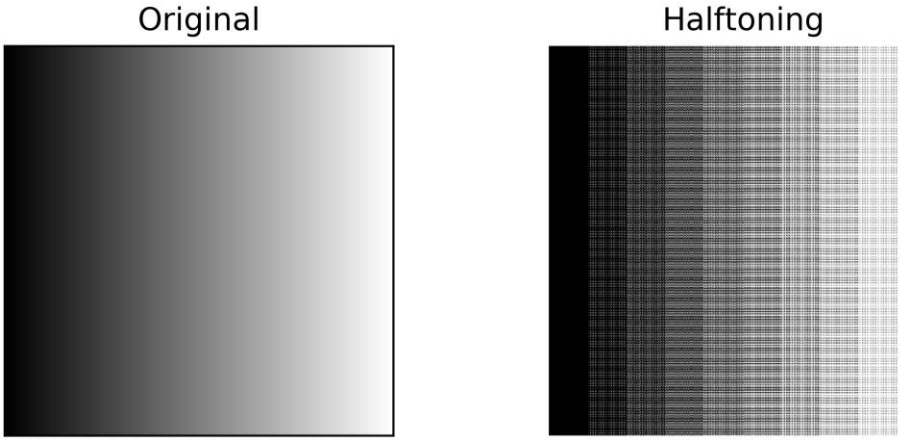

图 7.Project 02-01 使用 Halftoning 方法显示渐变灰度图像（dpi=600）

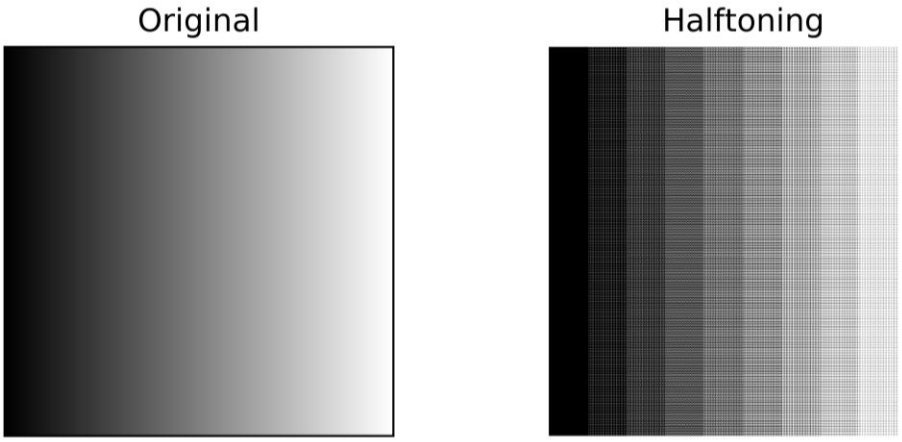

图 8.Project 02-01 使用 Halftoning 方法显示渐变灰度图像（dpi=1200）

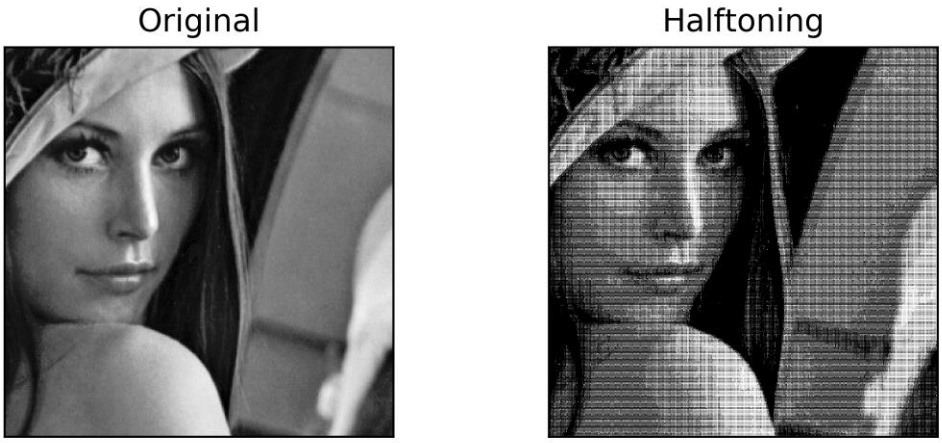

图 9.Project 02-01 使用 Halftoning 方法显示 Figs.2.22(a)

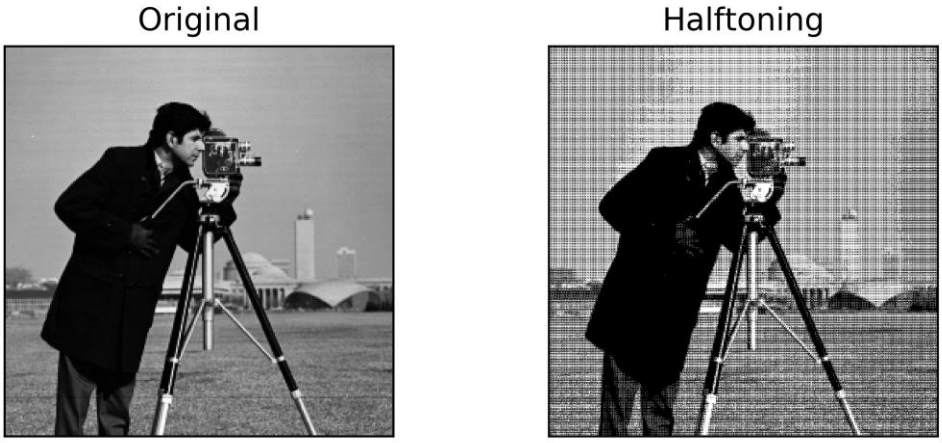

图 10.Project 02-01 使用 Halftoning 方法显示 Figs.2.22(b)

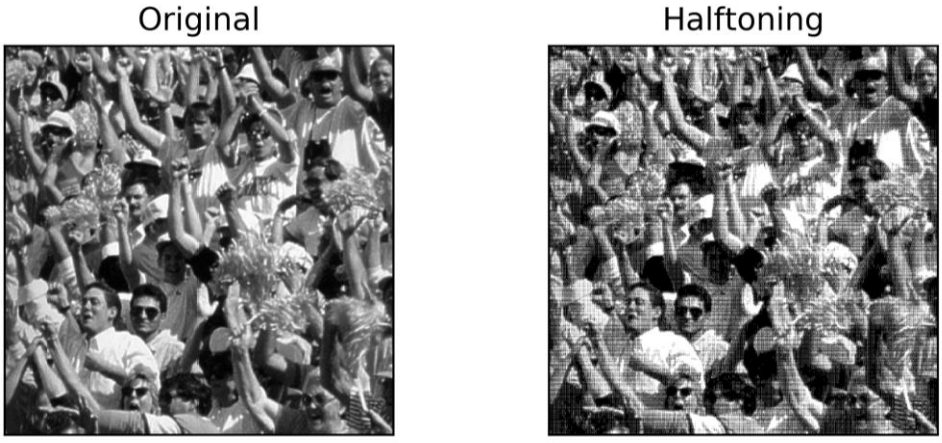

图 11.Project 02-01 使用 Halftoning 方法显示 Figs.2.22(c)

4. Appendix

```
import matplotlib.pylab as plt
import numpy as np

IMAGE_FOLDER='images'

def calHist(im):
    # calculate and return the histogram of a given image
    histogram = np.zeros((256,), dtype=np.uint32)
    for row in im:
        for pixel in row:
            histogram[pixel] += 1
    return histogram

def histChart(histogram):
    # draw the histogram chart
    plt.bar(np.arange(0, 256), histogram, color='g')

def histEqual(histogram):
    # apply equalization to a histogram and return the mapping
    length = len(histogram)
    acc = np.zeros(length, dtype=np.float32)

    acc[0] = histogram[0]
    tot = np.float32(histogram[0])
    for idx in range(1, length):
        acc[idx] = acc[idx - 1] + histogram[idx]
        tot += histogram[idx]

    mapping = np.zeros(length, dtype=np.uint8)
    for idx in range(length):
        mapping[idx] = np.uint8(length * (acc[idx] / tot) - 0.5)

    return mapping

def spatialFilter3x3(kernel, im, co=1.):
    # operating convolution on a given image with 3x3 kernel
    kernel = np.array(kernel)
    im = np.array(im)
    new_im = np.zeros(im.shape, dtype=np.uint8)

    ix = im.shape[0]
    iy = im.shape[1]
```

```
    for x in range(ix):
        for y in range(iy):
            temp = 0.
            for i in range(-1, 2):
                for j in range(-1, 2):
                    if x + i >= 0 and x + i < ix and \
                       y + j >= 0 and y + j < iy:
                       temp += kernel[i + 1, j + 1] * im[x + i, y + j]
            temp *= co
            if temp < 0:
                temp = 0
            elif temp >= 255:
                temp = 255
            new_im[x, y] = np.uint8(temp)

    return new_im

def spatialFilter3x3_(kernel, im, co=1.):
    # unbound version
    # operating convolution on a given image with 3x3 kernel
    kernel = np.array(kernel)
    im = np.array(im)
    new_im = np.zeros(im.shape, dtype=np.int32)

    ix = im.shape[0]
    iy = im.shape[1]

    for x in range(ix):
        for y in range(iy):
            temp = 0.
            for i in range(-1, 2):
                for j in range(-1, 2):
                    if x + i >= 0 and x + i < ix and \
                       y + j >= 0 and y + j < iy:
                       temp += kernel[i + 1, j + 1] * im[x + i, y + j]
            temp *= co
            new_im[x, y] = np.int16(temp)

    return new_im

def bound(im):
    # truncate the values out of [0, 255]
    im = np.array(im)
    ix, iy = im.shape
```

```
    for i in range(ix):
        for j in range(iy):
            if im[i, j] < 0:
                im[i, j] = 0
            elif im[i, j] >= 255:
                im[i, j] = 255

    return np.uint8(im)

def halftoning(im):
    im = np.array(im)
    row, col = im.shape
    res = np.zeros((row * 3, col * 3), dtype=np.uint8)

    def grayLevel(x, y, level):
        if level >= 1:
            res[x][y + 1] = 1      # (0, 1)
        if level >= 2:
            res[x + 2][y + 2] = 1  # (2, 2)
        if level >= 3:
            res[x][y] = 1          # (0, 0)
        if level >= 4:
            res[x + 2][y] = 1      # (2, 0)
        if level >= 5:
            res[x][y + 2] = 1      # (0, 2)
        if level >= 6:
            res[x + 1][y + 2] = 1  # (1, 2)
        if level >= 7:
            res[x + 2][y + 1] = 1  # (2, 1)
        if level >= 8:
            res[x + 1][y] = 1      # (1, 0)
        if level >= 9:
            res[x + 1][y + 1] = 1  # (1, 1)

    for iidx in range(0, row):
        for jidx in range(0, col):
            grayLevel(3*iidx, 3*jidx, 10. * im[iidx][jidx] / 255.)

    return res

def proj0302(image_name='Fig3.08(a).jpg'):
    im1 = plt.imread(IMAGE_FOLDER + '/' + image_name)
    im2 = np.zeros(im1.shape, dtype=np.uint8)
```

```
    # histogram equalization
    histogram1 = calHist(im1)
    mapping = histEqual(histogram1)
    for row in range(im1.shape[0]):
        for col in range(im1.shape[1]):
            im2[row, col] = mapping[im1[row, col]]
    histogram2 = calHist(im2)

    ax11 = plt.subplot(2, 2, 1)
    ax12 = plt.subplot(2, 2, 2)
    ax21 = plt.subplot(2, 2, 3)
    ax22 = plt.subplot(2, 2, 4)

    plt.sca(ax11)
    plt.title("Original")
    ax11.set_xticks([])
    ax11.set_yticks([])
    plt.imshow(im1, cmap=plt.cm.gray)

    plt.sca(ax12)
    plt.title("Histogram of the Original")
    histChart(histogram1)

    plt.sca(ax21)
    plt.title("Result")
    ax21.set_xticks([])
    ax21.set_yticks([])
    plt.imshow(im2, cmap=plt.cm.gray)

    plt.sca(ax22)
    plt.title("Histogram of the Result")
    histChart(histogram2)

    plt.subplots_adjust(0.125, 0.1, 0.9, 0.9, 0.4, 0.4)
    plt.savefig('proj0302.png', dpi=600, cmap=plt.cm.gray)
    plt.show()

def proj0305(image_name='Fig3.40(a).jpg'):
    im = plt.imread(IMAGE_FOLDER + '/' + image_name)

    # Laplacian Enhancement
    kernel = np.array(
        [[-1, -1, -1],
         [-1, 9, -1],
```

```
         [-1, -1, -1]]
    )
    new_im = spatialFilter3x3(kernel, im)

    ax11 = plt.subplot(1, 2, 1)
    ax11.set_xticks([])
    ax11.set_yticks([])
    ax12 = plt.subplot(1, 2, 2)
    ax12.set_xticks([])
    ax12.set_yticks([])

    plt.sca(ax11)
    plt.title('Original')
    plt.imshow(im, cmap=plt.cm.gray)

    plt.sca(ax12)
    plt.title('Result')
    plt.imshow(new_im, cmap=plt.cm.gray)

    plt.subplots_adjust(0.125, 0.1, 0.9, 0.9, 0.4, 0.4)
    plt.savefig('proj0305.png', dpi=600, cmap=plt.cm.gray)
    plt.show()

def proj0306(image_name='Fig3.43(a).jpg'):
    im = plt.imread(IMAGE_FOLDER + '/' + image_name)
    A = input()

    avgKernel = np.array(
        [[1, 1, 1],
         [1, 1, 1],
         [1, 1, 1]])
    co = 1. / 9.

    avg_im = spatialFilter3x3_(avgKernel, im, co)

    lapKernel = np.array(
        [[-1, -1, -1],
         [-1, A + 8, -1],
         [-1, -1, -1]])
    lap_im = spatialFilter3x3_(lapKernel, im)

    res_im = bound(np.int16(lap_im))

    ax11 = plt.subplot(1, 2, 1)
```

```
    ax11.set_xticks([])
    ax11.set_yticks([])

    ax12 = plt.subplot(1, 2, 2)
    ax12.set_xticks([])
    ax12.set_yticks([])

    plt.sca(ax11)
    plt.title('Original')
    plt.imshow(im, cmap=plt.cm.gray)

    plt.sca(ax12)
    plt.title('Result with A=' + str(A))
    plt.imshow(res_im, cmap=plt.cm.gray)

    plt.subplots_adjust(0.125, 0.1, 0.9, 0.9, 0.4, 0.4)
    plt.savefig('proj0306_'+str(A)+'.png', dpi=600, cmap=plt.cm.gray)
    plt.show()

def proj0201(image_name='Fig2.22(a).jpg', image=None):
    if not image == None:
        im = image
        image_name = ''
    else:
        im = plt.imread(IMAGE_FOLDER + '/' + image_name)

    res = halftoning(im)

    ax11 = plt.subplot(1, 2, 1)
    ax11.set_xticks([])
    ax11.set_yticks([])
    ax12 = plt.subplot(1, 2, 2)
    ax12.set_xticks([])
    ax12.set_yticks([])
    plt.sca(ax11)
    plt.title('Original')
    plt.imshow(im, cmap=plt.cm.gray)
    plt.sca(ax12)
    plt.title('Halftoning')
    plt.axis('off')
    plt.imshow(res, cmap=plt.cm.gray, interpolation='none')
    plt.subplots_adjust(0.125, 0.1, 0.9, 0.9, 0.4, 0.4)
    plt.savefig('proj0201_'+image_name+'.png', dpi=1200, cmap=plt.cm.gray)
    plt.show()
```