**实验一：数字图像空域增强实验**

**作业一：彩色图像的直方图均衡化**

问题1：提取RGB三个通道图像，计算其平均直方图结果，然后再进行均衡化；

def default\_loader(path):

    return cv2.imread(path, cv2.IMREAD\_COLOR)

def rgb\_equalization(self, index):

    try:

        image\_path = os.path.join(self.path, self.image\_list[index - 1])

    except:

        print("ERROR！ 并不包含你想要进行RGB处理的这张图片")

    else:

        image\_name = re.findall(r'(.+?)\.', self.image\_list[index - 1])

        image = default\_loader(image\_path)

        [b, g, r] = cv2.split(image)

        fig = plt.figure(figsize=(10, 6), num="Red Color Histogram")

        plt.hist(r.flatten(), 256, [0, 256], color="red")

        plt.title("Red Color Histogram", fontsize=24)

        plt.xlim([0, 256])

        plt.tick\_params(labelsize=14)

        plt.savefig(str(\*image\_name) + " Red Color Histogram.png")

        fig = plt.figure(figsize=(10, 6), num="Green Color Histogram")

        plt.hist(g.flatten(), 256, [0, 256], color="green")

        plt.title("Green Color Histogram", fontsize=24)

        plt.xlim([0, 256])

        plt.tick\_params(labelsize=14)

        plt.savefig(str(\*image\_name) + " Green Color Histogram.png")

        fig = plt.figure(figsize=(10, 6), num="Blue Color Histogram")

        plt.hist(b.flatten(), 256, [0, 256], color="blue")

        plt.title("Blue Color Histogram", fontsize=24)

        plt.xlim([0, 256])

        plt.tick\_params(labelsize=14)

        plt.savefig(str(\*image\_name) + " Blue Color Histogram.png")

        # 直方图均衡化处理

        equ\_r = cv2.equalizeHist(r)

        equ\_g = cv2.equalizeHist(g)

        equ\_b = cv2.equalizeHist(b)

        fig = plt.figure(figsize=(10, 6),

                            num="Red Color Histogram (Equed)")

        plt.hist(r.flatten(), 256, [0, 256], color="red")

        plt.title("Red Color Histogram (Equed)", fontsize=24)

        plt.xlim([0, 256])

        plt.tick\_params(labelsize=14)

        plt.savefig(str(\*image\_name) + " Red Color Histogram (Equed).png")

        fig = plt.figure(figsize=(10, 6),

                            num="Green Color Histogram (Equed)")

        plt.hist(g.flatten(), 256, [0, 256], color="green")

        plt.title("Green Color Histogram (Equed)", fontsize=24)

        plt.xlim([0, 256])

        plt.tick\_params(labelsize=14)

        plt.savefig(

            str(\*image\_name) + " Green Color Histogram (Equed).png")

        fig = plt.figure(figsize=(10, 6),

                            num="Blue Color Histogram (Equed)")

        plt.hist(b.flatten(), 256, [0, 256], color="blue")

        plt.title("Blue Color Histogram (Equed)", fontsize=24)

        plt.xlim([0, 256])

        plt.tick\_params(labelsize=14)

        plt.savefig(str(\*image\_name) + " Blue Color Histogram (Equed).png")

        equ = cv2.merge((equ\_b, equ\_g, equ\_r))

        cv2.imwrite(str(\*image\_name) + " RGB Equ.png", equ)

问题2：RGB空间转为HSI空间图像，对I（亮度，Intensity）通道进行直方图均衡化，再转为RGB图像。

def hsv\_loader(path):

    img = cv2.imread(path, cv2.IMREAD\_COLOR)

    return cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

def hsv\_equalization(self, index):

    try:

        image\_path = os.path.join(self.path, self.image\_list[index - 1])

    except:

        print("ERROR！ 并不包含你想要进行HSV处理的这张图片")

    else:

        image\_name = re.findall(r'(.+?)\.', self.image\_list[index - 1])

        image = default\_loader(image\_path)

        hsv = cv2.cvtColor(image, cv2.COLOR\_BGR2HSV)

        [h, s, v] = cv2.split(hsv)

        fig = plt.figure(figsize=(10, 6), num="HSV Hue Histogram")

        plt.hist(h.flatten(), 256, [0, 256], color="blue")

        plt.title("HSV Hue Histogram", fontsize=24)

        plt.xlim([0, 256])

        plt.tick\_params(labelsize=14)

        plt.savefig(str(\*image\_name) + " HSV Hue Histogram.png")

        fig = plt.figure(figsize=(10, 6), num="HSV Saturation Histogram")

        plt.hist(s.flatten(), 256, [0, 256], color="green")

        plt.title("HSV Saturation Histogram", fontsize=24)

        plt.xlim([0, 256])

        plt.tick\_params(labelsize=14)

        plt.savefig(str(\*image\_name) + " HSV Saturation Histogram.png")

        fig = plt.figure(figsize=(10, 6), num="HSV Value Histogram")

        plt.hist(v.flatten(), 256, [0, 256], color="red")

        plt.title("HSV Value Histogram", fontsize=24)

        plt.xlim([0, 256])

        plt.tick\_params(labelsize=14)

        plt.savefig(str(\*image\_name) + " HSV Value Histogram.png")

        # 直方图均衡化处理

        equ\_h = cv2.equalizeHist(h)

        equ\_s = cv2.equalizeHist(s)

        equ\_v = cv2.equalizeHist(v)

        fig = plt.figure(figsize=(10, 6), num="HSV Hue Histogram (Equed)")

        plt.hist(equ\_h.flatten(), 256, [0, 256], color="blue")

        plt.title("HSV Hue Histogram (Equed)", fontsize=24)

        plt.xlim([0, 256])

        plt.tick\_params(labelsize=14)

        plt.savefig(str(\*image\_name) + " HSV Hue Histogram (Equed).png")

        fig = plt.figure(figsize=(10, 6),

                            num="HSV Saturation Histogram (Equed)")

        plt.hist(equ\_s.flatten(), 256, [0, 256], color="green")

        plt.title("HSV Saturation Histogram (Equed)", fontsize=24)

        plt.xlim([0, 256])

        plt.tick\_params(labelsize=14)

        plt.savefig(

            str(\*image\_name) + " HSV Saturation Histogram (Equed).png")

        fig = plt.figure(figsize=(10, 6),

                            num="HSV Value Histogram (Equed)")

        plt.hist(equ\_v.flatten(), 256, [0, 256], color="red")

        plt.title("HSV Value Histogram (Equed)", fontsize=24)

        plt.xlim([0, 256])

        plt.tick\_params(labelsize=14)

        plt.savefig(str(\*image\_name) + " HSV Value Histogram (Equed).png")

        equ = cv2.merge((equ\_h, equ\_s, equ\_v))

        cv2.imwrite(str(\*image\_name) + " HSV Equ.png", equ)

**作业二：彩色图像的去噪**

问题1：对一副彩色图像分别添加高斯噪声和椒盐噪声，输出结果图像；

def add\_noise(

        self,

        index,

        noise="gaussian",

):

    try:

        image\_path = os.path.join(self.path, self.image\_list[index - 1])

    except:

        print("ERROR！ 并不包含你想要进行添加噪声处理的这张图片")

    else:

        image\_name = re.findall(r'(.+?)\.', self.image\_list[index - 1])

        image = default\_loader(image\_path)

        noise\_image = skimage.util.random\_noise(image, mode=noise) \* 256

        cv2.imwrite(

            str(\*image\_name) + " " + noise.title() + ".png", noise\_image)

问题2（编写KNN均值滤波对二种噪声图像进行去噪，测试3组不同的参数，输出结果图像，并分析实验效果；）& 问题3（编写KNN中值滤波对二种噪声图像进行去噪，测试3组不同的参数，输出结果图像，并分析实验效果。）

def delete\_noise(self,

                    index,

                    noise="gaussian",

                    modal="mean",

                    kernal\_size=3):

    self.local\_image\_list = [

        x for x in listdir(".") if is\_noise\_image\_file(x, index, noise)

    ]

    for image in self.local\_image\_list:

        image\_name = re.findall(r'(.+?)\.', image)

        img = default\_loader(image)

        if (modal == "mean"):

            image\_blur = cv2.blur(img, (kernal\_size, kernal\_size))

            cv2.imwrite(

                str(image\_name[0]) + " " + str(modal.title()) +

                str(kernal\_size) + ".png", image\_blur)

        elif (modal == "middle"):

            image\_blur = cv2.medianBlur(img, kernal\_size)

            cv2.imwrite(

                str(image\_name[0]) + " " + str(modal.title()) +

                str(kernal\_size) + ".png", image\_blur)

**作业三：图像锐化**

问题1（自己编写程序，实现Lapalician算子，Sobel算子，提取图像的细节信息；）&问题2（自己设计一个模板，与传统的Lapalician算子和Sobel算子不同（检测x和y方向），能够检测斜方向上的图像细节）&问题3（将上述步骤提取的图像细节，叠加到原图上，分析和比较锐化后的效果。）

class Dataset():

    def \_\_init\_\_(self, path):

        self.path = path

        self.image\_list = [x for x in listdir(path) if is\_image\_file(x)]

        self.image\_list = sorted(self.image\_list)

        self.lap\_kernal = np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]])

        self.sobel\_kernal\_x = np.array([[-3, 0, 3], [-10, 0, 10], [-3, 0, 3]])

        self.sobel\_kernal\_y = np.array([[-3, -10, -3], [0, 0, 0], [3, 10, 3]])

        self.fangnan\_kernal\_left = np.array([[3, -2, 0], [-2, 2, -2],

                                             [0, -2, 3]])

        self.fangnan\_kernal\_right = np.array([[0, -2, 3], [-2, 2, -2],

                                              [3, -2, 0]])

    def Laplacian(self, index):

        try:

            image\_path = os.path.join(self.path, self.image\_list[index - 1])

        except:

            print("ERROR！ 并不包含你想要进行RGB处理的这张图片", index)

        else:

            image\_name = re.findall(r'(.+?)\.', self.image\_list[index - 1])

            c\_image = default\_loader(image\_path)

            image = gray\_loader(image\_path)

            lap = signal.convolve2d(image,

                                    self.lap\_kernal,

                                    boundary="symm",

                                    mode='same')

            lap = np.absolute(lap)

            # lap = cv2.merge((b\_lap, g\_lap, r\_lap))

            lap = np.array(lap, dtype=np.uint8)

            cv2.imwrite(str(\*image\_name) + " Lap.png", lap)

            # 与原图做叠加

            stacked\_lap = np.stack((lap, ) \* 3, axis=-1)

            added\_img = cv2.add(c\_image, stacked\_lap)

            cv2.imwrite(str(\*image\_name) + " Lap ADD.png", added\_img)

    def Sobel(self, index, dir="x"):

        try:

            image\_path = os.path.join(self.path, self.image\_list[index - 1])

        except:

            print("ERROR！ 并不包含你想要进行RGB处理的这张图片", index)

        else:

            image\_name = re.findall(r'(.+?)\.', self.image\_list[index - 1])

            c\_image = default\_loader(image\_path)

            image = gray\_loader(image\_path)

            # [b, g, r] = cv2.split(image)

            if (dir == "x"):

                kernal = self.sobel\_kernal\_x

            elif (dir == "y"):

                kernal = self.sobel\_kernal\_y

            lap = signal.convolve2d(image, kernal, boundary="symm", mode='same')

            lap = np.absolute(lap)

            # lap = cv2.merge((b\_lap, g\_lap, r\_lap))

            lap = np.array(lap, dtype=np.uint8)

            cv2.imwrite(str(\*image\_name) + " " + str(dir) + " Sobel.png", lap)

            # 与原图做叠加

            stacked\_lap = np.stack((lap, ) \* 3, axis=-1)

            added\_img = cv2.add(c\_image, stacked\_lap)

            cv2.imwrite(

                str(\*image\_name) + " " + str(dir) + " Sobel ADD.png", added\_img)

    def fangnan(self, index, fdir="left"):

        try:

            image\_path = os.path.join(self.path, self.image\_list[index - 1])

        except:

            print("ERROR！ 并不包含你想要进行RGB处理的这张图片", index)

        else:

            image\_name = re.findall(r'(.+?)\.', self.image\_list[index - 1])

            c\_image = default\_loader(image\_path)

            image = gray\_loader(image\_path)

            if (fdir == "left"):

                kernal = self.fangnan\_kernal\_left

            elif (fdir == "right"):

                kernal = self.fangnan\_kernal\_right

            lap = signal.convolve2d(image, kernal, boundary="symm", mode='same')

            lap = np.absolute(lap)

            lap = np.array(lap, dtype=np.uint8)

            cv2.imwrite(str(\*image\_name) + " " + str(fdir) + " Fangnan.png", lap)

            # 与原图做叠加

            stacked\_lap = np.stack((lap, ) \* 3, axis=-1)

            added\_img = cv2.add(c\_image, stacked\_lap)

            cv2.imwrite(

                str(\*image\_name) + " " + str(fdir) + " Fangnan ADD.png", added\_img)