**实验二：数字图像频域增强实验**

**作业一：彩色图像的频域滤波器**

问题1：采用高斯低通滤波器对彩色图像进行滤波操作, 取半径为5, 20, 50, 80和250, 分别输出空域和频域的结果图像。

def fft\_lowpass(self, index, radius=250):

    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)

        # 使用快速傅里叶变换求频域

        r\_f = np.fft.fft2(r)

        g\_f = np.fft.fft2(g)

        b\_f = np.fft.fft2(b)

        # 将低频信息移到图像的中间位置

        r\_fshift = np.fft.fftshift(r\_f)

        g\_fshift = np.fft.fftshift(g\_f)

        b\_fshift = np.fft.fftshift(b\_f)

        # 创建蒙版

        rows, cols, \_ = image.shape

        crow, ccol = rows // 2, cols // 2

        if radius > crow or radius > ccol:

            radius = min(crow, ccol)

        mask = np.zeros((rows, cols), np.uint8)

        mask[crow - radius:crow + radius, ccol - radius:ccol + radius] = 1

        r\_ffshift = r\_fshift \* mask

        g\_ffshift = g\_fshift \* mask

        b\_ffshift = b\_fshift \* mask

        ffshift = cv2.add(r\_ffshift, g\_ffshift, b\_ffshift)

        # 构建振幅图

        r\_magnitude\_spectrum = 20 \* np.log(np.abs(r\_ffshift) + 1) / 3

        g\_magnitude\_spectrum = 20 \* np.log(np.abs(g\_ffshift) + 1) / 3

        b\_magnitude\_spectrum = 20 \* np.log(np.abs(b\_ffshift) + 1) / 3

        magnitude\_spectrum = cv2.add(b\_magnitude\_spectrum,

                                        g\_magnitude\_spectrum,

                                        r\_magnitude\_spectrum)

        cv2.imwrite(

            str(\*image\_name) + " " + str(radius) + " Lowpass (FD).png",

            magnitude\_spectrum)

        # 逆向傅里叶变换

        r\_fishift = np.fft.ifftshift(r\_ffshift)

        g\_fishift = np.fft.ifftshift(g\_ffshift)

        b\_fishift = np.fft.ifftshift(b\_ffshift)

        r\_i = np.fft.ifft2(r\_fishift)

        g\_i = np.fft.ifft2(g\_fishift)

        b\_i = np.fft.ifft2(b\_fishift)

        r\_i = np.abs(r\_i)

        g\_i = np.abs(g\_i)

        b\_i = np.abs(b\_i)

        image\_back = cv2.merge((b\_i, g\_i, r\_i))

        cv2.imwrite(

            str(\*image\_name) + " " + str(radius) + " Lowpass.png",

            image\_back)

问题2：自行选择一种频域的高通滤波器对彩色图像进行滤波操作, 取3组不同的参数进行实验，根据实验效果进行参数的比较分析。

def fft\_highpass(self, index, radius=50):

    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)

        # 使用快速傅里叶变换求频域

        r\_f = np.fft.fft2(r)

        g\_f = np.fft.fft2(g)

        b\_f = np.fft.fft2(b)

        # 将低频信息移到图像的中间位置

        r\_fshift = np.fft.fftshift(r\_f)

        g\_fshift = np.fft.fftshift(g\_f)

        b\_fshift = np.fft.fftshift(b\_f)

        # 创建蒙版

        rows, cols, \_ = image.shape

        crow, ccol = rows // 2, cols // 2

        if radius > crow or radius > ccol:

            radius = min(crow, ccol)

        mask = np.ones((rows, cols), np.uint8)

        mask[crow - radius:crow + radius, ccol - radius:ccol + radius] = 0

        r\_ffshift = r\_fshift \* mask

        g\_ffshift = g\_fshift \* mask

        b\_ffshift = b\_fshift \* mask

        # 构建振幅图

        r\_magnitude\_spectrum = 20 \* np.log(np.abs(r\_ffshift) + 1) / 3

        g\_magnitude\_spectrum = 20 \* np.log(np.abs(g\_ffshift) + 1) / 3

        b\_magnitude\_spectrum = 20 \* np.log(np.abs(b\_ffshift) + 1) / 3

        magnitude\_spectrum = cv2.add(b\_magnitude\_spectrum,

                                        g\_magnitude\_spectrum,

                                        r\_magnitude\_spectrum)

        cv2.imwrite(

            str(\*image\_name) + " " + str(radius) + " Highpass (FD).png",

            magnitude\_spectrum)

        # 逆向傅里叶变换

        r\_fishift = np.fft.ifftshift(r\_ffshift)

        g\_fishift = np.fft.ifftshift(g\_ffshift)

        b\_fishift = np.fft.ifftshift(b\_ffshift)

        r\_i = np.fft.ifft2(r\_fishift)

        g\_i = np.fft.ifft2(g\_fishift)

        b\_i = np.fft.ifft2(b\_fishift)

        r\_i = np.abs(r\_i)

        g\_i = np.abs(g\_i)

        b\_i = np.abs(b\_i)

        image\_back = cv2.merge((b\_i, g\_i, r\_i))

        cv2.imwrite(

            str(\*image\_name) + " " + str(radius) + " Highpass.png",

            image\_back)

**作业二：灰度和彩色图像的快速傅立叶变换**

问题1（对输入的灰度和彩色图像进行快速傅立叶变换,显示频谱图;）&问题2（进行逆变换,观察结果。）

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)