程序报告

一、问题重述

图像是一种非常常见的信息载体，但是在图像的获取、传输、存储的过程中可能由于各种原因使得图像受到噪声的影响。
如何去除噪声的影响，恢复图像原本的信息是计算机视觉中的重要研究问题。

常见的图像恢复算法有基于空间域的中值滤波、基于小波域的小波去噪、基于偏微分方程的非线性扩散滤波等，在本次实验中，我们要对图像添加噪声，并对添加噪声的图像进行基于模型的去噪。

A. 生成受损图像。

- 受损图像（X）是由原始图像（$I \in R^{H*W*C}$）添加了不同噪声遮罩（noise masks）（$M \in R^{H*W*C}$）得到的（$X = I \bigodot M$），其中$\bigodot$是逐元素相乘。
- 噪声遮罩仅包含 {0,1} 值。对原图的噪声遮罩的可以每行分别用 0.8/0.4/0.6 的噪声比率产生的，即噪声遮罩每个通道每行 80%/40%/60% 的像素值为 0，其他为 1。

B. 使用你最擅长的算法模型，进行图像恢复。

C. 评估误差为所有恢复图像（R）与原始图像（I）的 2-范数之和，此误差越小越好。
$\text{error} = \sum_{i=1}^{3} \text{norm}(\text{R}_i(:) - \text{I}_i(:), 2)$，其中(:)是向量化操作，其他评估方式包括 Cosine 相似度以及 SSIM 相似度。

二、设计思路

noise_mask_image()

输入的`noise_ratio`是一个有三个值的列表，分别是channel1、2、3的ratio。

对于每一个像素（按照行`i`，列`j`来遍历）的每一个channel（按`k`来遍历），生成一个随机数。如果这个随机数比该channel对应的ratio小，将`noise_img[i][j][k]`设为0，否则，设为`img[i][j][k]`

restore_image()

对于图片中的每一个像素点而言，都有3个channel。

对于每一个像素点中的每一个channel而言，都需要在以其为中心、边长为`size*2`的正方形区域内，寻找未被noise影响的样本点作为训练集，来对该像素点进行线性拟合。

三、代码内容

noise_mask_image()

```
def noise_mask_image(img, noise_ratio):
    """
    根据题目要求生成受损图片
    :param img: 图像矩阵，一般为 np.ndarray
    :param noise_ratio: 噪声比率，可能值是0.4/0.6/0.8
    :return: noise_img 受损图片, 图像矩阵值 0-1 之间，数据类型为 np.array,
             数据类型对象 (dtype): np.double, 图像形状:(height,width,channel),通道(channel) 顺序为RGB
    """
```

```
    # 受损图片初始化
    noise_img = None

    # -------------实现受损图像答题区域-----------------
    height,width,channel=img.shape

    for i in range(height):
        for j in range(width):
            for k in range(channel):
                r=random.random()
                if r<noise_ratio[k]:
                    noise_img[i][j][k]=0
                else:
                    noise_img[i][j][k] = img[i][j][k]

    # ------------------------------------------------

    return noise_img

```

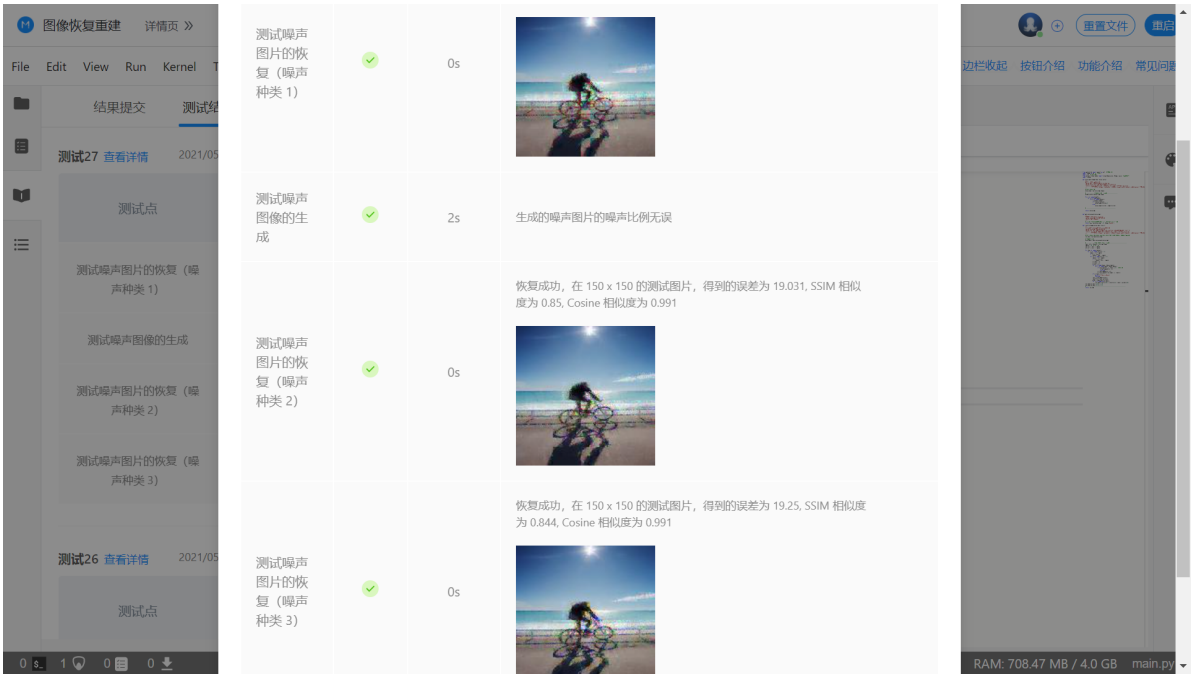
restore_image()

```
def restore_image(noise_img, size=4):
    """
    使用 区域二元线性回归模型 进行图像恢复。
    :param noise_img: 一个受损的图像
    :param size: 输入区域半径，长宽是以 size*size 方形区域获取区域，默认是 4
    :return: res_img 恢复后的图片，图像矩阵值 0-1 之间，数据类型为 np.array,
            数据类型对象 (dtype): np.double, 图像形状:(height,width,channel), 通道(channel) 顺序为RGB
    """
    # 恢复图片初始化，首先 copy 受损图片，然后预测噪声点的坐标后作为返回值。
    res_img = np.copy(noise_img)

    # 获取噪声图像
    noise_mask = get_noise_mask(noise_img)

    # -------------实现图像恢复代码答题区域----------------------------
    rows, cols, channel = res_img.shape
    region = 10  # 10 * 10
    row_cnt = rows // region
    col_cnt = cols // region

    # channel
    for chan in range(channel):
        # row
        for rn in range(row_cnt + 1):
            ibase = rn * region
            if rn == row_cnt:
                ibase = rows - region
            # col
            for cn in range(col_cnt + 1):
                jbase = cn * region
                if cn == col_cnt:
                     jbase = cols - region
                # x是训练集
                x_train = []
```

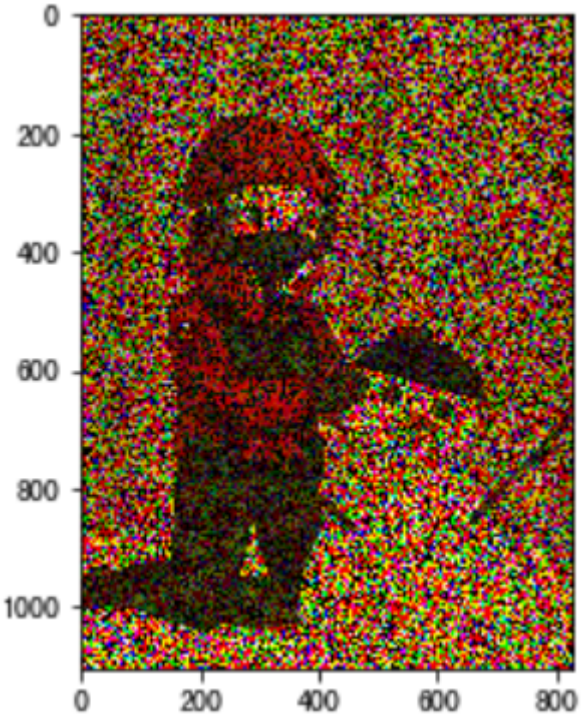
```
                # y是带了label的训练集
                y_train = []
                x_test = []

                # 对于每一个像素而言，都需要在以其为中心、
                # 边长为`size*2`的正方形区域内，寻找未被noise影响的样本点作为训练集，
                # 来对该像素点进行线性拟合。
                for i in range(ibase, ibase+region):
                    for j in range(jbase, jbase+region):
                        # 被noise影响的点，不算在训练集之内。
                        if noise_mask[i, j, chan] == 0:
                            x_test.append([i, j])
                            continue
                        x_train.append([i, j])
                        y_train.append([res_img[i, j, chan]])
                if x_train == []:
                    print("x_train is None")
                    continue

                #线性回归
                reg = LinearRegression()
                #训练
                reg.fit(x_train, y_train)
                pred = reg.predict(x_test)
                for i in range(len(x_test)):
                    res_img[x_test[i][0], x_test[i][1], chan] = pred[i][0]
    res_img[res_img > 1.0] = 1.0
    res_img[res_img < 0.0] = 0.0
    # --------------------------------------------------------------
    return res_img
```

四、实验结果

在mo平台上的结果如下：



用A.png测试，输出的噪声图片如下：



五、总结

是否达到目标预期：是

可能改进的方向：提高算法性能

实验过程中遇到的困难：对于线性回归模型的一知半解，对于本实验中需要使用的几个包的不熟悉，刚开始对于题意也不太明白。