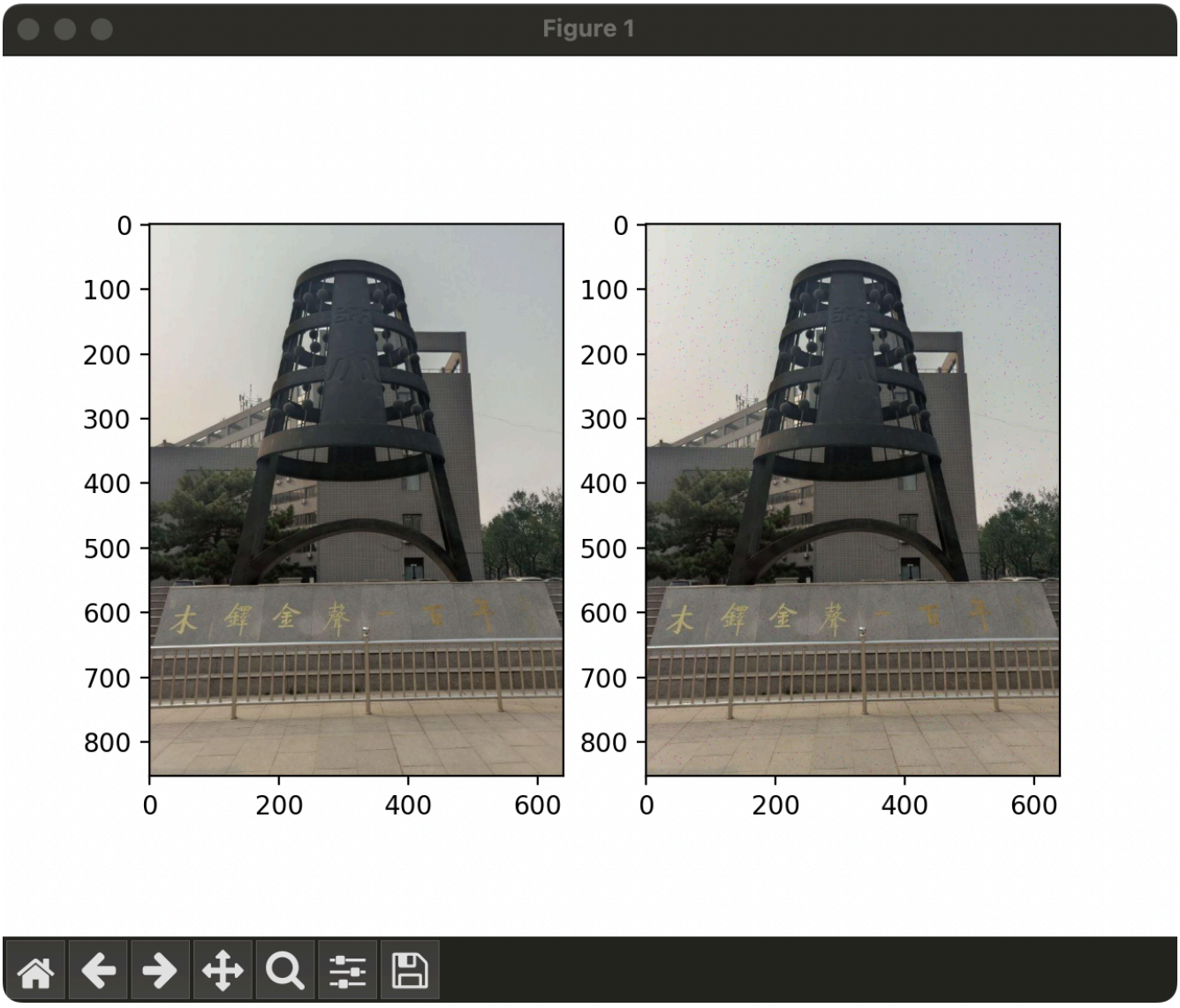
实验报告

实验四：基于区域多元线性回归的图像恢复

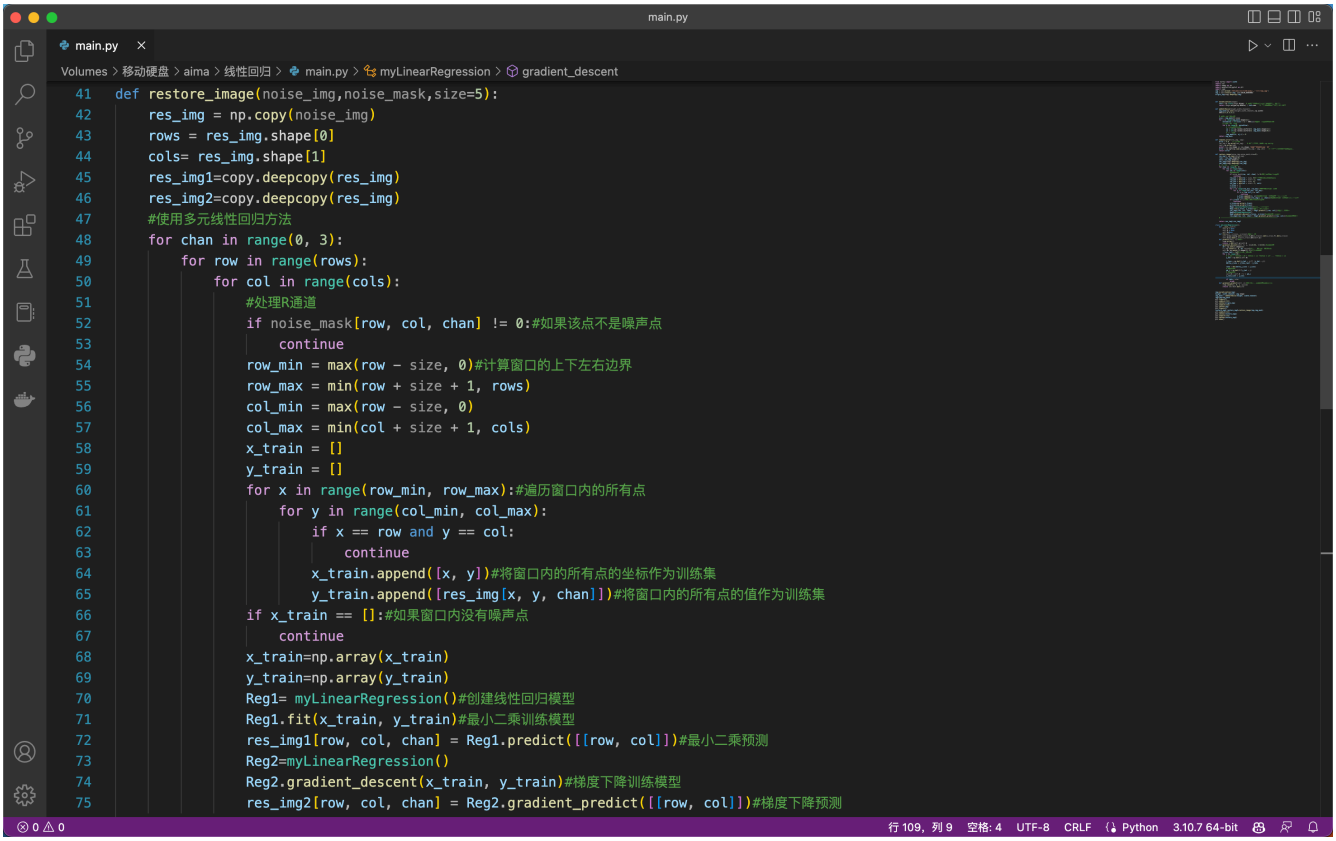
姓名　姚冠宇　　学号　202011260070　专业　人工智能

1. 为原始图像添加噪声的代码，并展示添加噪声后图像(两幅图的结果放在一起即可):

```
17  def addSaltNoise(height,width,channel):
18      img_mask=np.ones((height,width,channel),np.uint8)
19      SNR=[0.4,0.4,0.2]
20
21      # 获取总共像素个数
22      size = img_mask[0].size
23      for i in range(img_mask.shape[2]):
24          noiseSize = int(size * (1 - SNR[i]))#需要加噪声的点的个数
25          # 对这些点加噪声
26          for k in range(0, noiseSize):
27              # 随机获取某个点
28              xj = int(np.random.uniform(0, img_mask.shape[1]))
29              xi = int(np.random.uniform(0, img_mask.shape[0]))
30              # 增加噪声
31              img_mask[xi, xj,i] = 0
32      return img_mask
```



2. 使用区域多元线性回归对受噪图像进行复原(分别注明梯度下降、最小二乘，适当增加注释)：



修复图像的代码

```python
class myLinearRegression():
    def __init__(self):
        self.w = None
        self.b = None
        self.W=None
    def fit(self, x_train, y_train):#最小二乘
        self.w=np.linalg.inv(x_train.T.dot(x_train)).dot(x_train.T).dot(y_train)
        self.b=np.mean(y_train-x_train.dot(self.w))
    def predict(self, x):#预测
        x=np.array(x)
        return x.dot(self.w)+self.b
```

最小二乘线性回归代码⬆️

```python
    def gradient_descent(self, X, y, eta=0.01, lr=0.001):#梯度下降
        x0 = np.ones(X.shape[0])
        X = np.insert(X, 0, x0, axis=1)#在第一列插入一列1充当b
        self.W= np.zeros([X.shape[1],1])#初始化权重
        j_loss_last = 1e5#初始化损失函数
        for i in range(5000):
            # y_hat = theta0 * x0 + theta1 * x1 +theta2 * x2+ ... +thetan * xn
            y_hat = np.dot(X,self.W)

            j_loss = np.dot((y_hat - y).T, (y_hat - y))
            delta_j_loss = j_loss_last - j_loss

            rate = abs(delta_j_loss / j_loss)
            # 得到梯度
            pd_j = np.dot(X.T,y_hat - y)
            # 更新权重
            self.W = self.W - lr * pd_j
            j_loss_last = j_loss

            if rate < eta:
                break
    def gradient_predict(self, x):#预测（基于梯度下降法的结果）
        x=np.insert(x, 0, 1, axis=1)
        return self.W.T.dot(x.T)

```

梯度下降代码⬆️（学习率0.001，误差小于0.01时退出循环，迭代次数5000）
实际运行时选迭代次数50不然跑的时间太长了，但是最后结果不会很好因为迭代次数太少

3、计算复原后图像与真实图像之间的误差(展示复原结果、误差实现代码以及误差结果):

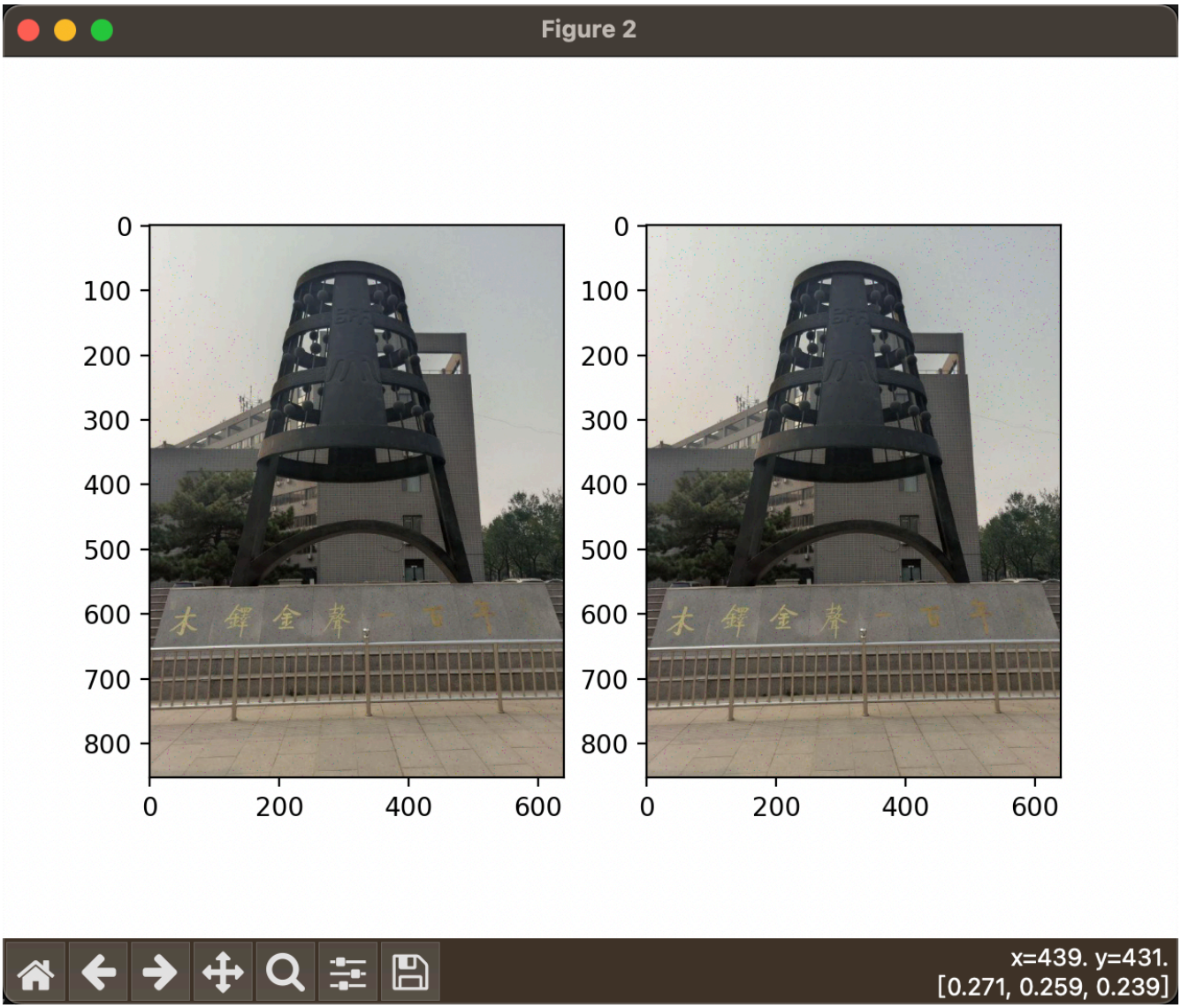
```python
def compute_error(res_img, img):
    error = 0.0    # 初始化
    res_img = np.array(res_img)    # 将图像矩阵转换成为np.narray
    img = np.array(img)
    assert res_img.shape == img.shape,'两张图片的尺寸不一致'
    error = np.sqrt(np.sum(np.power(res_img - img, 2)))    # 计算图像矩阵之间的评估误差
    return error
```

计算误差代码

误差结果

```
最小二乘法的均方误差为：  24.456376635786853
梯度下降法的均方误差为：  1.375149237310897e+71
```

复原结果（左最小二乘右梯度下降）



4、实验总结

本实验实现了基于线性回归的图像修复，主要原理是通过受损点周围一定半径内的像素点来预测当前图像的像素值。其中实现线性回归参数估计的方法有两种。基于最小二乘和基于梯度下降。最小二乘在矩阵较小时可以很快计算出结果，梯度下降法在矩阵求逆成本过高时有很好的效果，同时可以增加正则化项来减少过拟合的风险。

由于本实验受损像素点过多，故运用梯度下降法时需要的迭代次数较多计算较慢，这里为了快速出结果将迭代轮次调小了，想得到更好的结果可以调大迭代轮次。

最终结果来看最小二乘法基本上消除了噪声，说明方法的有效性。