Project 2: 图像恢复

序号	学号	专业班级	姓名	性别
1	**3140103367**	**软工 1401**	**杨奕辉**	**男**

1. Project Introduction

给定 3 张受损图像，尝试恢复他们的原始图像。

1. 原始图像包含 1 张黑白图像（A.png）和 2 张彩色图像（B.png, C.png）。
2. 受损图像（$\mathbf{X}$）是由原始图像（$\mathbf{I} \in \mathcal{R}^{H*W*C}$）添加了不同噪声遮罩（noise masks）（$\mathbf{M} \in \mathcal{R}^{H*W*C}$）得到的（$\mathbf{X} = \mathbf{I} \odot \mathbf{M}$），其中$\odot$是逐元素相乘。
3. 噪声遮罩仅包含{0,1}值。对应原图（A/B/C）的噪声遮罩的每行分别用 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。

2. Technical Details

操作系统：Windows 10
编程语言：Python 2.7
核心库：numpy,sklearn

算法核心思想：对图片每个像素点的每个通道进行遍历。若当前像素点的当前通道是受损点，则将其周围特定尺寸正方形内的像素点通道作为训练集，**做二维多项式线性回归**，得到拟合函数。最后，将受损点的行列坐标带入， 预测出该通道的值。

主要实验过程：（输入输出图像算法略过，图像点阵为 0-1 的 double 型矩阵）

1. 首先获取图像的受损点遮罩层，即一个 0,1 三维矩阵。

```
# get the noise mask of corrImg
def getNoiseMask(corrImg):
    return np.array(corrImg!=0,dtype='double')
```

2. 遍历像素点的每个通道，对每个通道进行以下处理。
   1) 格式化训练集：将像素点周围特定半径的像素点组装成一个二维数组。
      x_train:[[x1,y1],[x2,y2]….]
      y_train:[[y1],[y2]….]

```
x_train=[]
y_train=[]
for i in range(row-radius,row+radius):
    if i<0 or i>=rows:
        continue
    for j in range(col-radius,col+radius):
        if j<0 or j>=cols or noiseMask[i,j,chan] == 0.:
            continue
        if i==row and j==col:
            continue
        x_train.append([i,j])
        y_train.append([img[i,j,chan]])
```

这里半径的计算策略是对错误率*0.8 再取整。

2) 利用训练集做多项式线性回归：

```
# quadratic linear regression
quadratic=PolynomialFeatures(degree=3)
x_train_quadratic=quadratic.fit_transform(x_train)
regressor_quadratic=LinearRegression()
regressor_quadratic.fit(x_train_quadratic,y_train)
```

3) 使用训练出来的多项式预测受损像素点的通道的值：

```
# predict
test=quadratic.transform([[row,col]])
resImg[row,col,chan]=regressor_quadratic.predict(test)
```

3. 对所有像素点的所有通道遍历完并赋予新值，输出图像。
4. 评估修复图像与原图像的误差，采用 2-范数之和评估法，代码如下：

```
# evaluate error rate
def computeError(resImg,corrImg,filename):
    noiseRatio=samples[filename]
    try:
        im1=readImg(filename+'_ori.png').flatten()
        im2=corrImg.flatten()
        im3=resImg.flatten()
        print((
            '{}({}):\n'
            'Distance between original and corrupted: {}\n'
            'Distance between original and reconstructed (regression): {}'
        ).format(filename, noiseRatio, norm(im1-im2, 2), norm(im1-im3, 2)))
    except Exception as error:
        print error
```

3. Experiment Results

Readme:脚本采用了多进程修复图像，故控制台可能有多次不同进程的输出。

操作说明：

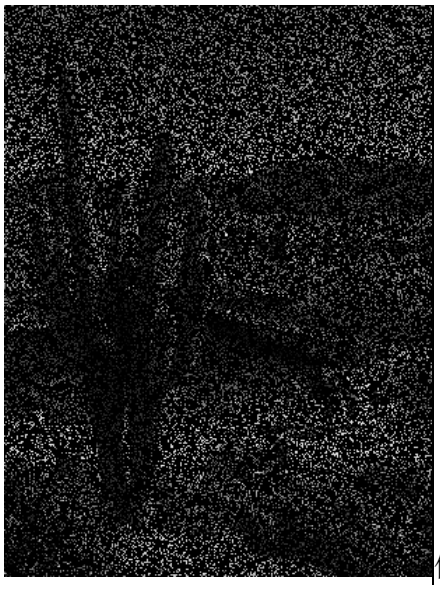
1. Run:命令行键入 python Lowes_restore.py 即可。默认修复’A’,’B’,’C’三张受损图像。若需要修改，请在该函数的 queue 数组中添加或替换**文件名字符串**即可。

```
def main():
    queue=['F']     # task img queue
    for img in queue:
        t=multiprocessing.Process(target=run,args=img)
        t.start()
```

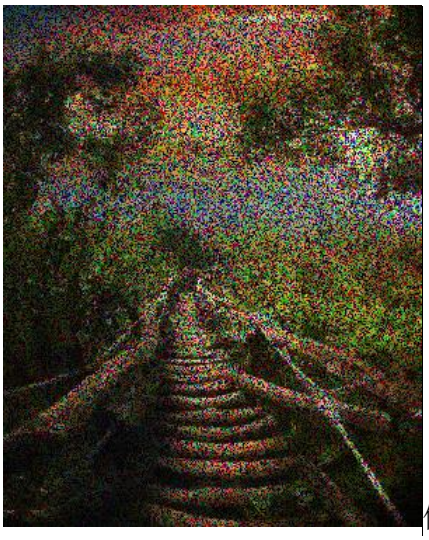
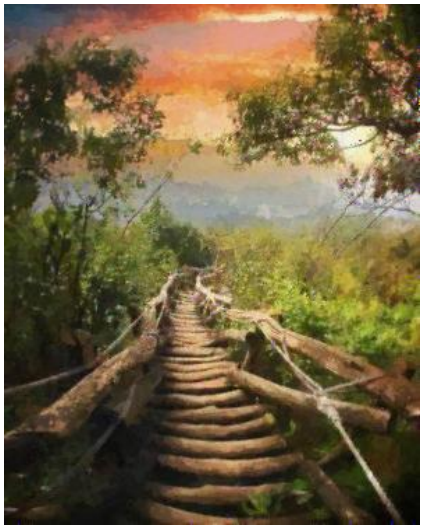
2. **评估误差：**修复完后会自动输出误差，也可以手动命令行键入 python computeError.py 即可。

修复结果：

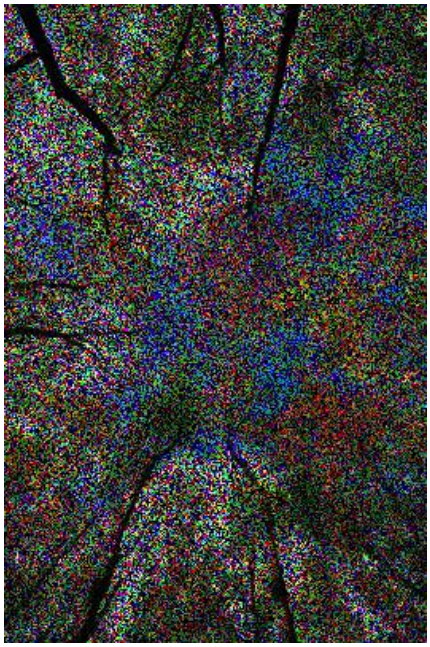
A.

修复前：  修复后： 

B.

修复前：  修复后： 

C.

修复前：  修复后： 

自用测试图片：（D，E，F）

图片在数据包中，下为评估误差：

```
D(0.8):
Distance between original and corrupted: 656.548186537
Distance between original and reconstructed (regression): 52.5881307437
```

```
E(0.8):
Distance between original and corrupted: 656.173666517
Distance between original and reconstructed (regression): 63.7936420801
```

```
F(0.8):
Distance between original and corrupted: 401.980127206
Distance between original and reconstructed (regression): 36.6220936487
```

References:

1. 《Python_sklearn 机器学习库学习笔记》
   http://www.cnblogs.com/wuchuanying/p/6264025.html
2. 《scikit-learn:线性回归，多元回归，多项式回归》
   http://blog.csdn.net/SA14023053/article/details/51703204
3. 《一种基于 HSV 空间的颜色相似度计算方法》 谢君廷、王小华 杭州电子科技大学计算机学院