

程序报告

学号: 3190100923

姓名: 陈志博

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。我们希望通过各种方法能实现对于已受损的图片的修复。我们通过对图片添加 rgb 随机噪声;来模拟图片的损失过程,然后通过对 rgb 噪声对去除来实现图片对修复工作。

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。

有空间滤波器,频域滤波器、线性回归、神经网络等方法对损坏的图片进行修复,不同的修复方法对于不同的是损失形式的效果也相应地不同。

二、设计思想

鉴于此次实验模拟的噪声图片是将 RGB 三个通道图层的随机像素点做遮盖,因此我们可以很轻易的从像素点数据中将经过遮盖的像素点提取出来。将没有进行遮盖的点集作为训练数据集,然后把训练出的模型套用到剩下的经过遮盖处理的噪声点处。为了兼顾效率和回复效果。我采用的是把图片逐小块小块的进行回归处理来预测各小块中的噪声点的原始数据。主函数中主要调用来 sklearn 库中的 linearregression 模块来做线性回归的运算。

三、代码内容

mask 生成模块的代码:

```
def noise_mask_image(img, noise_ratio=[0.8,0.4,0.6]):
```

```
    noise_img = None
    x,y=img.shape[:2]
    noise_img = np.random.rand(x,y,3)
    for i in range(3):
        noise_img[:, :, i]=np.where(noise_img[:, :, i]<noise_ratio[i],0,1)
    noise_img=img*noise_img
    return noise_img
```

恢复模块的代码:

```
from sklearn import linear_model
def restore_image(noise_img, size=4):
```

```

# 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)

height,width=noise_img.shape[:2]
#print(height,width)
reg=linear_model.LinearRegression(fit_intercept=True,normalize=False)
for i in range(height-size+1):

    for j in range(width-size+1):
        for l in range(3):
            x=[[i+k,j+t] for k in range(size) for t in range(size) if i+k<height and
j+t<width and noise_mask[i+k,j+t,l]==1]
            y=[noise_img[i+k,j+t,l] for k in range(size) for t in range(size) if i+k<height
and j+t<width and noise_mask[i+k,j+t,l]==1]
            #print(x)
            #print(y)
            #print('#####')
            if len(y)>0:
                reg.fit(x,y)
                k=reg.coef_
                #print(k)
                for a in [[i+k,j+t] for k in range(size) for t in range(size) if i+k<height
and j+t<width and noise_mask[i+k,j+t,l]==0]:
                    res_img[a[0],a[1],l]=reg.predict(np.array(a).reshape((1, -1)))
res_img=np.where(res_img>1,1,res_img)
res_img=np.where(res_img<0,0,res_img)
return res_img

```


四、实验结果

the noise_ratio = [0.4, 0.6, 0.8] of original image



恢复图片与原始图片的评估误差: 73.864
恢复图片与原始图片的 SSIM 相似度: 0.7268151975995507
恢复图片与原始图片的 Cosine 相似度: 0.998102002945583

restore image



测试噪声 图片的恢 复 (噪声 种类 2)	✓	39s	恢复成功, 在 150 x 150 的测试图片, 得到的误差为 21.161, SSIM 相 似度为 0.851, Cosine 相似度为 0.989 
测试噪声 图片的恢 复 (噪声 种类 1)	✓	42s	恢复成功, 在 150 x 150 的测试图片, 得到的误差为 26.447, SSIM 相 似度为 0.78, Cosine 相似度为 0.983 
测试噪声 图片的恢 复 (噪声 种类 3)	✓	40s	恢复成功, 在 150 x 150 的测试图片, 得到的误差为 22.004, SSIM 相 似度为 0.837, Cosine 相似度为 0.988 

五、总结

实验主要思想是将输入的噪声图片逐小块小块地分割后对个小块进行线性回归。实质上是一种将图片各小块的像素点进行平滑化处理的过程。同时这一方法的实现也是基于所有的噪声 mask 都是把像素点的值变为 0 的基础上。一开始测试样例给出的图片噪声存在有不为零的情况, 这样的话对其回归化处理后图片与原图的相似度反而下降了。对于图片中存在非零噪声的情况就无法简单的将有噪声的点和无噪声的点分割出来了, 需要一些其它的方法来对其进行修复。