第一次大作业技术报告

- **环境配置**
  - visual studio 2022
  - C++ OpenCV 3.4.1

概述

- 本次作业完成了以下的工作：
  - 首先使用OpenCV将**图片读入**，其次利用亮度公式完成彩色图片到**灰度图片的转化**
  - 利用离散直方图均衡公式完成图片的**均衡化**
  - 利用**线性变换**完成灰度修正，给出了全范围变换和部分区间变换的样例
  - 利用**非线性变换（指数变换）**完成灰度修正
- 在完成后将生成样例进行对比，并从原理层面给出了解释

图片读入

- OpenCV的 **imread()** 函数可以实现将图片读入，并保存为一个Mat类型变量。其中**Mat**类型变量就是一个矩阵，其data元素是指向像素数组的指针，通过对于相应内存的读写，便可以完成对于图片的读取、复制、更改。

```
Mat in_img;
Mat grey_img(size512, size512, CV_8UC1);
in_img = imread("D:\\files\\北航\\大三下\\图像处理\\snow.jpg", 1);
unsigned char* ptr1 = grey_img->data;
```

灰度转化

- 将图片读取为Mat之后，由于是彩色图片，其像素有三个通道，按顺序分别是**BGR**，依次排列在内存中。而将其转化为灰度图片的思路如下：
  - 新建一个空的Mat对象，其size与原图片相等
  - 利用公式，将每个像素的BGR数值带入，计算出亮度值，保存在新图片的对应像素位置

$$Y（灰度）=0.3R+0.59G+0.11B$$

    所得到的Mat对象如果用单通道显示，便是原图像的灰度图，至此完成了转化。

```
void change2grey(Mat* in_img, Mat* grey_img) {

    unsigned char* optr = in_img->data;
    unsigned char* nptr = grey_img->data;

    for (int i = 0; i < size512 ; i++) {
        for (int j = 0; j < size512 ; j++) {
            *nptr = 0.11 * (float)*optr + 0.59 * (float)*(optr+1) + 0.3
* (float)*(optr+2); //转换
            optr += 3;
            nptr += 1;
        }
    }
}
```

直方图均衡化

- 直方图均衡化实际上能有两个作用
  - 其首先能将不同灰度的像素点的数量进行**均衡**，灰度分配更为均匀
  - 也可以将原先没有使用的灰度曲线进行使用，实际上也完成了**全域线性**灰度修正
- 按照老师给出的ppt，实现均衡化有如下的步骤：
  - 首先**统计**出 0-255 灰度区间上 每个值对应的像素点的数量，并完成概率计算
  - 利用**离散均衡公式**，得到每个 旧灰度 值对应的 新灰度值

$$S_k(\text{新的灰度}) = \sum_{j=0}^{k} \frac{n_j}{N} \times 255$$

  - 将原灰度图的点，转化为新灰度值，**映射**到新图上，完成均衡化

```
statistic(&grey_img, grey_num);
void balance(int* data, Mat* grey, Mat* balance) {
    int sum = 0;
    float p[256] = { 0 };
    float balance_range[256] = {0};
    for (int i = 0; i < 256; i++) {
        sum = 0;
        for (int j = i; j >= 0; j--) {
            sum += data[j];                                    //统计个数
        }
        p[i] = (float)sum / (float)(size512 * size512);        //计算概率
        balance_range[i] = p[i] * 255;                         //计算新灰度
    }
    unsigned char* ptr1 = grey->data;
    unsigned char* ptr2 = balance->data;
    for (int i = 0; i < size512; i++) {
        for (int j = 0; j < size512; j++) {
            *ptr2 = balance_range[(int)*ptr1];                 //完成映射
            ptr1++;
            ptr2++;
        }
    }
}
```

线性变换&非线性变换

- **线性变换**
  - 原理比较简单，类似均衡化，但是直接操作像素点灰度，完成计算，映射给新图像即可。其中涉及到新图像灰度的区间，可以自己设定。

$$g(x,y)=\frac{d-c}{b-a}[f(x,y)-a]+c$$

```
void linear_stretch(int* data, Mat* grey, Mat* linear) {
    float min = 0, max = 0;
    //找到最小、最大灰度值
    unsigned char* ptr1 = grey->data;
    unsigned char* ptr2 = linear->data;
    for (int i = 0; i < size512; i++) {
        for (int j = 0; j < size512; j++) {
            *ptr2 = (float)((*ptr1 - min)/(max-min)*(255-155))+155; //完
成映射
            ptr1++;
            ptr2++;
        }
    }
}
```

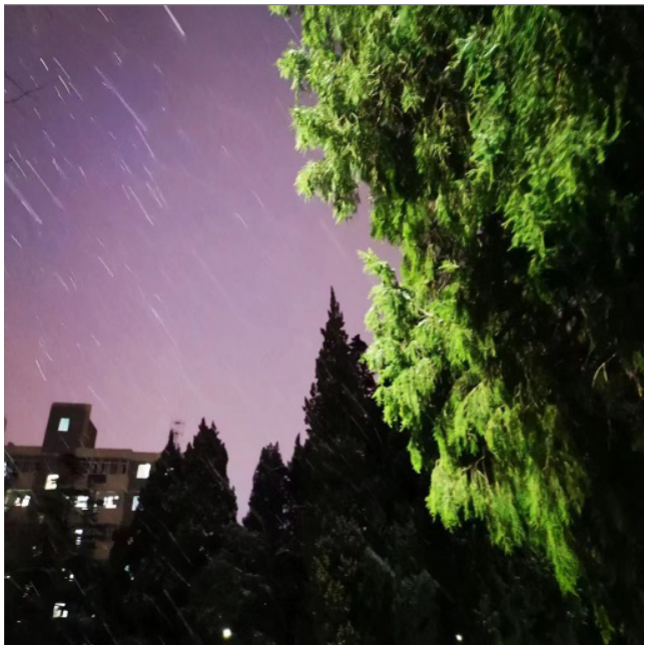
- **非线性变换**
  - 采用的是**指数变换**。先用每个点的灰度值映射到0-1，指数计算后映射即可。但是指数需要自己判断。

$$g(x,y)=f(x,y)^{\gamma}$$

```
void nonelinear_stretch(int* data, Mat* grey, Mat* nonelinear) {
    unsigned char* ptr1 = grey->data;
    unsigned char* ptr2 = nonelinear->data;
    for (int i = 0; i < size512; i++) {
        for (int j = 0; j < size512; j++) {
            *ptr2 = (float)pow((float)*ptr1 / (float)255, 0.5) * 255;
//完成计算
            ptr1++;
            ptr2++;
        }
    }
}
```

结果对比&分析

- **样例对比**
  - 原图



- 灰度图

- 均衡化

- 线性变换（155-255）

- 指数变换 指数为0.5

- **样例分析**
  - 可以看到，均衡化之后的图片对比原图对比度更加明显，下方松树的纹理细节清晰可见。但是由于均衡化，导致树叶边缘、夜空过亮，有一些不自然。
  - 线性变换就是整体变量（变暗），但是保留了原来的图片结构。
  - 指数变换效果很好。0.5的指数使得亮的地方变化不大，但是暗的地方变亮的很明显。对比均衡化，其效果更加自然，夜空和树叶边缘也没有过亮。

总结

- 在本次大作业中，我理解了一个图片的具体存储细节，掌握了基本的读取、保存、更改图片的方法。
- 其次我明白了三通道RGB图像和灰度图像间的转化关系，证实了公式的正确性。
- 进一步，我实践了课上所讲的 均衡化、线性变化、非线性变化 等灰度图片变化方式，加深了理解。