作业要求

打开一幅图像，进行直方图均衡．将灰度线性变化，将灰度拉伸．

打开图像

原始图像为jpg图像，经查阅资料，opencv作为优秀的图像处理开源库集成了图像打开的办法

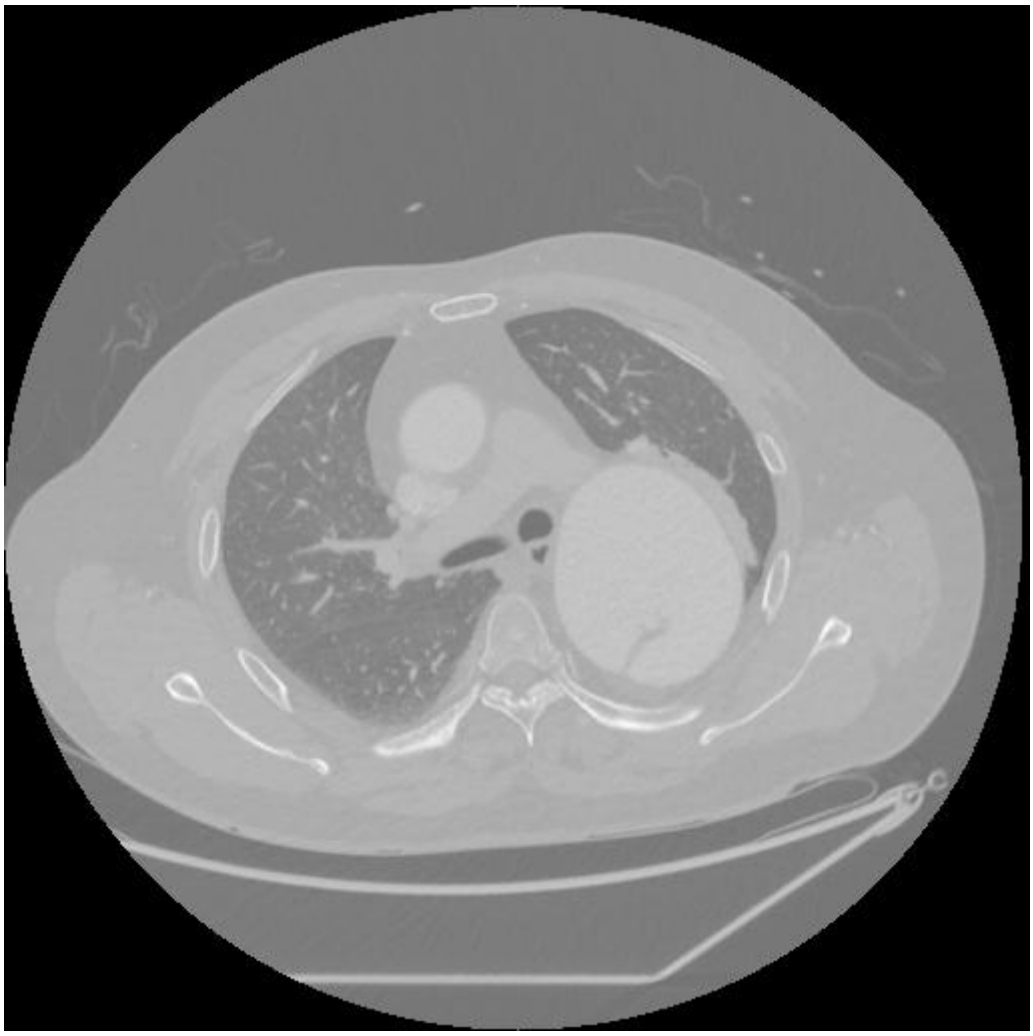
```
cv::Mat img = cv::imread(path, 0);
```

这次作业，主要使用了mat对象的`at(i,j)`取矩阵中的值，使用`mat.cols, mat.rows`取得长和宽，使用`mat.ptr(i)[j]`修改矩阵元素

图片的展示使用

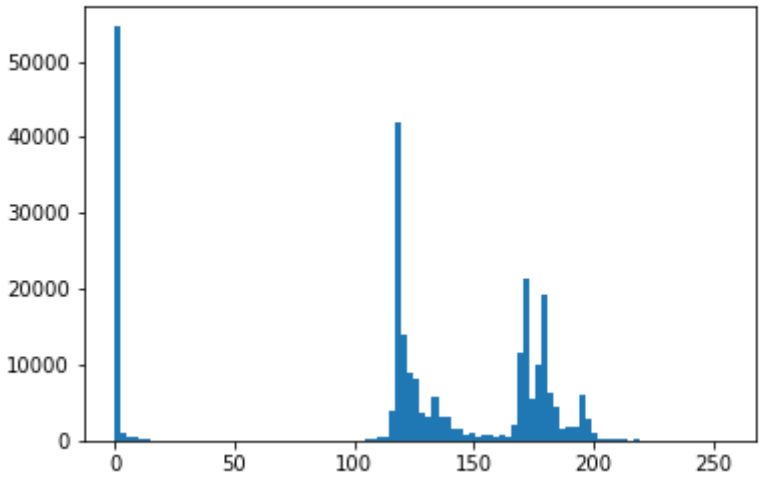
```
namedWindow("before");
imshow("before", img);
waitKey(0);
```

原图如图所示：



原图是经典的医学图像，较亮部分由于padding数值过小而特征不明显

采用其他手段获取图像的直方图如下：



实现直方图均衡：

算法思路：

首先统计每个灰度值出现的次数，从而计算得到每个灰度值的概率，以及累计概率。

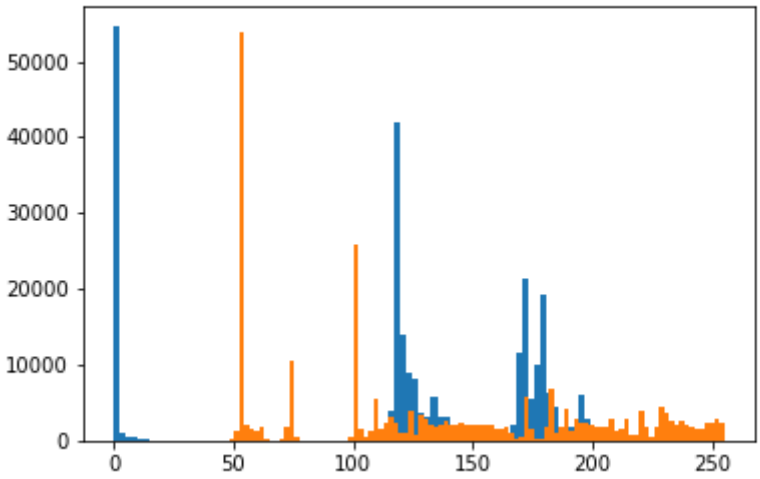
`after_gray = 255.0 * sum_prob[ori_gray]+0.5`

其中0.5是为了四舍五入，将灰度分布根据概率重新拉伸。

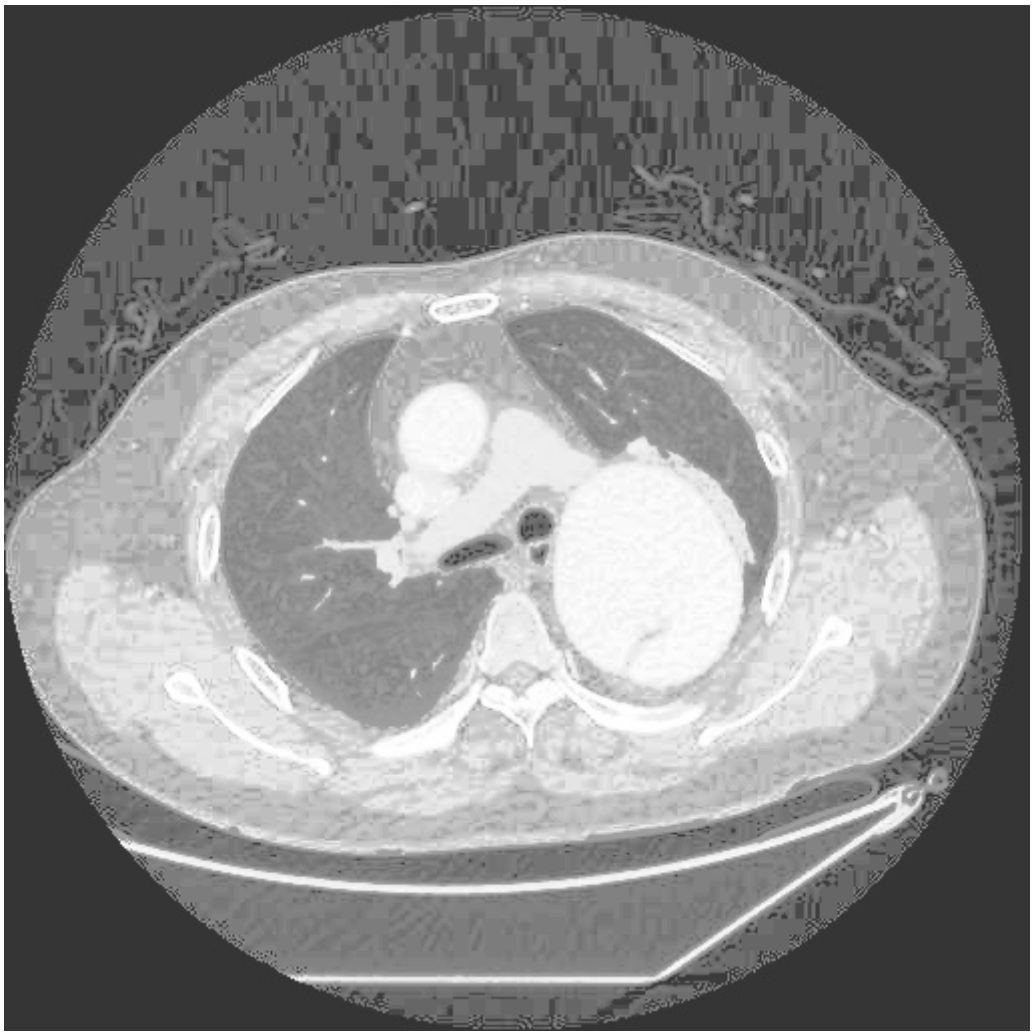
```
int histogram[256] = { 0 };
    double gray_prob[256] = { 0 };
    double gray_sum[256] = { 0 };

    for (int i = 0; i < W; i++) {
        for (int j = 0; j < H; j++) {
            if (img.at<uchar>(i, j) < 0 || img.at<uchar>(i, j) > 255)
                cout << img.at<uchar>(i, j) << " overflower" << endl;
            else histogram[img.at<uchar>(i, j)] ++;
        }
    }
    double prob = 0;
    double sum_prob = 0;
    for (int i = 0; i < 256; i++) {
        gray_prob[i] = prob = 1.0 * histogram[i] / (W * H);
        sum_prob += prob;
        gray_sum[i] = sum_prob;
        //cout << gray_sum[i] << endl;
    }
    Mat balance = img.clone();
    for (int i = 0; i < W; i++) {
        uchar* p = balance.ptr<uchar>(i);
        for (int j = 0; j < H; j++) {
            uchar after_gray = gray_sum[img.at<uchar>(i, j)] * 255 + 0.5; //四舍
五入
            // cout << int(after_gray) << "  ";
            p[j] = after_gray;
        }
    }
```

下图为均衡后的直方图对比：



可以看到图像在保留原有分布的基础上，各种灰度值的概率大致相等了。但由于0灰度像素太多，效果仍不理想。变化后的图像如下，可以看到细节更加清晰，但部分不必要的细节变成了噪音：



线性变换

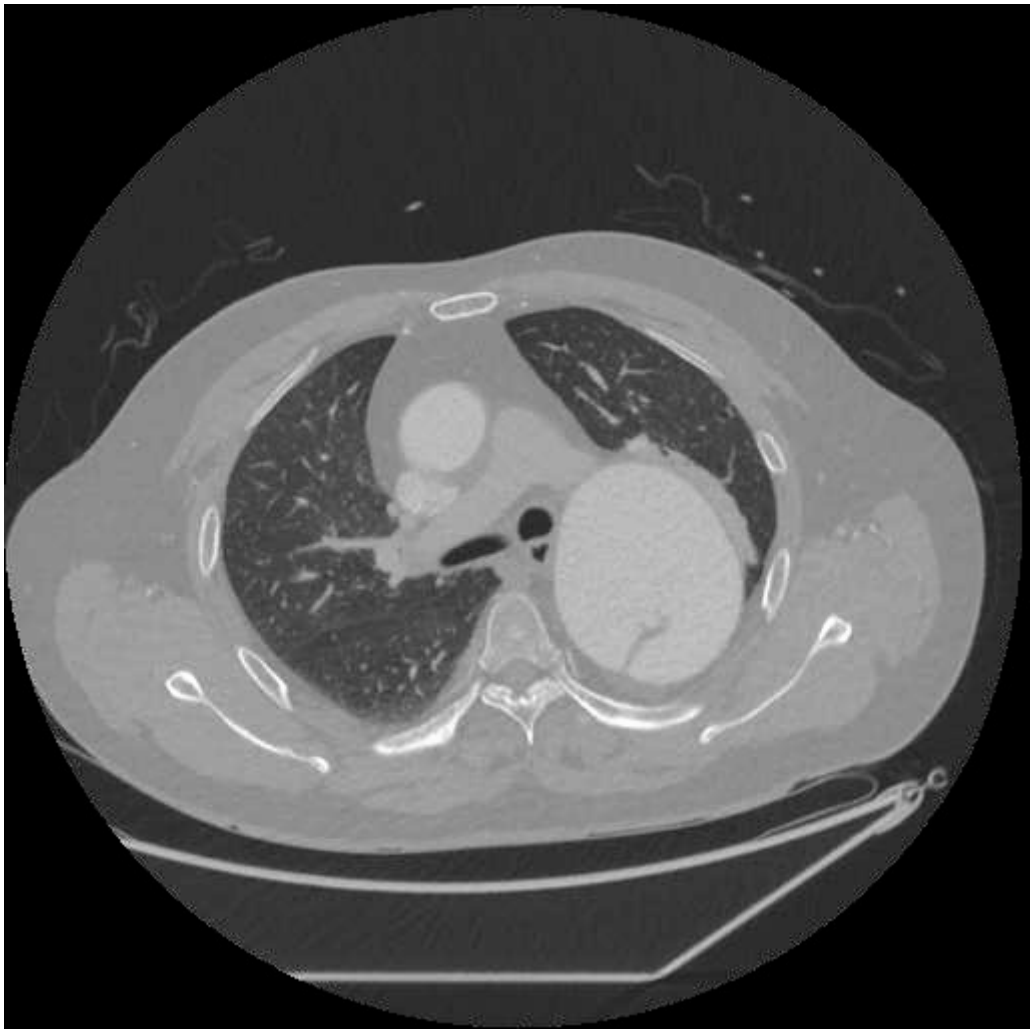
算法思路：

基于直方图，我们看到低于100的部分几乎只有0，也就是底色，这不是我们关注的重点，高于230的也几乎没有。但是为了保留这部分信息，我们采用分段函数。低于100的重映射到[0,10]高于230重映射到[250,255]，原[100,230]重映射到[10,250]

```
void draw_line(Mat& img, int min_be, int max_be, int min_af, int max_af) {
    Mat liner_img = img.clone();
    for (int i = 0; i < W; i++) {
        uchar* p = liner_img.ptr<uchar>(i);
        for (int j = 0; j < H; j++) {
            if (img.at<uchar>(i, j) <= min_be)
                p[j] = (uchar)(1. * min_af * img.at<uchar>(i, j) / min_be);
            else if (img.at<uchar>(i, j) >= max_be)
                p[j] = (uchar)(1. * (255 - max_af) * (img.at<uchar>(i, j) -
max_be) / (255 - max_be) + max_af);
            else
                p[j] = (uchar)(1. * (max_af - min_af) * (img.at<uchar>(i, j) -
min_be) / (max_be - min_be) + min_af);

        }
    }
    //show img
}
```

转化后的图像如下。可以看到我们最想关注的部分有了质的提升，且依然保留了不重要信息。

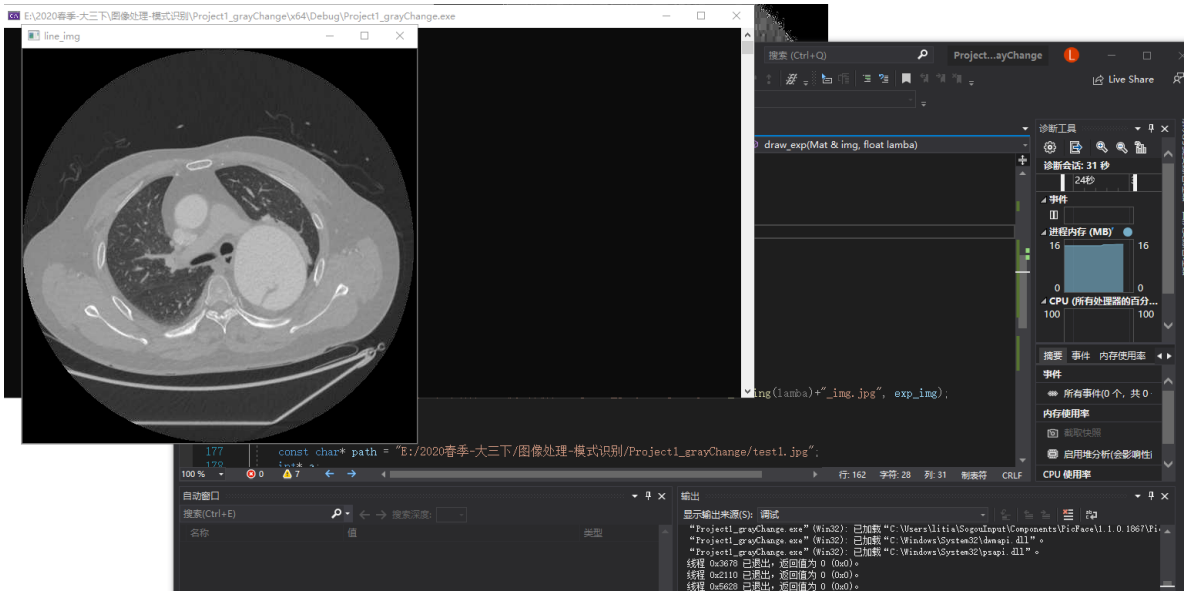


非线性变换：

除了线性变换，我还尝试了指数变换，发现在lamba=2.0的时候，图像增强效果较好，但是鲁棒性较差，引入许多噪声。如图：

```
Mat exp_img = img.clone();
    for (int i = 0; i < W; i++) {
        uchar* p = exp_img.ptr<uchar>(i);
        for (int j = 0; j < H; j++) {
            uchar pixel = (uchar)(pow(img.at<uchar>(i, j), lamba));
            p[j] = pixel;
        }
    }
```

运行过程截图：


E:\2020春季-大三下\图像处理-模式识别\Project1_grayChange\x64\Debug\Project1_grayChange.exe
line_img
draw_exp(Mat & img, float lamba)
const char* path = "E:/2020春季-大三下/图像处理-模式识别/Project1_grayChange/test1.jpg";
诊断工具
诊断会话: 31 秒
事件
进程内存 (MB)
CPU (所有处理器的百分...
自动窗口
输出
显示输出来源(S): 调试
线程 0x3678 已退出，返回值为 0 (0x0)。
线程 0x2110 已退出，返回值为 0 (0x0)。