实验目的：

通过p率阈值化对图像进行二值处理

实验原理：

基于图像直方图选择阈值T，使得1/p的图像面积具有比T小的灰度值。

实验步骤：

1. 将彩色图转化为灰度图
2. 求出图像的直方图
3. 确定p的值
4. 通过不断调整阈值T，使得图像中灰度值小于T的像素点数量与总像素点数量之比为1/p
5. 对图像进行遍历，将灰度值小于T的像素点的灰度设为0，将灰度值大于T的像素点的灰度设为255
6. 保存并输出新生成的图像

程序代码：

#include <iostream>

#include <fstream>

#include "opencv2/opencv.hpp"

#include "opencv2/imgproc/imgproc.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

using namespace cv;

using namespace std;

int sumchar(int a[256], int j) {//构造一个求数组前j项和的函数

int p = 0;

for (int i = 0; i < j; i++) {

p = p + a[i];

}

return p;

}

Mat myThresholdP(Mat img)

{

Mat ThresholdPImg;

ThresholdPImg = img;

int H[256] = { 0 };//创建一维数组

for (int i = 0; i < img.rows; i++) {//统计各个灰度值的数目

for (int j = 0; j < img.cols; j++) {

int tmp = img.at<uchar>(i, j);

for (int p = 0; p < 256; p++) {

if (p == tmp) { H[p]++; }

}

}

}

float p;//设置p率的值，由用户自己输入

cout << "请输入p率的值：p = " << endl;

cin >> p;

int T;

int tmp=0;

int sum=0;

for (int i = 0; i < 256; i++) {//求出所有像素点的个数

sum = sum + H[i];

}

int u =-1;

for (int i = 0; (i < 256)&&(u<0); i++) {

u = sumchar(H, i) \* p - sum; T = i;//检测是否达到了要求的阈值，从而确定T

};

for (int i = 0; i < ThresholdPImg.rows; i++) {

for (int j = 0; j < ThresholdPImg.cols; j++) {

if (ThresholdPImg.at<uchar>(i, j) < T) { ThresholdPImg.at<uchar>(i, j) = 0; };//对图像进行二值化处理

if (ThresholdPImg.at<uchar>(i, j) >= T) { ThresholdPImg.at<uchar>(i, j) = 255; };

}

}

/\*

完善p率阈值化的计算过程

\*/

//返回原图像img经过p率阈值化后的结果,即图像ThresholdPImg。

return ThresholdPImg;

}

void main()

{

Mat input = imread("2233.jpg");

Mat gray;

//彩色图转为灰度图

cvtColor(input, gray, COLOR\_BGR2GRAY);

//灰度图p率阈值化，需编程实现

Mat ThresholdPImg = myThresholdP(gray);

namedWindow("input", 0);

imshow("input", input);

namedWindow("ThresholdPImg", 0);

imshow("ThresholdPImg", ThresholdPImg);

waitKey(0);

}

实验结果显示：

原图：

p=1.3时：



p=1.7时：



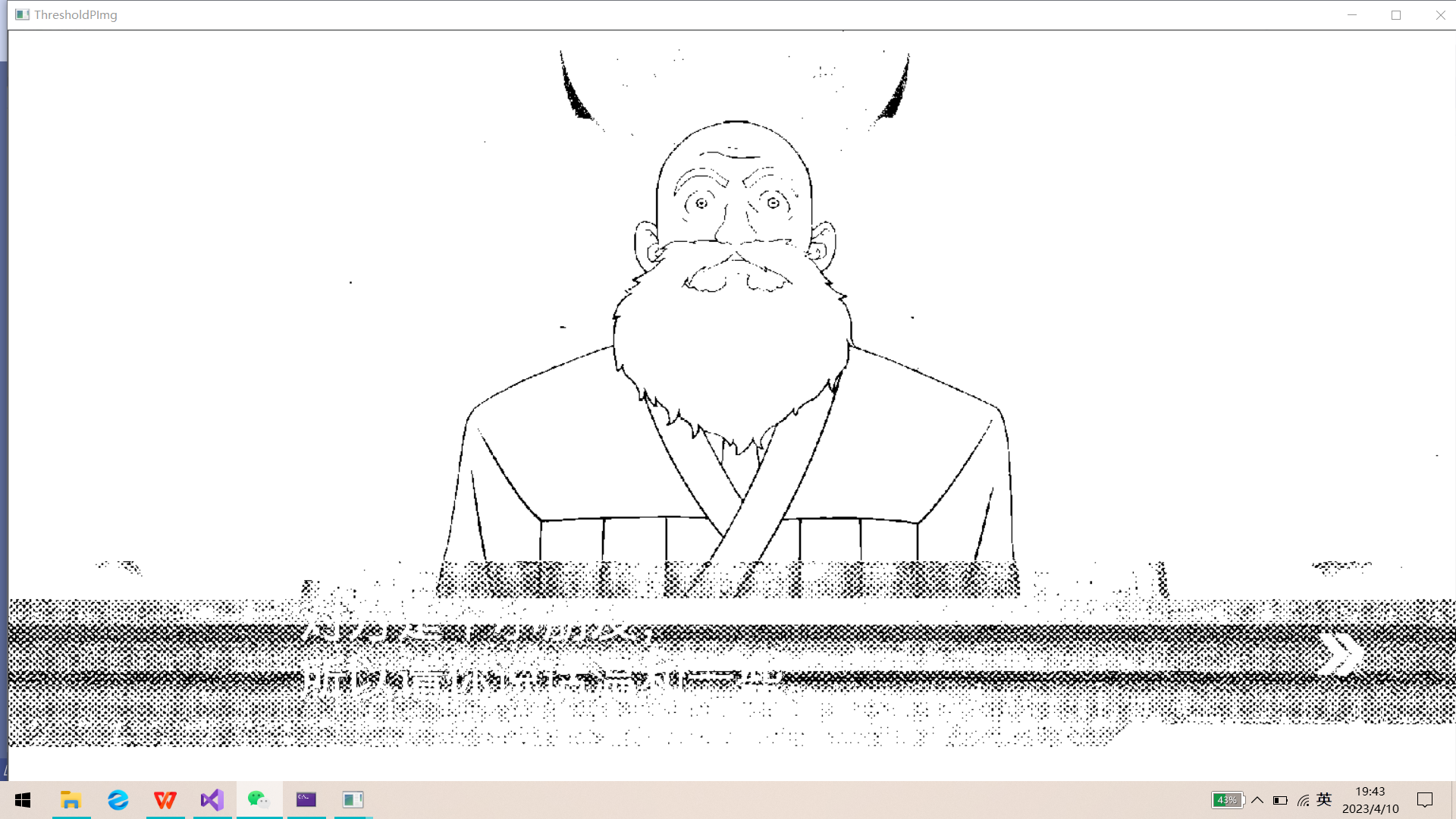
p=3时：



p=9时：



p=20时：



实验分析总结：

通过改变p的值，可以得知，当p从小逐渐变大时，阈值T的值从大逐渐变小。同时处理后图像的整体灰度值也逐渐从大变小。