|  |  |
| --- | --- |
| **成绩** |  |

**重庆邮电大学**

**实验报告**

**班级：13111507**

**姓名：陈继平**

**学号：2015214128**

**指导老师：刘玲慧**

**课程名称：数字媒体设计与开发**

**实验时间：2017/10/28**

**实验地点：A501**

**一、实验名称**

**实验二**

**二、实验目的**

**1、 掌握在VS+OpenCV开发环境下读取和显示图像的方法**

**2、 访问图像中任意一个像素点的颜色信息**

**3、 掌握图像的灰度化方法和二值化方法**

**三、实验内容**

**1、掌握基础图像容器Mat的用法；**

**2、访问图像中任意一个像素点的颜色信息；**

**3、编写程序实现减小颜色空间；**

**4、编写程序实现图像的灰度化和二值化。**

**四、实验步骤**

**要求1：编写完整代码实现颜色空间缩减算法，要求分别实现3种不同的元素访问方法，各执行100次，统计每种方法的平均运行时间，并输出平均运行时间和缩减颜色后的结果图。**

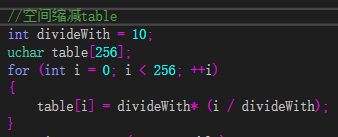
（1）实现三种不同的Mat元素访问方法：

指针访问C操作符[ ]法；Mat& ScanImageAndReduceC(Mat& I, const uchar\* const table);

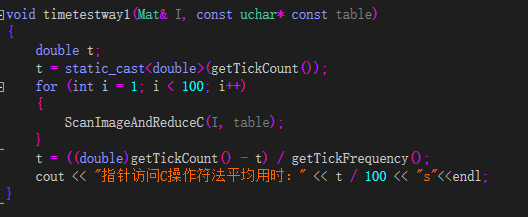
迭代器iterator法；Mat& ScanImageAndReduceIterator(Mat& I, const uchar\* const table);

动态地址计算法；Mat& ScanImageAndReduceRandomAccess(Mat& I, const uchar\* const table);

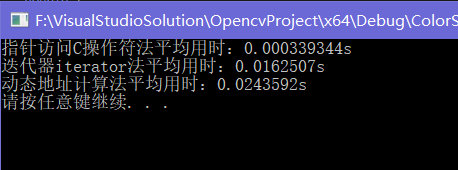
（2）初始化空间缩减矩阵：



（3）将三种元素访问方法加到计时算法中，用for()循环执行100次，计算其平均运行时间



（4）得到的输出结果如图所示：



**要求2：编程实现图像的灰度化和二值化，要求使用3种元素访问方法之一，不能使用OpenCV自带的灰度化或二值化函数。**

#include "stdafx.h"

#include<iostream>

#include<opencv2/opencv.hpp>

using namespace std;

using namespace cv;

Mat grayscale(Mat& I);

Mat twovalue(Mat& I);

int main()

{

Mat grayscaleImg;

const char\* filename = "input.jpg";

Mat srcImg = imread(filename, CV\_LOAD\_IMAGE\_COLOR);

if (srcImg.empty())

return -1;

int channels = srcImg.channels();

int nRows = srcImg.rows;

int nCols = srcImg.cols; //每行的元素个数

//灰度化

grayscaleImg = grayscale(srcImg);

imshow("source", grayscaleImg);

waitKey(0);

return 0;

//二值化

grayscaleImg = twovalue(srcImg);

imshow("source", grayscaleImg);

waitKey(0);

return 0;

}

Mat grayscale(Mat& I)

{

int channels = I.channels();

int nRows = I.rows;

int nCols = I.cols; //每行的元素个数

Mat GrayscaleImg(nRows, nCols, CV\_8UC1);

int i, j,k;

//双重循环，遍历所有像素值

for (i = 0; i < nRows; ++i)

{

for (j = 0,k = 0; j < nCols; ++j,k = k+3)

{

GrayscaleImg.ptr(i)[j] = (int)(I.ptr(i)[k] \* 0.11 + I.ptr(i)[k+1] \* 0.59 + I.ptr(i)[k+2] \* 0.30);

//GrayscaleImg.at<uchar>(i, j) = (int)( I.at<Vec3b>(i, j)[0] \* 0.11 + I.at<Vec3b>(i, j)[1]\*0.59 + I.at<Vec3b>(i, j)[2] \* 0.30);

//cout <<(int)( I.at<Vec3b>(i, j)[0] \* 0.11 + I.at<Vec3b>(i, j)[1]\*0.59 + I.at<Vec3b>(i, j)[2] \* 0.30) << " ";

}

}

return GrayscaleImg;

}

Mat twovalue(Mat& I)

{

int nRows = I.rows;

int nCols = I.cols;

Mat Twovalue(nRows, nCols, CV\_8UC1);

I = grayscale(I);

for (int i = 0; i < nRows; ++i)

{

for (int j = 0; j < nCols; ++j)

{

if (I.ptr(i)[j] < 105)

{

Twovalue.ptr(i)[j] = 0;

}

else

{

Twovalue.ptr(i)[j] = 255;

}

}

}

return Twovalue;

}

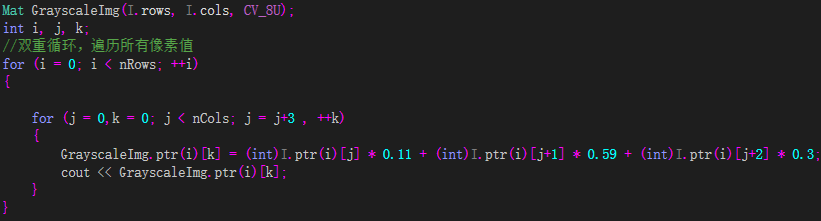
**五、实验结论**

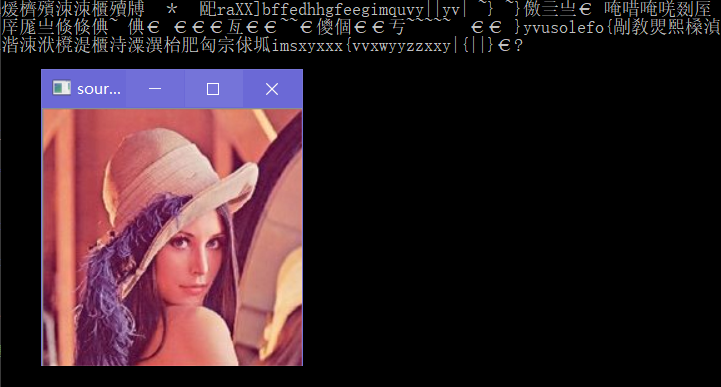
**1. 颜色空间缩减算法：**

此题目三种Mat遍历算法都由指导书给出，时间统计算法和颜色空间缩减矩阵也都已经给出，没有难点和需要设计的算法，只要使用循环让缩减算法执行100次，统计出总时间，再计算平均用时即可。

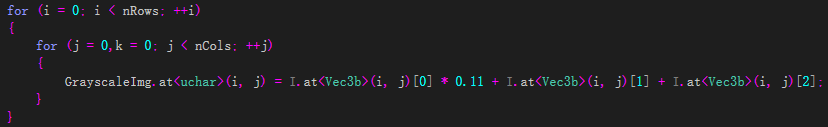
**2. 灰度化和二值化：**

原图片是8位3通道的图片，要对其进行二值化必须要将通道数缩减为1，如何将一个像素的三个通道值合并成一个便是难点所在。我的解决办法是使用上一题中的最快的遍历算法将Mat对象的所有元素遍历一遍，对每个像素的BGR三个通道做一次加权求平均值的运算，用一个新建的Mat对象（8位1通道）来接收每个像素的值，从而得到灰度化的图像。然而我还是Too Young Too Simple,



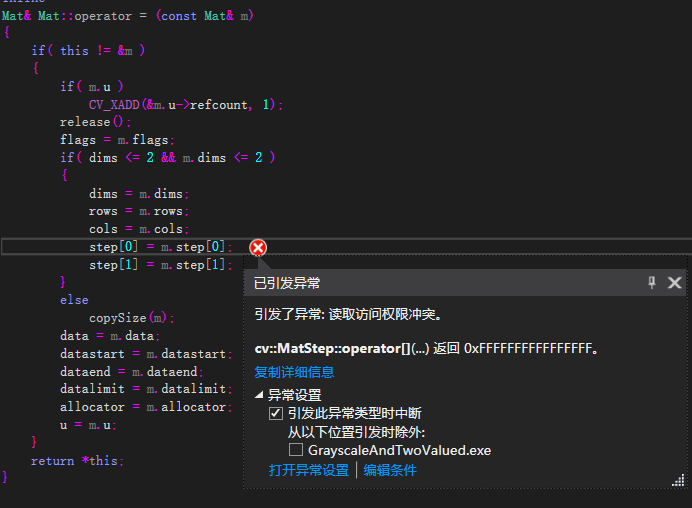


使用Mat.ptr(i)[j]可以取到每个元素的值，虽然是uchar，但还是可以强制转换成int类型，但是int类型没办法赋值给Mat.ptr(i)[j]啊啊啊！问了下百度大佬，学到了一种新的赋值方法：Img.at<uchar>(i, j)，

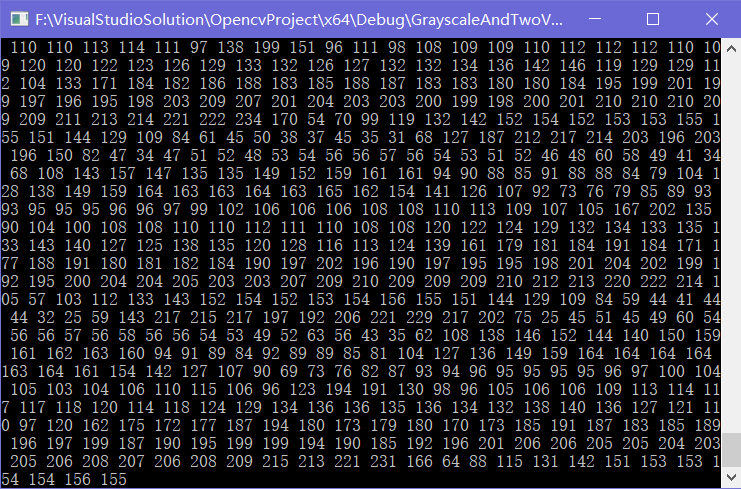


用I.at<Vec3b>(i, j)[0] I.at<Vec3b>(i, j)[1] I.at<Vec3b>(i, j)[2]分别取到三个通道的值，计算后赋给单通道图像Img对应的（i，j）位置的像素，再次运行……

然后就又尴尬了……



这异常出在Opencv源码中啊。我勒个天，我就想用一个矩阵画张图啊，于是我又一次检查我的矩阵是否有问题，



矩阵没问题数值都介于[0-255]之间，但是它依旧抛出了同样的异常，这就说明，是使用I.at<Vec3b>(i, j)[0]读取时出现了问题，emmmmmmm...

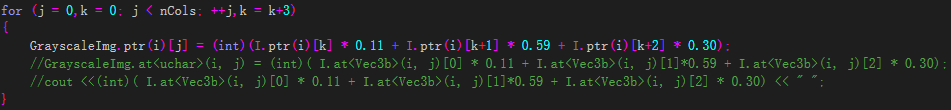
等等，

就在我绝望的时候，我发现了一个很不愉快的东西

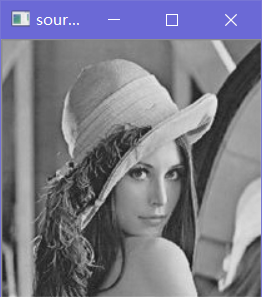


我这函数返回值类型好像有点别扭的样子，我return的是一个Mat对象，但是函数的返回值类型却是Mat&。哇……的一声就哭了出来

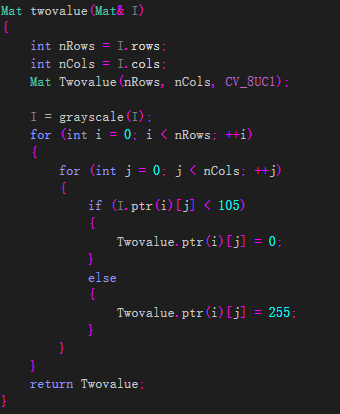
修改函数之后，两种算法都可以用了



成功实现灰度化



完成灰度化后，二值化就简单了，将灰度图像中的数值分成两类：大于阈值和小于阈值的，大的全部归到255，小的全部归到0；经过数次测试，发现阈值取105左右效果最佳

**3.回答指导书中的问题1-3：**

问题1. 通过如下例子观察Mat对象A,B,C的变化，并得出结论

答:A、B、C三张图片显示效果完全相同，使用blur（）函数对C图片进行模糊化处理后，A1、B1、C1都产生相同程度的模糊，由此可以得出使用Mat B(A); C = A;两种方式复制的Mat对象是指针类型，指向被复制的Mat对象地址，并没有开辟新的内存空间。

问题2：通过如下例子观察Mat对象A,B,C的变化，并得出结论

答：A为源图像，B显示为模糊化后的图像，C显示为灰度图像，B、C是由A.clone(),A.copyTo()得到，对BC的修改是对BC各自Mat对象的修改，对A无影响，由此可以得出使用clone(),copyTo()两个函数复制的Mat对象是数值对象，而非指针，开辟了新的内存空间

问题3：分析如下代码，并得到实验结果，对结果进行分析

*#include<iostream> //包含输入输出流头文件*

*#include<opencv2/opencv.hpp> //包含opencv2头文件*

*using namespace std; //使用std命名空间*

*using namespace cv; //使用cv命名空间*

*int main(int argc, char \*\*argv)//程序入口*

*{*

*const char\* filename = "input.jpg";//定义一个指向字符常量的指针filename*

*Mat srcImg = imread(filename, CV\_LOAD\_IMAGE\_COLOR);//创建一个Mat对象*

*if(srcImg.empty()) //判断Mat是否为空*

*return -1; //为空则退出*

*imshow("source", srcImg); //显示图片*

*int channels = srcImg.channels(); //获取图片的通道数*

*int nRows = srcImg.rows ; //获取图片的行数*

*int nCols = srcImg.cols\* channels; //计算每一行的元素个数*

*if (srcImg.isContinuous()) //判断图片是否连续*

*{*

*nCols \*= nRows; //计算矩阵大小*

*nRows = 1; //将行数赋值为1，把图片看做在内存中连续的一个一维数组*

*}*

*int i,j; //设置测试点（i,j）*

*i=5;*

*j=2;*

*cout<<int(srcImg.ptr(i)[j])<<"\n"; //输出该图像第i+1行，j/3（若有余数则+1）*

*//列位置的像素的第j%3+1个通道的值*

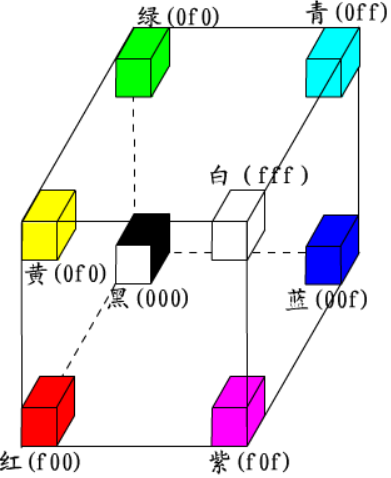
*waitKey(0);*

*return 0;*

*}*

**4.** **附加作业：**

题目给出的图片： RGB三维矩阵模型：





观察可以看出，要求的渐变效果是红->黄-绿->青->蓝->紫->红;

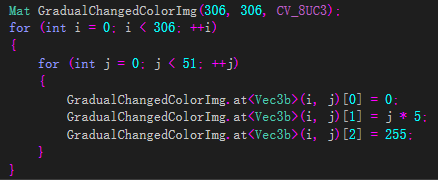
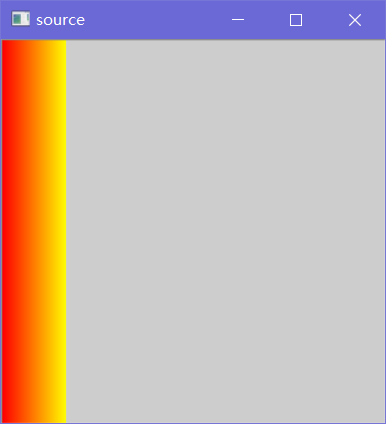
分别对应RGB矩阵的6个顶点

(255,0,0)(255,255,0)(0,255,0)(0,255,255)(0,0,255)(255,0,255)(255,0,0)

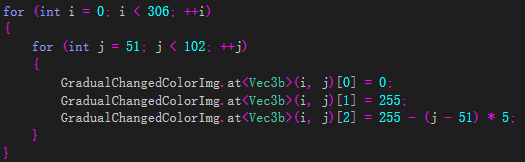
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **R** | **255** | **255** | **0** | **0** | **0** | **255** | **255** |
| **G** | **0** | **255** | **255** | **255** | **0** | **0** | **0** |
| **B** | **0** | **0** | **0** | **255** | **255** | **255** | **0** |

从上表的值变化可以看出，每次只有一个通道发生改变，所以我分六次将整张图片绘制完成。选择渐变梯度为5，每次变化共执行255/5=51次，所以创建一个306\*306dp的图片：

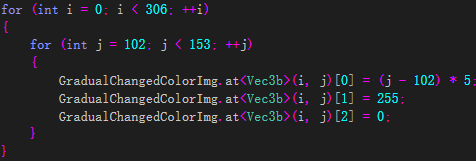
第一次：绘制第1-51列

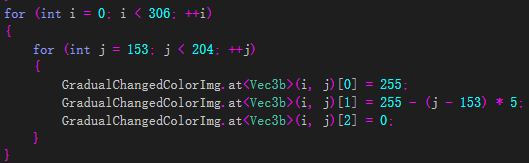
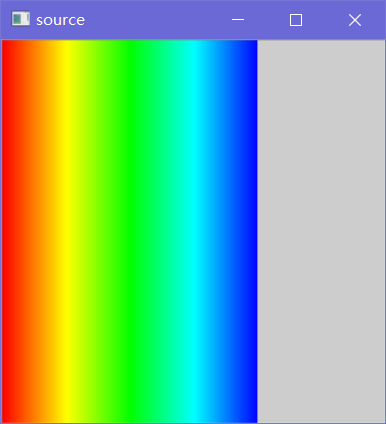
第二次：绘制第52-102列

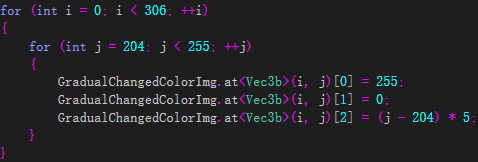
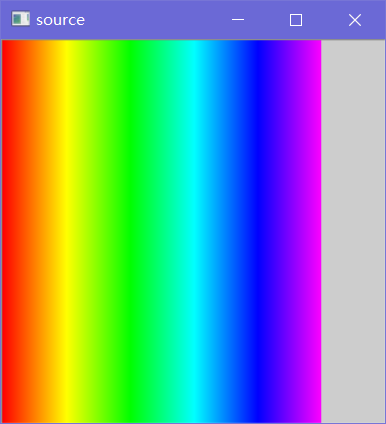
第三次：绘制第103-153列

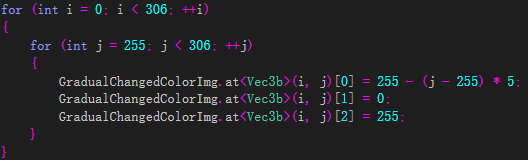
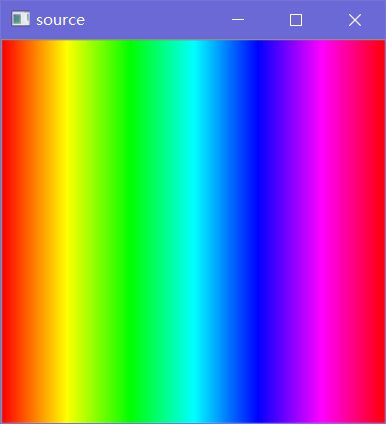
第四次：绘制第154-204列

第五次：绘制第205-255列

第六次：绘制第256-306列

**最终效果：**

