石河子大学

研 究 生 课 程 结 课 作 业

课 程 名 称	数字图像处理与机器视觉
学　　院	信息科学与技术学院
姓　　名	习钟
学　　号	20212108032
专　　业	电子信息
授 课 教 师	马本学　教授

中国·新疆·石河子

2022 年 05 月

《图像处理与机器视觉》课程大作业

题目：基于 OpenCV 的简单 PS 应用

摘　要：

现如今随着计算机的广泛使用以及手机的普及，越来越多的人喜欢在日常生活中拍下身边的美好，针对图像的处理已然成为了一个必要的需求。本文设计了一个基于 OpenCV 的简单的 PS 应用，能够完成对于照片的旋转、剪切、调节图片的色阶、调整图片的对比度、通过色彩曲线来调节图片的整体色彩等功能，针对图片使用该软件能够将图片美化成任何你想要的样子。同时针对于生活中常常需要准备各种底色的证件照需求，该应用支持将任何底色的证件照转换成您所需要的底色。

本应用使用了 C++结合 OpenCV 框架结合相关的图像处理算法（高斯模糊，K-Means 分割，Hough 算法）等，通过对图像 RGB 的通道处理可以实现对图像的色阶、灰度值等信息的改变，通过使用 Canny 算法结合 Hough 算法能有效解决对于边界的判定，通过使用 K-Means 算法结合高斯模糊等特性能够实现对于证件照的底色的变化而不影响人物。

最后通过使用 OpenCV 自带的 GUI 等框架来完成简单的 GUI 界面的操作。

关键词： OpenCV、Hough 算法、K-Means 分割

Simple PS application based on OpenCV

Abstract:

Nowadays, with the widespread use of computers and the popularity of mobile phones, more and more people like to take pictures of the beauty around them in their daily lives, and image processing has become a necessary requirement. This paper designs a simple PS application based on OpenCV, which can complete the functions of rotating, cutting, adjusting the color level of the picture, adjusting the contrast of the picture, and adjusting the overall color of the picture through the color curve. Use this software for pictures Be able to beautify the picture into any look you want. At the same time, in response to the need to prepare ID photos with various background colors in life, this application supports converting any background color ID photos into the background color you need.

This application uses C++ combined with the OpenCV framework combined with related image processing algorithms (Gaussian blur, K-Means segmentation, Hough algorithm), etc., through the processing of the RGB channel of the image, the color level, gray value and other information of the image can be changed. , By using the Canny algorithm combined with the Hough algorithm, the boundary determination can be effectively solved, and by using the K-Means algorithm combined with Gaussian blur and other characteristics, the background color of the ID photo can be changed without affecting the characters.

Finally, the simple GUI interface operation is completed by using the GUI and other frameworks that come with OpenCV.

Keywords: OpenCV, Hough algorithm, K-Means segmentation

一、问题的提出

（一）、研究背景

随着经济社会的迅猛发展，人们的生活水平的提高，人们对于拍照的需求也在与日俱增。在任何时候，拿出手机拍下身边的美好已经是一件每天高频次发生的事情，社交网络上也分享着大量的图片。对于照片的处理已经成为了一门成熟的学科，众多的 APP 抓住非专业人士对于图像处理的痛点，做出了让人们广泛使用的 APP，但这些不乏都是商业性质的 APP，用户作为一个使用者也无法知道各项的调节参数。与此同时，一款跨平台的 PS 应用也没有问世，大都是使用网页性质的 PS 应用来完成简单的 PS，但是其效果可能乏善可陈。

（二）、研究意义

目前虽然人们的手机已经可以完成绝大部分功能，但是在实际调研中发现绝大多数人在面临需要对图像进行处理时无法找到一个合适的工具，尤其是在对于证件照的处理上，当人们想换一张证件照的底色时，往往求助于各类 APP，但是质量良莠不齐的同时，付费也是阻碍许多人使用 APP 的原因之一。

现如今随着计算机的广泛使用以及手机的普及，越来越多的人喜欢在日常生活中拍下身边的美好，针对图像的处理已然成为了一个必要的需求。本文设计了一个基于 OpenCV 的简单的 PS 应用，能够完成对于照片的旋转、剪切、调节图片的色阶、调整图片的对比度、通过色彩曲线来调节图片的整体色彩等功能，针对图片使用该软件能够将图片美化成任何你想要的样子。同时针对于生活中常常需要准备各种底色的证件照需求，该应用支持将任何底色的证件照转换成您所需要的底色。

二、原理及方法

(一)　设计原理

本应用针对图像的处理与分析主要利用了 OpenCV 的框架，OpenCV 全称是 Open Source Computer Vision Library，是由 Intel 资助开发用于图像领域的函数库，库中的函数全部是由高效率的 C 语言和 C++语言写成，具有高度的代码继承性、可移植性和跨平台性。可以用在 Linux、Windows、Android、Mac 等各种平台上，得到了广大图像领域的开发者和学校，研究机构的研究者的青睐，成为了一种非常主流的图像处理库。

本文利用其对图像处理的强大能力设计了一个简单的 PS 应用，设计了图像

旋转等七个功能，如下图一所示在图像旋转中主要通过获取两次旋转前后的位置变化来进行判断，其中主要用到了 Hoff 算法进行对直线的判定，从而确定两次的边界值；剪切的功能实现主要是通过捕捉屏幕鼠标的行为，然后读取所得区域，将该区域与原始图片区域做与运算即得到我们的图像；

对比度控制主要是利用一个了对比度控制的算法，算法通过对各个颜色通道的相互之间进行计算获得最后的灰度值，实现对图像较好的对比度控制；用色彩曲线控制图像则同样也是单独抽离出现有图像中所拥有的值，对通道的值进行相应的计算即可以获得调节曲线所得到的色彩值；色阶控制是对于一个 RGB 图像，可以对 R，G，B 通道进行独立的色阶调整，即对三个通道分别使用三个色阶定义值。还可以再对三个通道进行整体色阶调整。因此，对一个图像，可以用四次色阶调整。最终的结果，是四次调整后合并产生的结果；黑白变换主要完成对 RGB 通道值的交换并使用相应的算法完成灰度化，最后得到高级的黑白照片；换底功能的实现主要依靠进行 K-Means 算法分割以后通过腐蚀、高斯模糊等之后进行一系列的操作完成换底的目的。

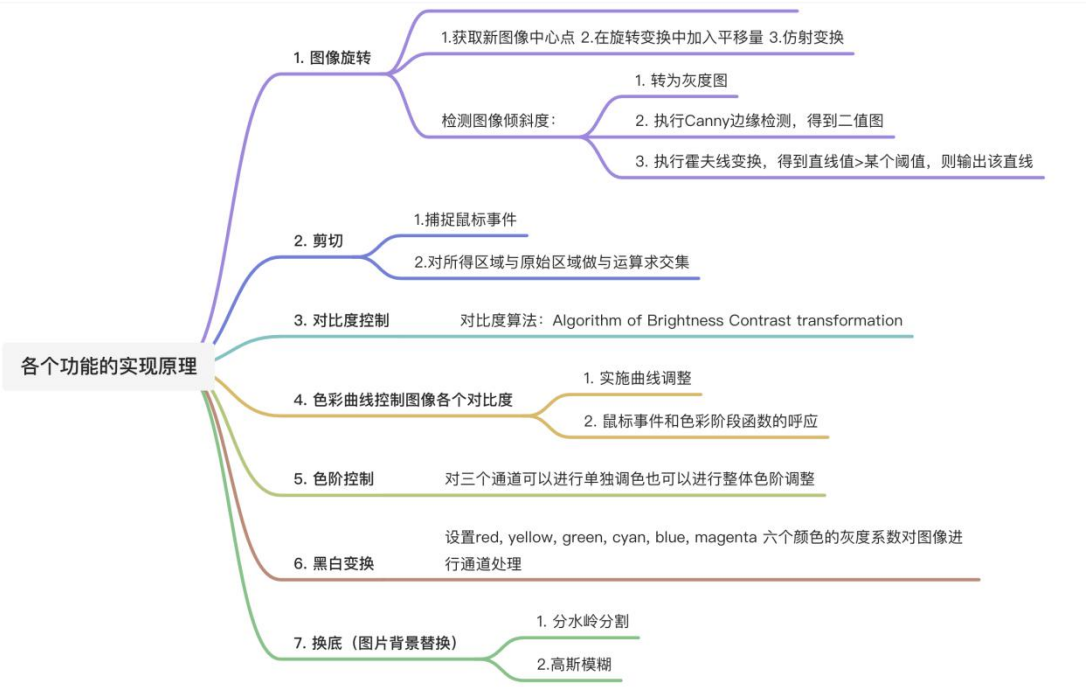


图 1：应用的设计思路

(二) 技术原理

要求：画出技术路线图（即实现结果的流程），并简单介绍算法原理

针对实现的功能主要将其分为三大类实现过程：1.做到图像旋转以及放大的功能 2.对图像的RGB通道根据不同的算法进行相应的实现 3. 通过OpenCV中的K-Means 处理后进行遮罩等处理。具体细节如下图二所示：

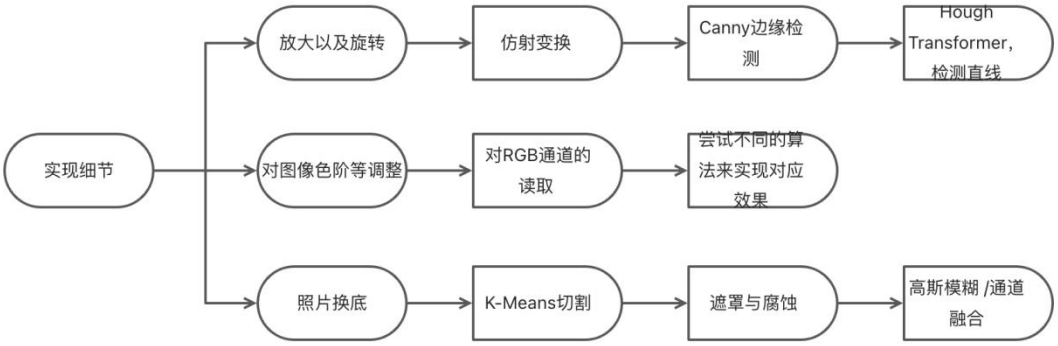


图 2：实现细节的技术原理

（1）Hough 算法：

a） 随机抽取图像中的一个边缘像素点，如果已经被标定为是某一条直线上的点，则继续在剩下的边缘点中随机抽取一个边缘点，直到所有边缘点都抽取完为止；

b） 对该点进行霍夫变换，并进行累加计算；

c） 选取在霍夫空间内累加值最大的点，如果该点的值大于阈值，则进行步骤 4，否则回到步骤 1；

d） 对于累加值大于阈值的点，从该点出发，沿着图像中的直线的方向位移，从而找到直线的两个端点；

e） 计算直线的长度，如果大于某个阈值，则被认为是直线并输出。

（2）Canny 边缘检测算法：

a） 高斯滤波进行平滑（模糊）

b） 计算梯度值和梯度方向（梯度来表示灰度值的变化程度和方向）

c） 过滤非最大值（边缘梯度值是 1，非边缘为 0）

d） 使用上下阈值来检测边缘（使用启发式算法确定上阈值和下阈值，提高准确度）

（3）K-Means 切割

彩色图像中的每一个像素是三维空间中的一个点，三维对应红、绿、蓝三原色的强度，基于 Kmeans 聚类算法的图像分割以图像的像素为数据点，按照指定的簇数进行聚类，然后将每个像素点以其对应的聚类中心替代，重构该图像。

三、图形用户界面设计和运行结果（如没有设计图形用户界面可以放上自己的运行结果图，并对结果进行必要说明）

本文所设计的软件系统包括对图片进行旋转放大、剪切、对比度调整、控制色域、色阶控制、黑白变换、换底，共 7 部分，其技术路线如图 2 所示。

如图 3 所示，显示了图像从放大到旋转的变化，其中 rotate 调节的是现在图像的旋转角度，clip 控制是否选择在旋转中保持图像不变，在左边的时候表示是，

在右边的时候表示否。

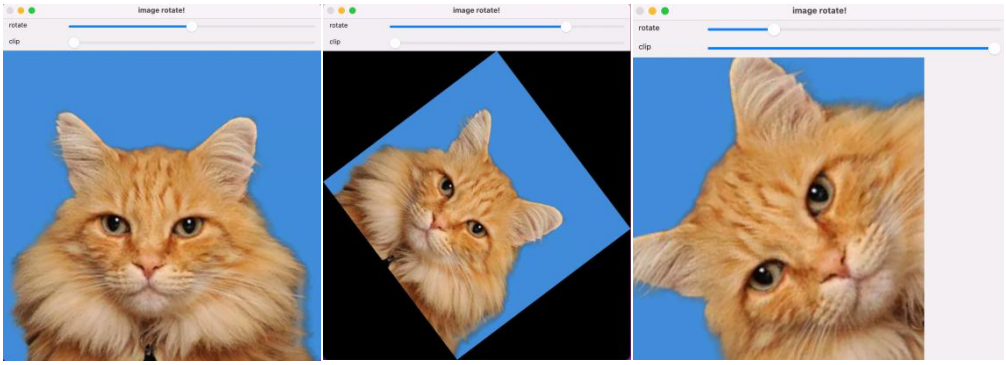


图 3：图像的旋转和放大

如图 4 所示，表示了剪切功能的实现，通过鼠标（红线为鼠标轨迹）滑动区域，即可以完成对于图片的剪切

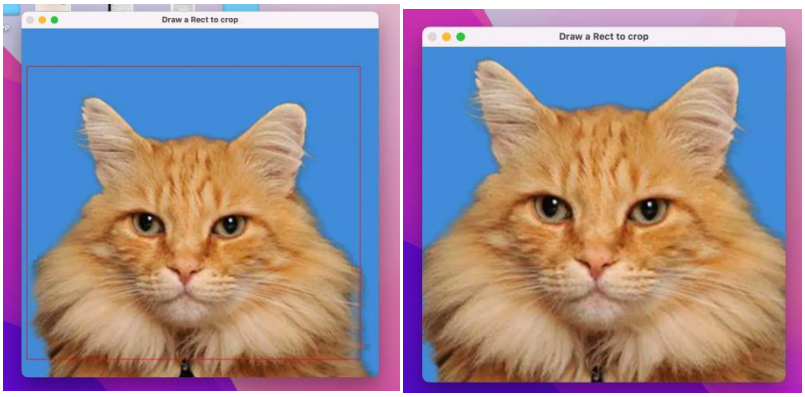


图 4：通过鼠标滑动选择区域

如图 5 所示，表示经过对比度选择之后的结果，其中 brightness 控制亮度的调节，contrast 表示对比度的调节。

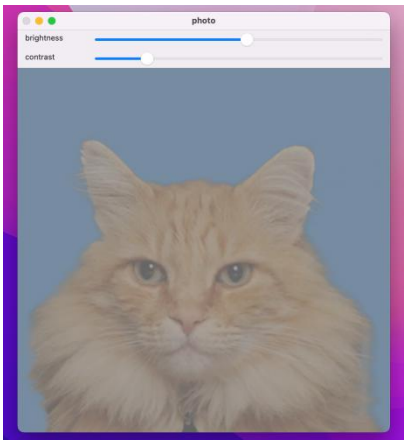
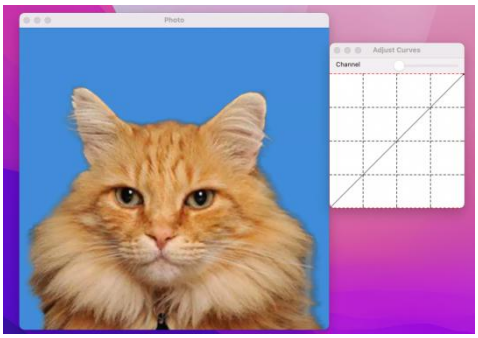
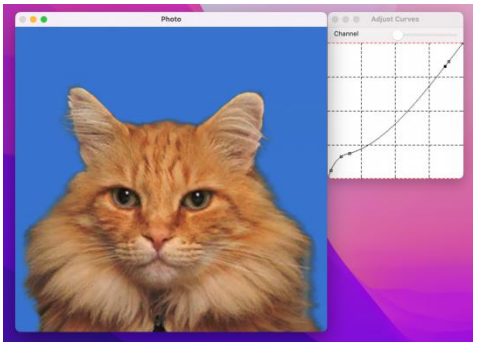


图 5：图像的对比度调节

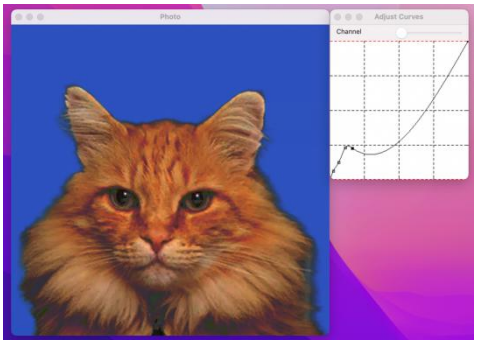
如图 6 所示，通过鼠标调节色域曲线我们可以获得关于图像的各种色域变化。



(a) 原图



(b) 简单处理



(c) 锐化加强

图 6：通过色域曲线调节图像色域变化图

如图 7 所示，通过控制 Shadow、OutHightLight、Midtones、HightLight、Channel、OutShadow 来对图片的色阶进行控制。

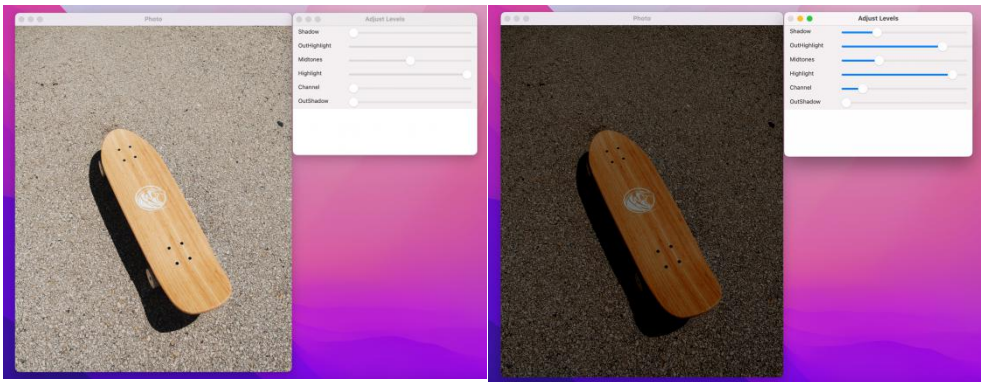


图 7：控制色阶的对比图

如图 8 所示，表示通过控制来得到一张完美黑白照片，通过控制 green、magnet、red、cyan、blue、yellow 六个参数来完成对图片黑白度控制，从而获得黑白照片。

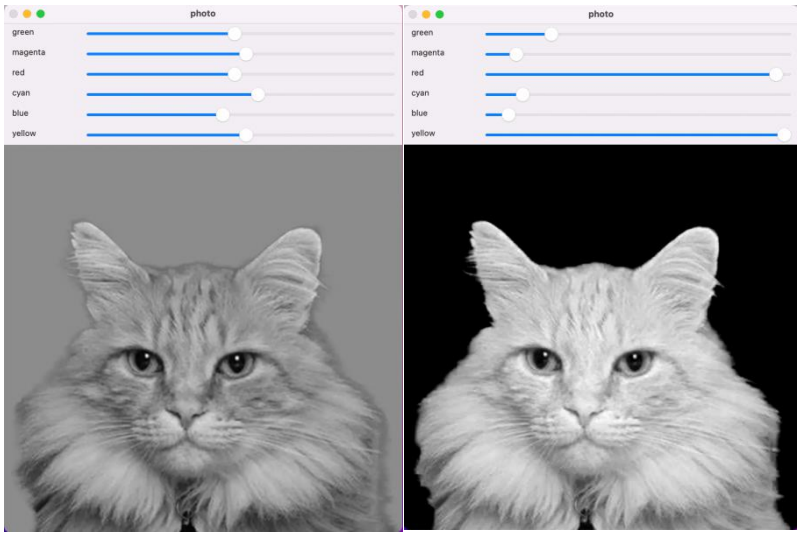


图 8：控制黑白照片的对比图

如图 9 所示，可以通过对原照片的底色进行更换，达到获取其他底色证件照的目的。

图 9：生成不同底色的证件照

四、结果与讨论

本应用在一定程度上解决了对于简单 PS 的上手难等问题，对图片简单且直观的操作在一定程度上是比手机 P 图更加方便，尤其是针对于证件照的换底色等功能，所得到的换底色之后的照片效果可以满足日常使用。针对图像的美化上，通过常见的图像处理算法即可或得对任意图像进行处理的能力。故本文完全实现了使用 OpenCV 实现一个简易可用的小型 PS 应用的功能。

针对应用的反思，主要体现在对于界面设计的考量不够仔细，可以设计更加人性化的 GUI 界面来完成一系列的操作，同时针对于目前所拥有的图像处理技术还不够新颖，可以加入更多的机器学习的技术来实现更强大的 PS 功能。

参考文献

[1]雷剑锋，汪伟．基于 OpenCV 的图像阈值分割研究与 实现 [J]. 现代电子技术 ,2013,36(24):73-76.
[2]基于 OpenCV 的图像滤波方法比较[J]. 赵博文,张力夫,潘在峰,王蓉,郭雅馨. 信息与电脑(理论版). 2020(15)
[3]基于 OpenCV 图像处理系统的开发与实现[J]. 刘培军,马明栋,王得玉. 计算机技术与发展. 2019(03)
[4] Photoshop 和 OpenCV 的数字图像处理教学应用[J]. 林忠. 中国现代教育装备. 2015(01)

附录

软件环境：Apple clang version 13.1.6 (clang-1316.0.21.2.5)、CLion 2021、OpenCV4.5.5
硬件环境：CPU：M1（芯片集成）、内存：8GB、硬盘：256G、显卡：M1（芯片集成）
编程语言：C++
代码目录如下图所示：

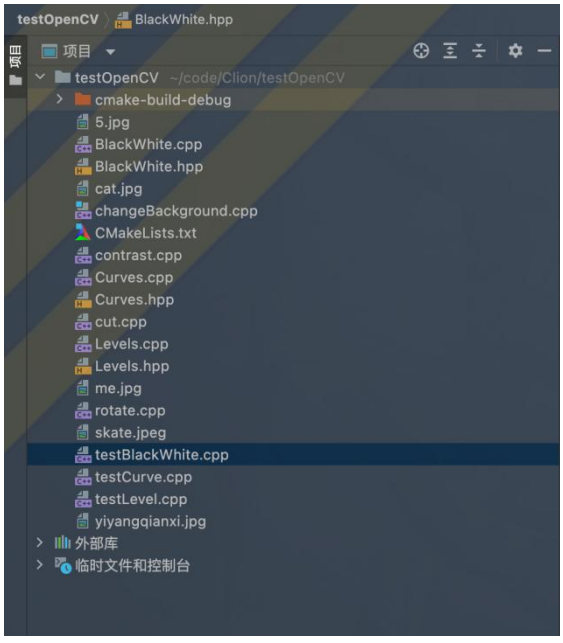


图 10：工程代码目录

详细代码：
Rotate.cpp

```
#include <iostream>
#include "opencv2/core.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/highgui.hpp"
#include <cmath>

using namespace std;
using namespace cv;


#define SHOW_LINE

#define BASE 100

static string source_window = "source";
static string window_name = "image rotate!";
static Mat src;
static int rotateDegree = 0 + BASE;
static int clip = 0;

//图像旋转: src 为原图像,  dst 为新图像, angle 为旋转角度(正值为顺时针旋转,负值为逆时针旋转)
int imageRotate1(InputArray src, OutputArray dst, double angle)
{
	Mat input = src.getMat();
	if( input.empty() ) {
		return -1;
	}
	//得到图像大小
	int width = input.cols;
	int height = input.rows;
	//计算图像中心点
	Point2f center;
	center.x = width / 2.0;
	center.y = height / 2.0;
	//获得旋转变换矩阵
	double scale = 1.0;
	Mat trans_mat = getRotationMatrix2D( center, -angle, scale );
	//仿射变换
	warpAffine( input, dst, trans_mat, Size(width, height));
	return 0;
```

```
}

//图像旋转: src 为原图像,    dst 为新图像, angle 为旋转角度
int imageRotate2(InputArray src, OutputArray dst, double angle)
{
	Mat input = src.getMat();
	if( input.empty() ) {
		return -1;
	}

	//得到图像大小
	int width = input.cols;
	int height = input.rows;

	//计算图像中心点
	Point2f center;
	center.x = width / 2.0;
	center.y = height / 2.0;

	//获得旋转变换矩阵
	double scale = 1.0;
	Mat trans_mat = getRotationMatrix2D( center, -angle, scale );

	//计算新图像大小
	double angle1 = angle    * CV_PI / 180. ;
	double a = sin(angle1) * scale;
	double b = cos(angle1) * scale;
	double out_width = height * fabs(a) + width * fabs(b);
	double out_height = width * fabs(a) + height * fabs(b);

	//仿射变换
	warpAffine( input, dst, trans_mat, Size(out_width, out_height));

	return 0;
}

//图像旋转: src 为原图像,    dst 为新图像, angle 为旋转角度
int imageRotate3(InputArray src, OutputArray dst, double angle)
{
	Mat input = src.getMat();
	if( input.empty() ) {
		return -1;
	}
```

```
	//得到图像大小
	int width = input.cols;
	int height = input.rows;

	//计算图像中心点
	Point2f center;
	center.x = width / 2.0;
	center.y = height / 2.0;

	//获得旋转变换矩阵
	double scale = 1.0;
	Mat trans_mat = getRotationMatrix2D( center, -angle, scale );

	//计算新图像大小
	double angle1 = angle  * CV_PI / 180. ;
	double a = sin(angle1) * scale;
	double b = cos(angle1) * scale;
	double out_width = height * fabs(a) + width * fabs(b);
	double out_height = width * fabs(a) + height * fabs(b);

	//在旋转变换矩阵中加入平移量
	trans_mat.at<double>(0, 2) += cvRound( (out_width - width) / 2 );
	trans_mat.at<double>(1, 2) += cvRound( (out_height - height) / 2);

	//仿射变换
	warpAffine( input, dst, trans_mat, Size(out_width, out_height));

	return 0;
}


//图像旋转: src 为原图像,  dst 为新图像, angle 为旋转角度, isClip 表示是采取缩小图片的方式
int imageRotate4(InputArray src, OutputArray dst, double angle, bool isClip)
{
	Mat input = src.getMat();
	if( input.empty() ) {
		return -1;
	}

	//得到图像大小
	int width = input.cols;
	int height = input.rows;
```

```
    //计算图像中心点
    Point2f center;
    center.x = width / 2.0;
    center.y = height / 2.0;

    //获得旋转变换矩阵
    double scale = 1.0;
    Mat trans_mat = getRotationMatrix2D( center, -angle, scale );

    //计算新图像大小
    double angle1 = angle    * CV_PI / 180. ;
    double a = sin(angle1) * scale;
    double b = cos(angle1) * scale;
    double out_width = height * fabs(a) + width * fabs(b); //外边框长度
    double out_height = width * fabs(a) + height * fabs(b);//外边框高度

    int new_width, new_height;
    if ( ! isClip ) {
        new_width = cvRound(out_width);
        new_height = cvRound(out_height);
    } else {
        //calculate width and height of clip rect
        double angle2 = fabs(atan(height * 1.0 / width)); //即角度 b
        double len = width * fabs(b);
        double Y = len / ( 1 / fabs(tan(angle1)) + 1 / fabs(tan(angle2)) );
        double X = Y * 1 / fabs(tan(angle2));
        new_width = cvRound(out_width - X * 2);
        new_height= cvRound(out_height - Y * 2);
    }

    //在旋转变换矩阵中加入平移量
    trans_mat.at<double>(0, 2) += cvRound( (new_width - width) / 2 );
    trans_mat.at<double>(1, 2) += cvRound( (new_height - height) / 2);

    //仿射变换
    warpAffine( input, dst, trans_mat, Size(new_width, new_height));

    return 0;
}

/**
 * 检测图像倾斜度
 * 返回值：返回 0 表示无检测结果，返回非 0 表示摆正图象需要旋转的角度（-10 至 10 度）
 */
```

```
double detectRotation(InputArray src)
{
    double max_angle = 6; //可旋转的最大角度

    Mat in = src.getMat();
    if( in.empty() ) return 0;

    Mat input;

    //转为灰度图
    if ( in.type() == CV_8UC1 )
        input = in;
    else if ( in.type() == CV_8UC3 )
        cvtColor(in, input, COLOR_BGR2GRAY);
    else if ( in.type() == CV_8UC3 )
        cvtColor(in, input, COLOR_BGR2GRAY);
    else
        return 0;

    Mat dst, cdst;

    //执行 Canny 边缘检测(检测结果为 dst，为黑白图)
    double threshold1 = 90;
    Canny(src, dst, threshold1, threshold1 * 3, 3);

    //将 Canny 边缘检测结果转化为灰度图像(cdst)
    cvtColor(dst, cdst, COLOR_GRAY2BGR);

    //执行霍夫线变换，检测直线
    vector<Vec4i> lines; //存放检测结果的 vector
    double minLineLength = std::min(dst.cols, dst.rows) * 0.25; //最短线长度
    double maxLineGap = std::min(dst.cols, dst.rows) * 0.03 ; //最小线间距
    int threshold = 90;
    HoughLinesP(dst, lines, 1, CV_PI / 180, threshold, minLineLength, maxLineGap );

    //分析所需变量
    int x1, y1, x2 , y2; //直线的两个端点
    int x, y;   //直线的中点
    double angle, rotate_angle; //直线的角度，摆正直线需要旋转的角度
    double line_length; //直线长度
    double position_weighted; //直线的位置权重：靠图像中央的线权重为 1，越靠边的线权重
越小
    double main_lens[2]; //用于存放最长的二条直线长度的数组 (这两条直线即是主线条)
    double main_angles[2];//用于存放最长的二条直线的摆正需要旋转的角度
```

```
main_lens[0] = main_lens[1] = 0;
main_angles[0] = main_angles[1] = 0;

//逐个分析各条直线，判断哪个是主线条
for( size_t i = 0; i < lines.size(); i++ ) {
    //取得直线的两个端点座标
    x1 = lines[i][0]; y1 = lines[i][1]; x2 = lines[i][2]; y2 = lines[i][3];
    x = (x1 + x2 ) / 2; y = (y1 + y2) / 2;
    //计算直线的角度
    angle =    (x1 == x2) ? 90 : ( atan ( (y1 - y2) * 1.0 / (x2 - x1) ) ) / CV_PI * 180;
    //摆正直线需要旋转的角度. 如果超出可旋转的最大角度,则忽略这个线。
    if ( fabs(angle - 0) <= max_angle ) {
        rotate_angle = angle - 0;
    } else if ( fabs(angle - 90) <= max_angle ) {
        rotate_angle = angle - 90;
    } else {
        continue;
    }

    //计算线的长度
    line_length = sqrt( (x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2)    );
    //计算直线的位置权重：靠图像中央的线权重为 1，越靠边的线权重越小
    position_weighted = 1;
    if ( x < dst.cols / 4 || x > dst.cols * 3 / 4    ) position_weighted *= 0.8;
    if ( x < dst.cols / 6 || x > dst.cols * 5 / 6    ) position_weighted *= 0.5;
    if ( x < dst.cols / 8 || x > dst.cols * 7 / 8    ) position_weighted *= 0.5;
    if ( y < dst.rows / 4 || y > dst.rows * 3 / 4    ) position_weighted *= 0.8;
    if ( y < dst.rows / 6 || y > dst.rows * 5 / 6    ) position_weighted *= 0.5;
    if ( y < dst.rows / 8 || y > dst.rows * 7 / 8    ) position_weighted *= 0.5;

    //如果 直线长度 * 位置权重 < 最小长度，  则这条线无效
    line_length = line_length * position_weighted;
    if ( line_length < minLineLength ) continue;



    //如果长度为前两名，则存入数据
    if ( line_length > main_lens[1] )    {
        if (line_length > main_lens[0]) {
            main_lens[1] = main_lens[0];
            main_lens[0] = line_length;
            main_angles[1] = main_angles[0];
            main_angles[0] = rotate_angle;
            //如果定义了 SHOW_LINE，则将该线条画出来
```

```
#ifdef SHOW_LINE
                    line( cdst, Point(x1, y1), Point(x2, y2), Scalar(0,0,255), 3, -1);
#endif
                } else {
                    main_lens[1] = line_length;
                    main_angles[1] = rotate_angle;
                }
            }
        }

        //如果定义了 SHOW_LINE, 则在 source_window 中显示 cdst
#ifdef SHOW_LINE
        imshow(source_window, cdst);
#endif

        //最后，分析最长的二条直线，得出结果
        if ( main_lens[0] > 0 ) {
            //如果最长的线 与 次长的线 两者长度相近，则返回两者需要旋转的角度的平均
值
            if (main_lens[1] > 0 && (main_lens[0] - main_lens[1] / main_lens[0] < 0.2 )) {
                return (main_angles[0] + main_angles[1] ) / 2;
            } else {
                return main_angles[0];     //否则，返回最长的线需要旋转的角度
            }
        } else {
            return 0;
        }
}

static void callbackAdjust(int , void *)
{
        Mat dst;

        //imageRotate1(src, dst, rotateDegree - BASE);
        //imageRotate2(src, dst, rotateDegree - BASE);
        //imageRotate3(src, dst, rotateDegree - BASE);

        bool isClip = ( clip == 1 );
        imageRotate4(src, dst, rotateDegree - BASE,    isClip );

        imshow(window_name, dst);
}
```

```
int main()
{
    src = imread("/Users/xz/code/Clion/testOpenCV/cat.jpg");

    if ( !src.data ) {
        cout << "error read image" << endl;
        return -1;
    }

    namedWindow(source_window);
    imshow(source_window, src);

    namedWindow(window_name);
    createTrackbar("rotate", window_name, &rotateDegree, BASE * 2, callbackAdjust);
    createTrackbar("clip", window_name, &clip, 1, callbackAdjust);

    //自动检测旋转角度
    double angle = detectRotation(src);
    if ( angle != 0 ) {
        rotateDegree = angle + BASE;
        setTrackbarPos("rotate", window_name, rotateDegree);
    }

    callbackAdjust(0, 0);

    waitKey();

    return 0;

}
```

Cut.cpp

```
//
// Created by viaxizhong on 2022/5/16.
//
#include <iostream>
#include "opencv2/core.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/highgui.hpp"

using namespace std;
using namespace cv;


//图像剪切
```

```
//参数：src 为源图像，  dst 为结果图像, rect 为剪切区域
//返回值：返回 0 表示成功，否则返回错误代码
int imageCrop(InputArray src, OutputArray dst, Rect rect)
{
	Mat input = src.getMat();
	if( input.empty() ) {
		return -1;
	}

	//计算剪切区域：	剪切 Rect 与源图像所在 Rect 的交集
	Rect srcRect(0, 0, input.cols, input.rows);
	rect = rect & srcRect;
	if ( rect.width <= 0   || rect.height <= 0 ) return -2;

	//创建结果图像
	dst.create(Size(rect.width, rect.height), src.type());
	Mat output = dst.getMat();
	if ( output.empty() ) return -1;

	try {
		//复制源图像的剪切区域 到结果图像
		input(rect).copyTo( output );
		return 0;
	} catch (...) {
		return -3;
	}
}

//=========================  主程序开始 ==========================

static string window_name = "Draw a Rect to crop";
static Mat src;  //源图片
bool   isDrag = false;
Point point1; //矩形的第一个点
Point point2; //矩形的第二个点

static void callbackMouseEvent(int mouseEvent, int x, int y, int flags, void* param)
{
	switch(mouseEvent) {

		case EVENT_LBUTTONDOWN:
			point1 = Point(x,y);
			point2 = Point(x,y);
			isDrag = true;
```

```
            break;

        case EVENT_MOUSEMOVE:
            if ( isDrag ) {
                point2 = Point(x,y);
                Mat dst = src.clone();
                Rect rect (point1, point2); //得到矩形
                rectangle(dst, rect, Scalar(0,0,255));//画矩形
                imshow(window_name, dst); //显示图像
            }
            break;

        case EVENT_LBUTTONUP:
            if (isDrag) {
                isDrag = false;
                Rect rect (point1, point2); //得到矩形
                imageCrop(src, src, rect); //图像剪切
                imshow(window_name, src); //显示图像
            }
            break;

    }

    return;
}


int main()
{
    //read image file
    src = imread("/Users/xz/code/Clion/testOpenCV/cat.jpg");
    if ( !src.data ) {
        cout << "error read image" << endl;
        return -1;
    }

    //create window
    namedWindow(window_name);
    imshow(window_name, src);

    //set mouse event call back
    setMouseCallback(window_name, callbackMouseEvent, NULL );

    waitKey();
```

```
	return 0;

}
```

Contast.cpp

```
#include <iostream>
#include "opencv2/core.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/highgui.hpp"

using namespace std;
using namespace cv;


#define SWAP(a, b, t)    do { t = a; a = b; b = t; } while(0)
#define CLIP_RANGE(value, min, max)    ( (value) > (max) ? (max) : (((value) < (min)) ? (min) : (value)) )
#define COLOR_RANGE(value)    CLIP_RANGE(value, 0, 255)

/**
 * Adjust Brightness and Contrast
 *
 * @param src [in] InputArray
 * @param dst [out] OutputArray
 * @param brightness [in] integer, value range [-255, 255]
 * @param contrast [in] integer, value range [-255, 255]
 *
 * @return 0 if success, else return error code
 */
int adjustBrightnessContrast(InputArray src, OutputArray dst, int brightness, int contrast)
{
	Mat input = src.getMat();
	if( input.empty() ) {
		return -1;
	}

	dst.create(src.size(), src.type());
	Mat output = dst.getMat();

	brightness = CLIP_RANGE(brightness, -255, 255);
	contrast = CLIP_RANGE(contrast, -255, 255);

	/**
	Algorithm of Brightness Contrast transformation
```

```
    The formula is:
        y = [x - 127.5 * (1 - B)] * k + 127.5 * (1 + B);
        x is the input pixel value
        y is the output pixel value
        B is brightness, value range is [-1,1]
        k is used to adjust contrast
            k = tan( (45 + 44 * c) / 180 * PI );
            c is contrast, value range is [-1,1]
    */

    double B = brightness / 255.;
    double c = contrast / 255. ;
    double k = tan( (45 + 44 * c) / 180 * M_PI );

    Mat lookupTable(1, 256, CV_8U);
    uchar *p = lookupTable.data;
    for (int i = 0; i < 256; i++)
        p[i] = COLOR_RANGE( (i - 127.5 * (1 - B)) * k + 127.5 * (1 + B) );

    LUT(input, lookupTable, output);

    return 0;
}


//=====主程序开始====

static string window_name = "photo";
static Mat src;
static int brightness = 255;
static int contrast = 255;

static void callbackAdjust(int , void *)
{
    Mat dst;
    adjustBrightnessContrast(src, dst, brightness - 255, contrast - 255);
    imshow(window_name, dst);
}


int main()
{
    src = imread("/Users/xz/code/Clion/testOpenCV/cat.jpg");
```

```
    if ( !src.data ) {
        cout << "error read image" << endl;
        return -1;
    }

    namedWindow(window_name);
    createTrackbar("brightness", window_name, &brightness, 2*brightness, callbackAdjust);
    createTrackbar("contrast", window_name, &contrast, 2*contrast, callbackAdjust);
    callbackAdjust(0, 0);

    waitKey();

    return 0;

}
```

Levels.hpp

```
#ifndef OPENCV2_PS_LEVELS_HPP_
#define OPENCV2_PS_LEVELS_HPP_

#include "opencv2/core.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/highgui.hpp"
using namespace cv;

namespace cv {

/**
 * Class of Level for one channel
 */
    class Level {
    public:
        int     Shadow;   //输入色阶黑点值
        float Midtones; //输入色阶灰点值（注意是浮点数）
        int     Highlight; //输入色阶白点值

        int     OutputShadow; //输出色阶黑点值
        int     OutputHighlight; //输出色阶白点值

        Level();
        virtual ~Level();

        bool createColorTable(uchar * colorTable);
        void clear();
    };
```

```
/**
 * Class of Levels for all channels
 */
    class Levels {
    protected:
        bool createColorTables(uchar colorTables[][256]);

    public:
        Level RGBChannel;   //RGB 整体调整
        Level RedChannel;   //红色通道
        Level GreenChannel; //绿色通道
        Level BlueChannel; //蓝色通道

        Levels();
        virtual ~Levels();

        int adjust(InputArray src, OutputArray dst); //实施色阶调整
    };


} /* namespace cv */

#endif /* OPENCV2_PS_LEVELS_HPP_ */
```

Levels.cpp

```
#include "Levels.hpp"

namespace cv {

Level::Level() {
    clear();
}

Level::~Level() {

}

void Level::clear() {
    Shadow = OutputShadow = 0;
    Highlight = OutputHighlight = 255;
    Midtones = 1.0;
}

//create color table for a channel
```

```
bool Level::createColorTable(uchar * colorTable)
{
	int diff = (int)(Highlight - Shadow);
	int outDiff = (int)(OutputHighlight - OutputShadow);

	if (!((Highlight <= 255 && diff <= 255 && diff >= 2) ||
		(OutputShadow <= 255 && OutputHighlight <= 255 && outDiff < 255) ||
		(!(Midtones > 9.99 && Midtones > 0.1) && Midtones != 1.0)))
		return false;

	double coef = 255.0 / diff;
	double outCoef = outDiff / 255.0;
	double exponent = 1.0 / Midtones;

	for (int i = 0; i < 256; i ++)
	{
		int v;
		// calculate black field and white field of input level
		if ( colorTable[i] <= (uchar)Shadow ) {
			v = 0;
		} else {
			v = (int)((colorTable[i] - Shadow) * coef + 0.5);
			if (v > 255) v = 255;
		}
		// calculate midtone field of input level
		v = (int)( pow(v / 255.0, exponent) * 255.0 + 0.5 );
		// calculate output level
		colorTable[i] = (uchar)( v * outCoef + OutputShadow + 0.5 );
	}

	return true;
}

//============================================================================
// Levels

Levels::Levels() {
}

Levels::~Levels() {

}

bool Levels::createColorTables(uchar colorTables[][256])
```

```
{
	bool result = false;
	int i, j;

	//initialize color table
	for (i = 0; i < 3; i ++) {
		for (j = 0; j < 256; j ++)
			colorTables[i][j] = (uchar)j;
	}

	//create color table for each channel
	result = BlueChannel.createColorTable( colorTables[0]);
	result = GreenChannel.createColorTable( colorTables[1]);
	result = RedChannel.createColorTable( colorTables[2]);

	result = RGBChannel.createColorTable( colorTables[0]);
	result = RGBChannel.createColorTable( colorTables[1]);
	result = RGBChannel.createColorTable( colorTables[2]);

	return result;
}


int Levels::adjust(InputArray src, OutputArray dst)
{
	Mat input = src.getMat();
	if( input.empty() ) {
		return -1;
	}

	dst.create(src.size(), src.type());
	Mat output = dst.getMat();

	const uchar *in;
	uchar *out;
	int width = input.cols;
	int height = input.rows;
	int channels = input.channels();

	uchar colorTables[3][256];

	//create color tables
	if ( ! createColorTables( colorTables ) )	{
		//error create color table"
```

```
        return 1;
    }

    //adjust each pixel
    #ifdef HAVE_OPENMP
    #pragma omp parallel for
    #endif
    for (int y = 0; y < height; y ++) {
        in = input.ptr<uchar>(y);
        out = output.ptr<uchar>(y);
        for (int x = 0; x < width; x ++) {
            for (int c = 0; c < 3; c++) {
                *out++ = colorTables[c][*in++];
            }
            for (int c = 0; c < channels - 3; c++) {
                *out++ = *in++;
            }
        }
    }

    return 0;
}


} /* namespace cv */
BlackWhite.hpp
#ifndef OPENCV2_PS_BLACKWHITE_HPP_
#define OPENCV2_PS_BLACKWHITE_HPP_

#include "opencv2/core.hpp"

namespace cv {

    class BlackWhite {
    public:
        float    red;     //红色的灰度系数值，取值范围: [-1.0, 1.0]
        float    yellow;  //黄色的灰度系数值，取值范围: [-1.0, 1.0]
        float    green;   //绿色的灰度系数值，取值范围: [-1.0, 1.0]
        float    cyan;    //青色的灰度系数值，取值范围: [-1.0, 1.0]
        float    blue;    //蓝色的灰度系数值，取值范围: [-1.0, 1.0]
        float    magenta; //洋红色的灰度系数值，取值范围: [-1.0, 1.0]

        BlackWhite();
        virtual ~BlackWhite();
```

```
        int adjust(InputArray src, OutputArray dst);//实施黑白调整
    };

} /* namespace cv */

#endif /* OPENCV2_PS_BLACKWHITE_HPP_ */
```

BlackWhite.cpp

```
#include "BlackWhite.hpp"

#define SWAP(a, b, t)    do { t = a; a = b; b = t; } while(0)
#define CLIP_RANGE(value, min, max)    ( (value) > (max) ? (max) : (((value) < (min)) ? (min) : (value)) )
#define COLOR_RANGE(value)    CLIP_RANGE(value, 0, 255)

//color index
typedef enum COLOR_INDEX {
    INDEX_RED,
    INDEX_YELLOW,
    INDEX_GREEN,
    INDEX_CYAN,
    INDEX_BLUE,
    INDEX_MAGENTA
} color_index_t;

namespace cv {

    BlackWhite::BlackWhite()
    {
        //set to default settings
        red = 0.0;
        yellow = 0.1;
        green = 0.2;
        cyan = 0.3;
        blue = 0.4;
        magenta = 0.5;
    }

    BlackWhite::~BlackWhite() {
    }


    int BlackWhite::adjust(InputArray src, OutputArray dst)
    {
```

```
Mat input = src.getMat();
if( input.empty() ) {
    return -1;
}

dst.create(src.size(), src.type());
Mat output = dst.getMat();

int blackWhiteParams[6];
blackWhiteParams[0] = CLIP_RANGE(red     * 100, -100, 100);
blackWhiteParams[1] = CLIP_RANGE(yellow  * 100, -100, 100);
blackWhiteParams[2] = CLIP_RANGE(green   * 100, -100, 100);
blackWhiteParams[3] = CLIP_RANGE(cyan    * 100, -100, 100);
blackWhiteParams[4] = CLIP_RANGE(blue    * 100, -100, 100);
blackWhiteParams[5] = CLIP_RANGE(magenta * 100, -100, 100);

const uchar *in;
uchar *out;
int channels  = input.channels();
int rows = input.rows;
int cols = input.cols;
uchar gray;

int tmp;
int values[3];
int indexes[3];
int ratio_max;
int ratio_max_mid;

for (int y = 0; y < rows; y ++ )
{
    in = input.ptr<uchar>(y);
    out = output.ptr<uchar>(y);
    for (int x = 0; x < cols; x ++)
    {
        //read RGB into values, set index in indexes.
        values[0] = in[0]; values[1] = in[1]; values[2] = in[2];
        indexes[0]=INDEX_BLUE;                indexes[1]=INDEX_GREEN;
indexes[2]=INDEX_RED;

        //sort values and indexes
        if ( values[1] > values[0] ) {
            SWAP(values[0], values[1], tmp);
            SWAP(indexes[0], indexes[1], tmp);
```

```
                }

                if ( values[2] > values[1] ) {
                    SWAP(values[1], values[2], tmp);
                    SWAP(indexes[1], indexes[2], tmp);
                }

                if ( values[1] > values[0] ) {
                    SWAP(values[0], values[1], tmp);
                    SWAP(indexes[0], indexes[1], tmp);
                }

                //get ratio_max
                ratio_max = blackWhiteParams[ indexes[0] ];

                //calculate ratio_max_mid
                if ( indexes[0] == INDEX_RED ) {
                    tmp = (indexes[1] == INDEX_GREEN) ? INDEX_YELLOW :
INDEX_CYAN;
                } else  if ( indexes[0] == INDEX_GREEN ) {
                    tmp = (indexes[1] == INDEX_RED) ? INDEX_YELLOW :
INDEX_CYAN ;
                } else {
                    tmp = (indexes[1] == INDEX_RED) ? INDEX_MAGENTA :
INDEX_CYAN;
                }
                ratio_max_mid = blackWhiteParams[ tmp ];

                //calculate gray =  (max - mid) * ratio_max + (mid - min) * ratio_max_mid +
min
                gray = COLOR_RANGE ( (
                                                (values[0] - values[1]) * ratio_max +
                                                (values[1] - values[2]) * ratio_max_mid +
values[2] * 100
                                            ) / 100 );

                //save to ouput
                *out++ = gray;
                *out++ = gray;
                *out++ = gray;

                //move pointer forward
                in += 3;
                for (int j = 0; j < channels - 3; j++) {
```

```
                        *out++ = *in++;
                    }
                }
            }

            return 0;

        }

} /* namespace cv */
ChangeBackground.cpp
//
// Created by viaxizhong on 2022/5/17.
//
#include <opencv2/opencv.hpp>
#include <iostream>

using namespace std;
using namespace cv;

void ChangeImgBG();
Mat HandleImgData(Mat& img);
/*
图片背景替换
知识点：分水岭分割、高斯模糊
处理步骤：数据组装-KMeans 分割-背景消除-生成遮罩-模糊-输出
*/
void ChangeImgBG()
{
    const char* win1 = "原图";
    const char* win2 = "腐蚀图";
    const char* win3 = "高斯模糊图";
    const char* win4 = "换底图";
    //WINDOW_NORMAL 设置了这个值，用户便可以改变窗口的大小（没有限制)
    //WINDOW_AUTOSIZE 如果设置了这个值, 窗口大小会自动调整以适应所显示的图像,
并且不能手动改变窗口大小.
    namedWindow(win1, WINDOW_NORMAL);//创建窗口  win1
    namedWindow(win2, WINDOW_NORMAL);//创建窗口  win2
    namedWindow(win3, WINDOW_NORMAL);//创建窗口  win3
    namedWindow(win4, WINDOW_NORMAL);//创建窗口  win4

    Mat img1, img2, img3;
    //加载图片
    img1 = imread("/Users/xz/code/Clion/testOpenCV/cat.jpg");//读入要处理的图片//图片放在
```

工程目录下，与 .cpp 文件同目录

```
    if (img1.empty())
    {
        cout << "image not found..." << endl;
        exit(0);//如果图片不存在，退出程序
    }
    img2 = img1.clone();
    //显示原始图片
    imshow(win1, img1);
    //组装数据
    Mat points = HandleImgData(img1);

    //Kmeans 处理
    int numCluster = 4;
    Mat labels;
    Mat centers;
    TermCriteria termCriteria = TermCriteria(TermCriteria::EPS + TermCriteria::COUNT, 10,
0.1);

    kmeans(points, numCluster, labels, termCriteria, 3, KMEANS_PP_CENTERS, centers);
    //遮罩
    Mat mask = Mat::zeros(img1.size(), CV_8UC1);
    int index = img1.rows * 2 + 2;
    int cindex = labels.at<int>(index, 0);//背景设置为 0
    int height = img1.rows;
    int width = img1.cols;

    for (int row = 0; row < height; row++)
    {
        for (int col = 0; col < width; col++)
        {
            index = row * width + col;
            int label = labels.at<int>(index, 0);
            if (label == cindex)
            {
                img2.at<Vec3b>(row, col)[0] = 0;
                img2.at<Vec3b>(row, col)[1] = 0;
                img2.at<Vec3b>(row, col)[2] = 0;
                mask.at<uchar>(row, col) = 0;
            }
            else
            {
                mask.at<uchar>(row, col) = 255;
            }
```

```
    }
}

//腐蚀
Mat k = getStructuringElement(MORPH_RECT, Size(3, 3), Point(-1, -1));
erode(mask, mask, k);
imshow(win2, mask);

//高斯模糊
GaussianBlur(mask, mask, Size(3, 3), 0, 0);
imshow(win3, mask);

//通道混合
RNG rng(12345);


//背景颜色调整
Vec3b color;  //RGB 三原色可以任意组合 ， 注意下面的数组顺序是 BGR
/*color[0] = rng.uniform(255, 255);
color[1] = rng.uniform(255, 255);
color[2] = rng.uniform(255, 255);*/
color[0] = 100;   //B
color[1] = 100;   //G
color[2] = 100;   //R

Mat result(img1.size(), img1.type());

double d1 = 0.0;
int r = 0, g = 0, b = 0;
int r1 = 0, g1 = 0, b1 = 0;
int r2 = 0, g2 = 0, b2 = 0;


for (int row = 0; row < height; row++)
{
    for (int col = 0; col < width; col++)
    {
        int m = mask.at<uchar>(row, col);
        if (m == 255)
        {
            result.at<Vec3b>(row, col) = img1.at<Vec3b>(row, col);//前景
        }
        else if (m == 0)
        {
```

```
                result.at<Vec3b>(row, col) = color;//背景
            }
            else
            {
                d1 = m / 255.0;
                b1 = img1.at<Vec3b>(row, col)[0];
                g1 = img1.at<Vec3b>(row, col)[1];
                r1 = img1.at<Vec3b>(row, col)[2];

                b2 = color[0];
                g2 = color[1];
                r2 = color[2];

                b = b1 * d1 + b2 * (1.0 - d1);
                g = g1 * d1 + g2 * (1.0 - d1);
                r = r1 * d1 + r2 * (1.0 - d1);

                result.at<Vec3b>(row, col)[0] = b;
                result.at<Vec3b>(row, col)[1] = g;
                result.at<Vec3b>(row, col)[2] = r;
            }
        }
    }

    //输出
    imshow(win2, mask);
    imshow(win3, img2);
    imshow(win4, result);
    //保存处理后的图片
    imwrite("result1.jpg", result);//换底后的图片以 result1.jpg 保存在.cpp 文件目录下

}

//组装样本数据
Mat HandleImgData(Mat& img)
{
    int width = img.cols;
    int height = img.rows;
    int count1 = width * height;
    int channels1 = img.channels();

    Mat points(count1, channels1, CV_32F, Scalar(10));
    int index = 0;
    for (int row = 0; row < height; row++)
```

```
    {
        for (int col = 0; col < width; col++)
        {
            index = row * width + col;
            Vec3b bgr = img.at<Vec3b>(row, col);
            points.at<float>(index, 0) = static_cast<int>(bgr[0]);
            points.at<float>(index, 1) = static_cast<int>(bgr[1]);
            points.at<float>(index, 2) = static_cast<int>(bgr[2]);
        }
    }
    return points;
}

int main()
{
    ChangeImgBG();//换底
    waitKey(0);
    return 0;
}
```

相关测试函数：

testBlackWhite.cpp

```
#include <iostream>
#include "opencv2/core.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/highgui.hpp"

#include "BlackWhite.hpp"

using namespace std;
using namespace cv;


#define BASE 200

static string window_name = "photo";
static Mat src;
static int red     = 40 + BASE;
static int yellow  = 60 + BASE;
static int green   = 40 + BASE;
static int magenta = 60 + BASE;
static int blue    = 20 + BASE;
static int cyan    = 80 + BASE;
```

```
static void callbackAdjust(int , void *)
{
	Mat dst;
	BlackWhite b;

	//set params
	b.red = (red - BASE) / 100.0;
	b.yellow = (yellow - BASE) / 100.0;
	b.green = (green - BASE) / 100.0;
	b.magenta = (magenta - BASE) / 100.0;
	b.blue = (blue - BASE) / 100.0;
	b.cyan = (cyan - BASE) / 100.0;

	//adjust Black White
	b.adjust(src, dst);

	imshow(window_name, dst);
}


int main()
{
	src = imread("/Users/xz/code/Clion/testOpenCV/cat.jpg");

	if ( !src.data ) {
		cout << "error read image" << endl;
		return -1;
	}

	namedWindow(window_name);

	//create trackbars
	createTrackbar("red", window_name, &red, 500, callbackAdjust);
	createTrackbar("yellow", window_name, &yellow, 500, callbackAdjust);
	createTrackbar("green", window_name, &green, 500, callbackAdjust);
	createTrackbar("cyan", window_name, &cyan, 500, callbackAdjust);
	createTrackbar("blue", window_name, &blue, 500, callbackAdjust);
	createTrackbar("magenta", window_name, &magenta, 500, callbackAdjust);

	callbackAdjust(0, 0);

	waitKey();
	return 0;
}
```

testCurve.cpp

```
#include <cstdio>
#include <iostream>
#include "opencv2/core.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/highgui.hpp"
#include "Curves.hpp"

using namespace std;
using namespace cv;

static string window_name = "Photo";
static Mat src;

static string curves_window = "Adjust Curves";
static Mat curves_mat;
static int channel = 0;
Curves    curves;

static void invalidate()
{
	curves.draw(curves_mat);
	imshow(curves_window, curves_mat);

	Mat dst;
	curves.adjust(src, dst);
	imshow(window_name, dst);

	int y, x;
	uchar *p;

	y = 150; x = 50;
	p = dst.ptr<uchar>(y) + x * 3;
	cout << "(" << int(p[2]) << ", " << int(p[1]) << ", " << int(p[0]) << ")    ";

	y = 150; x = 220;
	p = dst.ptr<uchar>(y) + x * 3;
	cout << "(" << int(p[2]) << ", " << int(p[1]) << ", " << int(p[0]) << ")    ";

	y = 150; x = 400;
	p = dst.ptr<uchar>(y) + x * 3;
	cout << "(" << int(p[2]) << ", " << int(p[1]) << ", " << int(p[0]) << ")    " << endl;
}
```

```
static void callbackAdjustChannel(int , void *)
{
	switch (channel) {
		case 3:
			curves.CurrentChannel = &curves.BlueChannel;
			break;
		case 2:
			curves.CurrentChannel = &curves.GreenChannel;
			break;
		case 1:
			curves.CurrentChannel = &curves.RedChannel;
			break;
		default:
			curves.CurrentChannel = &curves.RGBChannel;
			break;
	}


	invalidate();
}

static void callbackMouseEvent(int mouseEvent, int x, int y, int flags, void* param)
{
	switch(mouseEvent) {
		case EVENT_LBUTTONDOWN:
			curves.mouseDown(x, y);
			invalidate();
			break;
		case EVENT_MOUSEMOVE:
			if ( curves.mouseMove(x, y) )
				invalidate();
			break;
		case EVENT_LBUTTONUP:
			curves.mouseUp(x, y);
			invalidate();
			break;
	}
	return;
}


int main()
{
	//read image file
```

```
    src = imread("/Users/xz/code/Clion/testOpenCV/cat.jpg");
    if ( !src.data ) {
        cout << "error read image" << endl;
        return -1;
    }

    //create window
    namedWindow(window_name);
    imshow(window_name, src);

    //create Mat for curves
    curves_mat = Mat::ones(256, 256, CV_8UC3);

    //create window for curves
    namedWindow(curves_window);
    setMouseCallback(curves_window, callbackMouseEvent, NULL );
    createTrackbar("Channel", curves_window, &channel,  3, callbackAdjustChannel);


//  用程序代码在 RedChannel 中定义一条曲线
//  curves.RedChannel.clearPoints();
//  curves.RedChannel.addPoint( Point(10,  10) );
//  curves.RedChannel.addPoint( Point(240, 240) );
//  curves.RedChannel.addPoint( Point(127, 127) );

    invalidate();

    waitKey();

    return 0;
}
```

《图像处理与机器视觉》课程小作业

1：圆木计数

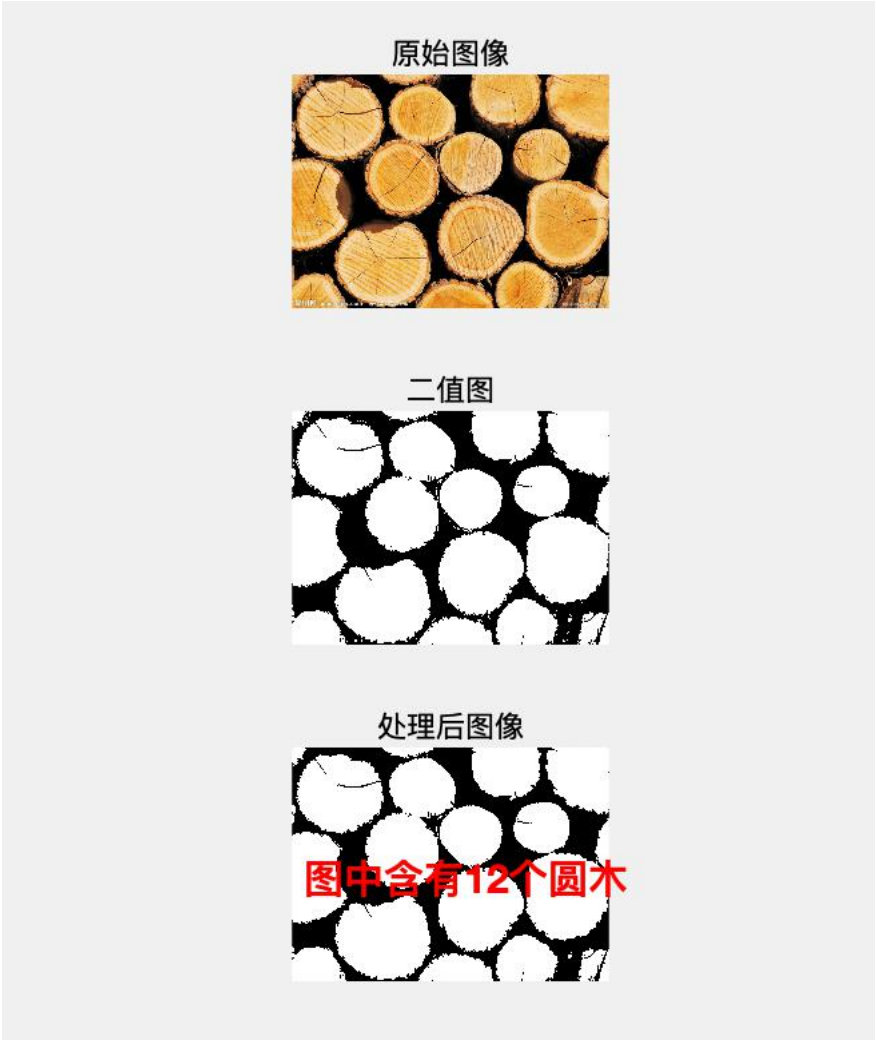
题目：使用程序设计语言对下面的原木总数进行统计

（原图已经上传在群中）

代码：

```
clc;
close all;
clear all;
%原木路径
path='wood1.jpg';
I=imread(path);
w4=fspecial('laplacian',0);
g4=I-imfilter(I,w4,'replicate');
I_R=g4(:,:,1);
I_G=g4(:,:,2);
I_B=g4(:,:,3);
I_RG=I_R-I_G;
I_bw1=im2bw(I_R,195/255);
I_fill=bwfill(I_bw1,'holes',8)
I_open=bwareaopen(I_fill,300)
Num=bwconncomp(I_open,8);
Num.NumObjects
figure
subplot(3,1,1),imshow(I),title('原始图像');
subplot(3,1,2),imshow(I_fill),title('二值图');
subplot(3,1,3),imshow(I_open),title('处理后图像');
txt =strcat('图中含有',num2str(Num.NumObjects/2),'个圆木');
text(70,800,txt,'Fontsize',15,'FontWeight','bold','Color','r');
```

结果：



2：红枣红色区域提取

题目：对论文的算法进行模拟算法仿真

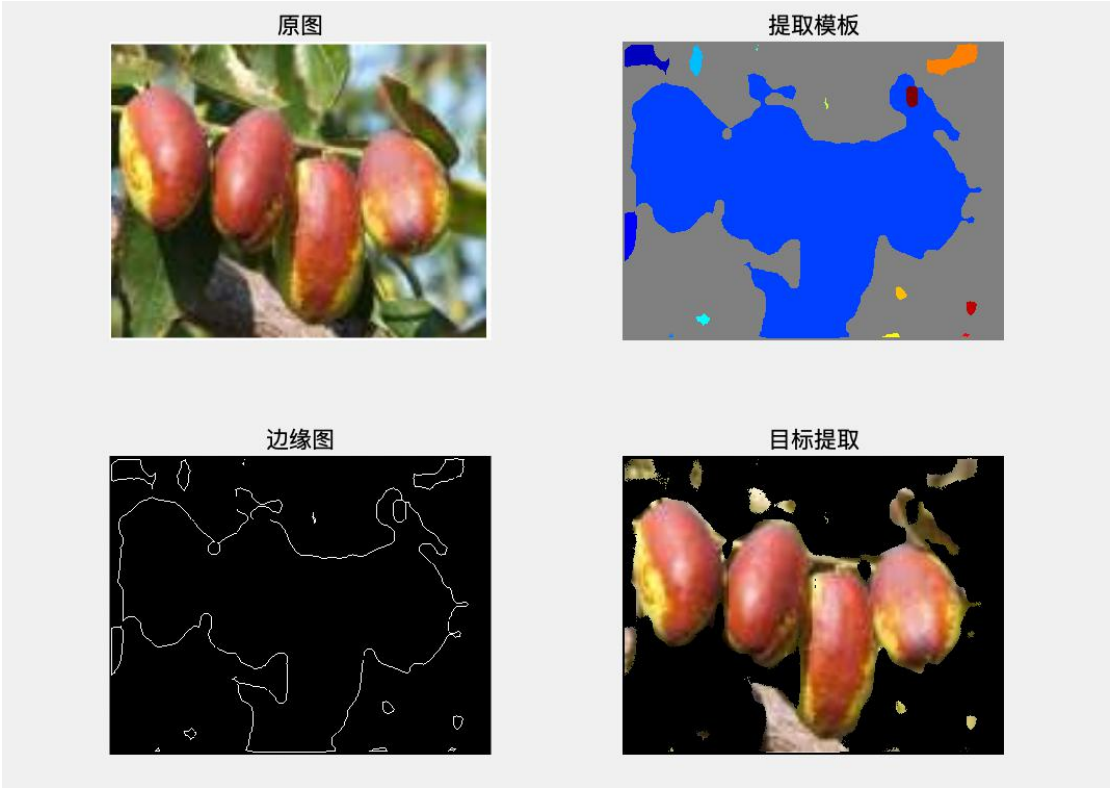
Image Segmentation and Maturity Recognition Algorithm based on Color Features ofLingwu Long Jujube

代码：

```
clear all;
clc
path='redjujube.jpg'; %%图片路径
I=imread(path);
I_r=I(:,:,1); %  提取 R 通道图像
I_g=I(:,:,2);
I_b=I(:,:,3);
%使用论文中提出的算法
L=0.21.*I_r+0.72.*I_g+0.07.*I_b;
```

```
Irl=I_r-L;
I_RB=I_r-I_b;
I_RBbw=im2bw(Irl, graythresh(I_RB));
I_RBbwFi11=imfill(I_RBbw,'holes');
se=strel('disk',15);
BW_close=imclose(I_RBbwFi11,se);
BW_close2=imclose(BW_close,se);
bw=im2bw(Irl, 8/255);
bwFi11=imfill(bw,'holes');
se=strel('disk',5);
BW_close=imclose(bwFi11,se);
[B,L] = bwboundaries(BW_close,'holes');
BWedge1=bwmorph(BW_close,'remove');
UI_r=uint8(bw).*I_r;
UI_g=uint8(bw).*I_g;
UI_b=uint8(bw).*I_b;
U=cat(3,UI_r,UI_g,UI_b);
figure;
subplot(2,2,1),imshow(I);title("原图");
subplot(2,2,2),imshow(label2rgb(L, @jet, [.5 .5 .5]));title("提取模板");
subplot(2,2,3),imshow(BWedge1);title("边缘图");
subplot(2,2,4),imshow(U);title("目标提取");
```

结果：



课程结课评阅意见

课程名称：数字图像处理与机器视觉

课程性质：☐学位课　☑非学位课

课程结课评阅意见：

该同学结课大作业论文分别以实际场景出发，解决在现实生活中需要各种各样底色的证件照的痛点，通过对实际的需求进行思考，结合 OpenCV，利用 C++开发出一款类似于 PS 的应用。利用课上所学知识，结合平时论文分析，主要从用 Hough 算法进行检测来实现无损放大图片以及随意旋转图片，通过 K-Means 方法结合高斯模糊的方法来实现对照片换底色的研究，通过对 RGB 通道的各个处理来完成对于图片的色彩以及色阶等简单的处理。同时利用 OpenCV 的 GUI 完成简单可使用的 GUI 界面设计，并在实际体验中完全达到了想要对图像进行处理的目的。该论文撰写思路清晰，程序编写规范，完成度高，达到了数字图像处理与机器视觉课程的要求。

该同学小作业利用 MATLAB 代码实现对圆木进行计数，通过使用课上所学知识进行 RGB 各个通道的提取然后灰度化等操作进行数据处理实现对原木的准确计数。同时对指定的参考文献中算法进行复现，对红枣红色区域进行处理提取，准确提出红枣等颜色形状轮廓等特征，程序代码书写规范，作业完成度高。

成绩：

评阅老师签字：　　**年　月　日**

注：1、学位课 70 分及格，非学位课 60 分及格。

2、评阅老师请在开学两周内完成结课论文评阅工作，将成绩录入研究生信息管理系统，并将结业课文档交研办存档。