```
#include<iostream>
#include<opencv2/opencv.hpp>
#include"function.h"
using namespace cv;
using std::cout;
using std::endl;

/*****************************************************************************************************************************************
第三周练习1：肤色提取实验
利用不同色域物体在HSV色彩空间上的不同色域，实现对人肤色的提取
*****************************************************************************************************************************************/
void skinRecognition()
{
VideoCapture cap(0);//打开0号摄像头
double scale = 0.5;

//肤色h
double i_minH = 0;
double i_maxH = 20;

//颜色饱和度s
double i_minS = 43;
double i_maxS = 255;

//颜色亮度v
double i_minV = 55;
double i_maxV = 255;

while (1)
{
//定义图像容器
Mat frame;
Mat hsvMat;
Mat detectMat;

cap >> frame;//读取当前帧的照片

//修改图片尺寸大小
Size ResImgSiz = Size(frame.cols*scale, frame.rows*scale);
Mat rFrame = Mat(ResImgSiz, frame.type());
resize(frame, rFrame, ResImgSiz, INTER_LINEAR);

//将原图转化为hsv类型的图片

/*********************************************
cvTColor（） 函数模型：
cvtColor( InputArray src, OutputArray dst, int code, int dstCn = 0 );

参数介绍：
. InputArray src: 输入图像即要进行颜色空间变换的原图像，可以是Mat类
. OutputArray dst: 输出图像即进行颜色空间变换后存储图像，也可以Mat类
. int code: 转换的代码或标识，即在此确定将什么制式的图片转换成什么制式的图片，
. int dstCn = 0: 目标图像通道数，如果取值为0，则由src和code决定

*********************************************/
cvtColor(rFrame, hsvMat, COLOR_BGR2HSV);

//对detecMat进行初始化
frame.copyTo(detectMat);
//利用inRange函数对图片进行hsv筛选

/*********************************************
inRange（） 函数模型：
inRange(InputArray src, InputArray lowerb, InputArray upperb, OutputArray dst);

参数介绍：
. InputArray src: 输入要处理的图像，可以为单通道或多通道
. InputArray lowerb: 包含下边界的数组或标量
. InputArray upperb: 包含上边界数组或标量
. OutputArray dst: 输出图像，与输入图像src 尺寸相同且为CV_8U 类型

*********************************************/
cv::inRange(hsvMat, Scalar(i_minH, i_minS, i_minV), Scalar(i_maxH, i_maxS, i_maxV), detectMat);


imshow("while:in the range", detectMat);
imshow("frame", rFrame);

waitKey(30);
}

}

/*****************************************************************************************************************************************
第三周练习2：
调用几种图像二值化的方法
*****************************************************************************************************************************************/
void binarization()
{
//读取图片并转化为灰度图
cv::Mat srcMat = cv::imread("d:\\timg.jpg", 0);

//判断图片是否提取成功
if (srcMat.empty())
{
cout << "fail to read !" << endl;
return;
```

```
}


//定义图像容器
cv::Mat bin_Mat;
cv::Mat otsu_Mat;
cv::Mat adap_Mat;

/***********************************************
threshold()  函数模型:
threshold(InputArray src, OutputArray dst, double thresh, double maxval, int type);

参数介绍:
. InputArray src: 输入要处理的图像，可以为单通道或多通道
. OutputArray dst: 输出图像，与输入图像src 尺寸相同且为CV_8U 类型
. double thresh:阈值
. double maxval:输出图像最大值
. int type:阈值类型
0: THRESH_BINARY  当前点值大于阈值时，取Maxval,否则设置为0
1: THRESH_BINARY_INV 当前点值大于阈值时，设置为0，否则设置为Maxval
2: THRESH_TRUNC 当前点值大于阈值时，设置为阈值，否则不改变
3: THRESH_TOZERO 当前点值大于阈值时，不改变，否则设置为0
4: THRESH_TOZERO_INV  当前点值大于阈值时，设置为0，否则不改变
...
8:THRESH_OTSU  通过大津法求出最佳阈值，大于阈值置255，小于置0


***********************************************/

//普通二值化方法
threshold(srcMat, bin_Mat, 100, 255, CV_THRESH_BINARY);

//大津法
threshold(srcMat, otsu_Mat, 100, 255, CV_THRESH_OTSU);

/***********************************************
adaptiveThreshold()  函数模型:
adaptiveThreshold( InputArray src, OutputArray dst,double maxValue, int adaptiveMethod,int thresholdType, int blockSize, double C );

参数介绍:
. 参数1：InputArray类型的src，输入图像，填单通道，单8位浮点类型Mat即可。
. 参数2：函数运算后的结果存放在这。即为输出图像（与输入图像同样的尺寸和类型）。
. 参数3：预设满足条件的最大值。
. 参数4：指定自适应阈值算法。可选择ADAPTIVE_THRESH_MEAN_C 或 ADAPTIVE_THRESH_GAUSSIAN_C两种。
. 参数5：指定阈值类型。可选择THRESH_BINARY或者THRESH_BINARY_INV两种。（即二进制阈值或反二进制阈值）。
. 参数6：表示邻域块大小，用来计算区域阈值，一般选择为3、5、7.....等。
    . 参数7：参数C表示与算法有关的参数，它是一个从均值或加权均值提取的常数，可以是负数。

***********************************************/
//区域自适应二值化
adaptiveThreshold(srcMat, adap_Mat, 255, ADAPTIVE_THRESH_GAUSSIAN_C, THRESH_BINARY, 15, 10);//适合对那些光照不均的图片进行二值化，因为它的二值化阈值是自适应的

//显示结果图像
cv::imshow("bin_Mat", bin_Mat);
cv::imshow("otsu_Mat", otsu_Mat);
cv::imshow("adap_Mat", adap_Mat);


cv::waitKey(0);
}

/*****************************************************************************************************************************
第三周练习3:
创建一个滑动条，可通过移动滑动条调节二值化阈值
*****************************************************************************************************************************/


//回调函数
void callback(int th, void* data)
{
//强制类型转换
cv::Mat src = *((cv::Mat*) data);

//定义输出图片容器
cv::Mat dst;

//二值化函数
threshold(src, dst, th, 255, CV_THRESH_BINARY);
cv::imshow("bar", dst);
}
//主函数
void trackbar()
{
//定义图像容器
cv::Mat srcMat;
cv::Mat gryMat;


//初始化滑动条的值，并设置滑动节的调节范围
int lowth = 30;
int maxth = 255;

//读取图片
srcMat = cv::imread("d:\\timg.jpg");

//判断图片读取是否成功
```

```
if (srcMat.empty())
{
cout << "fail to read!" << endl;
}


//转化为灰度图
cvtColor(srcMat, gryMat, CV_BGR2GRAY);

//显示灰度图
namedWindow("bar");
cv::imshow("bar", gryMat);

/*********************************************
createTrackbar（） 函数模型:
createTrackbar(const String& trackbarname, const String& winname,
                               int* value, int count,
                               TrackbarCallback onChange = 0,
                               void* userdata = 0);

参数介绍:
. 参数1：const String&类型的trackbarname，用来代表轨迹条的名字
. 参数2：onst String&类型的winname,滑动空间用于依附的图像窗口的名称。
. 参数3：int* 类型的value，初始化阈值。
. 参数4：int类型count，滑动空间的刻度范围。
. 参数5： TrackbarCallback类型的onChange，这是一个指向回调函数的指针。
. 参数6：void* 类型的userdata，用户传给回调函数的数据，用来处理轨迹条事件。

*********************************************/
//创建滑动条函数
createTrackbar("threshold", "bar", &lowth, maxth, callback, &gryMat);
cv::waitKey(0);

}
```