**版本信息**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本信息记录表** | | | | |
| 版本号 | 修订内容 | 修订人 | 修订日期 | 审核人 |
| 1.0 | 初版 | 党慧杰 | 2018.08.20 |  |
| 2.0 | 修订 | 党慧杰 | 2018.10.15 |  |
|  |  |  |  |  |

# 颜色识别

## 【实验目的】

1、学习OpenCV颜色空间的相互转换。

2、在图像中识别指定的颜色。

## 【实验原理】

**1. OpenCV简介**

OpenCV是一个开放源代码的计算机视觉应用平台，由英特尔公司下属研发中心俄罗斯团队发起该项目，开源BSD证书，OpenCV的目标是实现实时计算机视觉，是一个跨平台的计算机视觉库。其应用领域包括：二维和三维特征工具箱、运动估算、人脸系统识别、姿势识别、人机交互、移动机器人、运动理解、对象鉴别、分割与识别、立体视觉、运动跟踪、增强实现(AR技术)。

**2. 颜色空间转换**

颜色空间也称彩色模型(又称彩色空间或彩色系统）它的用途是在某些标准下用通常可接受的方式对彩色加以说明。本质上，彩色模型是[坐标系统](https://baike.baidu.com/item/%E5%9D%90%E6%A0%87%E7%B3%BB%E7%BB%9F)和子空间的阐述。位于系统的每种颜色都有单个点表示。采用的大多数[颜色模型](https://baike.baidu.com/item/%E9%A2%9C%E8%89%B2%E6%A8%A1%E5%9E%8B)都是面向硬件或面向应用的。

**1）RGB颜色空间**

RGB是通过红绿蓝三原色来描述颜色的颜色空间，R=Red、G=Green、B=Blue。RGB颜色空间以R(Red红)、G(Green绿)、B(Blue蓝)三种基本色为基础，进行不同程度的叠加，产生丰富而广泛的颜色，所以俗称三基色模式。

**2）CIE-XYZ颜色空间**

XYZ系统，就是在RGB系统的基础上，用数学方法，选用三个理想的原色来代替实际的三原色，从而将CIE-RGB系统中的光谱三刺激值和色度坐标r、g、b均变为正值。

与RGB颜色空间相互转换关系如下图所示：

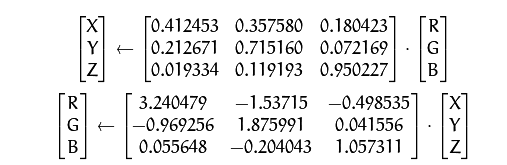


图 1

**3）YCrCb颜色空间**

YUV主要用于优化彩色视频信号的传输，使其向后相容老式黑白电视。与RGB视频信号传输相比，它最大的优点在于只需占用极少的频宽（RGB要求三个独立的视频信号同时传输）。其中“Y”表示明亮度（Luminance或Luma），也就是灰阶值；而“U”和“V” 表示的则是色度（Chrominance或Chroma），作用是描述影像色彩及饱和度，用于指定像素的颜色。“亮度”是透过RGB输入信号来建立的，方法是将RGB信号的特定部分叠加到一起。“色度”则定义了颜色的两个方面─色调与饱和度，分别用Cr和Cb来表示。其中，Cr反映了RGB输入信号红色部分与RGB信号亮度值之间的差异。而Cb反映的是RGB输入信号蓝色部分与RGB信号亮度值之同的差异。

与RGB转换关系如下：

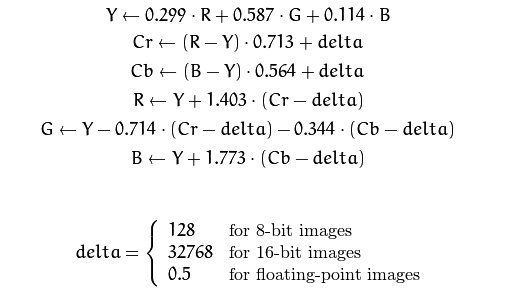


图 2

**4）HSV颜色空间**

HSV是一种将RGB色彩空间中的点在倒圆锥体中的表示方法。HSV即色相(Hue)、饱和度(Saturation)、明度(Value)，又称HSB(B即Brightness)。色相是色彩的基本属性，就是平常说的颜色的名称，如红色、黄色等。饱和度（S）是指色彩的纯度，越高色彩越纯，低则逐渐变灰，0-100%的数值。明度（V），取0-max。HSV颜色空间可以用一个圆锥空间模型来描述。圆锥的顶点处，V=0，H和S无定义，代表黑色。圆锥的顶面中心处V=max，S=0，H无定义，代表白色。

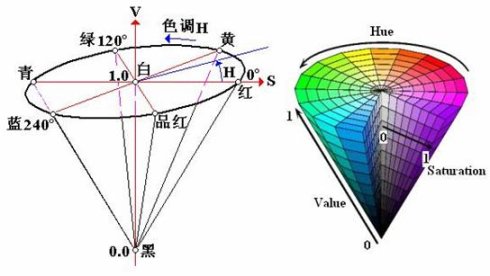


图 3

RGB到HSV的转换如下：设max等于r、g和b中的最大者，min为最小者。对应的HSV空间中的(h,s,v)值为

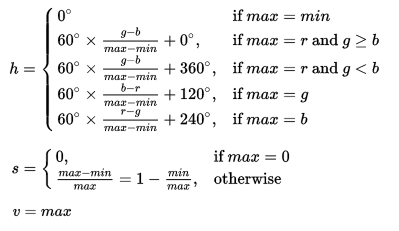


图 4

h在0到360°之间，s在0到100%之间，v在0到max之间。

HSV到RGB的转换如下：

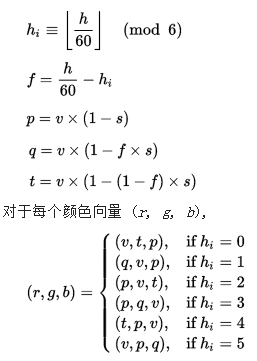


图 5

在OpenCV中提供了了各个颜色空间相互转换的函数，其定义如下：

void cvtColor(InputArray src, OutputArray dst, int code, int dstCn=0 );

其中：

* src: 输入图像；
* dst: 输出图像；
* 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；
* dstCn = 0: 目标图像通道数，如果取值为0，则由src和code决定。

下图表示各个颜色空间相互转换时使用的标识码：



图 6

**3. 颜色识别（提取）**

多少图片都使用的RGB颜色空间，而RGB通道不能很好的反映出物体的颜色信息。而相对于RGB空间，HSV空间能够非常直观的表达色彩的明暗，色调，以及鲜艳程度，方便进行颜色之间的对比，

HSV各种颜色的范围如下图所示：



图 7

颜色识别的过程如下：

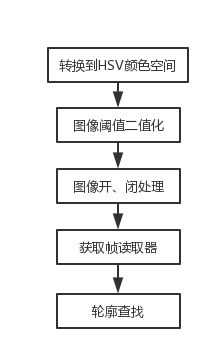


图 8

## 【实验环境】

操作系统

Windows 系列 36位/64位

开发环境

Visual Studio 2012、OpenCV

开发语言

C++

设备

PC（配备USB3.0）、Kinect

## 【实验步骤】

### 一、新建Win32控制台应用程序

1. 点击菜单栏【文件】->【新建】->【项目】或快捷件Ctrl + Shift + N 新建项目，如下图所示：



图 9

2. 在弹出的【新建项目】框中选择模板Visual C++，在中间的栏目中选择【Win32 控制台应用程序】。输入工程名称和保存路径，点击确定。

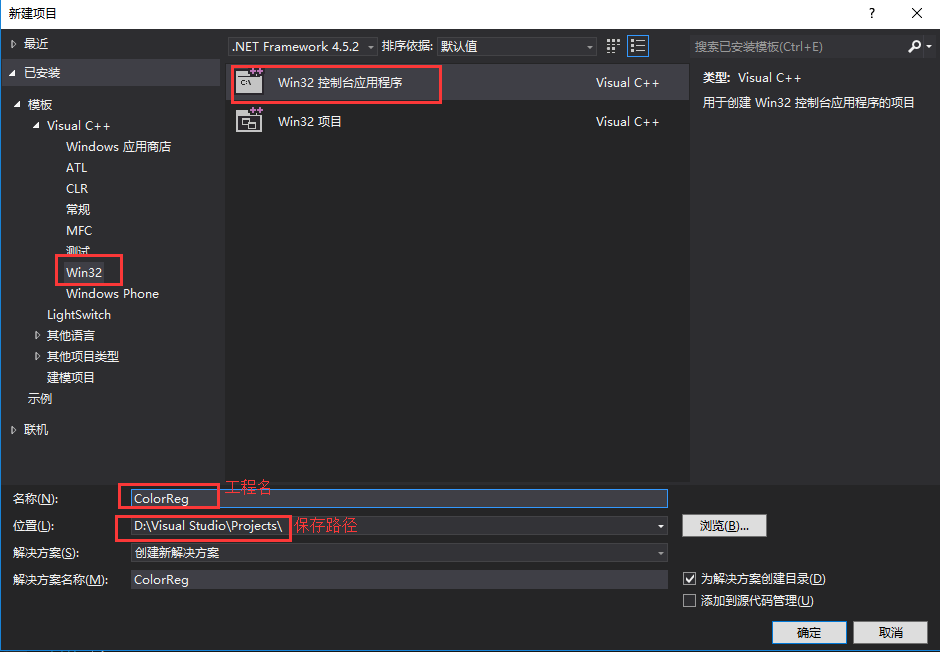


图10

3. 在弹出的【Win32 应用程序向导】对话框中，保持默认配置，直至完成。

### 二、OpenCV配置

在Visual Studio 2012项目中配置OpenCV，参见文档《Opencv的配置》。Kinect的配置，参见文档《Kinect使用说明》。

### 三、颜色空间转换

**1. 新建ImageHandler类**

1）右击项目，选择【添加】->【类】，如下图所示：

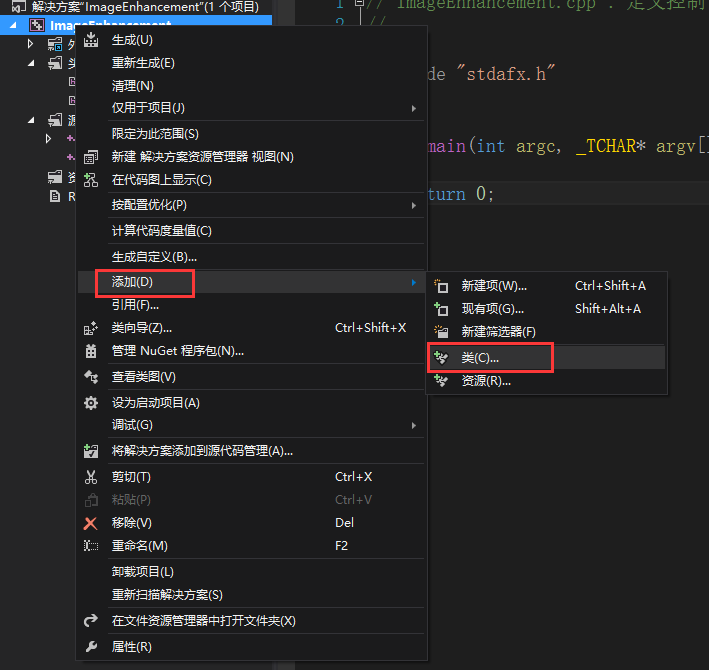


图11

2）选择C++类，点击【添加】

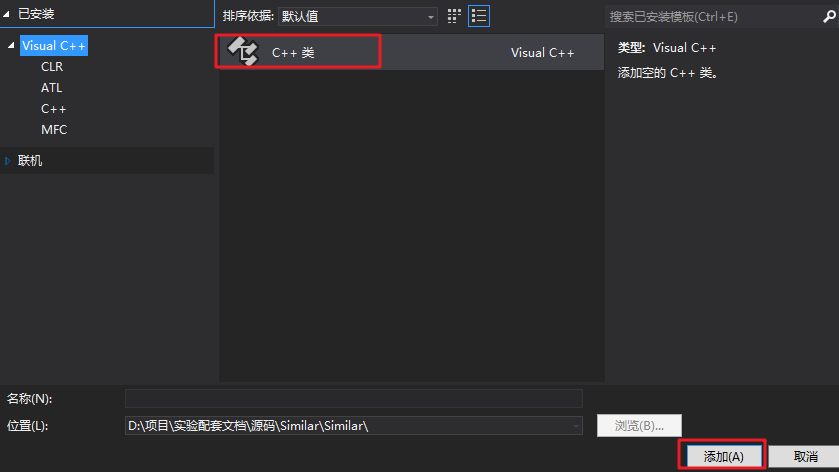


图12

3）输入类名，点击确定，新类就创建完成。

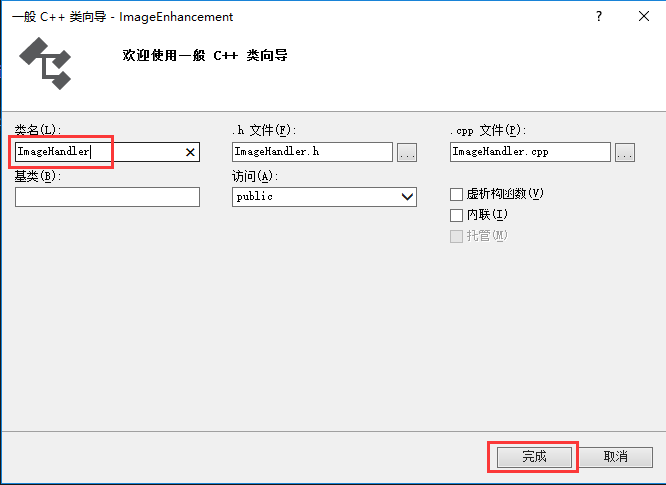


图13

**2. 引入必需的头文件和命名空间**

该方法主要是利用OpenCV提供的函数来完成，因而首先必需添加相关的头文件。打开‘ImageHandler.h’文件。

添加如下头文件：

#include <opencv2/opencv.hpp>

引入命名空间：

using namespace cv;

如下图所示：

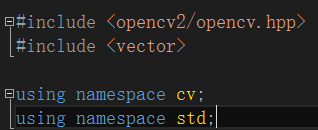


图14

注意：如果不能导入头文件，请检查OpenCV的配置是否正确。

**3. 颜色空间转换的实现**

1）声明函数

打开‘ImageHandler.h’文件，在 ImageHandler 类中添加如下函数声明：

void ConvertColor(Mat frame);

其中，参数frame即为需要颜色空间转换图像。

2）函数实现

打开‘ImageHandler.cpp’文件，键入如下程序：

void ConvertColor(Mat frame);void ImageHandler::ConvertColor(Mat frame)

{

// RGB 转 灰度

Mat gray;

cvtColor(frame, gray, CV\_RGB2GRAY);

imshow("灰度", gray);

// RGB 转 CIE XYZ

Mat xyz;

cvtColor(frame, xyz, CV\_RGB2XYZ);

imshow("XYZ", xyz);

// RGB 转 YCrCb

Mat yCrCb;

cvtColor(frame, yCrCb, CV\_RGB2YCrCb);

imshow("YCrCb", yCrCb);

// RGB 转 HSV

Mat hsv;

cvtColor(frame, hsv, CV\_RGB2HSV);

imshow("HSV", hsv);

}

该函数实现了RGB空间到灰度、XYZ、YCrCb和HSV的转换。

### 五、颜色识别

**1. 定义表示HSV颜色值的结构体**

打开‘ImageHandler.h’头文件，在其中定义结构体‘HSVBoundary’该结构体表示了RGB颜色在HSV空间中取值范围。代码如下：

struct HSVBoundray

{

int HLow;

int SLow;

int VLow;

int HUp;

int SUp;

int VUp;

};

**2. 定义识别颜色的枚举类型**

打开‘ImageHandler.h’头文件，定义枚举‘ColorType’，该类型规定了程序可以识别的颜色。代码如下：

enum ColorType

{

Red,

Blue,

Green,

Yello,

Purple,

Black,

White,

Gray

};

**3. 初始识别颜色在HSV中的取值**

根据图7中的HSV各种颜色的范围，初始待识别颜色的取值范围。在ImageHandler类中定义如下变量：

// 待识别颜色HSV

HSVBoundray HsvRed, HsvBlue, HsvGreen, HsvYellow, HsvPurple, HsvBlack, HsvWhite, HsvGray;

在ImageHandler的构造函数中初始化这些值：

// 红色

HsvRed.HLow=156;

HsvRed.SLow=43;

HsvRed.VLow=46;

HsvRed.HUp=180;

HsvRed.SUp=255;

HsvRed.VUp=255;

// 蓝色

HsvBlue.HLow=100;

HsvBlue.SLow=43;

HsvBlue.VLow=46;

HsvBlue.HUp=124;

HsvBlue.SUp=255;

HsvBlue.VUp=255;

// 绿色

HsvGreen.HLow=35;

HsvGreen.SLow=43;

HsvGreen.VLow=46;

HsvGreen.HUp=77;

HsvGreen.SUp=255;

HsvGreen.VUp=255;

其它颜色，按照这个格式进行初始化。

**4. 申明函数**

在ImageHandler类中，申明如下函数：

private HSVBoundray GetColorBoundary(ColorType);

该函数获取待识别颜色在HSV中的取值范围。和函数：

public int RecogColor(Mat& frame, ColorType color);

该函数实现在图像中识别指定的颜色，frame即为待处理的图像；color为待识别的颜色。

**5. 函数的实现**

打开‘ImageHandler.cpp’文件。首先实现GetColorBoundary函数，实现代码如下：

// 获取待识别颜色在HSV上的范围值

HSVBoundray ImageHandler::GetColorBoundary(ColorType color)

{

HSVBoundray hsvResult;

switch (color)

{

case Red:

hsvResult = HsvRed;

break;

case Blue:

hsvResult = HsvBlue;

break;

case Green:

hsvResult = HsvGreen;

break;

case Yello:

hsvResult = HsvYellow;

break;

case Purple:

hsvResult = HsvPurple;

break;

case Black:

hsvResult = HsvBlack;

break;

case White:

hsvResult = HsvWhite;

break;

case Gray:

hsvResult = HsvGray;

break;

default:

break;

}

return hsvResult;

}

接着实现RecogColor函数：

int ImageHandler::RecogColor(Mat &sourceFrame, ColorType color)

{

Mat imgHSV, resultImage;

//颜色空间转换

cvtColor(sourceFrame, imgHSV, COLOR\_BGR2HSV);

//颜色过滤

HSVBoundray hsv = GetColorBoundary(color);

inRange(imgHSV, Scalar(hsv.HLow, hsv.SLow, hsv.VLow), Scalar(hsv.HUp,hsv.SUp,hsv.VUp), resultImage);

//去噪

Mat element = getStructuringElement(MORPH\_RECT, Size(5, 5));

morphologyEx(resultImage, resultImage, MORPH\_OPEN, element, Point(-1, -1), 1);

morphologyEx(resultImage, resultImage, MORPH\_CLOSE, element, Point(-1, -1), 4);

// 查找区域

vector < vector < cv::Point> > contourAll;

vector < Vec4i> hierarchy;

findContours(resultImage, contourAll, hierarchy, RETR\_TREE, CV\_CHAIN\_APPROX\_NONE);

// 绘制识别区域

int shapeCount = contourAll.size();

Mat lineImage = Mat(resultImage.rows,resultImage.cols,CV\_8UC1,1);

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

{

drawContours(lineImage,contourAll,i,Scalar(255), CV\_FILLED);

}

imshow("识别颜色区域", lineImage);

return 0;

}

### 六、从Kinect获取图像并识别指定颜色

**1. 引入头文件**

在主程序入口函数之前，添加如下的头部文件：

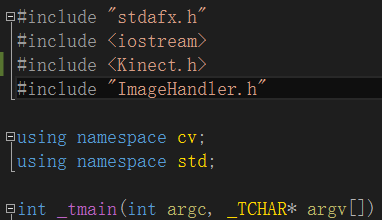


图 15

**2. 获取Kinect彩色图像**

从Kinect获取图像的一般过程为：

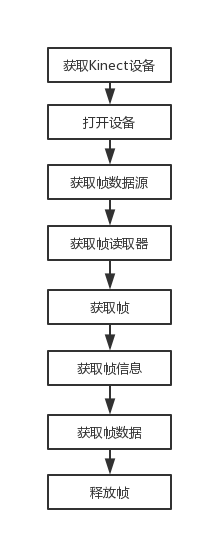


图 16

相关代码如下：

IKinectSensor\* pKinectSensor = NULL;

IColorFrameSource\* pColorFrameSource = NULL;

IColorFrameReader\* pColorFrameReader = NULL;

IFrameDescription\* pFrameDescription = NULL;

ColorImageFormat imgFormat = ColorImageFormat\_None;

int width, height;

uchar\* pBuffer = NULL;

UINT bufferSize = 0;

// 获取设备

HRESULT hr = GetDefaultKinectSensor(&pKinectSensor);

if (FAILED(hr))

{

cout << "获取Kinect设备失败" << endl;

goto exit;

}

// 打开设备

hr = pKinectSensor->Open();

if (FAILED(hr))

{

cout << "打开Kinect设备失败" << endl;

goto exit;

}

// 获取彩色帧数据源

hr = pKinectSensor->get\_ColorFrameSource(&pColorFrameSource);

if (FAILED(hr))

{

cout << "获取Kinect彩色数据源失败" << endl;

goto exit;

}

// 打开颜色帧读取器

hr = pColorFrameSource->OpenReader(&pColorFrameReader);

if (FAILED(hr))

{

cout << "打开Kinect颜色帧读取器失败" << endl;

goto exit;

}

while (1)

{

IColorFrame\* pColorFrame = NULL;

hr = pColorFrameReader->AcquireLatestFrame(&pColorFrame);

if (FAILED(hr)) continue;

// 获取帧信息

pColorFrame->get\_FrameDescription(&pFrameDescription);

pFrameDescription->get\_Width(&width);

pFrameDescription->get\_Height(&height);

// 获取图像格式

pColorFrame->get\_RawColorImageFormat(&imgFormat);

// 拷贝数据

Mat img(height, width, CV\_8UC4);

pBuffer = img.data;

bufferSize = img.rows \* img.step;

pColorFrame->CopyConvertedFrameDataToArray(bufferSize, reinterpret\_cast<BYTE\*>(pBuffer), ColorImageFormat\_Bgra);

// 释放帧

pColorFrame->Release();

imshow("Kinect彩色数据", img);

if (waitKey(30) == 27) // 按ESC退出

{

break;

}

}

**3. 颜色空间转换和颜色识别**

1）在入口程序\_tmain中添加ImageHnadler实例对象。如下所示：

ImageHandler imgHandler;

2）在while（1）中，定位到‘释放帧’代码。在该代码下添加如下处理函数调用：

// 缩小尺寸

resize(img, img, cv::Size(width / 4, height / 4));

// 颜色转换

cvtColor(img, img, CV\_BGRA2BGR);

// 图像颜色空间转换

imgHandler.ConvertColor(img);

// 颜色识别

imgHandler.RecogColor(img, ColorType::Green);

**4. 运行程序**

点击vs工具栏中的【本地Windows调试器】（如下图）运行该程序。



图 17

程序成功运行后。将看到颜色空间转换的结果：

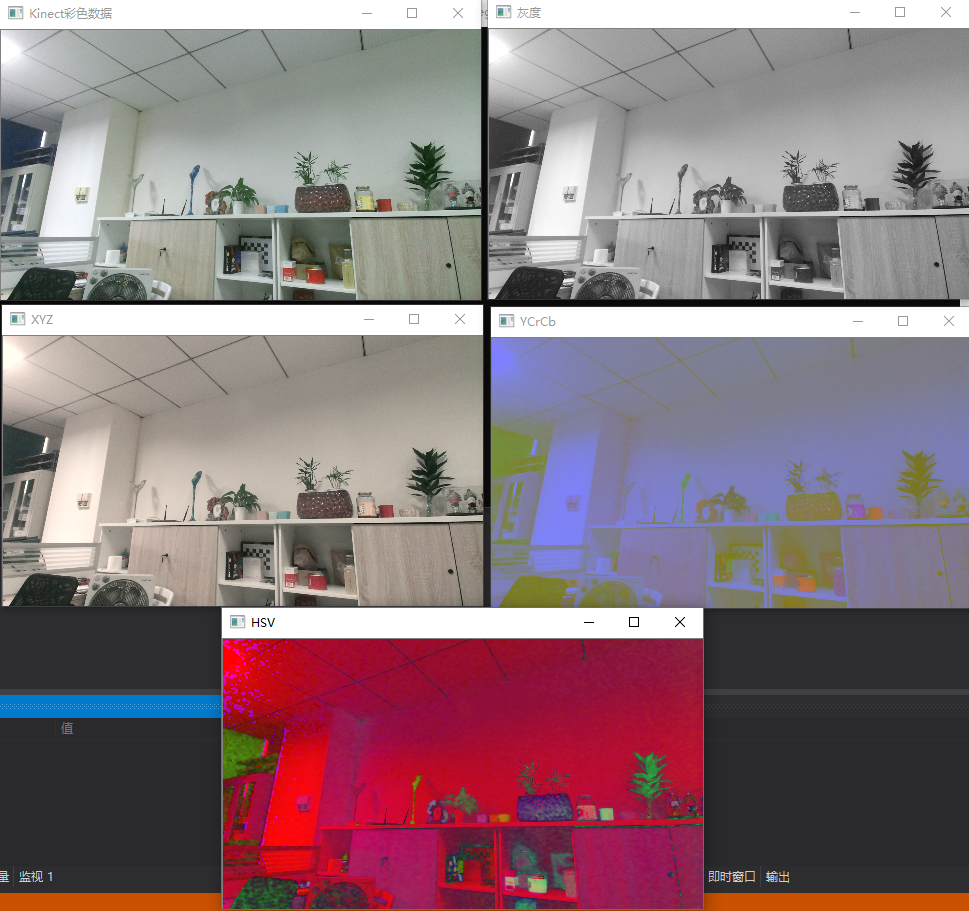


图 18

图中展示了RGB颜色空间到灰度空间、XZT空间，YCrCb空间和HSV空间转换的结果。

找一个绿色的物体（本实验识别的颜色），放置再Kinect摄像头前面，将看到颜色识别的处理结果：

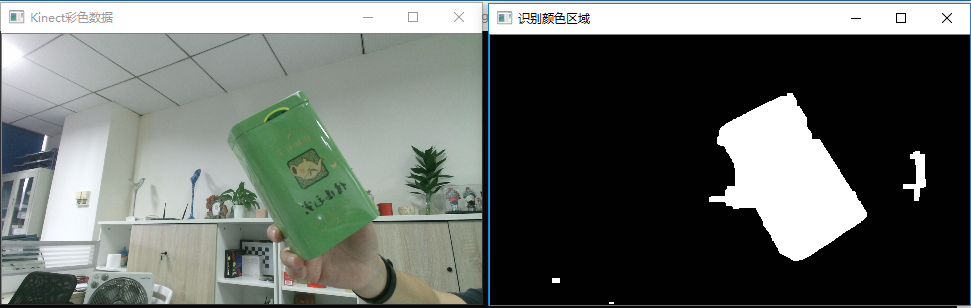


图 19

图中展示了对绿色物体识别的结果。

## 【思考题】

### 选择题

题目1：色彩的基本物理特性 (B)

A、间色 复色 补色

B、明度 纯度 色相

C、色调 固有色 环境色

D、色调 间色 复色

题目2：紫色在HSV空间中各个分量经验取值范围为：（C）

A、H：35-77，S：43-255，V：46-255

B、H：11-25，S：43-255，V：46-255

C、H：125-155，S：43-255，V：46-255

D、H：0-180，S：0-30，V：221-255

### 2、简答题

题目1：简述HSV颜色模型

HSV(Hue, Saturation, Value)是根据颜色的直观特性由A. R. Smith在1978年创建的一种颜色空间, 也称六角锥体模型(Hexcone Model)。这个模型中颜色的参数分别是：色调（H），饱和度（S），亮度（V）。

色调H：用角度度量，取值范围为0°～360°，从红色开始按逆时针方向计算，红色为0°，绿色为120°,蓝色为240°。它们的补色是：黄色为60°，青色为180°,品红为300°；

饱和度S：取值范围为0.0～1.0；

亮度V：取值范围为0.0(黑色)～1.0(白色)。