深 圳 大 学

本 科 毕 业 论 文（设计）

题目: 水果自助识别系统

姓名: 刘志鹏

专业: 自动化

学院: 机电与控制工程学院

学号: 2013110068

指导教师: 潘剑飞

职称： 教授

2017 年 04 月 11 日

深圳大学本科毕业论文（设计）诚信声明

本人郑重声明：所呈交的毕业论文（设计），题目《水果自助识别系统》是本人在指导教师的指导下，独立进行研究工作所取得的成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式注明。除此之外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。本人完全意识到本声明的法律结果。

毕业论文（设计）作者签名：

日期：　　年　　月　　日

摘要

近年来，随着人工智能领域的兴起，机器视觉等相关应用领域的技术越来越成熟。无论是当下火热的无人机技术、自动驾驶技术，还是现在已经在中国各大城市普及应用的车牌识别技术，机器视觉都在其中扮演着重要 角色。本次毕业设计是基于计算机视觉的物体分类——水果自助识别系统。这其中运用的具体知识为图像处理和图像识别。本文借助 OpenCV 开源计算机视觉库，在个人电脑上，进行水果分类的算法编写。最终，通过图像分割技术、特征提取技术以及支持向量机技术实现对于多种水果的识别。

关键字：水果识别；计算机视觉；机器学习；图像处理与识别

Abstract

In recent years, with the development of the artificial intelligence, there are many fields closely related to it have become increasingly matured especially computer vision. Moreover, computer vision has played an important role of lots of new technology including UAV, automatic driving and license plate detection which has been widely used in big cities in China. This graduation design is based on computer vision, completing object classification—a fruit recognition system. The main knowledge applied in it is image processing and recognition. Additionally, the recognition algorithm was completed with the aid of OpenCV, the open source computer vision library, in my personal computer. Finally, it can identify a variety of fruits via image segmentation, feature extraction and SVM..

Key words: Fruit recognition; Computer vision; Machine learning; Image processing and recognition

目 录

第一章 绪论

蒸汽机的改良，电能的应用，以及计算机的出现，都代表科技的创新与发展正在不断的提高着人类的工作效率和生活品质。从 20 世纪末开始，人工智能（Artificial intelligence）这一新兴名词被人们所知悉，由于其所涉及的领域广泛，人们当时主要是在理论上进行研究。但在近几年，人工智能的应用也逐渐兴起，在语音识别，视觉识别等方面都具有极高的应用价值。截止目前，车牌的视觉识别技术，文字的视觉识别技术，均已较为成熟，这些应用都是计算机视觉与人工智能相结合的结果。另外，在自动驾驶技术的研究中，已经有多家科技创新型企业取得突破性的进展，如中国的百度，美国的 Google、Tesla 等等。

1.1 课题研究背景与意义

计算机视觉作为人工智能的重要应用领域之一，随着机器学习的兴起而飞速发展。本文所研究的课题正是基于计算机视觉的水果识别技术。此技术可以用于各大型超市的水果打价中，通过水果识别技术可以让人们自助完成水果的打价，而不需要专门前往称重台排队进行人工打价。这样不仅可以节省消费者的时间提高消费者的购物体验，更可以为超市减少人工成本，不用再聘请专门的员工负责水果的打价。

1.2 国内外研究状况

1.2.1 国外研究状况

近几年，将有监督学习与计算机视觉进行结合的领域已经取得突破性的进展。在世界闻名的 ImageNet 大规模图像识别竞赛中，2012 年，图像识别成功率突破 80%，2013 年达到 88.8%，2014 年达到 93.3%。2015 年，在 1000 类的图像识别中，由微软设计的残差网（ResNet）识别算法更是达到了 96.43%的 Top5 正确率，首次超过 94.9%（人眼的识别成功率）。

在图像检测领域中，即是指将图像中的物体分类后并用矩形框给圈起来。从 14 到 16 年，无论是在检测精度上还是速度上都发展迅速，检测精度从 68.4% 到 75.1%，而检测速度从一幅图像需要处理 2 秒到一秒可以处理 23 帧图像。如图 1-1，为图像中不同物体的检测。

图 1-1 图像检测

另外，计算机视觉在自动驾驶领域也发挥着重要作用。在以美国为首世界各地，都已经开展了无人驾驶汽车的道路实测。而 Tesla 等汽车制造公司，更是已经在其部分新款汽车中安装了辅助驾驶功能。尽管目前的自动驾驶技术距离大范围推广还有一段距离，但其未来发展空间巨大。如图 1-2，为未来自动驾驶技术设想图。

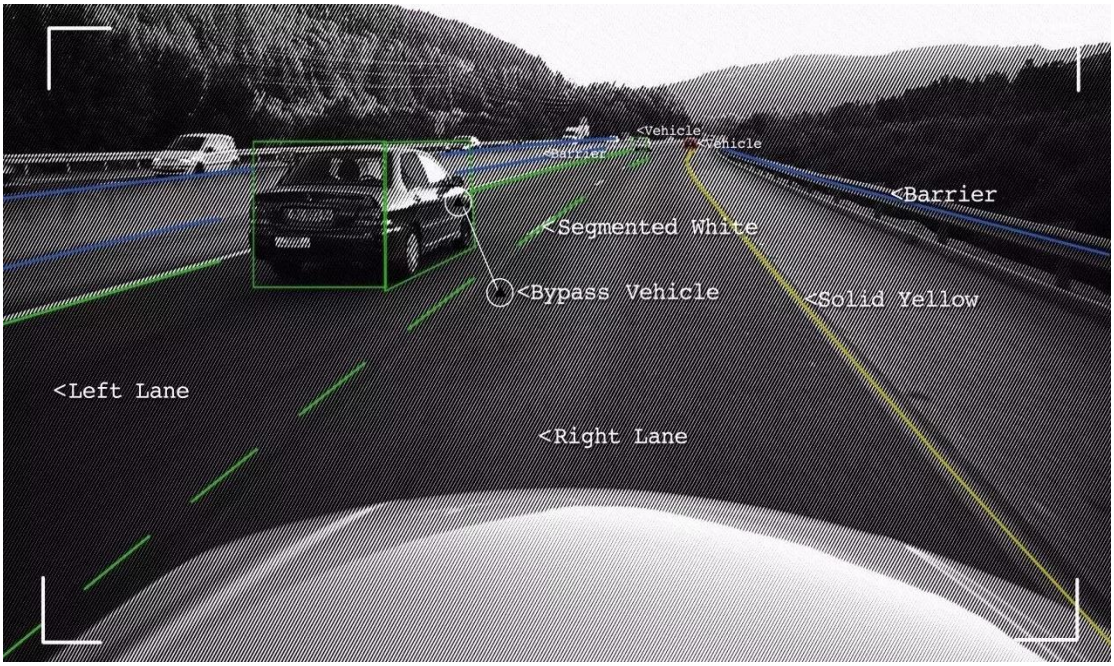

图 1-2 自动驾驶技术

1.2.2 国内研究状况

在国内，百度引领着计算机视觉领域的研究。2017 年 2 月 21 日，在麻省理工科技评论公布的十大突破技术中，百度入围了强化学习、自动驾驶货车、刷脸支付等类别，成为中国入围类别最多的公司，其评分更是超过了美国 IBM、苹果等科技创新型公司。

在百度开发出的实际应用产品中，人脸识别的准确率已经达到 99.7%，处于世界领先水平。在自动驾驶方面，百度的无人驾驶汽车也已经在城市道路、环路及高速道路混合路况下进行测试。

1.3 本文主要研究内容

本次毕业设计以水果作为对象，根据不同种类水果之间的颜色、形状差异来达到识别水果种类的目的。主要使用的软件工具为 OpenCV 、Visual Studio 和 Matlab；主要使用的硬件工具为摄像头支架、摄像头和个人电脑。研究内容主要有以下几个方面:（下图 1-3 为研究内容树状结构图）

1、使用图像分割技术去除背景，得到目标物体（水果）。

2、使用基本的图像处理技术对图像进行优化，便于特征提取和识别。

3、使用多物体特征提取技术同时得到所有目标水果的特征参数。

4、使用支持向量机技术，实现多个物体同时识别。

5、编写 MFC 人机交互界面。

6、使用 Matlab 进行仿真。

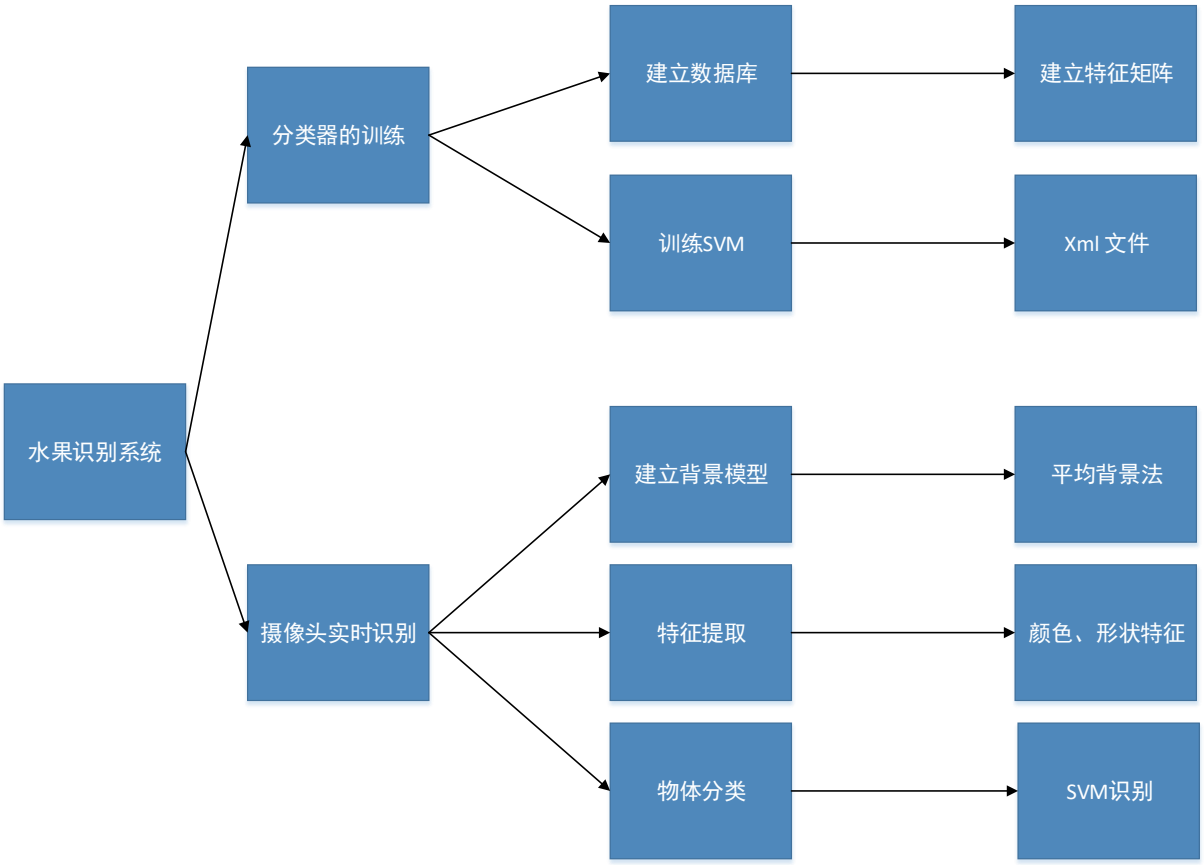

图 1-3 研究内容树状图

第二章　基本的图像处理与识别技术

2.1 概述

本章，将介绍水果识别系统中用到的基本图像处理与识别技术，它们包括图像格式的转换、图像滤波、形态学处理、Canny 算子、轮廓提取与绘制函数、特征提取函数、图像的点运算、支持向量机（SVM）等。

2.2 图像格式的转换

摄像头获取的图像默认为 RGB 三通道格式。RGB 是一种对彩色图像编码的方法，即“颜色模型”。几乎所有的颜色都可以通过 蓝（B）、绿（G）、红（R）三个颜色通道的变化和叠加来组成。在实际处理图像时，经常需要转换为单通道图像进行处理，比如转换为灰度图和二值化的图像。另外，对图像像素值的深度进行变化也是十分常用的方法，比如 8U 与 32F 图像之间的转换。这些图像格式的变化用于本文的背景累加、边缘检测、特征提取、图像显示等情况之中。

2.2.1 RGB2GRAY

此为将三通道的 RGB 图像转换为灰度图，灰度是亮度的量化值，使用图片像素遍历的方法。转化公式、实现代码以及实现效果图如下：

$$Gray = 0.29900 * R + 0.58700 * G + 0.11400 * B \qquad \text{公式（2-1）}$$

```
cvtColor(frame, frameGRAY, CV_BGR2GRAY);
```

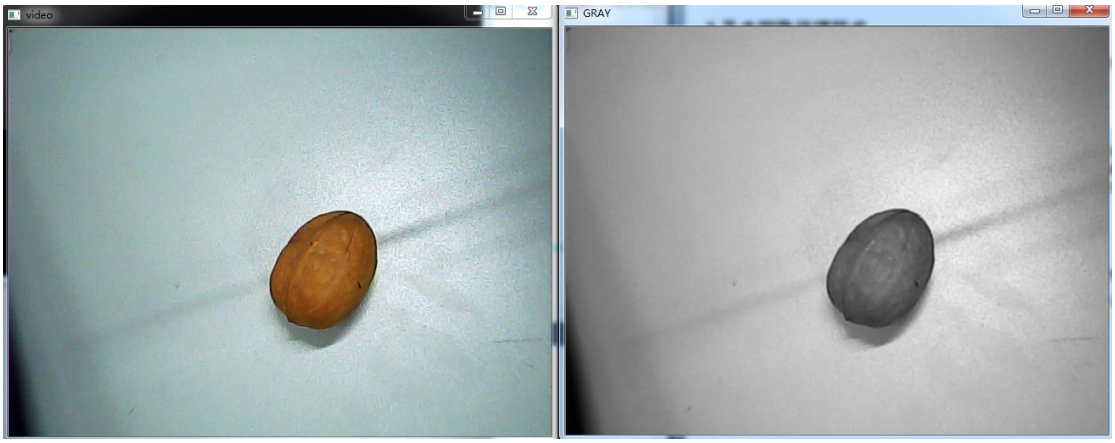

图 2-1 灰度化效果图

2.2.2 图像二值化

在本文中，图像的二值化处理应用在了水果数据库的建立中，使用阙值化操作提取图片中的目标物体水果，再进行特征提取存入数据库。

二值化操作是指，给定一幅图像和一个阙值，将图像中每个像素点的值与阙值比较，大于阙值的像素点赋值 255（白色）， 小于阙值的像素点赋值 0（黑色），最终产生一副黑白二值化图像。使用函数 Threshold() 实现。二值化效果图如下：

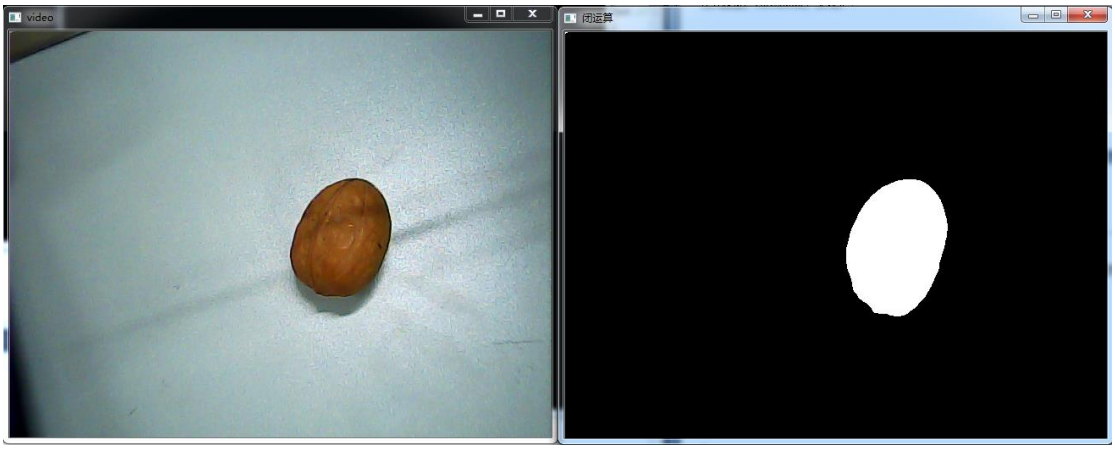

图 2-2 图像二值化效果图

2.2.3 图像像素值深度(depth)转换

通过适时的进行像素值深度的转换，不仅能提高程序运行的效率，而且可以提高图像识别的准确率。

本文中，摄像头采集的图像是 8 位的，就是像素值范围在 0~255。但是，在做背景累加时，在对像素值进行累加、乘除法等操作时，需要将 8 位图像转换为 32 位图像处理以提供更大数值范围。并且，在图像背景模型建立成功后，必须转换回 8 位图像用于 imshow() 显示。本文中，使用 convertTo() 函数进行像素值深度的转换。

2.3 图像滤波

图像的滤波也叫图像平滑处理。在本文中，摄像头读取的图像都必须进行滤波。图像滤波主要是为了消除图像的噪声，不同的噪声也有着不同的滤波方法。常见的滤波方法有方框滤波（BoxBlur() 函数）、均值滤波（Blur() 函数）、高斯滤波（medianBlur() 函数）、双边滤波（bilateralFilter() 函数）。

本文使用的是均值滤波，均值滤波是对图像中的每一个像素值重新赋值为核窗口内所有像素的平均值（所有像素加权系数相等）。实现代码以及效果图如下：

```
blur(frame, frame, Size(3, 3));
```

图 2-3 均值滤波效果图

2.4 形态学处理

形态学操作是基于形状的一系列图像处理操作，最基本的形态学操作为腐蚀（erode）和膨胀（dilate）。 其操作还包括：二值开运算、二值闭运算、骨架抽取等等。

在本文中，使用到了二值腐蚀、二值膨胀、二值开运算以及二值闭运算的形态学操作。

2.4.1 二值膨胀

膨胀（dilate）是求局部最大值的操作。从数学的角度来讲，膨胀是将图像与核进行卷积，也即计算核覆盖区域的像素点的最大值，并将最大值赋值给核内指定的像素，使图像中的高亮区域不断扩大。

在代码实现中，需要设定核的大小（element），核面积越大膨胀效果也就越明显。膨胀实现代码以及效果图如下：

```
Mat element1 = getStructuringElement(MORPH_ELLIPSE, Size(30, 30));
morphologyEx(frame, dst, MORPH_DILATE, element1);
```

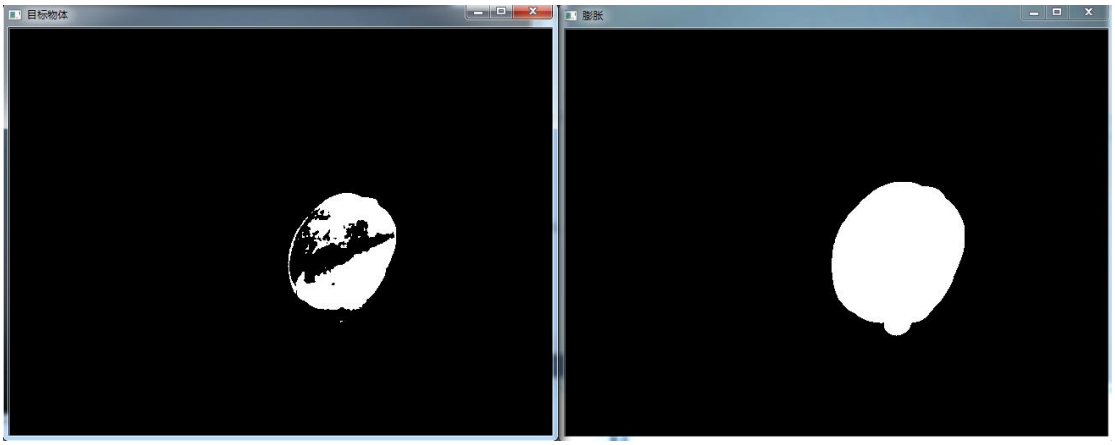

图 2-4 图像膨胀效果图

2.4.2 二值腐蚀

腐蚀（erode），顾名思义，是与膨胀效果相反的一对操作，腐蚀是求局部最小值。同样，腐蚀也是将图像与核进行卷积，计算核所覆盖的区域的最小值，并把这个最小值赋值给参考点指定的像素，使图像中小块（比核小）的高亮区域消除，达到去除杂质的目的。

另外，核的任意形状和大小都可以设定。通常情况下，核是一个小的，中间带有参考点的实心正方形或者圆盘。

由效果图可以看出，腐蚀（erode）可以很好的去除小块的杂志，避免后续的特征提取出现错误。但同时，腐蚀同样也使本就残缺不全的核桃（目标物体），更加不完整。实现代码以及效果图如下：

```
Mat element2 = getStructuringElement(MORPH_ELLIPSE, Size(10, 10));
morphologyEx(dst, dst2, MORPH_ERODE, element2);
```

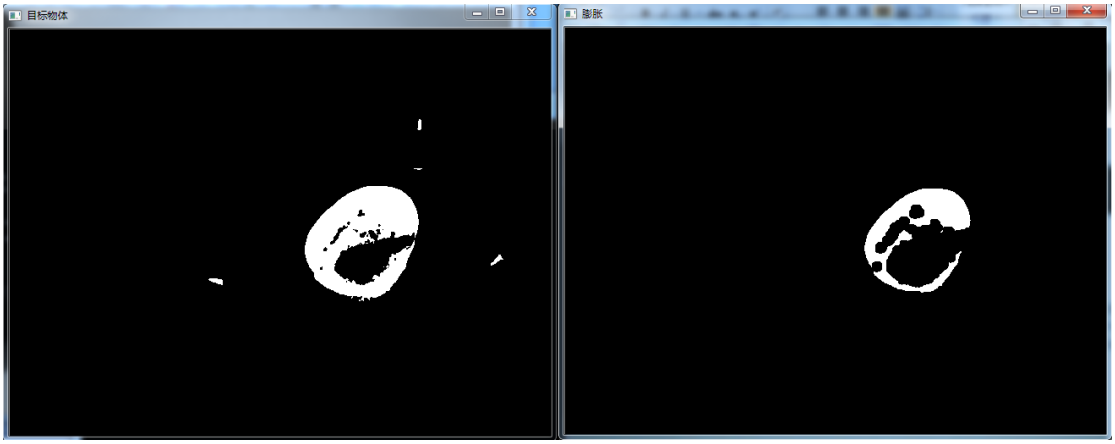

图 2-5 腐蚀（erode）效果图

2.4.3 二值开运算

在形态学处理中，腐蚀（erode）和膨胀（dilate）犹如两块基石，很多高级形态学操作，都是建立在它们的基础之上的。

开运算（Opening Operation），其实就是先腐蚀后膨胀的过程。它既拥有腐蚀的特点，消除小块杂质，又能利用膨胀的能力不让核桃（目标物体）的孔洞进一步扩大。最为重要的是，开运算不会像腐蚀膨胀一样明显改变核桃（目标物体）的面积，提高了特征提取参数的准确性。

由下图 2-6 可以看出，开运算在消除小块杂质的同时，基本保留了核桃（目标物体）的原貌。

代码实现如下，同样需要先定义内核，再使用 OpenCV 封装好的函数。

```
Mat element2 = getStructuringElement(MORPH_ELLIPSE, Size(10, 10));
morphologyEx(dst, dst2, MORPH_OPEN, element2);
```

图 2-6 开运算（Opening Operation）效果图

2.4.4 二值闭运算

二值闭运算（Closing Operation），与开运算相反，为先膨胀后腐蚀的过程。由于其先膨胀的特性，所以闭运算无法消除小块杂质，但是能够在填补核桃（目标物体）的内部孔洞的同时，不明显改变核桃（目标物体）的面积，提高了特征提取参数的准确性。

由效果图 2-7 可以发现，与单纯膨胀相比，闭运算在弥补了内部孔洞的同时，对面积参数的影响极低。闭运算实现代码以及效果图如下：

```
Mat element1 = getStructuringElement(MORPH_ELLIPSE, Size(30, 30));
morphologyEx(frame, dst, MORPH_CLOSE, element1);
```

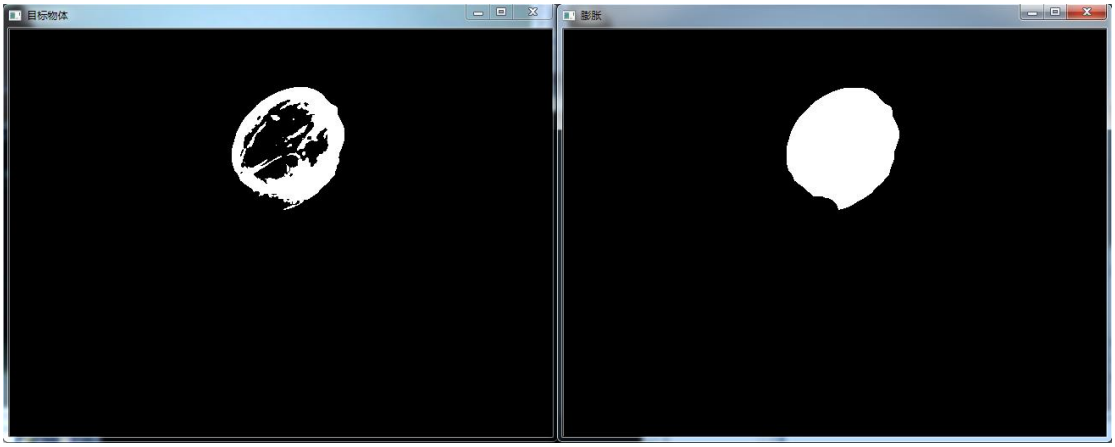

图 2-7 闭运算（Closing Operation）效果图

2.4.5 核心 API 函数介绍

API，全称 Application Programme Interface，作用是为程序员提供了可以操作系统的编程接口。在形态学处理中，OpenCV 将常用形态学处理功能封装在了 morphologyEx() 函数中，其函数原型如下:

```
C++: void morphologyEx( InputArray src, OutputArray dst,
                        int op, InputArray kernel,
                        Point anchor=Point(-1,-1), int iterations=1,
                        int borderType=BORDER_CONSTANT,
                        const Scalar&
                        borderValue=morphologyDefaultBorderValue() );
```

- 参数一：输入图像（Mat类对象即可）。
- 参数二：输出图像，必须和输入图像的尺寸和类型一致。
- 参数三：使用的形态学运算的类型，有 MORPH_OPEN（开运算）、MORPH_CLOSE（闭运算）、MORPH_ERODE（腐蚀）、MORPH_DILATE（膨胀）等等。
- 参数四：进行形态学运算的内核，若设置为 NULL，则使用参考点位于中心 3x3 内核，或者也可以使用 getStructuringElement() 函数创造内核。

2.5 Canny 算子

Canny 算子是 OpenCV 中边缘检测算子的一种，其余的还有 sobel 算子、Laplacian'算子、scharr 滤波器等等。

Canny 的目标是找到一个最优的边缘检测算法。利用 Canny 算法，实现对目标物体边缘的增强与提取。其工作原理如下：

第一步：消除噪声

边缘检测算法是基于图像强度的一阶和二阶导数的算法，其检测的原理就是通过图像中相邻像素点是否突变来判断此处是否为边缘。所以，这种方法对于图像中噪声极为敏感，非常容易将噪声所产生的突变像素点认为是边缘。因此，Canny 算子的第一步就是使用高斯平滑滤波器对图像进行卷积降噪。以下显示了一个 size = 5 的高斯内核示例：

$$K = \frac{1}{139}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \qquad \text{公式（2-2）}$$

第二步：计算梯度幅值和方向

计算梯度幅值实质上是增强边缘，通过确定图像各点邻域强度的变化值，将图像中邻域强度值有显著变化的像素点凸显出来。此外，计算梯度幅值具有方向性，所以需要分别从 X 和 Y 方向进行计算，最终效果为两个方向的效果合成。

运用一对卷积阵列（分别作用于 x 和 y 方向）：

$$G(x)=\begin{bmatrix}-1 & 0 & +1\\ -2 & 0 & +2\\ -1 & 0 & +1\end{bmatrix}\quad G(y)=\begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ +1 & +2 & +1\end{bmatrix} \qquad \text{公式（2-3）}$$

使用下列公式计算梯度幅值和方向：

$$G=\sqrt{G_x^2+G_y^2} \qquad \text{公式（2-4）}$$

$$\theta=\arctan\left(\frac{G_y}{G_x}\right) \qquad \text{公式（2-5）}$$

第三步：非极大值抑制

这一步排除非边缘像素，仅仅保留细线条状的边缘。

第四步：阈值检测

经过增强的图像，往往有许多像素点处的梯度值较大，但这些并不都一定是所需要的边缘。所以，需要采用阈值化的方法对这些边缘像素点进行取舍。此处结合 Canny() 函数的原型进行理解：

```
C++: void Canny( InputArray image, OutputArray edges,
                 double threshold1, double threshold2,
                 int apertureSize=3, bool L2gradient=false );
```

- 参数一：输入图像（Mat 类对象），且必须为单通道八位图像。
- 参数二：输出边缘图像，必须和输入图像的尺寸、类型一致。
- 参数三：第一个滞后性阈值（double 类型）。
- 参数四：第二个滞后性阈值（double 类型）。

在函数原型中，最重要的是对于参数三、四两个滞后阈值的设置，这将决定哪些像素点是边缘以及所获得的边缘的完整性。在这两个滞后阈值中，较高的阈值决定边缘的起点。图像中像素的梯度幅值大于高阈值的均作为边缘的起点像素点，然后以起点为核心对周围像素的梯度幅值进行检测，若周围像素的梯度幅值大于低阈值的判定为边缘像素点，否则该像素点舍弃。

另外，高低阈值一般保持在 2:1 ~ 3:1 之间。

2.6 轮廓提取与绘制函数

2.6.1 findContours() 轮廓提取函数

findContours()函数具有在二值图像中提取轮廓的功能。另外，还具有提取轮廓之间拓扑关系等实用性功能。函数原型如下：

C++： void findContours(InputOutputArray image, OutputArrayOfArrays contours, OutputArray hierarchy, int mode, int method, Point offset=Point());

- 参数一：输入图像，为8通道 Mat 类对象。
- 参数二：轮廓存储容器，每个轮廓存储为一个点向量。
- 参数三：存储图像轮廓的拓扑信息。每个轮廓对应有四个hierarchy元素，分别代表后一个轮廓、前一个轮廓、父轮廓和内嵌轮廓的索引编号。
- 参数四：轮廓检索模式。
- 参数五：轮廓近似办法。

2.6.2 drawContours() 轮廓提取函数

drawContours()函数具有在图像中绘制外部或内部轮廓的功能。另外，通过设置可以选择绘制轮廓的粗细以及是否被填充。函数原型如下：

C++：void drawContours(InputOutputArray image, InputArrayOfArrays contours, int contourIdx, const Scalar& color, int thickness=1, int lineType=8, InputArray hierarchy=noArray(), int maxLevel=INT_MAX, Point offset=Point());

- 参数一：目标图像（Mat类对象）。
- 参数二：保存轮廓的点向量集合。
- 参数三：制定需要画出的轮廓编号。
- 参数四：轮廓的颜色。
- 参数五：线的粗细程度，若为负数则填充整个轮廓。
- 参数六：连通性。

2.7 特征提取函数

2.7.1 contourArea()函数

contourArea()函数具有计算整个轮廓或者部分轮廓面积的功能。函数原型如下：

C++：double contourArea(InputArray contour, bool oriented=false);

- 参数一：输入需要计算面积的轮廓，可以为点向量或者 Mat 类型
- 参数二：面向区域的标示符，若为 true，则求得的面积参数带有正负号，符号取决于轮廓的方向（顺时针 or 逆时针）。

2.7.2 arcLength()函数

arcLength()函数具有计算封闭轮廓的周长或者曲线的长度的功能，函数原型如下：

C++：double arcLength(InputArray curve, bool closed);

- 参数一：输入需要计算周长的轮廓，可以为点向量或者 Mat 类型
- 参数二：一个用于指示曲线是否封闭的标示符

2.8 图像的点运算

2.8.1 基本原理

图像的点运算，顾名思义，就是对图像中每个像素点进行操作。这是一种最原始的图像处理方法，也是功能最强大的方法。一切的图像处理技术，最终都是源于对于像素点的操作，只是各自操作的方法不一。本文中，也多次使用到了图像的点运算，也叫图像遍历法。

在彩色模型中，有 RGB 模型、HSI 模型等。不同的颜色模型，各个通道之间的意义不一样，但通道数均为三个。所以要想实现对于每个像素点的操作，不仅要对每一行，每一列进行操作，还要对每一通道进行操作。所以图像遍历法的基础是三个循环嵌套。基本代码实现如下：

```
for (int x = 0; x < draw.rows; x++)//遍历所有行数
{
    for (int y = 0; y < draw.cols; y++)//遍历所有列数
    {
        for (int c = 0; c < 3; c++)//遍历所有通道数
        {
            draw.at<Vec3b>(x, y)[c] = 250;//像素点操作
        }
    }
}
```

2.8.2 像素点的类型

创建 Mat 图像时，不同的类型，其像素点的值具有不同的范围。在使用像素点运算时，at() 函数的调用也不同。本文中使用的像素点类型有 CV_8UC1、 CV_8UC3 和 CV_32F。在八通道图像中，像素点的范围为 0~255，所以在进行图像点运算时，必须改为 32F，这样不仅可以在乘除运算时通过保留更多小数点提高精度，还可以为加减运算提供更大的数值范围。at() 函数的调用规则为：CV_8UC1 → <uchar> 、 CV_8UC3 → <Vec3b> 、 CV_32FC1 → <float>。

2.9 支持向量机（SVM）

2.9.1 SVM 的基本原理

SVM 的核心思想是利用物体特征的参数，绘制一个能将尽可能多得不同物体区分开的超平面作为分类器的分类依据。超平面即是指一个能距离每个物体的特征参数都尽可能远的多维平面。下图 2-8 是基于两个特征（x1、x2）的二分类 SVM 示意图，中间那条线即为最优分类平面，距离两类物体（圆圈、方框）都最远。

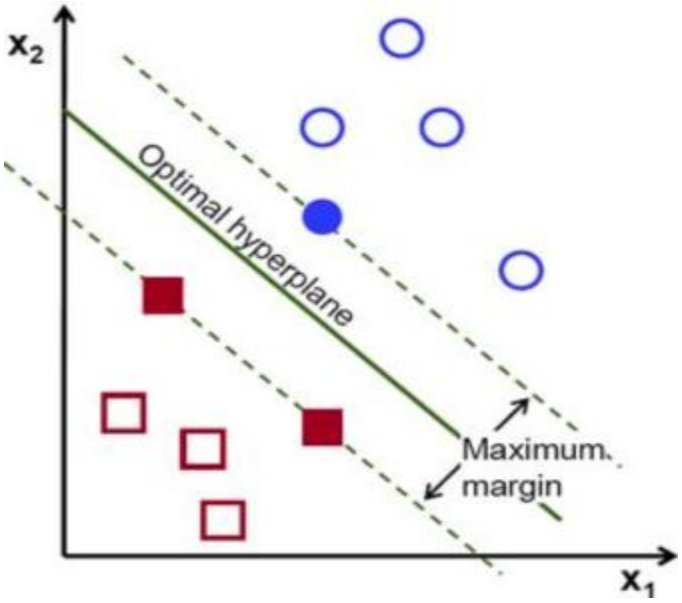

图 2-8 二分类器 SVM

2.9.2 SVMParams()函数

在 OpenCV 中，有专门的 SVM 类用于支持向量机的创建。另外，它还具有成员函数 SVMParams 进行初始化参数设置。函数原型如下

C++：CvSVMParams(int svm_type, int kernel_type,

double degree, double gamma, double coef0,

double Cvalue, double nu, double p,

CvMat* class_weights, CvTermCriteria term_crit);

- 参数一：指定 SVM 的类型，一共有五种 SVM 的类型，本文使用的是 C_SVM。
- 参数二：SVM 的内核类型，一共有四种内核，分别为线性内核、多项式内核、径向基函数内核、以及 Sigmoid 函数内核。
- 参数四：内核函数的参数 gamma。
- 参数六：内核函数的参数 C。
- 参数十：SVM 的迭代训练过程的中止条件，也可以设置的最大迭代次数。

2.9.3 train()函数

成员函数 train() 用于分类器的训练，函数原型如下：

```
C++: train( const cv::Mat& trainData, const cv::Mat& responses,
        const cv::Mat& varIdx=cv::Mat(),
        const cv::Mat& sampleIdx=cv::Mat(),
        CvSVMParams params=CvSVMParams() );
```

- 参数一：待训练的特征向量矩阵。
- 参数二：标签矩阵。
- 参数五：SVM 的参数设置

第三章 水果识别技术

3.1 概述

本章，将介绍水果识别系统具体的实现方法。并且，介绍方法的概念、实现方式、实现代码以及实现效果图。主要涉及的方面有，图像分割技术、多物体特征提取技术和支持向量机识别技术。

3.2 图像分割技术 —— 平均背景法

在图像分割领域中，针对不同的具体情况有不同的方法。主要分为两大方法，分别是静态背景的图像分割和动态物体的检测。前者是指在背景不变的情况下，将新出现的物体与背景分离，后者是指背景模型在不断变化且目标物体也处于运动状态中，后者的分割算法有三帧差法、混合高斯方法等。本文研究的是水果种类的识别，使用的是基于静态背景模型的平均背景图像分割法。

3.2.1 基本概念

平均背景法是一种建立背景模型的方法。通俗的说，就是将摄像头获取的一定数量的帧进行累加再求出平均值后，再将这幅平均图像加上一定的阙值范围后的模型，作为背景模型。

在此后新读入的图像，与平均图像进行比较（做差），差值大于阙值范围的，则认为是前景物体。

3.2.2 实现步骤

1、摄像头读取实时图像，每一帧作为一幅图像，循环读取

```
VideoCapture capture;
capture.open(0);
capture >> frame;
```

2、制作背景模型，通过遍历图像所有像素点的方法将前 30 帧图像累加并求平均值，实现代码如下：

```
if (count < 30)//前30帧，用于建立背景模型
{
    Mat frameGRAY;
    capture >> frame;
    blur(frame, frame, Size(3, 3));
    cvtColor(frame, frameGRAY, CV_BGR2GRAY);
    frameGRAY.convertTo(frameGRAY, CV_32FC1);

    for (int i = 0; i < frameGRAY.rows; i++)//通过遍历图像所有像素点进行图像累加
    {
        for (int j = 0; j < frameGRAY.cols; j++)
        {
            //像素点累加
            imgAll.at<float>(i, j) = imgAll.at<float>(i, j) + frameGRAY.at<float>(i, j);
        }
    }
    waitKey(30);
}
if(count == 30)//等于30帧时，求累加后图像的平均值作为背景模型
{
    cout << imgAll.at<float>(50, 50) << " " << count << endl;
    for (int i = 0; i < frame.rows; i++)//遍历图像所有像素点
    {
        for (int j = 0; j < frame.cols; j++)
        {
            imgAll.at<float>(i, j) = imgAll.at<float>(i, j) / count;//求平均值
        }
    }
    cout << imgAll.at<float>(50, 50) << " " << count << endl;
    imgAll.convertTo(imgAll2, CV_8UC1);
    imshow("背景模型", imgAll2);//显示背景模型
    waitKey(0);
}
```

3、目标物体的分割。30 帧过后，通过新图像与平均图像做差，若超出设定阈值，则认定为目标物体。（此处阈值设定为 90，阈值的设定根据不同环境，如在不同光照条件下，需要进行适当调整）实现代码如下：

```
if(count>30)//超过30帧，背景模型建立结束，寻找目标物体
{
    Mat diff, frameGRAY, canny, contours, dst;
    capture >> frame;
    blur(frame, frame, Size(3, 3));
    cvtColor(frame, frameGRAY, CV_BGR2GRAY);
    frameGRAY.convertTo(diff, CV_32FC1);
    absdiff(imgAll, diff, diff);

    //通过图像像素遍历法做差，若超出设定阈值，则认定为目标物体
    for (int i = 0; i < frame.rows; i++)
    {
        for (int j = 0; j < frame.cols; j++)
        {
            if (diff.at<float>(i, j) < 90)//阈值此时设定为90
                diff.at<float>(i, j) = 0.0;//背景设置为黑色
            else
                diff.at<float>(i, j) = 255.0;//目标物体设置为白色
        }
    }

    diff.convertTo(frame, CV_8UC1);
    imshow("目标物体", frame);//显示目标物体
```

3.2.3 实现效果截图

1、摄像头实时图像（图 3-1）

图 3-1 摄像头实时图像

2、背景模型（图 3-2）

图 3-2 背景模型

3、提取到的目标物体（核桃），白色为目标，黑色为背景（图 3-3）

图 3-3 目标物体（核桃）

3.3 多物体特征提取技术

通过平均背景法，得到二值化的目标物体后，开始对目标物体进行特征提取。本文将目标物体的颜色、形状特征作为分类依据。

3.3.1 形态学处理

特征提取的前提是获得完整的目标轮廓且不含杂质，从上图 3-3 容易看出，杂质和目标物体内部的孔洞都是在图像分割时，无法避免的问题。所以，特征提起前必须进行形态学处理。

另外，为了不影响形态学处理对于形状特征参数的影响，本文采用了先开运算，再闭运算的处理办法。达到消除杂质，填补物体内部孔洞的目的。需要注意的是，开运算的内核面积必须比闭运算的内核面积要小。实现代码以及效果图如下：

```
Mat element1 = getStructuringElement(MORPH_ELLIPSE, Size(10, 10));
morphologyEx(frame, dst, MORPH_OPEN, element1);

Mat element2 = getStructuringElement(MORPH_ELLIPSE, Size(30, 30));
morphologyEx(dst, dst2, MORPH_CLOSE, element2);
```

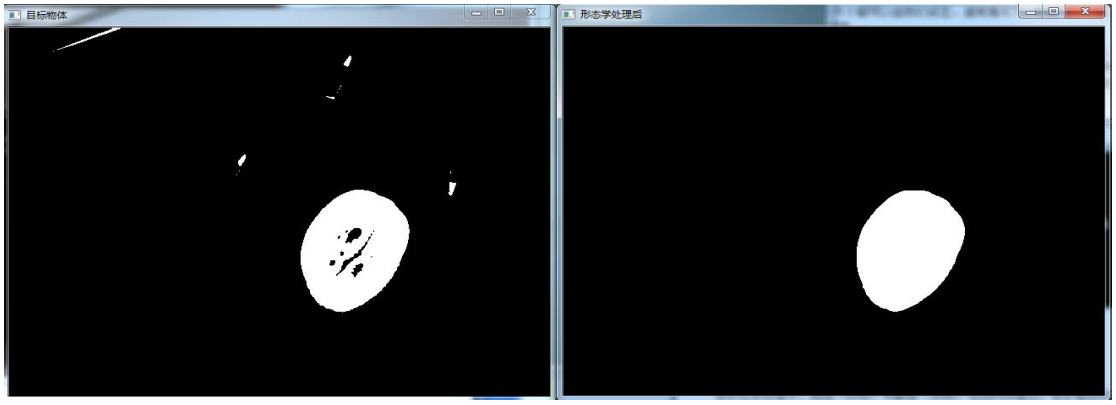

图 3-4 开闭运算综合效果图

3.3.2 提取形状特征参数

本文中，总共使用三种参数描述物体的形状特征，它们分别是目标物体的面积、周长和弧度。首先使用 Canny 算子进行边缘增强以及边缘检测，再使用 findContours() 和 drawContours() 函数进行轮廓的保存和标记，最后使用 contoursArea() 和 arcLength() 函数提取特征。

1、Canny 算子

使用 Canny 算子对目标物体进行边缘增强，此处选用的高低阈值分别为 60、30。实现代码以及效果图如下：

```
Canny(dst2, canny, 30, 60);
```

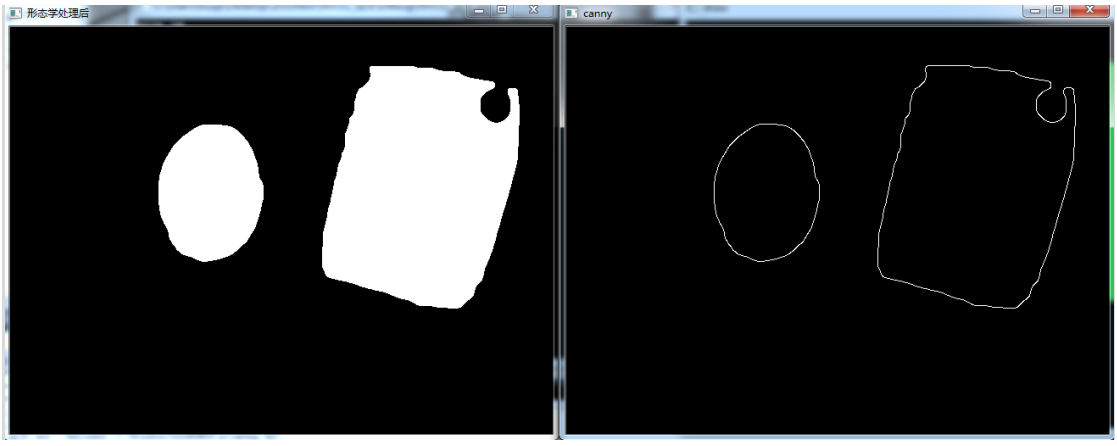

图 3-5 Canny 算子效果图

2、目标物体轮廓的提取

经过 Canny 算子后，增强了目标物体的轮廓。下一步是把轮廓分别提取出来，再进行形状的特征提取，为最终实现多目标物体识别提供可能。

使用 fingContours() 函数提取轮廓，轮廓检索模式选用的是 CV_RETR_EXTERNAL，即仅仅检索外层轮廓，因为外部轮廓即可得到水果的面积、周

长形状特征信息。轮廓的近似方法选用的是 CV_CHAIN_APPROX_NONE，即获取每个轮廓的像素点，这样可使提取的特征参数更加精确。实现代码如下：

```
vector<vector<Point>> contours;
vector<Vec4i> hierarchy;
findContours(canny, contours, hierarchy, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_NONE);
```

3、计算形状特征参数

通过 findContours() 函数完成对多个轮廓的提取后，对每个物体分别计算面积、周长以及弧度的形状特征参数。

此处，使用 contoursArea() 函数和 arcLength() 函数分别计算目标物体（水果）的面积和周长。而弧度参数则是由面积和周长参数计算得来。弧度计算公式如下：

$$\text{弧度} = 4\pi * \text{面积} / \text{周长}^2 \qquad \text{公式（3-1）}$$

代码实现以及效果图如下：

```
Vec<double, 5> Area;//形状特征参数初始化
Vec<double, 5> Perimeter;
Vec<double, 5> Radian;
```

```
Area[index] = contourArea(contours[index]);//求取形状特征参数
Perimeter[index] = arcLength(contours[index], true);
Radian[index] = 4 * 3.14*Area[index] / (Perimeter[index] * Perimeter[index]);
```

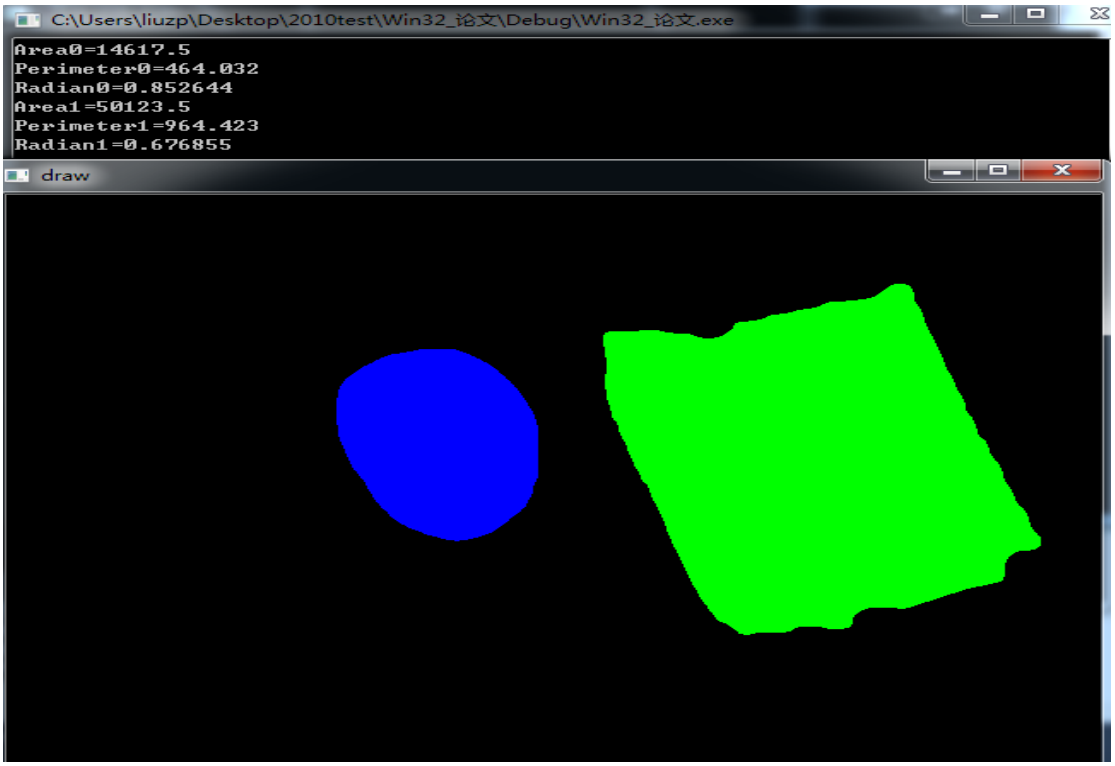

图 3-6 得到面积、周长、弧度参数效果图

3.3.3 提取颜色特征参数

1、方法综述

在 OpenCV 中，定义有多个彩色模型，即用来精确标定和生成各种颜色的规则。比较常用的彩色图像的编码方式有摄像机使用的 BGR（蓝、绿、红）模型，有用于彩色打印机的 CMY（青、深红、黄）和 CMYK（青、深红、黄、黑）模型，还有 HSV（色调、饱和度、亮度）这种最为符合人们主观对颜色的判断和描述方式的模型。这几种彩色模型均可以实现相互转换。

本文中，使用的是基于摄像头使用的 BGR（红、绿、蓝）彩色模型。

在 BGR 模型中，一共有蓝、绿、红三个通道，每个通道都可以独立成一幅图像分量，三幅图像分量的合成，也就是一幅彩色图像。每幅图像分量，都有着相同的行数、列数以及像素个数，而每个像素点的像素值即代表着该图像分量的亮度。在 8 位 3 通道（8UC3）的图像中，每个像素点的像素值范围是 0~255。

本文中，目标物体（水果）的颜色特征一共由 3 个特征参数表示，它们分别为 B、G、R。通过使用目标物体（水果）的三个通道分量之间的比例大小关系来区分不同水果种类的颜色信息。

2、实现方法

第一步，对颜色参数的初始化，实现代码如下：

```
double B0 = 0.0, G0 = 0.0, R0 = 0.0, Sum0 = 0.0;
double B1 = 0.0, G1 = 0.0, R1 = 0.0, Sum1 = 0.0;
double B2 = 0.0, G2 = 0.0, R2 = 0.0, Sum2 = 0.0;
```

第二步，使用 drawContours() 函数对不同物体进行颜色标记，以获取不同物体的位置信息。并且将参数五设置为填充整个轮廓。实现代码如下：

```
Mat draw = Mat::zeros(dst2.rows, dst2.cols, CV_8UC3);
for( int index = 0;index<contours.size();index++ )
{

    if(index==0)//目标物体1
    {
    Scalar color( 255,0,0 );//纯蓝色标记
    drawContours( draw, contours, index, color, CV_FILLED, 8 );
    }
    if(index==1)//目标物体2
    {
    Scalar color( 0,255,0 );//纯绿色标记
    drawContours( draw, contours, index, color, CV_FILLED, 8 );
    }
    if(index==2)//目标物体3
    {
    Scalar color( 0,0,255 );//纯红色标记
    drawContours( draw, contours, index, color, CV_FILLED, 8 );
    }
```

第三步，使用像素点遍历法，读取目标物体（水果）位置信息，再提取原摄像头图像对应位置下的颜色参数。实现代码如下：

```
for (int x = 0; x<draw.rows; x++)//遍历所有行数
{
    for (int y = 0; y<draw.cols; y++)//遍历所有列数
    {
        for (int c = 0; c<3; c++)//遍历所有通道数
        {
            if (draw.at<Vec3b>(x, y)[c]>250)//判定是否为目标物体位置
            {
                if (c == 0)//目标物体1
                {
                    //提取原摄像头图像对应位置下的颜色参数
                    B0 = B0 + frame_video.at<Vec3b>(x, y)[0];
                    G0 = G0 + frame_video.at<Vec3b>(x, y)[1];
                    R0 = R0 + frame_video.at<Vec3b>(x, y)[2];
                }
                if (c == 1)//目标物体2
                {
                    B1 = B1 + frame_video.at<Vec3b>(x, y)[0];
                    G1 = G1 + frame_video.at<Vec3b>(x, y)[1];
                    R1 = R1 + frame_video.at<Vec3b>(x, y)[2];
                }
                if (c == 2)//目标物体3
                {
                    B2 = B2 + frame_video.at<Vec3b>(x, y)[0];
                    G2 = G2 + frame_video.at<Vec3b>(x, y)[1];
                    R2 = R2 + frame_video.at<Vec3b>(x, y)[2];
                }
            }
        }
    }
}
```

第四步，计算 B、G、R 占总像素值的比例来作为最终的颜色信息参数。此外，通过比例的方式能够极大的减小光照变化对于目标物体的颜色信息参数的影响。经过试验发现，若直接使用 R、G、B 的值来作为特征参数，在不同的光照条件下，分类器几乎无法正确识别水果。实现代码如下：

```
Sum0 = B0 + G0 + R0;//物体1总像素值
if (Sum0 >10000)
{
    B0 = B0 / Sum0;//物体1最终颜色信息参数
    G0 = G0 / Sum0;
    R0 = R0 / Sum0;
    cout << "B1=" << B0 << " G0=" << G0 << " R0=" << R0 << endl;
}

Sum1 = B1 + G1 + R1;//物体2总像素值
if (Sum1 >10000)
{
    B1 = B1 / Sum1;//物体2最终颜色信息参数
    G1 = G1 / Sum1;
    R1 = R1 / Sum1;
    cout << "B1=" << B1 << " G1=" << G1 << " R1=" << R1 << endl;
}

Sum2 = B2 + G2 + R2;//物体3总像素值
if (Sum2 >10000)
{
    B2 = B2 / Sum2;//物体3最终颜色信息参数
    G2 = G2 / Sum2;
    R2 = R2 / Sum2;
    cout << "B2=" << B2 << " G2=" << G2 << " R2=" << R2 << endl;
}
```

3、实现效果图如下，成功获得颜色、形状特征参数

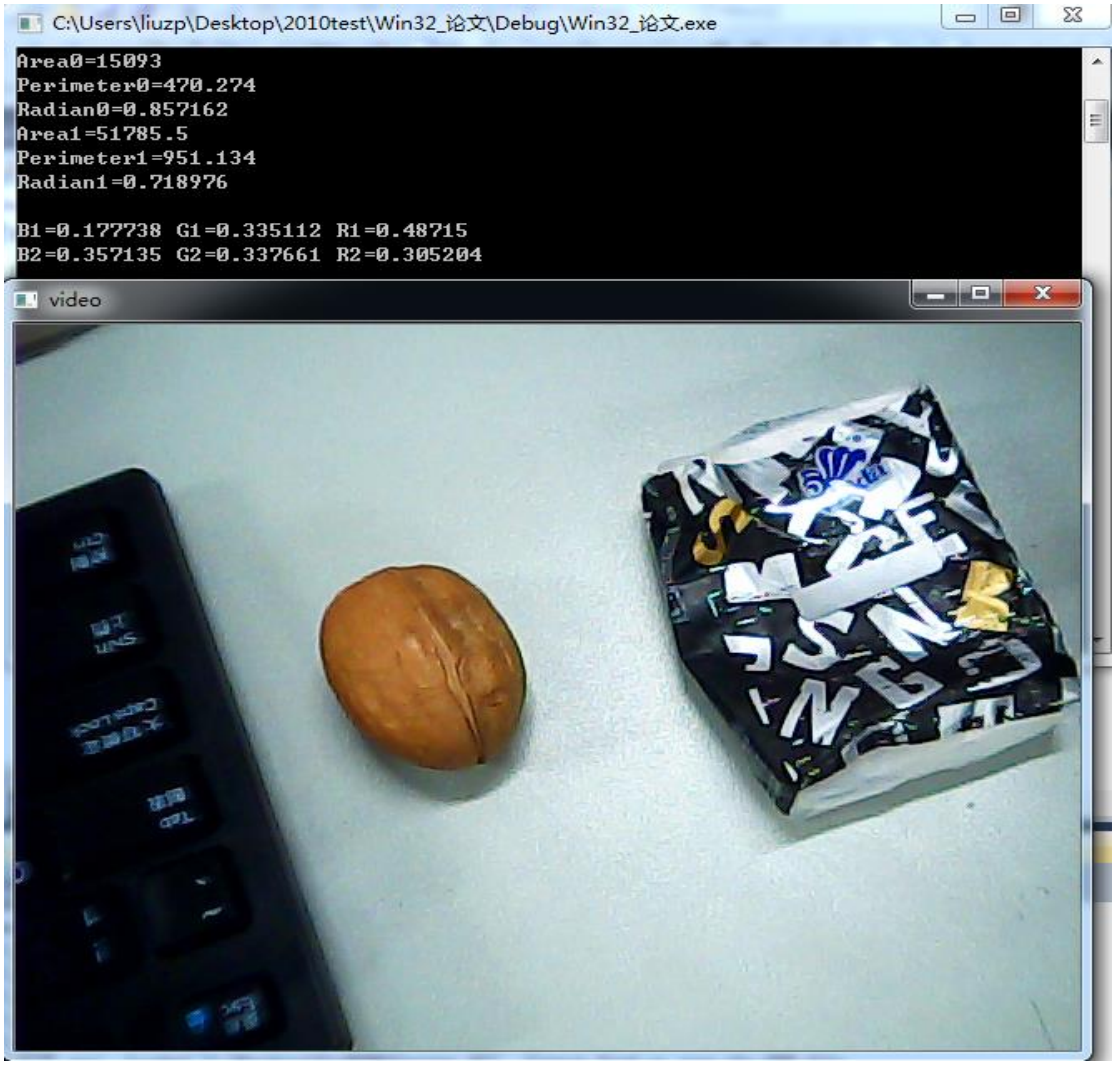

图 3-7 获得颜色、形状参数效果图

3.4 多物体识别技术 —— 支持向量机（SVM）

使用图像分割技术和特征提取技术得到实时水果特征后，需要对特征进行有效的识别，进而确定水果的种类。本文使用的识别方法是支持向量机的技术。本文中，使用的是监督学习的方式让支持向量机识别水果的种类。首先，建立需要进行识别的水果的特征数据库，对识别算法进行训练，再使用测试样本对算法进行验证。

3.4.1 建立数据库

建立数据库即是建立水果的特征矩阵，将训练图像的特征值保存在同一个矩阵（Mat类对象）中作为待训练的特征向量。另外，还需要对特征矩阵进行数据归一化，保证各个特征在用于识别时具有相同的权值。

1、提取一幅训练图像的特征

第一步，通过使用摄像头拍照的方式，储备水果的训练样本。在操作中，为了便于训练样本中目标物体（水果）特征的提取，统一在待拍摄水果下放置一张白纸。再对于每张训练图像进行形状、颜色特征的提取。

首先，使用 Canny() 函数检测目标物体（水果）的轮廓，利用水果与白纸具有较大色差的特点，直接达到图像分割的效果。经过多次试验，这里的 Canny() 函数使用 75、150 作为高低阈值时，可以获得杂质较少且较为完整的目标轮廓。实现代码以及效果图如下：

```
Canny(img, canny, 75, 150);
```

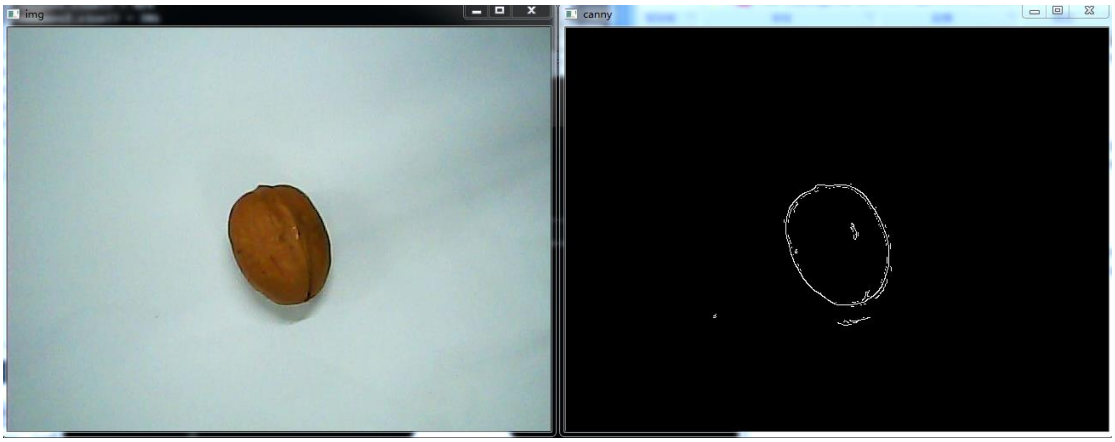

图 3-8 左为一幅待训练图像，右为直接使用 Canny()函数的效果

第二步，由于提取特征的前提是拥有完整的轮廓。而使用 Canny() 函数进行的图像分割，不可避免的在部分轮廓处出现断裂。所以，需要使用形态学方法进行初步的处理。对图像进行膨胀处理，以保证获得完整目标图像的轮廓。实现代码以及效果图如下：

```
Mat element1 = getStructuringElement(MORPH_ELLIPSE, Size(10, 10));
morphologyEx(canny, dst, MORPH_DILATE, element1);
```

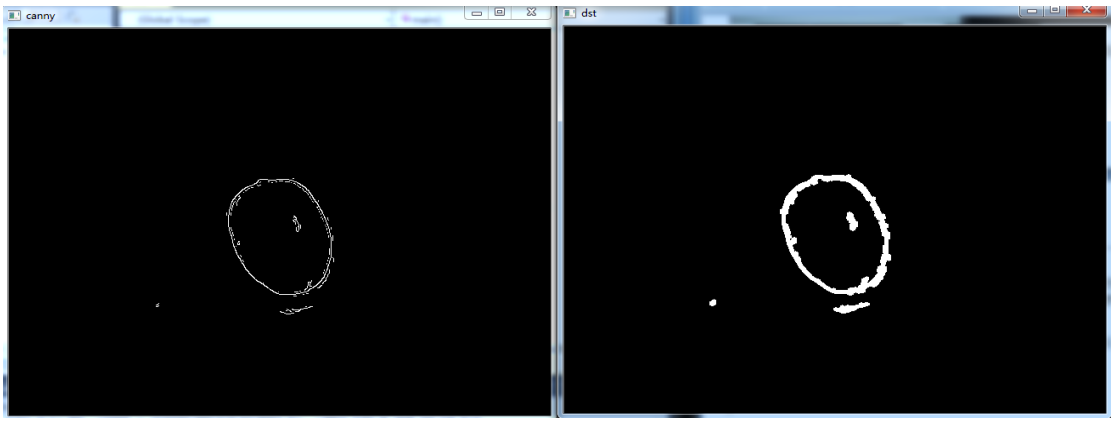

图 3-9 膨胀效果图

第三步，从上面的效果图可以发现，在获得了完整的目标物体轮廓后，杂质同样由于膨胀的关系变的更为明显。而且，此处不能用腐蚀操作去除杂质，否则可能损坏目标物体轮廓的完整性，导致重新出现断点。基于这种情况，本文利用了 findcontours() 和 drawcontours() 函数的特性进行杂质去除。其基本原理是，在使用 findcontours() 函数获取了图像中所有轮廓后，使用 drawcontours() 函数仅仅画出较大的轮廓，而那些杂质也就作为较小的轮廓被有效去除。实现代码以及效果图如下:

```
findContours(dst, contours, hierarchy, RETR_CCOMP, CHAIN_APPROX_SIMPLE);

int n = 0;
Mat draw = Mat::zeros(img.rows, img.cols, CV_8UC1);
for (int index = 0; index<contours.size(); index++)
{
    if (contours[index].size()>200)//通过设置条件，仅画出大轮廓，达到去除杂质目的
    {
        Scalar color(255);//纯白色标记轮廓
        drawContours(draw, contours, index, color, FILLED, 8, hierarchy);
        n++;
    }
}
```

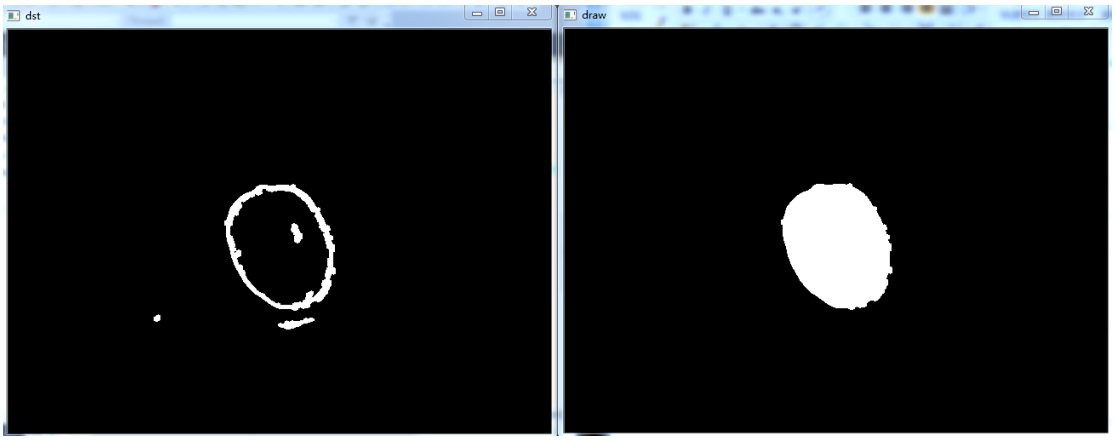

图 3-10 去除杂质后的效果图

第四步，由于之前进行了膨胀操作，为了不影响形状特征参数的准确性，对图像进行腐蚀操作。另外，为避免仍然有杂质存在，此处又进行了一次二值开运算。实现代码以及效果图如下：

```
Mat element2 = getStructuringElement(MORPH_ELLIPSE, Size(30, 30));
morphologyEx(draw, final, MORPH_OPEN, element2);//二值开运算
morphologyEx(final, final, MORPH_ERODE, element1);//二值腐蚀
```

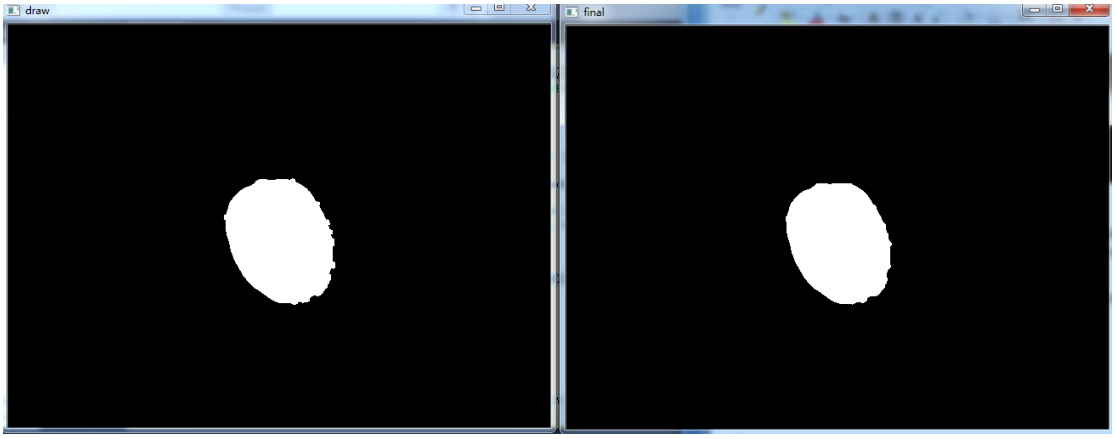

图 3-11 再次形态学处理后的效果图

第五步，在获得理想目标物体的二值化图像后，使用 contoursArea() 和 arcLength() 函数提取训练样本的形状特征参数。使用 3.2.2 节中的方法提取训练样本的颜色特征参数。

2、多幅训练图像逐一读取

水果特征数据库的建立，需要将所有训练图像逐一读取出来并一一进行特征提取。多幅训练图像的逐一读取代码如下，其中 i 代表水果的种类数，j 代表每种水果的训练图像的个数。

```
for (int i = 1; i <= 3; i++)
{
    for (int j = 1; j <= 10; j++)
    {
        string filename = "Fruit" + Int_String(2) + "/" + "s" + Int_String(i) + "/" + Int_String(j) + ".jpg";
        Mat img = imread(filename);
    }
}
```

3、生成特征向量和标签矩阵

特征向量矩阵是用于 SVM 的训练，它包含所有训练图像的特征参数。其中，矩阵的每一行储存一幅训练图像的特征参数，矩阵的列数与训练图像总个数相同。本文中，使用了 B、G、R、Area、Perimeter、Radian 六个特征参数，因此特征向量矩阵的列数为六。另外，标签矩阵用于告诉 SVM 每一幅训练图像中水果所属类别。因此，标签矩阵的列数为一，行数与特征向量矩阵相同。初始化特征向量矩阵和标签矩阵代码如下：

```
Mat feature = Mat::zeros(30, 6, CV_32FC1);
Mat feature_s = Mat::zeros(32, 6, CV_32FC1);
Mat feature_norm = Mat::zeros(30, 6, CV_32FC1);
Mat label = Mat::zeros(30, 1, CV_32FC1);
```

将得到的训练图像特征参数，输入到特征向量矩阵，同时生产标签矩阵，实现代码如下：

```
feature.at<float>((i - 1) * 10 + j - 1, 0) = B;
feature.at<float>((i - 1) * 10 + j - 1, 1) = G;
feature.at<float>((i - 1) * 10 + j - 1, 2) = R;
feature.at<float>((i - 1) * 10 + j - 1, 3) = Area[0];
feature.at<float>((i - 1) * 10 + j - 1, 4) = Perimeter[0];
feature.at<float>((i - 1) * 10 + j - 1, 5) = Radian[0];

label.row((i - 1) * 10 + j - 1) = i;
```

4、特征向量矩阵归一化

特征数据的归一化是物体分类识别中必不可少的一部分。SVM 是基于欧氏距离的分类器，若不使用特征数据的归一化，那些处于较大的数字范围的特征将很容易压倒那些处在相对较小的数字范围的特征，导致分类器几乎只根据处于较大数字范围的特征来分类。

另外，在使用 SVM 识别测试样本时，同样需要对测试样本的特征数据进行归一化处理。因此，在训练时，还需要提取出原始特征向量矩阵中每个特征的最大值与最小值并保存为 xml 文件。实现代码如下：

```
double minValue, maxValue;
Mat Min(1, 6, CV_32FC1);
Mat Max(1, 6, CV_32FC1);
for (int i = 0; i<feature.cols; i++)
{
    minMaxLoc(feature.col(i), &minValue, &maxValue);
    Min.col(i) = minValue;
    Max.col(i) = maxValue;
    normalize(feature.col(i), feature_norm.col(i), -1, 1, NORM_MINMAX);
}
FileStorage fs("other.xml", FileStorage::WRITE);
fs << "Min" << Min;
fs << "Max" << Max;
fs.release();
```

3.4.2 SVM 的训练

1、初始化 SVM 的参数设置

本文中，SVM 的类型设置为 C_SVM，内核类型为径向基函数，内核参数 C = 128，内核参数 gamma = 0.0078125，最大迭代次数为 100 次。实现代码如下：

```
CvSVMParams params;
params.svm_type    = SVM::C_SVC;
params.C           = 128;
params.gamma       = 0.0078125;
params.kernel_type = SVM::RBF;
params.term_crit   = TermCriteria(CV_TERMCRIT_ITER, 100, 1e-6);
```

2、训练并保存训练结果

使用 SVMParams 设置好参数后，使用 3.4.1 归一化的特征向量矩阵，进行 SVM 的训练，再使用 save 成员函数保存训练结果。实现代码如下：

```
CvSVM svm;
svm.train(feature_norm, label, Mat(), Mat(), params);
svm.save("svmdata.xml");
```

3.4.3 SVM 的识别

摄像头实时识别时，需要加载 SVM 训练时保存的参数，再对摄像头下水果进行识别，实现代码如下：（feature1 为归一化后的特征矩阵）

```
CvSVM svm;
svm.load("svmdata.xml");

FileStorage fs("other.xml", FileStorage::READ);
Mat Min;
fs["Min"]>>Min;
Mat Max;
fs["Max"]>>Max;

result1 = svm.predict(feature1);
```

3.4.4 注意事项

1、使用的摄像头对水果样本进行拍照时，摄像头相对于目标水果的距离大体上必须与识别时摄像头相对于目标水果的距离一致，否则将影响形状特征参数的数值，导致识别不准确。

2、训练图像采集时所使用的摄像头分辨率必须与识别时所使用的摄像头分辨率一致，否则同样会影响形状特征参数的数值，导致识别不准确。

3、整个过程中，摄像头不可以使用自动补光功能，否则将影响颜色特征参数的数值，导致识别不准确。

第四章 MFC 界面程序编写

4.1 概述

完成以上的背景建模，目标物体（水果）的提取，形态学处理，多物体特征参数提取，以及目标物体的识别后，整个水果识别算法完成。最后一步，编写人机交互界面进行可视化的操作。

由于本文使用的 C++ 编程完成的视觉算法设计，所以使用 MFC 编程来设计水果识别系统的操作界面。文中使用的是基于对话框的 MFC 程序，所涉及的研究内容主要有对话框的设计、控件的使用，和 MFC 与 OpenCV 的结合编程。

4.2 对话框的设计

在对话框设计中，本文一共使用了 4 个图片控件，7 个按钮控件，13 个静态文本控件，和 22 个编辑控件。对话框的设计效果图如下：

图 4-1 对话框效果图

4.3 控件的使用

4.3.1 静态文本控件

静态文本控件是用来显示在程序运行中不会发生变化的字符或者数字的控件，可以起到标示、提醒或者分割的作用，而且控件中的内容在程序运行过程中始终保持显示状态。默认情况下，静态文本控件的 ID 均为 IDC_STATIC，若需要为它们添加专属的消息处理函数，则必须要重新为其设定唯一的 ID 名称。

静态文本控件的功能由微软封装好的 CStatic 类来提供，虽然在实际操作中可以直接从工具箱中直接将静态文本控件拖动到对话框中来达到创建的目的，但在代码中，是先使用构造函数构造 CStatic 对象，再调用 Create 成员函数创建静态文本控件并添加到正在使用的对话框中，也就是 CStatic 的对象。

本文中主要使用了静态文本控件的显示字符串的功能，用于标示编辑控件中参数的具体含义。

4.3.2 编辑控件

编辑控件不仅具有静态文本控件所具有的输出接口，还提供有输入接口。在程序运行时，用户可以对编辑控件直接输入相应参数信息，也可以使用编辑控件随时显示程序运行当中的变量信息。编辑控件可以是单行显示，也可以由多行显示，换行标示符为“\r\n”，默认情况下，编辑控件的属性设置为单行显示。

编辑控件的功能由微软封装好的 CEdit 类来提供。在创建方式上，可以直接从对话框模板上进行创建也可以使用代码创建。

本文中，通过给每一个编辑控件添加成员变量来达到显示目标物体（水果）的特征参数以及种类信息的目的。在程序运行时，当我们得到形状、颜色特征参数后，将他们赋值给相应编辑控件中的成员变量（value 种类的 double 类型）后，使用 UpdateData(FALSE) 来将颜色、形状以及水果种类的参数信息显示在编辑控件中。另外，本文还使用编辑控件来显示程序进程（位于人机交互界面的右上角）。同样，对右上角编辑控件添加 value 种类的 CString 类型的成员变量，当程序完成按钮任务时，通过对 CString 进行字符赋值，来告诉用户程序的实时进程。效果图如下：

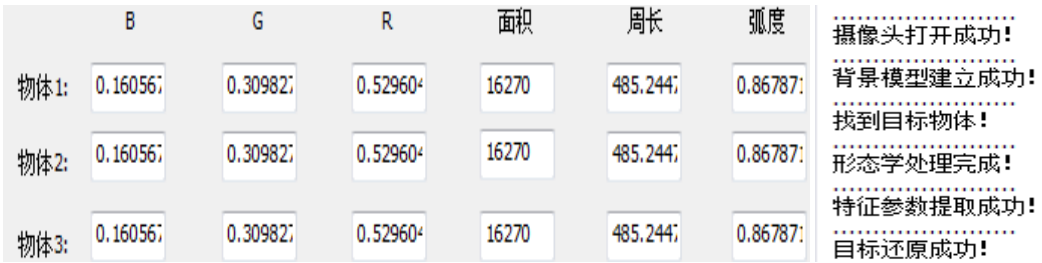

图 4-2 左为特征参数编辑控件，右为程序进程显示在编辑控件

4.3.3 按钮控件

按钮控件也叫命令控件，是界面程序开发中最常用的控件之一。当按钮控件被按下时，即执行相应操作代码。按钮控件有三种类型，分别是普通按钮 Button，复选框 Check Box，和单选按钮 Radio Button，按钮的类型可以在属性当中更改。本文使用的是普通按钮 Button 。

按钮控件功能由微软封装的 CButton 类提供，创建方式可以由工具箱直接创建也可以代码创建。

本文中一共使用七个按钮，均用来触发相应代码来执行不同的功能。它们用来触发打开摄像头、建立背景模型、提取前景目标、形态学处理、提取特征参数、目标还原以及识别的功能。通过双击按钮即可显示出当按钮控件被按下后，所需要执行的代码。

4.3.4 图片控件

图片控件和静态文本控件都属于静态控件，本文通过 MFC 与 OpenCV 的结合编程，使图片控件具有实时的输入输出功能。

本文中，图片控件用于显示图像处理后的效果以及摄像头下的实时情况。在按钮控件触发并执行相应的代码程序后，使用图片控件将处理效果显示出来。一共使用 4 个图片控件，分别用于显示摄像头下实时场景、显示利用图像分割技术提取出的前景目标图像、显示形态学处理后的前景目标图像和显示三通道前景目标图像。效果图如下：

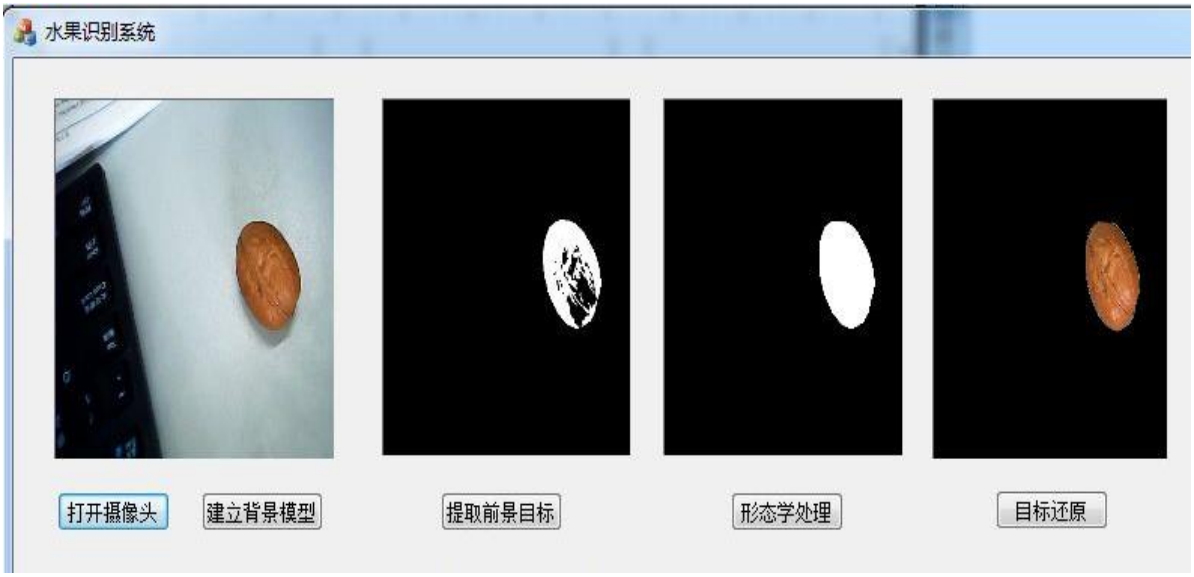

图 4-3 图片控件效果图

4.4 MFC 与 OpenCV 混合编程

本文中，通过 MFC 与 OpenCV 的结合，能够更好的完成算法实现。利用 imshow()、imread() 等图像显示与读取的函数，可以有效避免 MFC 本身对于图像操作的繁琐步骤，使代码更为简洁高效。

4.4.1 MFC 程序头文件修改

由于需要在 MFC 中使用 OpenCV 的库函数，在已经将 MFC 工程与 OpenCV 配置完成的情况下，首先需要修改对话框所对应的头文件，将所需要的库以及命名空间包含进来。程序截图如下：

```
// MFC_论文Dlg.h : header file
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/core/core.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/opencv.hpp"

using namespace cv;
using namespace std;
//
```

4.4.2 MFC 图片控件的修改

在 MFC 中，将图片控件用于图像处理以及实时操作是较为繁琐的。所以，本文利用了 OpenCV 中的 imshow() 以及 namedWindow() 函数使图片控件具有实时输入输出的能力。

实现原理为，在 MFC 程序的初始化阶段。首先使用 namedWindow() 函数创建一个 OpenCV 自带的图片显示窗口，并且获取该窗口的句柄以及父句柄，再将该父句柄设置为对话框中图片控件的父句柄，最后隐藏 OpenCV 自带窗口。最终，达到使用图片控件的窗口来显示 OpenCV 自带窗口显示内容的效果。初始化的实现代码如下：

```
namedWindow("view1",WINDOW_AUTOSIZE);//生成 OpenCV 自带窗口
HWND hWnd = (HWND)cvGetWindowHandle("view1");// 获取窗口句柄
HWND hParent = ::GetParent(hWnd);//获取父句柄
::SetParent(hWnd,GetDlgItem(IDC_STATIC1)->m_hWnd);//设置为图片控件父句柄，图片控件ID为 IDC_STATIC1
::ShowWindow(hParent,SW_HIDE);//隐藏 OpenCV 创建的窗口
```

在初始化阶段修改对话框中的图片控件属性后，需要在显示图像时进行尺寸修改。因为 OpenCV 自带的图像显示窗口具有自适应图像大小的能力，而 MFC 中的图片控件大小尺寸固定，如果不对输入图像大小进行调整，输出出来图像极有可能仅为源图像的一部分。本文中通过 OpenCV 中的 resize() 和 imshow() 函数来实现目的。

实现思路是，获取图片控件的尺寸，对于即将输入的图像按照控件大小进行尺寸变换，再使用 imshow() 函数在图片控件上输出图像。实现代码如下：

```
CRect rect;
GetDlgItem(a)->GetClientRect(&rect);//获取图片控件的尺寸
resize(img,img_s,Size(rect.Width(),rect.Height()));//尺寸归一化
imshow(view,img_s);//显示归一化后的图像
```

4.4.3 MFC 程序主体结构

实际上，为了使程序代码更具可观性以及为了避免变量在不同的成员函数下传递时出现错误。本文将主体程序代码全部写在了 Button1（打开摄像头）中，然后利用循环 while() 函数使程序始终运行。另外，在主体程序中多处设置标记位，其他按钮的作用就是通过修改标记位，决定主体程序具体执行哪段代码。程序截图如下：

```
void CMFC_论文Dlg::OnBnClickedButton2()
{
    // TODO: Add your control notification handler code here
    begin_bg = 1;//按下 Button2，将标记位修改为1
}


void CMFC_论文Dlg::OnBnClickedButton3()
{
    // TODO: Add your control notification handler code here
    begin_bg = 3;//按下 Button3，将标记位修改为3
}


void CMFC_论文Dlg::OnBnClickedButton4()
{
    // TODO: Add your control notification handler code here
    begin_bg = 4;//按下 Button4，将标记位修改为4
}


void CMFC_论文Dlg::OnBnClickedButton5()
{
    // TODO: Add your control notification handler code here
    begin_bg = 5;//按下 Button5，将标记位修改为5
}
```

由上图可以得知，当我们按下不同按钮时，标记位具有不同的数值，而不同的数值将决定主体程序执行不同的代码。在主体程序中，利用 if() 函数来判断标记位并决定执行哪部分代码。在每执行完一次标记位代码后，需要重置标记位，以等待下一次的按钮命令。主体程序代码在附录中贴出。

4.5 MFC 界面运行效果图

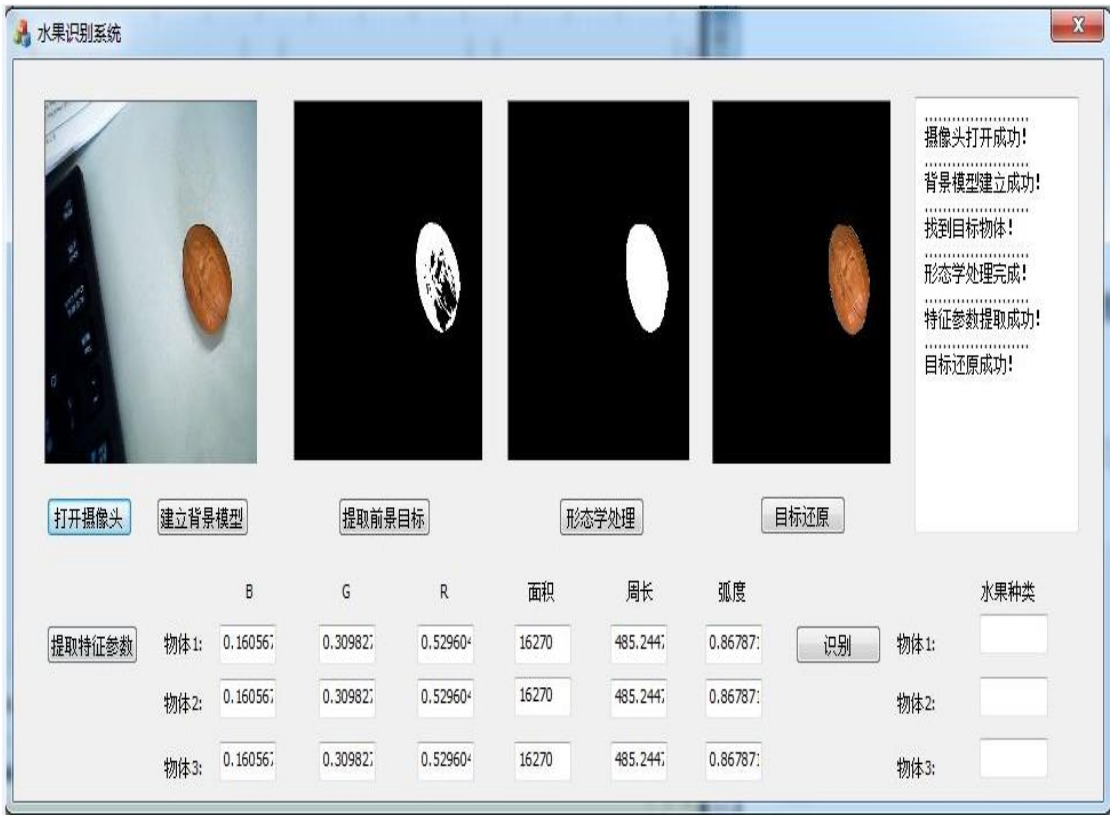

图 4-4 MFC 运行效果图

第五章 结论

在本文的最后，使用六种水果验证了水果自助识别系统的可行性。它们分别是核桃、苹果、香蕉、橘子、奇异果以及车厘子。通过实验发现，在不同的环境下，识别效果差异较大。但是只要在光源充足（无单侧光），且识别背景的反光效果不强烈的情况下，对程序当中的参数设置进行适当的修改，均可以获得较为准确的识别效果。下面给出本次毕业设计得到的主要成果。

1、摄像头实时识别（硬件图）

图 5-1 硬件图

2、水果数据库图片（部分）

图 5-2 水果数据库图片

3、水果特征向量矩阵（部分）

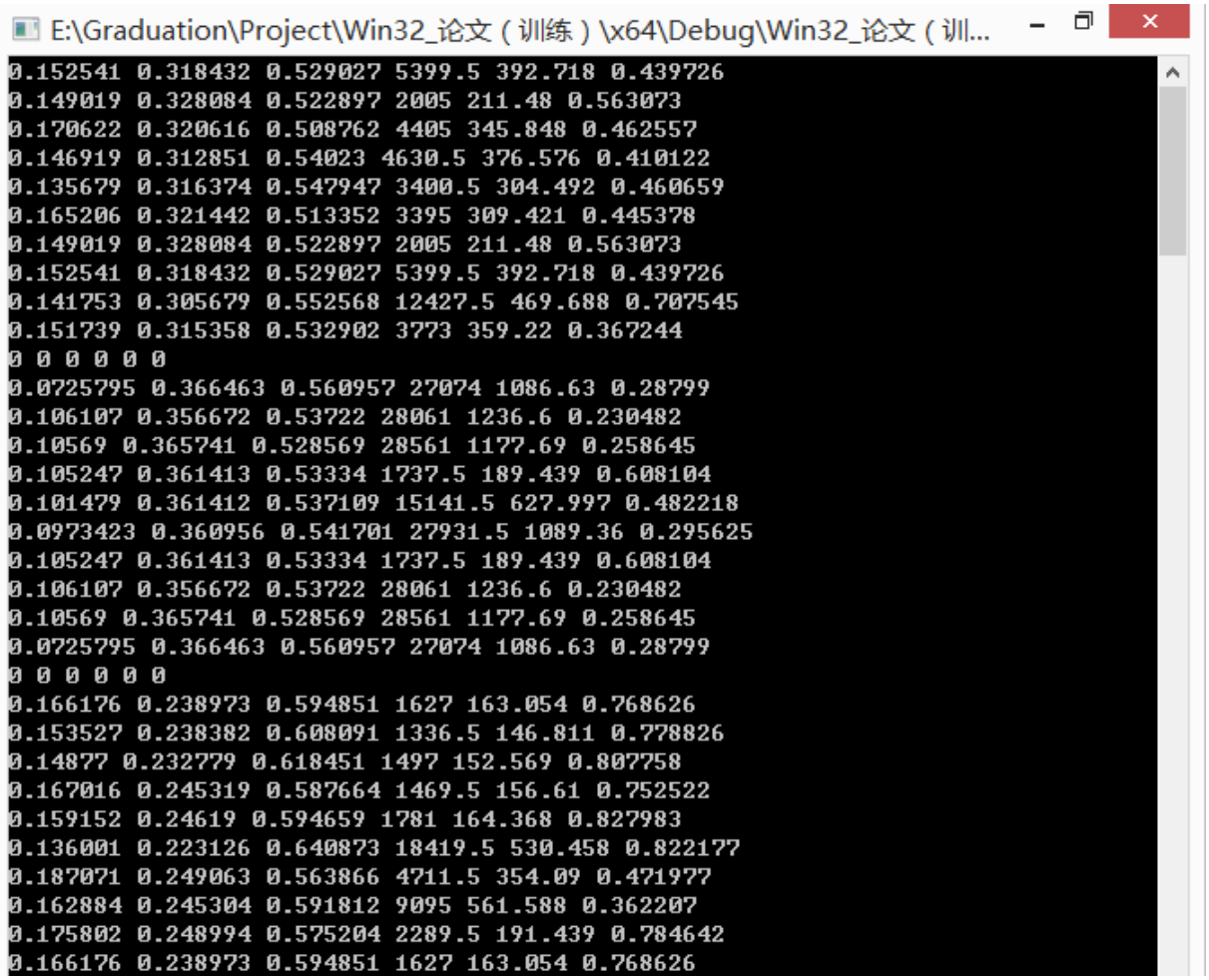

图 5-3 特征向量矩阵

4、成功识别的截图

图 5-4 识别出奇异果和苹果

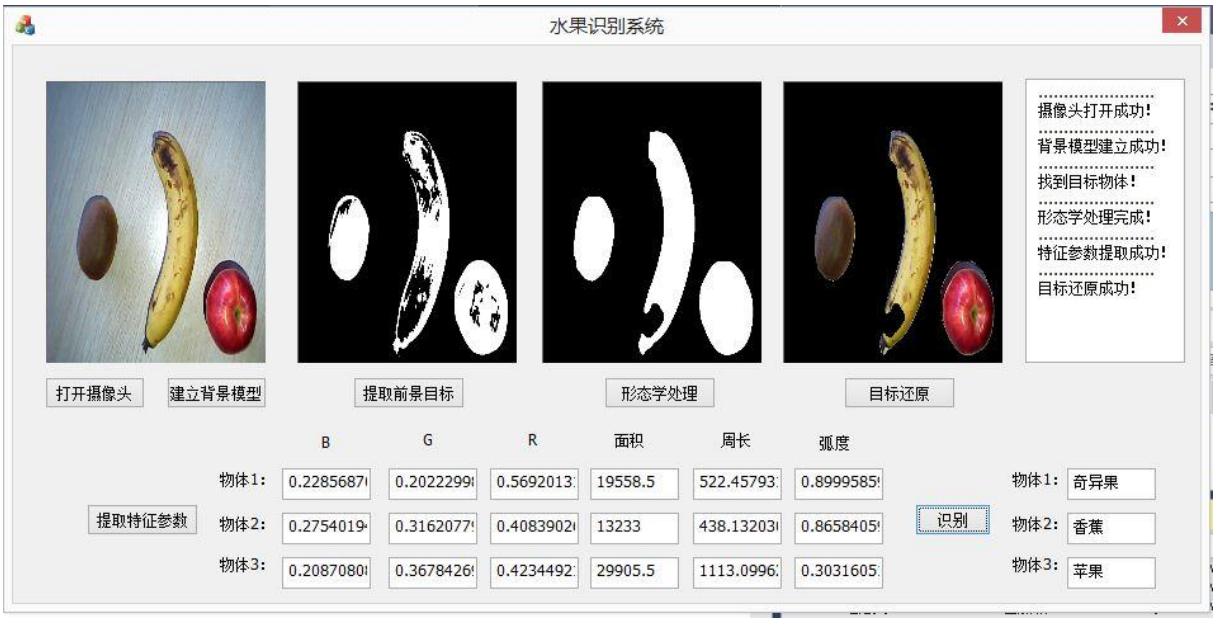

图 5-5 识别出奇异果、香蕉和苹果

【参考文献】

[1] 毛星云，冷雪飞，王碧辉，吴松森. OpenCV3 编程入门[M]. 北京: 电子工业出版社，2015: 157，187-188,191，248-250，304-306，328.

[2] 梁海英，李淑梅，谭晓东，胡世洋，罗琳. 基于案例的 Visual C++程序设计[M]. 北京: 清华大学出版社，2016: 76-88,，96-98.

[3] （美）Prateek Joshi，（西）David Millan Escriva，（巴西）Vinicius Godoy 著. OpenCV 实例精解[M]. 呆萌院长，李凤明，李翰阳译. 北京: 机械工业出版社，2016: 104，117.

[4] 张铮，倪红霞，苑春苗，杨立红. 精通 Matlab 数字图像处理与识别[M]. 北京: 人民邮电出版社，2013: 307-310，383-392.

[5] 付金霞，苏健民，刘嘉新，王健. 基于 RGB 模型的颜色特征在树种识别系统中的应用[D]. 哈尔滨: 东北林业大学，2007.

[6] 张中良. 基于机器视觉的图像目标识别方法综述[J]. 科技与创新，2016，（14）：32-33.

[7]侯宾，张文志，戴源成，田洪强. 基于 OpenCV 的目标物体颜色及轮廓的识别方法[J]. 现代电子技术，2014，37（24）：76-83.

[8] 陈源，张长江. 水果自动识别的 BP 神经网络方法[J]. 微型机与应，2010，29(22):40-48.

[9] 韩思奇，王蕾. 图像分割的阈值法综述[J]. 系统工程与电子技术，2002，24（6）：91-102

[10] Ergun Gumus, Niyazi Kilic, Ahmet Sertbas, Osman N. Ucan. Evaluation of face recognition techniques using PCA, wavelets and SVM[J]. Expert Systems with Applications, 2010 (37).

附录

MFC 对话框主程序（Visual studio 2015 版）：

```
#include "stdafx.h"
#include "MFC15_论文.h"
#include "MFC15_论文 Dlg.h"
#include "afxdialogex.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

class CAboutDlg : public CDialogEx
{
public:
	CAboutDlg();

#ifdef AFX_DESIGN_TIME
	enum { IDD = IDD_ABOUTBOX };
#endif

	protected:
	virtual void DoDataExchange(CDataExchange* pDX);

protected:
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialogEx(IDD_ABOUTBOX)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialogEx::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialogEx)
END_MESSAGE_MAP()

CMFC15_论文 Dlg::CMFC15_论文 Dlg(CWnd* pParent /*=NULL*/)
	: CDialogEx(IDD_MFC15__DIALOG, pParent)
	, begin_bg(0)
	, B_0(0)
	, B_1(0)
	, B_2(0)
	, G_0(0)
	, G_1(0)
	, G_2(0)
	, R_0(0)
	, R_1(0)
	, R_2(0)
	, Area_0(0)
	, Area_1(0)
	, Area_2(0)
```

```
	, Perimeter_0(0)
	, Perimeter_1(0)
	, Perimeter_2(0)
	, Radian_0(0)
	, Radian_1(0)
	, Radian_2(0)
	, Result_0(_T(""))
	, Result_1(_T(""))
	, Result_2(_T(""))
	, cout_string(_T(""))
{
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CMFC15_论文 Dlg::DoDataExchange(CDataExchange* pDX)
{
	CDialogEx::DoDataExchange(pDX);
	DDX_Text(pDX, IDC_EDIT7, begin_bg);
	DDX_Text(pDX, IDC_EDIT2, B_0);
	DDX_Text(pDX, IDC_EDIT18, B_1);
	DDX_Text(pDX, IDC_EDIT12, B_2);
	DDX_Text(pDX, IDC_EDIT1, G_0);
	DDX_Text(pDX, IDC_EDIT16, G_1);
	DDX_Text(pDX, IDC_EDIT10, G_2);
	DDX_Text(pDX, IDC_EDIT4, R_0);
	DDX_Text(pDX, IDC_EDIT17, R_1);
	DDX_Text(pDX, IDC_EDIT11, R_2);
	DDX_Text(pDX, IDC_EDIT5, Area_0);
	DDX_Text(pDX, IDC_EDIT15, Area_1);
	DDX_Text(pDX, IDC_EDIT8, Area_2);
	DDX_Text(pDX, IDC_EDIT3, Perimeter_0);
	DDX_Text(pDX, IDC_EDIT13, Perimeter_1);
	DDX_Text(pDX, IDC_EDIT9, Perimeter_2);
	DDX_Text(pDX, IDC_EDIT19, Radian_0);
	DDX_Text(pDX, IDC_EDIT14, Radian_1);
	DDX_Text(pDX, IDC_EDIT6, Radian_2);
	DDX_Text(pDX, IDC_EDIT21, Result_0);
	DDX_Text(pDX, IDC_EDIT23, Result_1);
	DDX_Text(pDX, IDC_EDIT22, Result_2);
	DDX_Text(pDX, IDC_EDIT20, cout_string);
}

BEGIN_MESSAGE_MAP(CMFC15_论文 Dlg, CDialogEx)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_BUTTON2, &CMFC15_论文 Dlg::OnBnClickedButton2)
	ON_BN_CLICKED(IDC_BUTTON1, &CMFC15_论文 Dlg::OnBnClickedButton1)
	ON_BN_CLICKED(IDC_BUTTON4, &CMFC15_论文 Dlg::OnBnClickedButton4)
	ON_BN_CLICKED(IDC_BUTTON3, &CMFC15_论文 Dlg::OnBnClickedButton3)
	ON_BN_CLICKED(IDC_BUTTON5, &CMFC15_论文 Dlg::OnBnClickedButton5)
	ON_BN_CLICKED(IDC_BUTTON6, &CMFC15_论文 Dlg::OnBnClickedButton6)
	ON_BN_CLICKED(IDC_BUTTON7, &CMFC15_论文 Dlg::OnBnClickedButton7)
END_MESSAGE_MAP()


BOOL CMFC15_论文 Dlg::OnInitDialog()
{
```

```
	CDialogEx::OnInitDialog();

	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		BOOL bNameValid;
		CString strAboutMenu;
		bNameValid = strAboutMenu.LoadString(IDS_ABOUTBOX);
		ASSERT(bNameValid);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	SetIcon(m_hIcon, TRUE);
	SetIcon(m_hIcon, FALSE);

	namedWindow("view1", WINDOW_AUTOSIZE);
	HWND hWnd = (HWND)cvGetWindowHandle("view1");
	HWND hParent = ::GetParent(hWnd);
	::SetParent(hWnd, GetDlgItem(IDC_STATIC1)->m_hWnd);
	::ShowWindow(hParent, SW_HIDE);

	namedWindow("view2", WINDOW_AUTOSIZE);
	HWND hWnd2 = (HWND)cvGetWindowHandle("view2");
	HWND hParent2 = ::GetParent(hWnd2);
	::SetParent(hWnd2, GetDlgItem(IDC_STATIC2)->m_hWnd);
	::ShowWindow(hParent2, SW_HIDE);

	namedWindow("view3", WINDOW_AUTOSIZE);
	HWND hWnd3 = (HWND)cvGetWindowHandle("view3");
	HWND hParent3 = ::GetParent(hWnd3);
	::SetParent(hWnd3, GetDlgItem(IDC_STATIC3)->m_hWnd);
	::ShowWindow(hParent3, SW_HIDE);

	namedWindow("view4", WINDOW_AUTOSIZE);
	HWND hWnd4 = (HWND)cvGetWindowHandle("view4");
	HWND hParent4 = ::GetParent(hWnd4);
	::SetParent(hWnd4, GetDlgItem(IDC_STATIC4)->m_hWnd);
	::ShowWindow(hParent4, SW_HIDE);

}

void CMFC15_论文 Dlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialogEx::OnSysCommand(nID, lParam);
	}
}
```

```
void CMFC15_论文 Dlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this);

		SendMessage(WM_ICONERASEBKGND,
		reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);

		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialogEx::OnPaint();
	}
}

HCURSOR CMFC15_论文 Dlg::OnQueryDragIcon()
{
	return static_cast<HCURSOR>(m_hIcon);
}

void CMFC15_论文 Dlg::OnBnClickedButton2()
{
	int count = 0;
	Mat frame, frame_video, frame_copy, dst2;
	Mat draw;
	VideoCapture capture;
	capture.open(1);
	capture >> frame;
	Mat imgAll = Mat::zeros(frame.rows, frame.cols, CV_32FC1);
	Mat imgAll2 = Mat::zeros(frame.rows, frame.cols, CV_8UC1);
	begin_bg = 0;

	while (true)
	{
		string str1 = "view1";
		capture >> frame_video;
		imshow_s(frame_video, IDC_STATIC1, str1);
		if (begin_bg == 0 || begin_bg == 2)
		{
			if (begin_bg == 0)
			{
				cout_string = " ..... \r\n 摄像头打开成功！";
				UpdateData(FALSE);
			}
			waitKey(30);
		}

		if (count < 30 && begin_bg == 1)
		{
			Mat frameGRAY;
```

```
        capture >> frame;
        blur(frame, frame, Size(3, 3));
        cvtColor(frame, frameGRAY, CV_BGR2GRAY);
        frameGRAY.convertTo(frameGRAY, CV_32FC1);

        for (int i = 0; i < frameGRAY.rows; i++)
        {
            for (int j = 0; j < frameGRAY.cols; j++)
            {
                imgAll.at<float>(i, j) = imgAll.at<float>(i, j) + frameGRAY.at<float>(i,
              j);
            }
        }
        waitKey(30);
    }
    if (count == 30 && begin_bg == 1)
    {
        for (int i = 0; i < frame.rows; i++)
        {
            for (int j = 0; j < frame.cols; j++)
            {
                imgAll.at<float>(i, j) = imgAll.at<float>(i, j) / count;
            }
        }
        imgAll.convertTo(imgAll2, CV_8UC1);
        cout_string = " ..... \r\n 摄像头打开成功！\r\n ..... \r\n 背
        景模型建立成功！";
        UpdateData(FALSE);
        begin_bg = 2;
    }

    if (count > 30 && begin_bg == 3)
    {
        Mat diff, frameGRAY, canny, dst, dst2;
        capture >> frame;
        blur(frame, frame, Size(3, 3));
        cvtColor(frame, frameGRAY, CV_BGR2GRAY);
        frameGRAY.convertTo(diff, CV_32FC1);
        absdiff(imgAll, diff, diff);

        for (int i = 0; i < frame.rows; i++)
        {
            for (int j = 0; j < frame.cols; j++)
            {
                if (diff.at<float>(i, j) < 40)
                    diff.at<float>(i, j) = 0.0;
                else
                    diff.at<float>(i, j) = 255.0;
            }
        }

        diff.convertTo(frame, CV_8UC1);
        string str2 = "view2";
        imshow_s(frame, IDC_STATIC2, str2);
        cout_string = " ..... \r\n 摄像头打开成功！\r\n ..... \r\n 背
        景模型建立成功！  \r\n ..... \r\n 找到目标物体！";
        UpdateData(FALSE);
        begin_bg = 2;
    }
```

```
if (count > 30 && begin_bg == 4)
{
	string str3 = "view3";
	Mat dst;
	Mat element1 = getStructuringElement(MORPH_ELLIPSE, Size(30, 30));
	morphologyEx(frame, dst, MORPH_CLOSE, element1);

	Mat element2 = getStructuringElement(MORPH_ELLIPSE, Size(10, 10));
	morphologyEx(dst, dst2, MORPH_OPEN, element2);
	imshow_s(dst2, IDC_STATIC3, str3);
	cout_string = " ..... \r\n 摄像头打开成功！\r\n ..... \r\n 背
	景模型建立成功！ \r\n ..... \r\n 找到目标物体！ \r\n .....
	\r\n 形态学处理完成！";
	UpdateData(FALSE);
	begin_bg = 2;
}

if (count > 30 && begin_bg == 5)
{
	Mat canny;
	Canny(dst2, canny, 30, 60);

	vector<vector<Point>> contours;
	vector<Vec4i> hierarchy;
	findContours(canny, contours, hierarchy, CV_RETR_EXTERNAL,
	CV_CHAIN_APPROX_NONE);
	Vec<double, 5> Area;
	Vec<double, 5> Perimeter;
	Vec<double, 5> Radian;

	findContours(canny, contours, hierarchy, CV_RETR_EXTERNAL,
	CV_CHAIN_APPROX_NONE);

	draw = Mat::zeros(dst2.rows, dst2.cols, CV_8UC3);
	for (int index = 0; index<contours.size(); index++)
	{

		if (index == 0)
		{
			Scalar color(255, 0, 0);
			drawContours(draw, contours, index, color, CV_FILLED, 8);
		}
		if (index == 1)
		{
			Scalar color(0, 255, 0);
			drawContours(draw, contours, index, color, CV_FILLED, 8);
		}
		if (index == 2)
		{
			Scalar color(0, 0, 255);
			drawContours(draw, contours, index, color, CV_FILLED, 8);
		}
		Area[index] = contourArea(contours[index]);
		Perimeter[index] = arcLength(contours[index], true);
		Radian[index] = 4 * 3.14*Area[index] / (Perimeter[index] *
		Perimeter[index]);
	}

	double B0 = 0.0, G0 = 0.0, R0 = 0.0, Sum0 = 0.0;
	double B1 = 0.0, G1 = 0.0, R1 = 0.0, Sum1 = 0.0;
```

```
double B2 = 0.0, G2 = 0.0, R2 = 0.0, Sum2 = 0.0;
for (int x = 0; x<draw.rows; x++)
{
	for (int y = 0; y<draw.cols; y++)
	{
		for (int c = 0; c<3; c++)
		{
			if (draw.at<Vec3b>(x, y)[c]>250)
			{
				if (c == 0)
				{
					B0 = B0 + frame_video.at<Vec3b>(x, y)[0];
					G0 = G0 + frame_video.at<Vec3b>(x, y)[1];
					R0 = R0 + frame_video.at<Vec3b>(x, y)[2];
				}
				if (c == 1)
				{
					B1 = B1 + frame_video.at<Vec3b>(x, y)[0];
					G1 = G1 + frame_video.at<Vec3b>(x, y)[1];
					R1 = R1 + frame_video.at<Vec3b>(x, y)[2];
				}
				if (c == 2)
				{
					B2 = B2 + frame_video.at<Vec3b>(x, y)[0];
					G2 = G2 + frame_video.at<Vec3b>(x, y)[1];
					R2 = R2 + frame_video.at<Vec3b>(x, y)[2];
				}
			}
		}
	}
}
Sum0 = B0 + G0 + R0;
if (Sum0 >10000)
{
	B0 = B0 / Sum0;
	G0 = G0 / Sum0;
	R0 = R0 / Sum0;
}

Sum1 = B1 + G1 + R1;
if (Sum1 >10000)
{
	B1 = B1 / Sum1;
	G1 = G1 / Sum1;
	R1 = R1 / Sum1;
}

Sum2 = B2 + G2 + R2;
if (Sum2 >10000)
{
	B2 = B2 / Sum2;
	G2 = G2 / Sum2;
	R2 = R2 / Sum2;
}
B_0 = B0; G_0 = G0; R_0 = R0; Area_0 = Area[0]; Perimeter_0 = Perimeter[0];
Radian_0 = Radian[0];
B_1 = B1; G_1 = G1; R_1 = R1; Area_1 = Area[1]; Perimeter_1 = Perimeter[1];
Radian_1 = Radian[1];
B_2 = B2; G_2 = G2; R_2 = R2; Area_2 = Area[2]; Perimeter_2 = Perimeter[2];
Radian_2 = Radian[2];
```

```
        cout_string = " ..... \r\n 摄像头打开成功！\r\n ..... \r\n 背
        景模型建立成功！ \r\n ..... \r\n 找到目标物体！ \r\n .....
        \r\n 形态学处理完成！ \r\n ..... \r\n 特征参数提取成功！";
        UpdateData(FALSE);
        begin_bg = 2;
}

if (count > 30 && begin_bg == 6)
{
        frame_copy = frame_video;
        vector<Mat> channels;
        split(frame_copy, channels);
        for (int x = 0; x<draw.rows; x++)
        {
                for (int y = 0; y<draw.cols; y++)
                {
                        if (draw.at<Vec3b>(x, y)[0] < 10 && draw.at<Vec3b>(x, y)[1] < 10 &&
                          draw.at<Vec3b>(x, y)[2] < 10)
                        {
                                channels[0].at<uchar>(x, y) = 0;
                                channels[1].at<uchar>(x, y) = 0;
                                channels[2].at<uchar>(x, y) = 0;
                        }
                }
        }
        string str4 = "view4";
        merge(channels, frame_copy);
        imshow_s(frame_copy, IDC_STATIC4, str4);
        cout_string = " ..... \r\n 摄像头打开成功！\r\n ..... \r\n 背
        景模型建立成功！ \r\n ..... \r\n 找到目标物体！ \r\n .....
        \r\n 形态学处理完成！ \r\n ..... \r\n 特征参数提取成功！
        \r\n ..... \r\n 目标还原成功！";
        UpdateData(FALSE);
        begin_bg = 2;
}

if (count > 30 && begin_bg == 7)
{
        Ptr<SVM> svm = SVM::create();
        svm->load("svmdata.xml");

        FileStorage fs("other.xml", FileStorage::READ);
        Mat Min;
        fs["Min"] >> Min;
        Mat Max;
        fs["Max"] >> Max;

        Mat feature1 = Mat::zeros(1, 6, CV_32FC1);
        feature1.col(0) = B_0;
        feature1.col(1) = G_0;
        feature1.col(2) = R_0;
        feature1.col(3) = Area_0;
        feature1.col(4) = Perimeter_0;
        feature1.col(5) = Radian_0;

        Mat feature2 = Mat::zeros(1, 6, CV_32FC1);
        feature2.col(0) = B_1;
        feature2.col(1) = G_1;
        feature2.col(2) = R_1;
```

```
	feature2.col(3) = Area_1;
	feature2.col(4) = Perimeter_1;
	feature2.col(5) = Radian_1;

	Mat feature3 = Mat::zeros(1, 6, CV_32FC1);
	feature3.col(0) = B_2;
	feature3.col(1) = G_2;
	feature3.col(2) = R_2;
	feature3.col(3) = Area_2;
	feature3.col(4) = Perimeter_2;
	feature3.col(5) = Radian_2;


	for (int i = 0; i<feature1.cols; i++)
	{
		feature1.col(i) = (feature1.col(i) - Min.col(i)) / (Max.col(i) - Min.col(i)) * 2;
		feature1.col(i) = feature1.col(i) - 1;
	}

	for (int i = 0; i<feature2.cols; i++)
	{
		feature2.col(i) = (feature2.col(i) - Min.col(i)) / (Max.col(i) - Min.col(i)) * 2;
		feature2.col(i) = feature2.col(i) - 1;
	}

	for (int i = 0; i<feature3.cols; i++)
	{
		feature3.col(i) = (feature3.col(i) - Min.col(i)) / (Max.col(i) - Min.col(i)) * 2;
		feature3.col(i) = feature3.col(i) - 1;
	}

	float result1 = svm->predict(feature1);
	 if(result1 == 1 && B_0!=0)
		Result_0 = “奇异果”;
	 if(result1 == 2 && B_0!=0)
		Result_0 = “香蕉”;
	 if(result1 == 3 && B_0!=0)
		Result_0 = “苹果”;

	float result2 = svm->predict(feature2);
	 if(result2 == 1 && B_1!=0)
		Result_1 = “奇异果”;
	 if(result2 == 2 && B_1!=0)
		Result_1 = “香蕉”;
	 if(result2 == 3 && B_1!=0)
		Result_1 = “苹果”;

	float result3 = svm->predict(feature3);
	 if(result3 == 1 && B_2!=0)
		Result_2 = “奇异果”;
	 if(result3 == 2 && B_2!=0)
		Result_2 = “香蕉”;
	 if(result3 == 3 && B_2!=0)
		Result_2 = “苹果”;

	UpdateData(FALSE);
	begin_bg = 2;
}
```

```
            if (begin_bg == 1 || begin_bg == 2 || begin_bg == 3)
            {
                count++;
            }
        }
}

void CMFC15_论文 Dlg::imshow_s(Mat img, int a, string view)
{
    Mat img_s;
    CRect rect;
    GetDlgItem(a)->GetClientRect(&rect);
    resize(img, img_s, Size(rect.Width(), rect.Height()));
    imshow(view, img_s);
}

void CMFC15_论文 Dlg::OnBnClickedButton1()
{
   begin_bg = 1;
}

void CMFC15_论文 Dlg::OnBnClickedButton4()
{
    begin_bg = 3;
}

void CMFC15_论文 Dlg::OnBnClickedButton3()
{
    begin_bg = 4;
}

void CMFC15_论文 Dlg::OnBnClickedButton5()
{
    begin_bg = 6;
}

void CMFC15_论文 Dlg::OnBnClickedButton6()
{
    begin_bg = 5;
}

void CMFC15_论文 Dlg::OnBnClickedButton7()
{
    begin_bg = 7;
}
```

致谢

本文能顺利完成，首先最要感谢的是我的导师潘剑飞。潘剑飞老师严谨的治学态度，精益求精的科研作风以及耐心细致的教学态度都使我终生受益。其实，早在大三的 L4 课程当中，在潘剑飞老师的指导下，我就已经开始了对于水果识别的研究。到最终论文的完成，潘剑飞老师总是能给予我耐心的指导。不仅如此，潘剑飞教授也是我的留学推荐信老师之一，为我出国读研的申请提供了极大的帮助。在此，向潘剑飞老师致以诚挚的感谢！

同时在大学期间，得到机电学院的钟小品老师在图像处理方面的多次指导。对于钟小品老师的无私帮助表示由衷的感谢。

还要感谢我的大学好友刘汇涛对于本人在完成毕业设计当中的帮助和支持。

最后，我要感谢我的家人。是你们的支持，让我能够顺利完成自己的学业。是你们的鼓励，让我拥有不断向前的勇气。

由于学识有限，本文还存在着诸多的不足和疏漏，敬请各位老师和同学批评指正。