
Broadview
www.broadview.com.cn
站在浪潮之巅，零距离接触最前沿的计算机视觉编程技术
全面涵盖OpenCV2、OpenCV3双版本的核心编程技巧
附赠OpenCV2、OpenCV3双版本总计200余个配套示例程序源代码
OpenCV3
编程入门
毛星云　冷雪飞　等编著
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

内 容 简 介

OpenCV 在计算机视觉领域扮演着重要的角色。作为一个基于开源发行的跨平台计算机视觉库，OpenCV 实现了图像处理和计算机视觉方面的很多通用算法。本书以当前最新版本的 OpenCV 最常用最核心的组件模块为索引，深入浅出地介绍了 OpenCV2 和 OpenCV3 中的强大功能、性能，以及新特性。书本配套的 OpenCV2 和 OpenCV3 双版本的示例代码包中，含有总计两百多个详细注释的程序源代码与思路说明。读者可以按图索骥，按技术方向进行快速上手和深入学习。

本书要求读者具有基础的 C/C++知识，适合研究计算机视觉以及相关领域的在校学生和老师、初次接触 OpenCV 但有一定 C/C++编程基础的研究人员，以及已有过 OpenCV 1.0 编程经验，想快速了解并上手 OpenCV2、OpenCV3 编程的计算机视觉领域的专业人员。本书也适合于图像处理、计算机视觉领域的业余爱好者、开源项目爱好者做为通向新版 OpenCV 的参考手册之用。

本书配套的【示例程序】、【.exe 可执行文件】、【书内彩图】的下载链接可通过扫描本书封底后勒口的二维码获取。

图书在版编目（CIP）数据

OpenCV3 编程入门 / 毛星云等编著. —北京：电子工业出版社，2015.2

ISBN 978-7-121-25331-7

Ⅰ. ①O… Ⅱ. ①毛… Ⅲ. ①图象处理软件－程序设计 Ⅳ. ①TP391.41

中国版本图书馆 CIP 数据核字（2014）第 310454 号

责任编辑：陈晓猛
印　　刷：北京京科印刷有限公司
装　　订：河北省三河市路通装订厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×1 092　1/16　印张：29.25　字数：653 千字
版　　次：2015 年 2 月第 1 版
印　　次：2015 年 2 月第 1 次印刷
定　　价：79.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

第 3 章
HighGUI 图形用户界面初步

导读

在 OpenCV 架构分析一节中我们讲过，HighGUI 模块为高层 GUI 图形用户界面模块，包含媒体的输入输出、视频捕捉、图像和视频的编码解码、图形交互界面的接口等内容。而在 1.5 节中讲到的 VideoCapture 视频类，就是出自此 HighGUI 模块。本章旨在为大家展开讲解 OpenCV 中最常用到的一些交互操作，包括图像的载入、显示和输出，为程序添加滑动条，以及鼠标操作等常用内容。

本章中，你将学到：

- 图像的载入、显示和输出到文件的详细分析
- 滑动条（Trackbar）的创建和使用
- OpenCV 中的鼠标操作

3.1 图像的载入、显示和输出到文件

学习过以往版本 OpenCV 的读者应该都清楚，对于 OpenCV1.0 时代的基于 C 语言接口而建的图像存储格式 IplImage*，如果在退出前忘记 release 掉的话，会造成内存泄露，而且用起来十分繁琐。我们在 debug 程序的时候，往往很大一部分时间会去纠结手动释放内存相关的问题。虽然对于小型的程序来说，手动管理内存不是什么难题，但一旦开发的项目日益庞大，代码量达到一定的规模，我们便会开始越来越多地纠缠于内存管理的问题，而不能把全部精力用于解决核心开发目标。因为不合适的图像存储数据结构而疲于维护日益庞大的项目，就有些舍本逐末的感觉了。

自踏入 2.0 版本的时代以来，OpenCV 采用了 Mat 类作为数据结构进行图像存取。这一改进使 OpenCV 变得和几乎零门槛入门的 Matlab 一样，很容易上手和用于实际开发。新版 OpenCV 中甚至有些函数名称都和 Matlab 中的一样，比如大家所熟知的 imread、imwrite、imshow 等函数。这对于广大图像处理和计算机视觉领域的研究者们来说，的确是一件可喜可贺的事情。而这一小节中，我们主要来详细讲解 OpenCV2、OpenCV3 入门最基本的问题，即图像的载入、显示和输出。

3.1.1 OpenCV 的命名空间

OpenCV 中的 C++类和函数都是定义在命名空间 cv 之内的，有两种方法可以访问：第一种，是在代码开头的适当位置加上 usingnamespace cv；这句代码，规定程序位于此命名空间之内；另外一种，是在使用 OpenCV 的每一个类和函数时，都加入 cv::命名空间。不过这种情况会很繁琐，每用一个 OpenCV 的类或者函数，都要多敲四下键盘写出 cv::。所以，推荐大家在代码开头的适当位置，加上 using namespace cv;这句。

比如在写简单的 OpenCV 程序的时候，以下三句可以作为标配：

```
#include <opencv2/core/core.hpp>
#include<opencv2/highgui/highgui.hpp>
using namespace cv;
```

3.1.2 Mat 类简析

Mat 类是用于保存图像以及其他矩阵数据的数据结构，默认情况下其尺寸为 0。我们也可以指定其初始尺寸，比如定义一个 Mat 类对象，就要写 cv::Mat pic（320,640,cv::Scalar(100)）；

Mat 类型作为 OpenCV2、OpenCV3 新纪元的重要代表，在稍后的章节中，笔者会花长篇幅详细讲解它，现在我们只要理解它是对应于 OpenCV1.0 时代的 IplImage，主要用来存放图像的数据结构就行了。对于本节，我们需要用到关于 Mat 的其实就简单的这样一句代码：

```
Mat srcImage= imread("dota.jpg");
```

这表示从工程目录下把一幅名为 dota.jpg 的 jpg 类型的图像载入到 Mat 类型的 srcImage 变量中。对于这里的 imread 函数，用于将图片读入 Mat 类型中，会在下文进行详细剖析。而 Mat 类，也会在第 4 章中进行更加全面的讲解。

3.1.3 图像的载入与显示概述

在新版本的 OpenCV2 中，最简单的图像载入和显示只需要两句代码，非常便捷。这两句代码分别对应了两个函数，它们分别是 imread()以及 imshow()。

3.1.4 图像的载入：imread()函数

首先来看 imread 函数，其用于读取文件中的图片到 OpenCV 中。可以在 OpenCV 官方文档中查到它的原型，如下。

```
Mat imread(const string& filename, intflags=1 );
```

（1）第一个参数，const string&类型的 filename，填我们需要载入的图片路径名。在 Windows 操作系统下，OpenCV 的 imread 函数支持如下类型的图像载入。

- Windows 位图：*.bmp, *.dib
- JPEG 文件：*.jpeg, *.jpg, *.jpe
- JPEG 2000 文件：*.jp2
- PNG 图片：*.png
- 便携文件格式：*.pbm, *.pgm, *.ppm
- Sun rasters 光栅文件：*.sr, *.ras
- TIFF 文件：*.tiff, *.tif

（2）第二个参数，int 类型的 flags，为载入标识，它指定一个加载图像的颜色类型。可以看到它自带默认值 1，所以有时候这个参数在调用时可以忽略。在看了下面的讲解之后，我们就会发现，如果在调用时忽略这个参数，就表示载入三通道的彩色图像。这个参数可以在 OpenCV 中标识图像格式的枚举体中取值。通过转到定义，我们可以在 higui_c.h 中发现这个枚举的定义是这样的：

```
enum
{
/* 8bit, color or not */
   CV_LOAD_IMAGE_UNCHANGED  =-1,
/* 8bit, gray */
   CV_LOAD_IMAGE_GRAYSCALE  =0,
/* ?, color */
   CV_LOAD_IMAGE_COLOR     =1,
/* any depth, ? */
   CV_LOAD_IMAGE_ANYDEPTH   =2,
/* ?, any color */
   CV_LOAD_IMAGE_ANYCOLOR   =4
};
```

对常用标识符相应的解释：

- CV_LOAD_IMAGE_UNCHANGED——等价取值为-1，这个标识在新版本中已被废置，忽略。
- CV_LOAD_IMAGE_GRAYSCALE——等价取值为 0，如果取这个标识的话，始终将图像转换成灰度再返回。
- CV_LOAD_IMAGE_COLOR——等价取值为 1，如果取这个标识，总是转换图像到彩色再返回。
- CV_LOAD_IMAGE_ANYDEPTH——等价取值为 2，如果取这个标识，且载入的图像的深度为 16 位或者 32 位，就返回对应深度的图像，否则，就转换为 8 位图像再返回。

需要说明的是，如果输入有冲突的标志，将采用较小的数字值。比如 CV_LOAD_IMAGE_COLOR | CV_LOAD_IMAGE_ANYCOLOR 将载入三通道图。而如果想要载入最真实无损的源图像，可以选择 CV_LOAD_IMAGE_ANYDEPTH | CV_LOAD_IMAGE_ ANYCOLOR。

OpenCV3 中此处列举的 CV_LOAD_IMAGE_UNCHANGED、CV_LOAD_IMAGE_GRAYSCALE 、 CV_LOAD_IMAGE_COLOR 、 CV_LOAD_IMAGE_ ANYDEPTH、CV_LOAD_IMAGE_ANYCOLOR 等宏已经失效，但在官方文档和源代码中暂时没有发现新的可以替换它们的宏，所以请暂时用与其等价的-1、0、1、2、4 代替。相信后续 OpenCV 版本中一定会进行修复。

因为 flags 是 int 型的变量，若我们不在这个枚举体中取固定的值，可以这样进行：

- flags >0 返回一个 3 通道的彩色图像；
- flags =0 返回灰度图像；
- flags <0 返回包含 Alpha 通道的加载图像。

输出的图像默认情况下不返回 Alpha 通道。如果想要载入 Alpha 通道，这里就需要取负值。比如，将 flags 取 1999 也是可以的，和取 1 的效果一样，它们同样表示返回一个 3 通道的彩色图像。
另外，需要额外注意，若以彩色模式载入图像，解码后的图像会以 BGR 的通道顺序进行存储，即蓝、绿、红的顺序，而不是通常的 RGB 的顺序。

经过上面详细的讲解，我们一起看几个载入示例，以便多方位地掌握 imread 函数的用法。

```
Mat image0=imread("1.jpg",2 | 4);//载入无损的源图像
Mat image1=imread("1.jpg",0);//载入灰度图
Mat image2=imread("1.jpg",199);//载入 3 通道的彩色图像
```

3.1.5 图像的显示：imshow()函数

imshow()函数用于在指定的窗口中显示一幅图像，函数原型如下。

```
void imshow(const string& winname, InputArray mat);
```

- 第一个参数：const string&类型的 winname，填需要显示的窗口标识名称。
- 第二个参数：InputArray 类型的 mat，填需要显示的图像。

imshow 函数用于在指定的窗口中显示图像。如果窗口是用 CV_WINDOW_AUTOSIZE（默认值）标志创建的，那么显示图像原始大小。否则，将图像进行缩放以适合窗口。而 imshow 函数缩放图像，取决于图像的深度，具体如下。

- 如果载入的图像是 8 位无符号类型（8-bit unsigned），就显示图像本来的样子。
- 如果图像是 16 位无符号类型（16-bit unsigned）或 32 位整型（32-bit integer），便用像素值除以 256。也就是说，值的范围是[0,255 x 256]映射到[0, 255]。
- 如果图像是 32 位浮点型（32-bit floating-point），像素值便要乘以 255。也就是说，该值的范围是[0, 1]映射到[0, 255]。

还有一点，在窗口创建的时候，如果设定了支持 OpenGL（WINDOW_OPENGL），那么 imshow 还支持 ogl::Buffer、ogl::Texture2D 以及 gpu::GpuMat 作为输入。

关于 imwrite 和 imshow 函数最精简的示例程序，可以参考 1.3.8 节“最终的测试”或 1.4.1 节“第一个程序：图像显示”中的代码。

3.1.6 关于 InputArray 类型

对于这里的 InputArray 类型，通过对 InputArray 转到定义，我们可以在 core.hpp 中查到一个 typedef 声明，如下：

```
typedef const _InputArray& InputArray;
```

这其实一个类型声明引用，就是说_InputArray 和 InputArray 是一个意思，因此我们就来做最后一步：对_InputArray 进行转到定义，可以在 core.hpp 头文件中发现 InputArray 的真身。InputArray 的源代码略显冗长，不在这里贴出，若读者自己去实践并通过转到定义的方法找到 InputArray 的真身，便可以看到_InputArray 类的里面首先定义了一个枚举，然后是各类的模板类型和一些方法。而很多时候，遇到函数原型中的 InputArray/OutputArray 类型，我们把它简单地当做 Mat 类型即可。

3.1.7 创建窗口：namedWindow()函数

namedWindow 函数用于创建一个窗口。若是简单地进行图片显示，可以略去 namedWindow 函数的调用，即先调用 imread 读入图片，然后用 imshow 直接指定出窗口名进行显示即可。但需要在显示窗口之前就用到窗口名时，比如我们后面会马上讲到滑动条的使用，要指定滑动条依附到某个窗口上，就需要 namedWindow 函数先创建出窗口，显式地规定窗口名称了。

namedWindow 的函数原型如下：

```
void namedWindow(const string& winname,int flags=WINDOW_AUTOSIZE );
```

（1）第一个参数，const string&型的 name，填写被用作窗口的标识符的窗口名称。

（2）第二个参数，int 类型的 flags ，窗口的标识，可以填如下几种值。

- WINDOW_NORMAL，设置这个值，用户可以改变窗口的大小（没有限制）。OpenCV2 中它还可以写为 CV_WINDOW_NORMAL。
- WINDOW_AUTOSIZE，设置这个值，窗口大小会自动调整以适应所显示的图像，并且用户不能手动改变窗口大小。OpenCV2 中它还可以写为 CV_WINDOW_AUTOSIZE。
- WINDOW_OPENGL，设置这个值，窗口创建的时候会支持 OpenGL。OpenCV2 中它还可以写为 CV_WINDOW_OPENGL。

首先需要注意的是，namedWindow 函数有默认值 WINDOW_AUTOSIZE，所以，一般情况下，这个函数我们填一个变量就行了。namedWindow 函数的作用是通过指定的名字，创建一个可以作为图像和进度条的容器窗口。如果具有相同名称的窗口已经存在，则函数不做任何事情。我们可以调用 destroyWindow()或者 destroyAllWindows()函数来关闭窗口，并取消之前分配的与窗口相关的所有内存空间。

但是事实上，对于代码量不大的简单程序来说，我们完全没有必要手动调用上述的 destroyWindow()或者 destroyAllWindows()函数，因为在退出时，所有的资源和应用程序的窗口会被操作系统自动关闭。

3.1.8 输出图像到文件：imwrite()函数

在 OpenCV 中，输出图像到文件一般采用 imwrite 函数，它的声明如下。

```
bool imwrite(const string& filename,InputArray img, const vector<int
>& params=vector<int>() );
```

（1）第一个参数，const string&类型的 filename，填需要写入的文件名。注意要带上后缀，如“123.jpg”。

（2）第二个参数，InputArray 类型的 img，一般填一个 Mat 类型的图像数据。

（3）第三个参数，const vector<int>&类型的 params，表示为特定格式保存的参数编码。它有默认值 vector<int>()，所以一般情况下不需要填写。而如果要填写的话，有下面这些需要了解的地方：

- 对于 JPEG 格式的图片，这个参数表示从 0 到 100 的图片质量（CV_IMWRITE_JPEG_ QUALITY），默认值是 95。
- 对于 PNG 格式的图片，这个参数表示压缩级别（CV_IMWRITE_PNG_ COMPRESSION）从 0 到 9。较高的值意味着更小的尺寸和更长的压缩时间，默认值是 3。
- 对于 PPM，PGM，或 PBM 格式的图片，这个参数表示一个二进制格式标志（CV_IMWRITE_PXM_BINARY），取值为 0 或 1，默认值是 1。

imwrite 函数用于将图像保存到指定的文件。图像格式是基于文件扩展名的，可保存的扩展名和 imread 中可以读取的图像扩展名一致。

下面是一个示例程序，讲解 imwrite 函数的用法——在 OpenCV 中生成一幅 png 图片，并写入到当前工程目录下。

```
#include<opencv2/opencv.hpp>
#include <vector>

using namespace cv;
using namespace std;

void createAlphaMat(Mat &mat)
{
    for(int i = 0; i < mat.rows; ++i) {
        for(int j = 0; j < mat.cols; ++j) {
         Vec4b&rgba = mat.at<Vec4b>(i, j);
         rgba[0]= UCHAR_MAX;
         rgba[1]= saturate_cast<uchar>((float (mat.cols - j)) /
((float)mat.cols) *UCHAR_MAX);
         rgba[2]= saturate_cast<uchar>((float (mat.rows - i)) /
((float)mat.rows) *UCHAR_MAX);
         rgba[3]= saturate_cast<uchar>(0.5 * (rgba[1] + rgba[2]));
        }
    }
}

int main()
{
    //创建带 Alpha 通道的 Mat
    Mat mat(480, 640, CV_8UC4);
    createAlphaMat(mat);

    vector<int>compression_params;
    /此句代码的 OpenCV2 版为：
    //compression_params.push_back(CV_IMWRITE_PNG_COMPRESSION);
    //此句代码的 OpenCV3 版为：
    compression_params.push_back(IMWRITE_PNG_COMPRESSION);
    compression_params.push_back(9);

    try{
        imwrite("透明 Alpha 值图.png", mat, compression_params);
        imshow("生成的 PNG 图",mat);
        fprintf(stdout,"PNG 图片文件的 alpha 数据保存完毕～\n 可以在工程目录下
查看由 imwrite 函数生成的图片\n");
        waitKey(0);
    }
    catch(runtime_error& ex) {
        fprintf(stderr,"图像转换成 PNG 格式发生错误：%s\n", ex.what());
        return 1;
    }

    return 0;
}
```

程序运行截图如图 3.1 所示。运行完毕，我们可以在工程目录下发现一张生成的名为“透明 Alpha 值图.png”的图片文件。

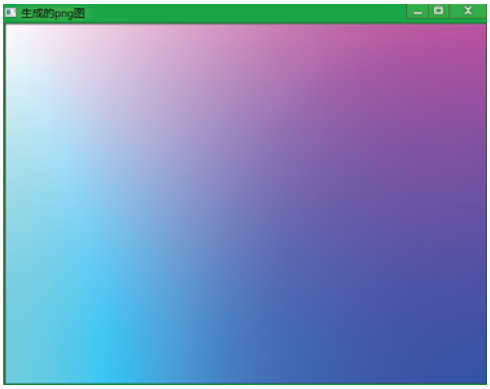

图 3.1 由 imwrite 生成的图

3.1.9 综合示例程序：图像的载入、显示与输出

本小节将给出一个综合示例，演示如何载入图像，进行简单的图像混合，显示图像，并且输出混合后的图像到 jpg 格式的文件中。

出于注重演示效果的原因，程序中图像混合的具体细节我们放到稍后的篇幅中再讲，现在先给大家看看混合的效果和源码。以下就是本节的综合示例程序，经过详细注释的代码非常简单明了。

```
//----------------------【头文件、命名空间包含部分】----------------------
//         描述：包含程序所使用的头文件和命名空间
//-------------------------------------------------------------------------
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
using namespace cv;


int main()
{
    //-----------------------【一、图像的载入和显示】-----------------------
    //  描述：以下三行代码用于完成图像的载入和显示
    //-------------------------------------------------------------------

    Mat girl=imread("girl.jpg"); //载入图像到 Mat
    namedWindow("【1】动漫图"); //创建一个名为 "【1】动漫图"的窗口
    imshow("【1】动漫图",girl);//显示名为 "【1】动漫图"的窗口

    //-----------------------【二、初级图像混合】-----------------------
    //  描述：二、初级图像混合
    //-------------------------------------------------------------------
    //载入图片
    Mat image= imread("dota.jpg",199);
    Mat logo= imread("dota_logo.jpg");
```

```
    //载入后先显示
    namedWindow("【2】原画图");
    imshow("【2】原画图",image);

    namedWindow("【3】logo 图");
    imshow("【3】logo 图",logo);

    // 定义一个 Mat 类型，用于存放，图像的 ROI
    Mat imageROI;
    //方法一
    imageROI= image(Rect(800,350,logo.cols,logo.rows));
    //方法二
    //imageROI=
image(Range(350,350+logo.rows),Range(800,800+logo.cols));

    // 将 logo 加到原图上
    addWeighted(imageROI,0.5,logo,0.3,0.,imageROI);

    //显示结果
    namedWindow("【4】原画+logo 图");
    imshow("【4】原画+logo 图",image);

    //-------------------------【三、图像的输出】-------------------------
    //  描述：将一个 Mat 图像输出到图像文件
    //----------------------------------------------------------------
    //输出一张 jpg 图片到工程目录下
    imwrite("由 imwrite 生成的图片.jpg",image);

    waitKey();

    return 0;
}
```

运行这个程序，会弹出 4 个在 OpenCV 中创建的窗口。

下面是运行截图。首先是图像载入和显示的演示，我们载入了一张动漫人物图，如图 3.2 所示。

图 3.2 动漫显示图

接着是载入一张 dota2 原画（图 3.3）和 dota2 的 logo（图 3.4），为图像融合做准备。

图 3.3 dota2 原画图窗口

图 3.4 logo 图

最终，经过处理，得到 dota2 原画+logo 的融合，并输出一张名为“由 imwrite 生成的图片.jpg”的图片到工程目录下。如图 3.5 所示。

图 3.5 融合后的效果图

3.2 滑动条的创建和使用

滑动条（Trackbar）是 OpenCV 动态调节参数特别好用的一种工具，它依附于窗口而存在。

由于 OpenCV 中并没有实现按钮的功能，所以很多时候，我们还可以用仅含 0-1 的滑动条来实现按钮的按下、弹起效果。

3.2.1 创建滑动条：createTrackbar()函数

createTrackbar 函数用于创建一个可以调整数值的滑动条（常常也被称作轨迹条），并将滑动条附加到指定的窗口上，使用起来很方便。需要记住，它往往会和一个回调函数配合起来使用。先看下它的函数原型，如下。

```
C++: int createTrackbar(conststring& trackbarname, conststring& winn
ame,
int* value, int count, TrackbarCallback onChange=0,void* userdata=0);
```

- 第一个参数，const string&类型的 trackbarname，轨迹条的名字，用来代表我们创建的轨迹条。
- 第二个参数，const string&类型的 winname，窗口的名字，表示这个轨迹条会依附到哪个窗口上，即对应 namedWindow()创建窗口时填的某一个窗口名。
- 第三个参数，int* 类型的 value，一个指向整型的指针，表示滑块的位置。在创建时，滑块的初始位置就是该变量当前的值。
- 第四个参数，int 类型的 count，表示滑块可以达到的最大位置的值。滑块最小位置的值始终为 0。
- 第五个参数，TrackbarCallback 类型的 onChange，它有默认值 0。这是一个指向回调函数的指针，每次滑块位置改变时，这个函数都会进行回调。并且这个函数的原型必须为 void XXXX(int，void*);，其中第一个参数是轨迹条的位置，第二个参数是用户数据（看下面的第六个参数）。如果回调是 NULL 指针，则表示没有回调函数的调用，仅第三个参数 value 有变化。
- 第六个参数，void*类型的 userdata，也有默认值 0。这个参数是用户传给回调函数的数据，用来处理轨迹条事件。如果使用的第三个参数 value 实参是全局变量的话，完全可以不去管这个 userdata 参数。

createTrackbar 函数为我们创建了一个具有特定名称和范围的轨迹条（Trackbar，或者说是滑块范围控制工具），指定一个和轨迹条位置同步的变量，而且要指定回调函数 onChange（第五个参数），在轨迹条位置改变的时候来调用这个回调函数，并且，创建的轨迹条显示在指定的 winname（第二个参数）所代表的窗口上。

至于回调函数，就是一个通过函数指针调用的函数。如果我们把函数的指针（地址）作为参数传递给另一个函数，当这个指针被用来调用其所指向的函数时，就称其为回调函数。回调函数不由该函数的实现方直接调用，而是在特定的事件

或条件发生时由另外的一方调用，用于对该事件或条件进行响应。

在函数讲解之后，给大家一个 createTrackbar 函数使用的小例子作为参照。

```
//创建轨迹条
  createTrackbar("对比度：", "【效果图窗口】",&g_nContrastValue,
        300,on_Change);// g_nContrastValue 为全局的整型变量，on_Change 为
回调函数的函数名（在 C/C++中，函数名为指向函数地址的指针）
```

接着，我们一起来欣赏一个完整的使用示例，它演示了如何用轨迹条来控制两幅图像的 Alpha 混合。

```
#include <opencv2/opencv.hpp>
#include "opencv2/highgui/highgui.hpp"
using namespace cv;

#define WINDOW_NAME "【线性混合示例】"        //为窗口标题定义的宏

//-------------------------【全局变量声明部分】-------------------------
//         描述：全局变量声明
//-----------------------------------------------------------------------
const int g_nMaxAlphaValue = 100;//Alpha 值的最大值
int g_nAlphaValueSlider;//滑动条对应的变量
double g_dAlphaValue;
double g_dBetaValue;

//声明存储图像的变量
Mat g_srcImage1;
Mat g_srcImage2;
Mat g_dstImage;


//-----------------------【on_Trackbar()函数】-----------------------
//         描述：响应滑动条的回调函数
//-----------------------------------------------------------------------
void on_Trackbar( int, void* )
{
    //求出当前 alpha 值相对于最大值的比例
    g_dAlphaValue = (double) g_nAlphaValueSlider/g_nMaxAlphaValue ;
    //则 beta 值为 1 减去 alpha 值
    g_dBetaValue = ( 1.0 - g_dAlphaValue );

    //根据 alpha 和 beta 值进行线性混合
    addWeighted( g_srcImage1, g_dAlphaValue, g_srcImage2, g_dBetaValue,
0.0, g_dstImage);

    //显示效果图
    imshow( WINDOW_NAME, g_dstImage );
}

//-------------------------【main()函数】-------------------------
//         描述：控制台应用程序的入口函数，我们的程序从这里开始执行
//-----------------------------------------------------------------------
```

```
int main( int argc, char** argv )
{
    //加载图像 (两图像的尺寸需相同)
    g_srcImage1 = imread("1.jpg");
    g_srcImage2 = imread("2.jpg");
    if( !g_srcImage1.data ) { printf("读取第一幅图片错误，请确定目录下是否有 imread 函数指定图片存在～！ \n"); return -1; }
    if( !g_srcImage2.data ) { printf("读取第二幅图片错误，请确定目录下是否有 imread 函数指定图片存在～！ \n"); return -1; }

    //设置滑动条初值为 70
    g_nAlphaValueSlider = 70;

    //创建窗体
    namedWindow(WINDOW_NAME, 1);

    //在创建的窗体中创建一个滑动条控件
    char TrackbarName[50];
    sprintf( TrackbarName, "透明值 %d", g_nMaxAlphaValue );

    createTrackbar( TrackbarName, WINDOW_NAME, &g_nAlphaValueSlider, g_nMaxAlphaValue, on_Trackbar );

    //结果在回调函数中显示
    on_Trackbar( g_nAlphaValueSlider, 0 );

    //按任意键退出
    waitKey(0);

    return 0;
}
```

运行此程序，我们可以通过调节滑动条的位置，来得到不同的混合效果。如图 3.6、图 3.7、图 3.8 所示。

图 3.6 透明值为 70 时的效果图

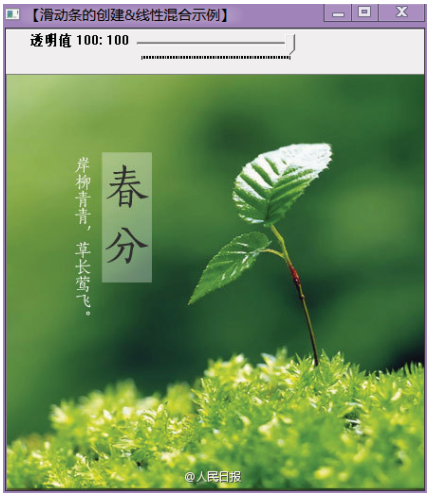

图 3.7 透明值为 100 时的效果图

图 3.8 透明值为 0 时的效果图

3.2.2 获取当前轨迹条的位置：getTrackbarPos()函数

本小节介绍一个配合 createTrackbar 使用的函数——getTrackbarPos()，它用于获取当前轨迹条的位置。

下面这个函数用于获取当前轨迹条的位置并返回。

```
C++: int getTrackbarPos(conststring& trackbarname, conststring& winn
ame);
```

- 第一个参数，const string&类型的 trackbarname，表示轨迹条的名字。
- 第二个参数，const string&类型的 winname，表示轨迹条的父窗口的名称。

3.3 鼠标操作

OpenCV 中的鼠标操作和滑动条的消息映射方式很类似，都是通过一个中介函数配合一个回调函数来实现的。创建和指定滑动条回调函数的函数为 createTrackbar，而指定鼠标操作消息回调函数的函数为 SetMouseCallback。下面一起来了解一下它。

```
SetMouseCallback 函数的作用是为指定的窗口设置鼠标回调函数，原型如下。
C++: void setMouseCallback(conststring& winname, MouseCallback onMouse,
void* userdata=0 )
```

- 第一个参数，const string&类型的 winname，窗口的名字。
- 第二个参数，MouseCallback 类型的 onMouse，指定窗口里每次鼠标时间发

生的时候，被调用的函数指针。这个函数的原型的大概形式为 void Foo(int event, int x, int y, int flags, void* param)。其中 event 是 EVENT_+变量之一，x 和 y 是鼠标指针在图像坐标系（需要注意，不是窗口坐标系）中的坐标值，flags 是 EVENT_FLAG 的组合，param 是用户定义的传递到 SetMouseCallback 函数调用的参数。如 EVENT_MOUSEMOVE 为鼠标移动消息、EVENT_LBUTTONDOWN 为鼠标左键按下消息等。

在 OpenCV2 中，上述“EVENT_”之前可以加上“CV_”前缀。

- 第三个参数，void*类型的 userdata，用户定义的传递到回调函数的参数，有默认值 0。

下面看一个详细注释的示例程序，在实战中了解此函数的用法以及如何在 OpenCV 中使用鼠标进行交互。

```
//----------------------【头文件、命名空间包含部分】----------------------
//         描述：包含程序所使用的头文件和命名空间
//-------------------------------------------------------------------
#include <opencv2/opencv.hpp>
using namespace cv;

#define WINDOW_NAME "【程序窗口】"        //为窗口标题定义的宏


//-------------------------【全局函数声明部分】-------------------------
//      描述：全局函数的声明
//-------------------------------------------------------------------
void on_MouseHandle(int event, int x, int y, int flags, void* param);
void DrawRectangle( cv::Mat& img, cv::Rect box );
void ShowHelpText();

//-------------------------【全局变量声明部分】-------------------------
//      描述：全局变量的声明
//-------------------------------------------------------------------
Rect g_rectangle;
bool g_bDrawingBox = false;//是否进行绘制
RNG g_rng(12345);


//---------------------------【main()函数】---------------------------
//      描述：控制台应用程序的入口函数，我们的程序从这里开始执行
//-------------------------------------------------------------------
int main( int argc, char** argv )
{

    //【1】准备参数
    g_rectangle = Rect(-1,-1,0,0);
    Mat srcImage(600, 800,CV_8UC3), tempImage;
    srcImage.copyTo(tempImage);
```

```
    g_rectangle = Rect(-1,-1,0,0);
    srcImage = Scalar::all(0);

    //【2】设置鼠标操作回调函数
    namedWindow( WINDOW_NAME );
    setMouseCallback(WINDOW_NAME,on_MouseHandle,(void*)&srcImage);

    //【3】程序主循环，当进行绘制的标识符为真时，进行绘制
    while(1)
    {
        srcImage.copyTo(tempImage);//复制源图到临时变量
        if( g_bDrawingBox ) DrawRectangle( tempImage, g_rectangle );//
当进行绘制的标识符为真，则进行绘制
        imshow( WINDOW_NAME, tempImage );
        if( waitKey( 10 ) == 27 ) break;//按下 ESC 键，程序退出
    }
    return 0;
}


//----------------------【on_MouseHandle()函数】----------------------
//      描述：鼠标回调函数，根据不同的鼠标事件进行不同的操作
//-----------------------------------------------------------------------
void on_MouseHandle(int event, int x, int y, int flags, void* param)
{

    Mat& image = *(cv::Mat*) param;
    switch( event)
    {
        //鼠标移动消息
        case EVENT_MOUSEMOVE:
         {
             if( g_bDrawingBox )//如果是否进行绘制的标识符为真，则记录下长和
宽到 RECT 型变量中
             {
                 g_rectangle.width = x-g_rectangle.x;
                 g_rectangle.height = y-g_rectangle.y;
             }
         }
              break;

        //左键按下消息
        case EVENT_LBUTTONDOWN:
         {
             g_bDrawingBox = true;
             g_rectangle =Rect( x, y, 0, 0 );//记录起始点
         }
             break;

        //左键抬起消息
        case EVENT_LBUTTONUP:
         {
```

```
                g_bDrawingBox = false;//置标识符为 false
                //对宽和高小于 0 的处理
                if( g_rectangle.width < 0 )
                {
                    g_rectangle.x += g_rectangle.width;
                    g_rectangle.width *= -1;
                }

                if( g_rectangle.height < 0 )
                {
                    g_rectangle.y += g_rectangle.height;
                    g_rectangle.height *= -1;
                }
                //调用函数进行绘制
                DrawRectangle( image, g_rectangle );
            }
                break;

    }
}


//-----------------------【DrawRectangle()函数】-----------------------
//      描述：自定义的矩形绘制函数
//-----------------------------------------------------------------------
void DrawRectangle( cv::Mat& img, cv::Rect box )
{
    rectangle(img,box.tl(),box.br(),Scalar(g_rng.uniform(0,255),
g_rng.uniform(0,255), g_rng.uniform(0,255)));//随机颜色
}
```

首先一起看看程序运行效果。我们可以通过鼠标左键的按下和松开来在黑色的窗口中绘制出一个一个彩色的矩形。如图 3.9、图 3.10 所示。

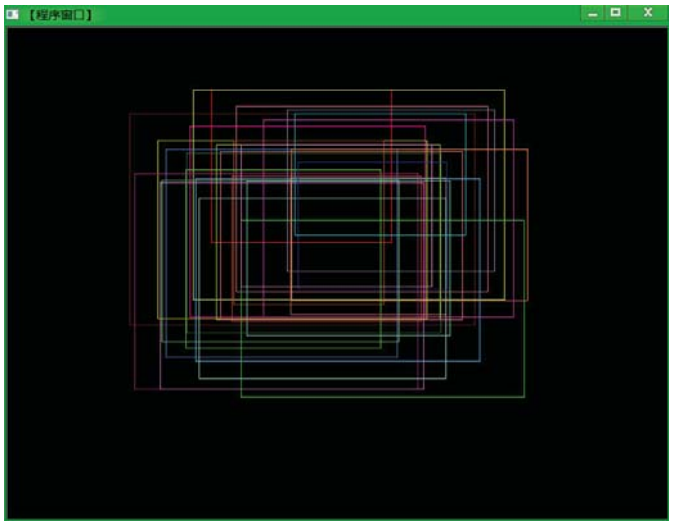

图 3.9 运行截图（1）

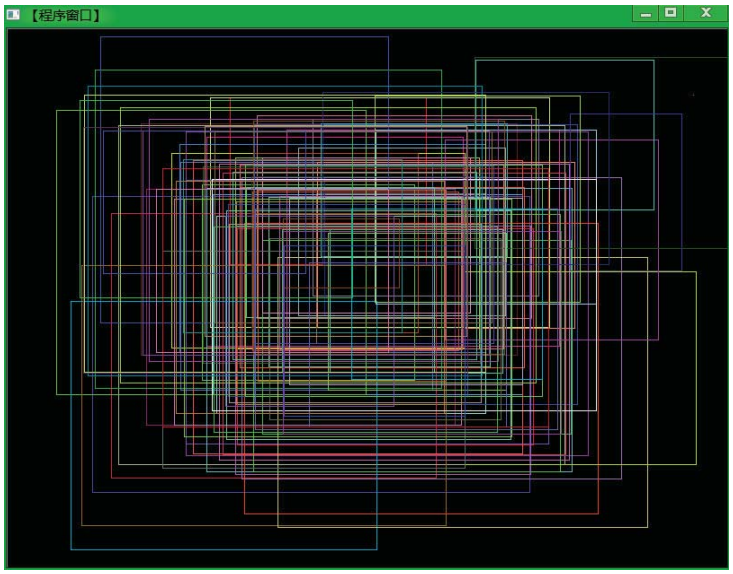

图 3.10 运行截图（2）

上述示例程序中的 on_MouseHandle 就是我们的鼠标消息回调函数。其中用一个 switch 语句指出了各种类型的鼠标消息，如鼠标移动消息 EVENT_MOUSEMOVE、左键按下消息 EVENT_LBUTTONDOWN、左键抬起消息 EVENT_LBUTTONUP，并对其进行了处理，以得到随机颜色的矩形绘制的功能。

3.4 本章小结

本章中，我们学习了 OpenCV 的高层 GUI 图形用户界面模块 highgui 中最重要的几个方面，分别是图像的载入、显示与输出图像到文件，以及如何使用滑动条如何进行鼠标操作。

本章核心函数清单

函数名称	用途	讲解章节
imread	用于读取文件中的图片到 OpenCV 中	3.1.4
imshow	在指定的窗口中显示一幅图像	3.1.5
namedWindow	用于创建一个窗口	3.1.7
imwrite	输出图像到文件	3.1.8
createTrackbar	用于创建一个可以调整数值的轨迹条	3.2.1
getTrackbarPos	用于获取轨迹条的当前位置	3.2.2
SetMouseCallback	为指定的窗口设置鼠标回调函数	3.3

本章示例程序清单

示例程序序号	程序说明	对应章节
15	用 imwrite 函数生成 png 透明图	3.1.8
16	综合示例程序：图像的载入、显示与输出	3.1.9
17	为程序界面添加滑动条	3.2.1
18	鼠标操作	3.3