《OpenCV3 编程入门》第三次勘误

2015.9

1

正文 P58 页。

将：

3.【写法三】返回值为 void 且不带参的 main 函数

```
int main()
{
   ……
   return 1;

}
```

修改为：

3 .【写法三】返回值为 void 且不带参的 main 函数

```
void main()
{
   ……
}
```

2

正文 P104 页。

将：

```
//【2.1】先绘制出椭圆
```

修改为：

```
//【2.1】先绘制出多边形
```

3

正文 P125 页。

将：

最后我们一起看一个示例。

```
Mat srcImage;
Mat imageROI;
vector<Mat> channels;
srcImage= cv::imread("dota.jpg");

// 把一个 3 通道图像转换成 3 个单通道图像
split(srcImage,channels);//分离色彩通道
      imageROI=channels.at(0);
      addWeighted(imageROI(Rect(385,250,logoImage.cols,logoImage.rows)),1.0,

logoImage,0.5,0.,imageROI(Rect(385,250,logoImage.cols,logoImage.rows)));

      merge(channels,srcImage4);

      namedWindow("sample");
      imshow("sample",srcImage);
```

将一个多通道数组分离成几个单通道数组的 split()函数的内容大概就是以上这些了，下面我们来看一下和它关系密切的 merge()函数。

替换为：

我们一起看一个示例。

```
vector<Mat> channels;
Mat imageBlueChannel;
Mat imageGreenChannel;
Mat imageRedChannel;
srcImage4= imread("dota.jpg");
// 把一个 3 通道图像转换成 3 个单通道图像
split(srcImage4,channels);//分离色彩通道
imageBlueChannel = channels.at(0);
imageGreenChannel = channels.at(1);
imageRedChannel = channels.at(2);
```

上面的代码先做了相关的类型声明，然后把载入的 3 通道图像转换成 3 个单通道图像，放到 vector<Mat>类型的 channels 中，接着进行引用赋值。

根据 OpenCV 的 BGR 色彩空间（Blue、Green、Red，蓝绿红），其中 channels.at（0）就表示引用取出 channels 中的蓝色分量，channels.at（1）就表示引用取出 channels 中的绿色色分量，channels.at

（2）就表示引用取出 channels 中的红色分量。

将一个多通道数组分离成几个单通道数组的 split()函数的内容大概就是以上这些了，下面我们来看一下和它关系密切的 merge()函数。

4

正文 P126 页。

将：

依然是一个示例，如下。

```
vector<Mat> channels;
Mat imageBlueChannel;
Mat imageGreenChannel;
Mat imageRedChannel;
srcImage4= imread("dota.jpg");
// 把一个 3 通道图像转换成 3 个单通道图像
split(srcImage4,channels);//分离色彩通道
imageBlueChannel = channels.at(0);
imageGreenChannel = channels.at(1);
imageRedChannel = channels.at(2);
```

上面的代码先做了相关的类型声明，然后把载入的 3 通道图像转换成 3 个单通道图像，放到 vector<Mat>类型的 channels 中，接着进行引用赋值。

根据 OpenCV 的 BGR 色彩空间（Bule、Green、Red，蓝绿红），其中 channels.at（0）就表示引用取出 channels 中的蓝色分量，channels.at（1）就表示引用取出 channels 中的绿色色分量，channels.at（2）就表示引用取出 channels 中的红色分量。

一对做相反操作的 split()函数和 merge()函数和用法就是这些。另外提一点，如果我们需要从多通道数组中提取出特定的单通道数组，或者说实现一些复杂的通道组合，可以使用 mixChannels()函数。

替换成：

依然是一个示例，综合了 split()函数和 merge()函数的使用，先将图像的通道进行拆分，再将通道合并。具体代码如下所示。

```
//定义一些 Mat 对象
Mat srcImage=imread("1.jpg");
Mat imageBlueChannel, imageGreenChannel, imageRedChannel,mergeImage;
 //定义 Mat 向量容器保存拆分后的数据
 vector<Mat> channels;
```

```
//通道的拆分
split(srcImage,channels);


//提取三色的通道的数据
imageBlueChannel = channels.at(0);
imageGreenChannel = channels.at(1);
imageRedChannel = channels.at(2);


//对拆分的通道数据合并
merge(channels,mergeImage);
//显示最终的合并效果
imshow("mergeImage",mergeImage);
```

一对做相反操作的 split()函数和 merge()函数和用法就是这些。另外提一点，如果我们需要从多通道数组中提取出特定的单通道数组，或者说实现一些复杂的通道组合，可以使用 mixChannels()函数。

5

正文 P132 页。

将：

这里的 a 也就是对比度，一般为了观察的效果，取值为 0.0 到 3.0 的浮点值，但是我们的轨迹条一般取值都会整数，所以在这里我们可以，将其代表对比度值的 nContrastValue 参数设为 0 到 300 之间的整型，在最后的式子中乘以一个 0.01，这样就可以完成轨迹条中 300 个不同取值的变化。

修改为：

这里的 a 也就是对比度，一般为了观察的效果，取值为 0.0 到 3.0 的浮点值，但是我们的轨迹条一般取值都会整数，所以在这里我们可以，将其代表对比度值的 g_nContrastValue 参数设为 0 到 300 之间的整型，在最后的式子中乘以一个 0.01，这样就可以完成轨迹条中 300 个不同取值的变化。

6

正文 P347 页。

如下公式中的 n=15 下标，修改为 n=16

$$[0,255]=[0,15]\cup[16,31]\cup\cdots\cup[240,255]$$
$$range=bin_1\cup bin_2\cup\cdots\cup bin_{n=15}$$