基于开源计算机视觉库 OpenCV 的图像处理

贾小军[1] 喻擎苍[2]
[1](嘉兴学院信息工程学院 浙江 嘉兴 314001)
[2](浙江理工大学信息电子学院 浙江 杭州 310018)

摘 要 讨论了 OpenCV(Open Source Computer Vision Library)相对于现有的计算机视觉软件包所具有的优势,描述了 OpenCV 的环境配置、数据定义、图像元素访问方式。OpenCV 成为一种源码开放、包含丰富的高级数学计算函数、图像处理函数和计算机视觉函数、不断更新和平台无关性的计算机视觉软件包。给出了两个实例,表明了其部分特性。

关键词 计算机视觉 OpenCV 图像处理

THE IMAGE PROCESSING BASED ON OPEN SOURCE COMPUTER VISION LIBRARY

Jia Xiaojun[1] Yu Qingcang[2]
[1](*School of Information Engineering, Jiaxing University, Jiaxing 314001, Zhejiang, China*)
[2](*School of Electrical and Information, Zhejiang Sci-Tech University, Hangzhou 310018, Zhejiang, China*)

Abstract The advantage of OpenCV over the existing computer vision library is discussed. The environment configuration, data definition and the access to image element of OpenCV are described. OpenCV has become an open source computer vision software package, which has advanced mathematical calculation function, image processing function and computer vision function. It is of ceaseless renewal and is platform independent. Two examples are demonstrated, and partial features of OpenCV are indicated.

Keywords Computer vison OpenCV Image processing

0 引 言

近年来,以图形、图像、视频等大容量为特征的图像数据处理广泛应用于医学、交通、工业自动化等领域。其最大特点是数据量大,有时要求实时处理。大多数图像处理软件包均用 C/C++来编写[1,2]。这些软件包为图像分析和机器视觉处理提供了极大的便利,但是却存在许多问题。如:大多数软件包只能作低层图像处理,无法实现诸如目标跟踪、摄像机标定、模式识别、三维重建、机器学习等高层处理;许多软件包作为商业出售而无法普及。

OpenCV 是 Intel 公司资助的开源计算机视觉库,与现有的计算机视觉库存相比,具有不可比拟的优势:

(1) 由一系列 C 函数和少量 C++类构成,包括 300 多个 C/C++函数;支持中、高层 API;可以使用外部库,也可以完全独立;为 IPP 提供了透明接口;对商业和非商业应用免费,用户可以进行二次开发;函数经过优化,执行效率高;

(2) 提供了丰富的图像及计算机视觉处理函数。提供了数组、列表、队列、集合、树等基本数据结构;包含了求矩阵积、特征值、奇异值、方程求解、特殊函数等众多高级数学计算函数;基本的图像处理,如滤波、边缘检测、形态操作等;高级图像视觉操作功能,如摄像机标定、运动分析、目标跟踪、模式识别、机器学习等等;

(3) 良好的跨平台性。支持 Windows、Linux、Unix 及 Mac OSX 操作系统;支持大多数 C/C++编译器,如:MSVC6.0、MSVC.NET2003、MSVC.NET2005 及 C++ BuilderX(简称 BCB)。可以轻易在不同平台之间进行移植;

(4) 更新速度快。2006 年 11 月发布了最新版本 OpenCV_1.0.exe,这是 Intel 推出的第一个正式版。扩充了大量函数;支持新编译器 GCC 4.X,支持 OpenMP;增加了新的大型视频监控模块,可以实现智能目标跟踪;增加了 ML(机器学习)类库;扩展了对 Python 的绑定。

本文给出 OpenCV 在各种编译环境下的配置及其语法结构,图像数据的存取方法,最后给出两个实例。

1 OpenCV 在几种编译器环境下的配置

使用 OpenCV,必须正确地配置其应用环境。不同的编译器,配置方法略有不同。下面给出了三种环境下的配置方法。

1.1 MSVC 6.0 环境下的配置

在 MSVC 6.0 环境下,启动 OpenCV 安装目录下的_make 文件夹中的 OpenCV.dsw 工程,用默认的 Win32 Debug 模式 Build 工程,将自动生成 Lib 及 Dll 文件,如 cvd.lib/highgyid.lib 及 cvd.dll/highguid.dll 等,将其拷贝到 Lib 目录下供以后使用。

要有效使用链接库,还需在程序中加入对应的头文件,如 cv.h/highgui.h,并且在工程中加入对应的链接文件及其它链接文件。打开工程环境下的菜单 Project,选择 Settings 项。在 C/C

收稿日期:2007-04-11。国家自然科学基金项目(50575211)。贾小军,讲师,主研领域:计算机视觉与数字图像处理。

++标签项中加入头文件所在的相对路径或绝对路径,如…\cv\include;在 Link 标签下加入 Lib 文件所在的路径及文件名,多个文件间以空格分开,如 cvd. lib highguid. lib 等;通过 Insert Project into Workspace 对话框加入 OpenCV 的现有工程,如 cv. dsp 等,即可应用 OpenCV。

1.2 MSVC. NET 2003 及 MSVC. NET 2005 环境下的配置

OpenCV 在 MSVC. NET 2003 及 MSVC. NET 2005 环境下的配置几乎完全相同。打开. NET 应用环境,选择菜单 Tools 中 Options,在打开的对话框中进行如下的设置:选择 Projects 中的 VC++ Directories,在 List Box 中添加库文件所在的路径、头文件所在的路径及源文件路径即可。在工程中的 Link 添加所需要的库文件,如 cv. lib/cvd. lib 等。经过上述的配置,可以在项目中使用 OpenCV 中的相关函数。

1.3 C++BuilderX 环境下的配置

OpenCV 的最新版本中取消了对 C++ BuilderX 的支持,但是仍然有一个脚本文件,位于 utils\gen_make. py,可以通过生成文件 makefile. bcc's 来编译实现。这里介绍,利用转换 OpenCV 静态库的方法实现其配置。OpenCV 的库在 MSVC 环境下可直接使用。然而,在 BCB 下必须进行转换,否则无法使用。

首先需将 OpenÇV 静态库文件进行转换。利用 BCB 安装目录 BIN 中的文件 coff2omf. exe 进行转换。还可以用另外两个工具 impdef. exe 或 implib. exe 进行转换。

其次在 BCB 下进行配置。选择菜单 Project 中的 Options,在对话框中选择 Directories/Conditionals 标签,在 Include Path 中添加 OpenCV 的头文件所在的路径。同时在 Library Path 中添加链接文件的路径。

最后在 BCB 工程中加入 OpenCV 的库文件。启动 Project Manager,选择 ADD,进行库文件添加。

虽然 OpenCV 在 BCB 下配置略嫌麻烦,但是 BCB 具有最成熟的图形界面可视化、功能强大的应用组件 VCL 的开发环境,对于有界面要求的图形开发者是个不错的选择。

2 OpenCV 函数及数据表示方法

OpenCV 函数命名方式比较简单,数据类型定义直观。

2.1 OpenCV 函数命名方式

通常,OpenCV 采用以下格式进行函数的命名:

cvActionTargetMod(Parameters)

cv 在每个函数名前都存在;Action 表示函数的核心功能,通常用一个英语动词表示,如:Set,Create;Target 表示目标图像区域,如轮廓 Contour,多边形 Polygon;Mod 为可选项,通常辅助说明操作的对象,如操作类型;Parameters 表示参数列表,不同的函数其参数个数及类型不完全相同。

2.2 矩阵数据表示方法

OpenCV 表示矩阵数据类型时,通常采用如下的形式:

CV_ < bit-depth > (S|U|F)C < number-of-channels >

bit-depth 表示数据长度,可以取值 8、16、32、64 位;S 表示带符号整型数据;U 表示无符号整型数据;F 表示实数型数据;number-of-channels 表示数据通道数,可以取值 1,2,3,4。

如:CV_8UC1 表示无符号 8 位单通道数据类型;CV_32FC2 表示 32 位双通道实数型数据。

2.3 图像数据表示方法

OpenCV 表示图像数据类型时,通常采用如下的形式:

IPL_DEPTH_ < bit-depth > (S|U|F)

如:IPL_EPTH_8U 指 8 位无符号整型数据,IPL_DEPTH_32F 指 32 位实数型数据。

3 OpenCV 数据结构及图像处理

OpenCV 提供了大量的动态数据结构,如二维点、三维点结构,列表、队列、集合、树、图、矩阵、视频、摄像机标定等数据结构。针对图像数据,提出了 IplImage 数据结构,其各元素的含义及应用可以参考文献[5]。

OpenCV 具有强大的图像处理功能。正确处理图像取决于对图像元素的合理访问;并且不同的访问方式,其执行效率不同。在实际应用过程中,许多使用者在存取图像元素采用单一、固定的格式。图像元素的存取具有较大的灵活性。

3.1 间接存取方式

间接存取方式适用于各种类型的图像,但是效率低下。假设图像定义为 IplImage * img。不同类型的图像元素存取方式为:

(1) 对 8 位单通道图像,像素 I(i,j)存取操作表示为:

```
CvScalar s = cvGet2D(img,i,j);                    //获取 I(i,j)像素值
s.val[0] = 120;cvSet2D(img,i,j,s);                //设置 I(i,j)像素值
```

(2) 对 8 位 3 通道图像,像素 I(i,j)存取操作表示为:

```
CvScalar s = cvGet2D(img,i,j); // 获取 I(i,j)像素,分别为 Blue
        //分量 s.val[0],Green 分量 s.val[1],Red 分量 s.val[2]
s.val[0] = 120;s.val[1] = 120;s.val[2] = 120;cvSet2D(img,i,j,s);
                                                  //设置 I(i,j)像素值
```

3.2 直接存取方式

这是一种最常用的、高效率存取方式。假设图像定义为 IplImage * img,不同类型的图像元素存取方式为:

(1) 对 8 位单通道图像,像素 I(i,j)存取操作表示为:

```
I(i,j) ~ ((uchar *)(img -> imageData + i * img -> widthStep))[j];
```

(2) 对 8 位 3 通道图像,像素 I(i,j)存取操作表示为:

```
I(i,j)B ~ ((uchar *)(img -> imageData + i * img -> widthStep))
[j * img -> nChannels + 0];                       //Blue 分量
I(i,j)G ~ ((uchar *)(img -> imageData + i * img -> widthStep))[j
* img -> nChannels + 1];                          //Green 分量
I(i,j)R ~ ((uchar *)(img -> imageData + i * img -> widthStep))[j
* img -> nChannels + 2];                          //Red 分量
```

3.3 带指针直接存取方式

存取方式简单,高效。假设图像定义为 IplImage * img,不同类型的图像元素存取方式为:

(1) 对 8 位单通道图像,像素 I(i,j)存取操作表示为:

```
int step = img -> widthStep/sizeof(uchar);
uchar * data = (uchar *) img -> imageData;
I(i,j) ~ data[i * step + j];
```

(2) 对 8 位 3 通道图像,像素 I(i,j)存取操作表示为:

```
int step = img -> widthStep/sizeof(uchar);
int channels = img -> nChannels;
uchar * data = (uchar *) img -> imageData;
I(i,j)B ~ data[i * step + j * channels + 0];      //Blue 分量
I(i,j)G ~ data[i * step + j * channels + 1];      //Green 分量
I(i,j)R ~ data[i * step + j * channels + 2];      //Red 分量
```

4 应用实例

以两个简单的实例说明 OpenCV 在图像处理方面的强大功能及其在图像处理中的应用。两个实例均在 Borland C ++ Builder 6.0 环境下实现。

4.1 实例 1

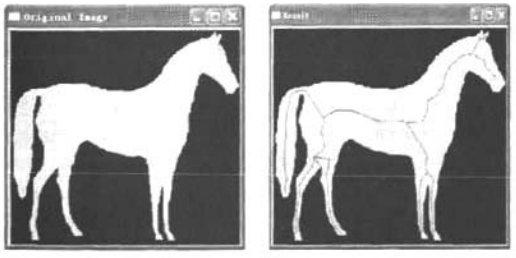
利用 OpenCV 实现图像的边缘提取。按照前面介绍的方法配置 OpenCV 应用环境,同时在程序的前面加上头文件 cv.h 及 highgui.h。该例简单,边缘检测可以调用函数 cvCanny,通过 cvLoadImage 调入图像及 cvShowImage 显示处理结果。原始的灰度图像及处理结果如图 1 所示。

(a) 灰度图像 (b)边缘检测结果

图 1 边缘检测实例

4.2 实例 2

本例展示了在任意多边形中实现提取骨架/中轴线的方法。提取骨架的方法主要有内切圆法、轮廓线法线相交法及 delaunay 三角形法。利用 OpenCV 提供的 cvFindCountours 函数寻找轮廓,cvApproxPoly 逼近多边形曲线,同时使用了大量的 OpenCV 数据结构进行数据表示。该例用 OpenCV 来实现,简单及执行速度快。原始图像及处理结果如图 2 所示。



(a) 原始图像 (b) 骨架提取结果

图 2 骨架提取实例

5 结 论

OpenCV 充分利用 C/C ++ 运行效率高的特点,开发了大量的几乎遍及所有的图像及计算机视觉处理函数。由于其代码完全开放,用户不但可以对源代码进行修改,加入新类,而且可以通过查看函数内的源代码来理解图像处理中很多经典算法的原理及实现过程,这将对从事图像的工作者提供有益的帮助。而且 OpenCV 操作方便,不但可以作为应用程序的后台处理程序,而且可以作为控制台程序进行操作。它在图像处理及计算机视觉的各个领域中具有广阔的应用前景。

参 考 文 献

[1] Yu Qingcang, Cheng Harry H, Cheng Wayne W, et al. CH OpenCV for Interactive Open Architecture Computer Vision[J]. Advances in Engineering Software, 2004, 35(9): 527 - 536.

[2] Yu Qingcang, Cheng Harry H, Cheng Wayne W, et al. Interactive Open Architecture Computer Vision[C]. 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03), Sacramento, California, USA, 11, 2003: 406 - 410.

[3] 黎松,平西建,丁益洪. 开放源代码的计算机视觉类库 OpenCV 的应用[J]. 计算机应用与软件,2005,22(8):134 - 136.

[4] 吕学刚,于明,刘翠响. 数字图像处理与计算机视觉编程的有力工具—IPL 和 OpenCV[J]. 现代计算机,2002,147:69 - 71.

[5] Intel® Open Source Computer Vision Librar Reference Manuals. 2003.

(上接第 247 页)

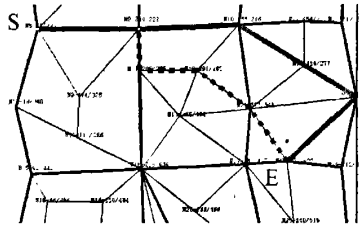


图 4 转变对最短距离的影响

3 结 论

以上是对 GIS 领域的最短路径计算所作的三方面的优化,可以看到它在虚拟地图的模拟计算中取得了相当好的效果。

在实际使用中,GIS 的地图可以以统一的格式分布在不同地区的网上。到不同的地区,就到指定的网站去搜索地图信息并进行计算。

最短路径的计算还有从起点出发要经过几个送货点如何走最有效,又如设计旅行的最短路线。此外,最短路径的计算除了平面以外还有在三维空间中的计算。如消防员如何从底楼最快地到达大楼中的着火地点。这些问题都是值得探讨的课题。

参 考 文 献

[1] 夏宽理. 算法基础. 高等教育出版社,2003.

[2] Car A, Taylor G, Brunsdon C. An analysis of the performance of a hierarchical wayfinding computational model using synthetic graphs Computers. Environment and Urban Systems, 2001, 25: 69 - 88.

[3] Tarantilis C D, et al. Combination of geographical information system and efficient routing algorithms for real life distribution operations. European Journal of Operational Research, 2004, 152: 437 - 453.

[4] Chris Upchurch, et al. Using GIS to generate mutually exclusive service areas linking travel on and off a network. Journal of Transport Geography, 2004, 12: 23 - 33.

[5] ZhongRen Peng, et al. Design and development of interactive trip planning for web-based transit information systems. Transportation Research Part C, 2000, 8: 409 - 425.