VC++与Matlab混合编程及其在轮辋裂纹检测中的应用

刘亚楠,郭三华,涂铮铮,罗 斌
(安徽大学 计算智能与信号处理教育部重点实验室,安徽 合肥 230039)

摘 要:文中结合轮辋裂纹检测系统介绍了几种VC++与Matlab混合编程的方式,并具体分析了它们在轮辋裂纹检测中的可行性,得出了在轮辋裂纹检测中可以采用Matlab引擎的结论。最后结合边缘检测、形态学运算、最优阈值分割等多种方法,采用Matlab引擎较好地实现了轮辋裂纹检测,取得了理想的效果。实验表明该方法在轮辋裂纹检测系统的开发中大大提高了编程效率。

关键词:混合编程;裂纹检测;Matlab引擎
中图分类号:TP391.41 **文献标识码**:A **文章编号**:1673-629X(2007)07-0233-03

Mixed Programming of VC++ with Matlab and Its Application in Wheel Crack Detection System

LIU Ya-nan, GUO San-hua, TU Zheng-zheng, LUO Bin
(Ministry of Education Key Lab. of Intelligent Computing and Signal Processing, Anhui University, Hefei 230039, China)

Abstract: Introduces several mixed programming methods of VC++ and Matlab based on the wheel crack detection system, and concretely analyses the feasibility of them in wheel crack detection system, eliciting the conclusion of using Matlab engine in crack detection system. To realize the wheel crack detection system, use the method of Matlab engine combined with edge detection algorithm, morphological operation, optimal threshold segmentation, etc, and get the perfect effect. The experiments indicate that the programming efficiency is improved in the development of the wheel crack detection system using the proposed method.

Key words: mixed programming; crack detection; Matlab engine

0 引 言

Matlab是当今最优秀的科技应用软件之一,它具有高效的科学计算功能与可视化功能,又有强大的图像处理功能。它简单易用,不需要用户有高深的数学知识和程序设计能力,也不需要深刻了解算法及编程技巧,具有开放式可扩展的工作环境。Matlab语句功能十分强大,一条语句可完成十分复杂的任务。Matlab软件中所包含的Matlab源代码相当于70万行C代码。特别是所附带的30多种面向不同领域的工具箱支持,使它成为许多科学领域的基本工具和首选平台。但是,它是以解释方式运行的高级语言,执行效率低;同时,Matlab程序不能脱离其环境运行,不能被用于开发商用软件。而对于另一种程序设计工具VC++6.0,它具有较高的编码效率,可以快速地开发出Windows环境下图形界面丰富的应用软件系统,但是,在进行较复杂的图像处理时,需要编出大量的代码。文中通过Matlab与VC++语言的接口,可以实现两种语言的混合编程,优势互补,提高编程效率,会大大地加快一些算法的实现,同时其可靠性也很高,可以开发出高质量的图像处理软件。

轮辋裂纹检测系统采用数字图像处理技术,利用CCD摄像头摄取车轮轮辋的运动图像,将其数字化后送往计算机处理,提取裂纹的特征,实现车轮轮辋表面裂纹的在线检测。如图1所示。

依据裂纹检测相关算法,采用Matlab与VC++语言混合编程来进行轮辋裂纹检测系统的开发,Matlab与VC++混合编程存在以下几种主要方式,根据这几种方式的各自特点来分析在轮辋裂纹检测系统中应该采用哪种方式来实现处理。

收稿日期:2006-09-17
作者简介:刘亚楠(1984-),女,山东济宁人,硕士研究生,研究方向为数字图像处理;罗 斌,教授,博士生导师,研究方向为数字图像处理与模式识别。

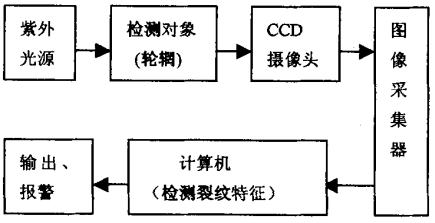

图 1 系统原理框图

1 VC++与 Matlab 混合编程的几种方法

混合编程的方法描述如下[1,2]:

(1)利用 Matlab 编译器实现与 VC 的连接。

这种方法需要编辑一个.m 文件,并且它必须是一个函数文件,而不能是一个脚本文件。假设为 lyn.m。利用 Matlab 提供的 mcc 编译器命令将.m 文件转化为 VC++下可以调用的动态链接库文件。这里需要注意的是由于 Matlab 6.5 下的编译器版本是 3.0 的,而在 Matlab 7.0 下编译器版本是 4.1 的,所以,所用编译器命令不一样,并且生成的文件也不一样。具体区别如下:

在 Matlab 6.5 中 mcc 编译器命令为:

mcc -t -W libhg:lyn03 -T link:lib -h libmmfile.mlib lyn.m

生成的文件中包含 lyn.c lyn.h 及.m 文件中调用的函数所对应的头文件和源文件。

所要调用的函数定义如下:

```
extern mxArray * mlfLyn(void);
extern void lyn03Initialize(void);
extern void lyn03Terminate(void);
```

在 Matlab7.0 中 mcc 编译器命令为:

```
mcc -W lib:lyn03 -T link:lib lyn.m
```

生成的文件中已经不再包含 lyn.c lyn.h。而且,对于所有.m 文件都只生成 8 个文件。.m 文件内所包含的函数对应的源文件和头文件都被封装起来,大大节省了存储空间。

所要调用的函数定义也有很大的不同,主要表现在 mlfLyn()函数上,函数定义如下:

extern void mlfLyn(int nargout,mxArray * * y);

其中 nargout 为所有函数的输出变量数目,y 为输出变量。对于一般只涉及到数学库函数的.m 文件只调用这三个函数即可。但是,对于包含图像处理工具箱中的函数的.m 文件,在 Matlab6.5 下是不能编译的。故在轮辋裂纹检测系统中不能采用这种方式。

(2)利用 matcom 实现与 VC 的连接。

有两种方法可以实现这种连接:运行 MIDEVA,在主界面上直接打开.m 文件,在菜单中选择 compile to dll,输入 lyn,找到 lyn.cpp lyn.h 加入工程[3];点击 VC 菜单项 Tools->Customize。在弹出的对话框中选择 Add-ins and Macro files,点击 Browse,选择插入文件类型为.dll 文件,在 Matcom 的 bin 目录下找到 mvcide.dll,点击 OK,VC 的工具栏上就会出现新的工具栏[4]。采用第二种方法点击工具栏上的按钮就可以把.m 文件以及与它相关函数的的源文件、头文件、库文件 v4501v.lib 自动加入工程中,非常方便。

Matcom 提供了 matlab 中.m 文件与其他高级语言的接口,使.m 文件可以编译为脱离 Matlab 环境独立执行的可执行性程序,这样提高了代码的复用率,提高了代码的执行速度。使纯文本的.m 文件变为二进制的可执行程序,增加了知识保护的安全性。它提供了近千个数学函数,对于其他高级语言编译器来说,提供了一个丰富的数学库,在 Matlab 上能用的常用函数基本上都可以在高级语言中直接调用。所以对于涉及数学函数的.m 文件来说,采用这种接口方法很方便。但是需要注意的是由于在.m 文件中所调用的函数文件也要与该.m 文件在同一目录下,同时进行编译。而对于轮辋裂纹检测系统,它的算法中调用的函数文件有很多,要把这些函数都放在同一个目录下需要一定的时间,另外,Matlab 中的图形、图像显示函数不能被调用,需要通过 VC 编程实现,所以,这种方法在轮辋裂纹检测系统中并不方便实现。

(3)用 Matlab 引擎调用 Matlab 函数。

Matlab 引擎采用客户/服务器计算方式。在运用中,C++语言的程序作为前端客户机,向 Matlab 引擎传递命令和数据信息,并从 Matlab 引擎接收数据信息。这种方式下,Matlab 相当于一个计算引擎。它以独立进程的形式工作在后台,有以下优点[3]:

①Matlab 引擎可以与应用程序运行在网络中不同的机器上,这样就能够将计算任务繁重的引擎程序放置到网络上计算速度较快、计算能力较强的机器上,充分利用网络资源,加快这个系统的速度,在另外的机器上实现应用程序的用户界面。

②由于应用程序和引擎是两个独立的进程,Matlab 引擎不需要将整个庞大的 Matlab 系统与程序连接,只需要把小部分的引擎通信库与程序相连,节省了大量的资源。

另外,和以上所介绍的接口方式相比较来说 Matlab 引擎所提供的 Matlab 功能最全面,可以完成任何计算和图像处理工作,而且利用 Matlab 引擎调用工具箱中的函数可节省大量的系统资源,虽然不可脱离 Matlab 的环境运行,但是应用程序整体性能较好并且它的设置很简单,只需要在源文件中加入 # include

"engine.h"，把 libeng.lib libmx.lib 引入工程即可。所以在轮辋裂纹检测系统中这种方法可行。下面就用这种方式来实现轮辋裂纹检测系统。

2　实例分析

文中基于 VC++6.0 和 Matlab7.0 利用 Matlab 引擎实现对轮辋裂纹检测。裂纹检测方法是：首先利用 Canny 算子检测图像边缘，从而可以提取连续而完整的边缘，然后用形态学的二值膨胀消除双边缘之间的间隙，再用腐蚀、细化算法平滑、细化粗边缘，使得边缘在宽度上更接近原始目标，得到第一个处理结果图；对原图进行傅里叶变换处理，再用最优阈值分割得到第二个处理结果图[5]；然后将两个结果图做点乘运算，最后使用模板去除残留的噪点[6]，得到最终的裂纹检测结果。

Matlab 实验中所用文件：tanshang.m。

在把 libeng.lib libmx.lib 引入工程时可以采用方法：在 C:\MATLAB701\extern\lib\win32\microsoft\msvc60 目录下找到 libeng.lib libmx.lib 引入工程。也可以采用文献[3]中的方法：利用命令 lib/def:＜Matlab 的安装路径＞C:\MATLAB701\extern\include*.def/machine:ix86/out:*.lib 来生成程序所需的静态链接库 libeng.lib libmx.lib，将 libeng.lib libmx.lib 所在的目录加入 VC++project/link/object/library modules 目录下即可。文中采用第一种方法。

主要代码如下：

```
Engine *ep;
if (!(ep=engOpen(NULL)))
{
  fprintf(stderr,"\n Can't start MATLAB engine\n");
  exit(-1);
}
engEvalString(ep,"a=imread('tansh.bmp');");
engEvalString(ep,"x=mat2gray(a);");
engEvalString(ep,"[M,N]=size(x);");
engEvalString(ep,"x=fenge(x,M,N);");
engEvalString(ep,"x=quban(x,M,N);");
engEvalString(ep,"imwrite(x,'kk1.bmp');");
engEvalString(ep,"BW1=edge(a,'canny',0.5);");
engEvalString(ep,"se1=strel('line',4,90);");
engEvalString(ep,"se2=strel('line',4,0);");
engEvalString(ep,"BWsdil=imdilate(BW1,[se1 se2]);");
engEvalString(ep,"y=imread('kk1.bmp');");
engEvalString(ep,"z=y&BWsdil;");
engEvalString(ep,"z=qugudian(z,M,N);");
engEvalString(ep,"figure;");
engEvalString(ep,"imshow(z)");
```

其中 fenge()，quban()，qugudian()为自定义的函数，分别实现分割、去斑、去孤点算法。函数定义如下：

Function x=fenge(*x*,*M*,*N*)

Function x=quban(*x*,*M*,*N*)

Function z=qugudian(*z*,*M*,*N*)

以上可以看出需要在 VC 下编很多代码才能实现的算法，用引擎就只需要几行，另外，也可以把整个 tanshang.m 文件封装成一个函数文件 function z=tanshang()，在 lyn.m 中直接调用，上述代码就减为一条：

```
engEvalString(ep,"z=tansh()");
```

这样更能看出使用 Matlab 引擎的优势，节省了在 VC 下编码的时间，提高了工作效率。

实验结果如图 2、图 3 所示。

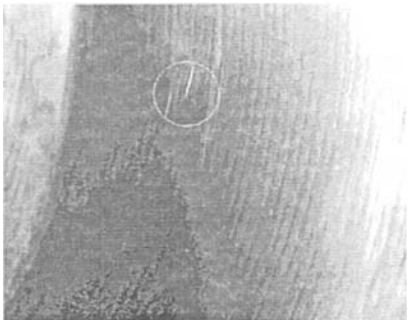

图 2　原始图像

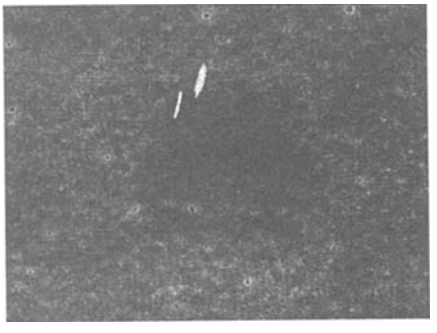

图 3　提取的裂纹

3　结束语

VC++和 Matlab 混合编程，弥补了单个软件开发应用程序功能上的不足，是一种时间、效率上的极好的折衷，它使得人们可以将更多的时间放在项目关键问题的思考上，而不是放在算法的实现和程序的编码中。

通过上述实例很好地应用了 VC++和 Matlab 的混合编程，实现了轮辋裂纹检测算法，在实际应用中取得了理想的效果。

（下转第 239 页）

系统正常模式下的关联规则集 S_1，然后挖掘系统在某暂态模式下的关联规则集 S_2,再计算两规则集的相似度 similarity(S_1,S_2),即用相似度来表示系统当前状态与正常状态的背离程度,以确定系统是否处于异常状态。

给定两个关联规则 $R_1: X \rightarrow Y, c, s, R_2: X' \rightarrow Y', c', s'$。如果 $X = X'$ 且 $Y = Y'$,两规则之间的相似度 similarity(R_1,R_2) = max[0,1, − max[| $c - c'$ | /c, | $s - s'$ | /s]], 两规则集 S_1、S_2 的相似度为 similarity(S_1,S_2) = μ^2/ | S_1 ‖ S_2 |, 其中,μ = $\sum$similarity(R_1,R_2), | S_1 ‖ S_2 | 为规则的数量。

而在滥用检测中运用模糊关联思路是将挖掘出的规则进行与已知匹配,对入侵行为进行识别。

在表 1 算例中网络流数据通过 F − 均值聚类方法求出隶属函数 F_k,然后转换为模糊数据。

取 $\lambda = 0.5$,minsup = 0.25,表 1 得到 L_1 = ({$A.L$},{$A.H$},{$B.L$},{$C.H$},{$D.H$}),然后按序连接通过扫描频繁项目的哈希链表得到 L_2,结果如表 2。

表 1　模糊数据集

	Ptcp		Pudp		Avgpacket/s		AvgMibt/s	
Tid	$A.L$	$A.H$	$B.L$	$B.H$	$C.L$	$C.H$	$D.L$	$D.H$
1	0	0.90	1	0	0	0.65	0	1
2	0	0.89	1	0	0	0.73	0	1
3	0	0.57	1	0	0	1	0	1
4	0	0.93	0	1	0	0.72	0	0.86
5	0	0.86	0.5	0	0.89	0	1	0
6	0	0	0	0.5	0	0.62	0	1
7	0	1	0.5	0	0	0.91	0	1
8	1	0	1	0	0	1	0	1
9	0.64	0	1	0	0.63	0	1	0
10	1	0	1	0	0	1	0	0.89
sup	0.26	0.51	0.7	0.15	0.15	0.66	0.2	0.78

表 2　等价类及模糊关联规则的计算

L_2	Sup
$A.H,B.L$	0.34
$A.H,C.H$	0.36
$A.H,D.H$	0.42
$B.L,C.H$	0.49
$B.L,D.H$	0.54
$C.H,D.H$	0.65

→

L_3	Sup
$A.H,B.L,C.H$	0.25
$A.H.B.L,D.H$	0.29
$A.H,C.H,D.H$	0.36
$B.L,C.H,D.H$	0.48

由表 2 中 L_2/R 得到等价类为:

$[A.H]$ = {$B.L$,$C.H$,$D.H$},$[B.L]$ = {$C.H$,$D.H$},$[C.H]$ = {$D.H$},$[A.H,B.L]$ = $[A.H] \cap [B.L]$ = {$C.H$,$D.H$},$[A.H,C.H]$ = $[A.H] \cap [C.H]$ = {$D.H$},$[B.L,C.H]$ = $[B.L] \cap [C.H]$ = {$D.H$},C_3({$A.H,B.L,C.H$},{$A.H,B.L,D.H$},{$A.H,C.H,D.H$},{$B.L,C.H,D.H$})。同理,可求 C_4。

取 minconf = 0.60 可挖掘出如下强模糊规则:

$A.H \wedge B.L \rightarrow C.H$　$s = 25\%$　$c = 74\%$

$A.H \wedge B.L \rightarrow D.H$　$s = 29\%$　$c = 85\%$

$A.H \wedge D.H \rightarrow B.L$　$s = 29\%$　$c = 70\%$ ……

挖掘出 $A.H \wedge B.L \rightarrow D.H$ 即"网络流中 TCP 包数量为高,且 UDP 包为低,则每秒平均数据位为高。"

5　结束语

文中运用模糊集理论中的截运算,采用 Hash 链表结构实现快速定位查找,减少了支持度的重复计算,且通过等价类优化算法避免了多余模式匹配。该算法的不足之处是新建哈希链表本身增加了一定的空间和时间开销。下一步工作是通过时间序列的模糊关联挖掘来识别入侵事件。

参考文献:

[1] Han Jiawei, Kamber M. 数据挖掘:概念与技术[M]. 范明,孟小峰译. 北京:机械工业出版社,2002.

[2] 张保稳,何华灿. 有效支持度和模糊关联规则挖掘[J]. 小型微型计算机系统,2002(9):1004 − 1006.

[3] Park J S, Chen M S, Yu P S. An effective hash − based algorithm for mining association rules[C]// In Proc. 1995 ACM Int. Conf. Management of Data. San Jose, CA: [s. n.], 1995:175 − 186.

[4] 王　翔,袁兆山. 基于等价类和最大完全图集聚类的关联规则发现算法[J]. 小型微型计算机系统,2000(6):614 − 616.

[5] 孙建勋,陈绵云,张曙红. 用模糊方法挖掘量化关联规则[J]. 计算机工程与应用,2003(18):190 − 192.

(上接第 235 页)

参考文献:

[1] 刘志俭. MATLAB 应用程序接口用户指南[M]. 北京:科学出版社,2000.

[2] 苏金明,黄国明,刘　波. MATLAB 与外部程序接口[M]. 北京:电子工业出版社,2004.

[3] 飞思科技产品研发中心. MATLAB7 基础与提高[M]. 北京:电子工业出版社,2005.

[4] 张友兵,田漫柳. 基于 Matcom 与 VC 混合编程的数字图像处理方法研究[J]. 湖北汽车工业学院学报,2005,19(1):38 − 41.

[5] Sonka M, Hlavac V, Boyle R. Image Processing, Analysis, and Machine Vision[M]. Second Edition. United States of America: Thomson Learning and PT Press, 2003.

[6] 周　军,彭培欣. 自动磁粉探伤系统中的图像技术[J]. 仪器仪表学报,2003,24(4):461 − 462.