文章编号:2095-6134(2015)01-0116-05

基于 VC++2008 的大容量数据曲线绘制方法*

马志刚[1,2], 刘文怡[1†]

(1 中北大学 仪器科学与动态测试教育部重点实验室, 电子测试技术重点实验室, 太原 030051;
2 山西农业大学信息科学与工程学院, 山西 太谷 030801)

(2014 年 3 月 19 日收稿; 2014 年 4 月 25 日收修改稿)

Ma Z G, Liu W Y. Curve plotting method for large-capacity data based on VC++2008[J]. Journal of University of Chinese Academy of Sciences, 2015,32(1):116-120.

摘 要 针对大容量数据曲线绘制中的数据提取问题,提出"逐块读取"→"拼接数据"→"提取数据",最后达到"完整提取"的方式,可以实现对超过 4 GB 数据文件的处理. 当数据量较大时,常规的绘图方法会暴露出效率低、绘图区域闪烁的问题. 利用 Polyline 函数优化绘图程序,提高了绘图效率;采用双缓冲法解决绘图区域闪烁的问题,并进一步缩短了绘图时间.

关键词 大容量数据; 曲线绘制; VC++ 2008; Polyline; 双缓冲

中图分类号:TP391 **文献标志码**:A **doi**:10.7523/j.issn.2095-6134.2015.01.019

Curve plotting method for large-capacity data based on VC++2008

MA Zhigang[1,2], LIU Wenyi[1]

(1 *Key Laboratory of Instrumentation Science & Dynamic Measurement of Ministry of Education, Science and Technology on Electronic Test & Measurement Laboratory, North University of China, Taiyuan* 030051, *China*;
2 *College of Information Science and Engineering, ShanXi Agricultural University, Taigu* 030801, *Shanxi, China*)

Abstract In view of the data extraction problems in large-capacity data curve plotting, the "data reading by-block" → "data join" → "data extraction" → achieving finally "full extraction" process is proposed, which can achieve the processing of the large-capacity data file over 4 GB. When the data quantity is large, conventional curve plotting methods may give rise to the low efficiency or the flicker problem in the plotting area. The "Polyline" function is employed to optimize the plotting program, and greatly improves the plotting efficiency. The dual-buffering method is adopted to eliminate the flicker in plotting area, and thus further shortens the drawing time.

Key words large-capacity data; curve plotting; VC++2008; Polyline; dual-buffering

在各类测试领域,经常需要采集、存储和处理大量数据. 例如,航天测试中的速变、振动、图像等信号,它们通常具有很高的采样率,因此需要处理的数据量非常大. 同时,原始数据晦涩难懂,如能将其转化为曲线或图像,将有助于后续分析和处理. 此外,将存储在数据文件中的数据以曲线形式再现,可以反映系统在过去一段时间内各类参数的变化情况,在研究系统的工作状态、进行故障诊

* 国家自然科学基金(51275491)资助

† 通信作者,E-mail:liuwenyi@nuc.edu.cn

断时具有重要作用[1].

目前，有大量软件开发工具、第三方控件等均可用于曲线绘制软件的开发，如 Visual Basic (VB)、Visual C++ (VC)、TeeChart 控件[2-3]等. 其中，使用 VB 绘图比较简单，在数据量较小时很有效；但在面对大容量数据时，由于 VB 效率较低，即便采用 API 绘图函数[4]，绘图效果和效率依然不佳. 此外，在绘图时经常会出现屏幕闪烁这一严重影响显示效果的问题. 文献[5-10]等均对闪烁问题及其解决方法作了研究，但很少有结合“如何提高程序执行效率”的讨论. 由于上述文献在讨论时大都未针对大容量数据，故对效率问题并不十分关注；而对于大容量数据处理[11]，则必须考虑程序的执行效率. 本文经过研究发现：屏幕闪烁与程序执行效率是 2 个有关联的问题，值得研究.

程序执行效率与开发工具的选择密切相关. 在当前流行的编程语言中，C/C++ 语言被普遍认为执行效率比较高[12]，如果再结合 API 相关函数，有望进一步提高绘图效率. 本文讨论的大容量数据大都超过了 4 GB，数据量很大. 为保证绘图效率，经过综合考虑，采用 VC++ 2008 作为开发工具.

1　数据文件和通道数据分离

在绘制曲线之前，应先将各通道测试数据从原始数据文件中分离出来，这就需要明确数据文件中测试数据的组织方式.

1.1　测试数据的帧格式

一般而言，不同领域的测试数据都有其常用的数据组织方式，工程上将这种数据组织方式称为“编帧”，编帧得到的一帧数据称为“帧结构”[13]. 图 1 是某测试数据的帧结构示意图.

帧头(2字节) 0X"EB90"	数据内容(共16字节)				帧计数 (4字节)	帧尾(2字节) 0X"146F"
	通道1数据 (2字节)	通道2数据 (2字节)	……	通道8数据 (2字节)		

图 1　某测试数据的帧结构

Fig. 1　A frame structure of test data

图 1 所示帧结构共 24 个字节，其中：“帧头”指明一帧数据的起始位置；“数据内容”是数据帧中的核心部分，其长度一般与采样路数、采样位数等有关. 该测试数据帧中包括 8 路模拟量数据，每路均为 16 位采样(各占 2 字节)；“帧计数”用于指定当前帧的序号，其作用在于判断相邻 2 帧是否连续；“帧尾”指明该帧的结束位置.

1.2　通道数据分离

通道数据分离是将各通道数据从原始数据中提取并写入相应通道数据文件的过程. 由于原始数据容量较大，不可能一次性将所有数据读入内存. 本文提出“逐块读取”→“拼接数据”→“分离数据”，最后达到“完整处理”的方式. 即：先读出一块数据(例如 1 MB)进行处理，并将该块剩余的未处理数据缓存至临时空间；再读出下一块数据，将上一块剩余数据与该块数据拼接后进行处理，并将剩余数据缓存；以此类推，直到处理完毕. 其中，“数据处理”是指找到该块数据中的有效数据帧，并将各通道数据写入相应文件. 通道数据分离流程如图 2 所示.

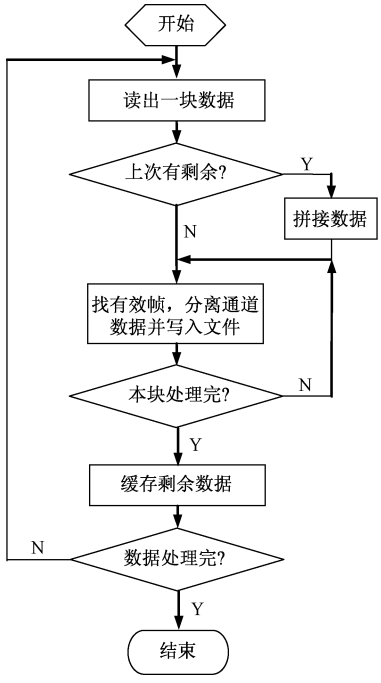

图 2　通道数据分离程序流程图

Fig. 2　Program flow diagram of data separation

此外，在数据分离时，每找到一个有效数据帧就要将其中的通道数据写入相应文件，而频繁写

文件(存盘),会严重影响程序执行效率. 本文采取“累积存储”法,即:当提取到的通道数据累积达到一定量(本文设计为2 MB)时才一次性写入文件,这样可以大幅减少写文件的次数,缩短程序执行时间.

2 大容量数据曲线绘制算法

VC是当前流行的基于Windows用户界面的编程工具,其功能非常强大. 本文在设计曲线绘制算法时,充分利用了MFC对Windows图形编程的支持.

2.1 从文件中按块提取数据

由于本文要处理的数据量很大,其曲线不可能一次性完整显示,只能通过分屏方式逐块显示. 绘制曲线时,可用滚动条控制要显示哪块数据,也可以设计为自动回放模式. 欲显示某块数据的曲线,需要确定起始位置和显示的数据点数,然后再从文件中取出相应的数据存入数组,以备显示之用.

以一个4 GB的通道数据文件为例,由于每个数据点各占2个字节,因此总的数据点数为2 G;如果每次显示4 K个数据点,则需分为512块.

2.2 曲线绘制算法设计

VC中使用MFC绘制曲线的一般过程为:“初始化绘图区域”→“定义设备描述表”→“定义并初始化绘图工具”→“调用函数绘制曲线”→“释放绘图资源”. 本文设计了绘图函数DrawWave,它包含3个输入参数:绘图区域、要显示的数据内容和显示的数据点数. 函数实现过程如下:

```
void DrawWave(CWnd* wWnd, USHORT* DisDat, ULONG DisLen)
{
//刷新绘图区域
pDC = wWnd -> GetDC();
pDC -> SetBkMode(TRANSPARENT);
pDC -> FillRect(&RectWave, &m_Brush);
//曲线的显示区域
dw = (X2 - X1) * 1.0/(DisLen - 1);
dh = (Y2 - Y1)/65 535.0;
//绘制曲线
Tmp = int(Y2 - DisDat[0] * dh);
pDC -> MoveTo(X1, Tmp);
for(i = 1; i < DisLen; i++)
{
Tmp = int(Y2 - DisDat[i] * dh);
pDC -> LineTo(int(X1 + i * dw), Tmp);
}
}
```

代码中“绘制曲线”部分的思路为:先通过MoveTo函数将画笔定位到第1个数据点处;然后再依次调用LineTo函数将剩余数据点用直线首尾相连. 由于VC绘图区的坐标系与常用的平面直角坐标系不同,因此数据点的纵坐标需经过换算,才能转换为常用的平面直角坐标系下的形式.

3 大容量数据曲线绘制遇到的问题

在对曲线绘制算法进行测试时发现:显示的数据量较大时,程序运行很慢,甚至会出现崩溃的情况,这说明绘图时占用资源过多且效率较低;另外,数据量较大时极易出现闪烁现象,影响曲线显示效果.

3.1 绘图效率低下问题

本文在设计绘图算法时已经做了数据结构、算法等方面的优化,但实际效果并不理想. 经过测试,绘图区域每次显示的数据点数超过1 M(1 048 576)时,就出现明显的迟滞现象,这说明绘图操作花费了程序的大量时间. 表1列出了绘图函数执行时间与显示数据点数的关系.

表1 程序执行时间与显示数据点数的关系(LineTo方法)

Table 1 Relationship between program execution time and data point count (LineTo method)

数据点数	32 768	65 536	131 072	262 144	524 288	1 048 576	1 310 720	2 097 152
时间/s	0.018 493	0.037 539	0.072 844	0.148 831	0.291 508	0.583 324	0.728 627	1.193 654

经过测算和分析,时间大都花费在LineTo函数上. 描绘点数越多,调用LineTo函数的次数就越多,因而积累的时间将较长. 为提高绘图效率,可以调用MFC中的API函数:Polyline,它能一次性将一组数据显示在绘图区域. 通过调用Polyline函数,“绘制曲线”部分的代码可改写为

```
for(i=0; DisLen; i++)
{
lpPoints[i].x=int(X1 + i * dw);
lpPoints[i].y=int(Y2 - DisDat[i] * dh);
}
pDC -> Polyline(lpPoints, DisLen);
```

调用 Polyline 函数绘图后，程序执行时间与显示数据点数的关系如表 2.2 种绘图方法在程序执行时间上的比较如图 3 所示. 经过计算，Polyline 方法比 LineTo 方法在绘图时间上平均缩短 94% 左右.

表 2　程序执行时间与显示数据点数的关系(Polyline 方法)

Table 2　Relationship between program execution time and data point count (Polyline method)

数据点数	32 768	65 536	131 072	262 144	524 288	1 048 576	1 310 720	2 097 152
时间/s	0.000 934	0.00 183	0.003 532	0.00 686	0.019 472	0.041 062	0.056 443	0.089 255

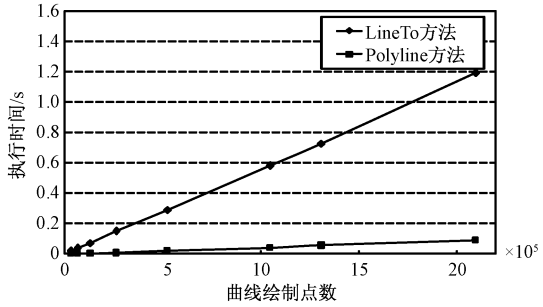

图 3　LineTo 方法和 Polyline 方法绘图执行时间比较

Fig. 3　Comparison of execution time between LineTo and Polyline methods

3.2　显示区域闪烁问题

在进行大容量数据曲线再现时，经常会出现显示屏幕闪烁的现象. 理论上讲，闪烁是由 VC 绘图机制引起并固有存在的，只是在多数情况下闪烁并不明显而不易被发觉. 但在图形切换频繁且反差较大时，此现象就会比较明显. 编程应用中有许多消除图形闪烁的方法，如页面切换法(page flipping)、增量更新法(incremental update)、双缓冲法(dual-buffering)等. 其中的双缓冲法比较常用，基本思路为：在内存中设置 2 块缓存，一块是显示区域设备描述表(前端缓冲区)，另一块是与设备描述表兼容的内存设备描述表(后备缓冲区). 绘图时，先将图形绘制在后备缓冲区中，然后将后备缓冲区中的图形拷贝到前端缓冲区中，最后由系统自动将前端缓冲区中的图形提交至显存进行显示[5, 10].

拷贝图形可使用 BitBlt 函数. 采用双缓冲法，绘图函数可改写为如下形式：

```
CDC * pDC;
CDC memDC; CBitmap memBmp;
memDC.CreateCompatibleDC(pDC);
memBmp.CreateCompatibleBitmap(pDC, RectWave.right, RectWave.bottom);
memDC.SelectObject(&memBmp);
void DrawWave(CWnd* wWnd, USHORT* DisDat, ULONG DisLen)
{
//刷新绘图区域
pDC=wWnd -> GetDC();
memDC.SetBkMode(TRANSPARENT);
CBrush m_Brush(BackCol);
memDC.FillRect(&RectWave, &m_Brush);
//绘制曲线
memDC.Polyline(lpPoints, DisLen);
pDC -> BitBlt(RectWave.left, RectWave.top, RectWave.right, RectWave.bottom, &memDC, 0, 0, SRCCOPY);
}
```

经过测试，采用双缓冲法可以有效消除绘图区域闪烁现象. 同时，采用该方法能进一步改善绘图效率. 以每次显示 2 097 152 个数据点为例，采用双缓冲法绘制曲线大约需要 0.068 8 s，比表 2 中的 0.089 255 s 缩短近 23%. 可见，采用双缓冲法既能解决显示闪烁问题，还有助于提高程序执行效率，这对大容量数据处理的意义极大.

4　大容量数据曲线绘制软件

本文在 VC++ 2008 环境下编写了大容量数据曲线绘制软件，其界面如图 4 所示. 软件有两大功能：数据分路和数据曲线显示. 图中曲线对应的数据来源于文件“DAT(Original).CH3.DAT”，这是一个通道数据文件，大小为 13.08 GB. 曲线纵

坐标是模拟量经采集、量化后的电压值,其范围为 0 ~ 5 V;横坐标是数据点数,在已知采样率的情况下可以显示为时间.

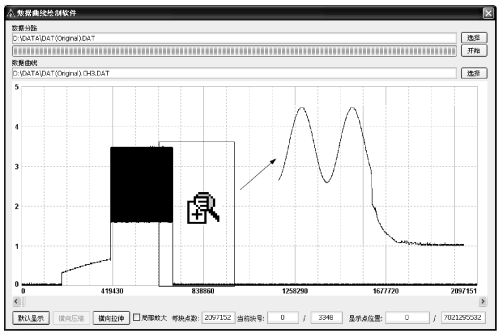

图 4 曲线绘制软件界面

Fig. 4 Curve plotting software interface

5 结论

本文利用 MFC 中对图形编程的支持,在 VC++ 2008编程环境下讨论大容量数据曲线绘制方法,并研究绘图程序的执行效率和显示闪烁问题. 对于提高绘图效率的问题,主要利用 API 中的 Polyline 函数,该方法比 LineTo 绘图法在时间上有大幅缩减,提高了程序执行效率. 针对显示区域闪烁的问题,本文采取双缓冲方法,一方面消除显示闪烁,同时进一步提高程序执行效率. 曲线绘制在各类测试领域均有广泛应用,本文设计的软件已在多项测试项目中得到成功应用.

参考文献

[1] 赵卫强, 鲁墨武. 基于 VB6.0 的监控系统软件中数据存储及数据曲线实现方法的研究[J]. 沈阳航空工业学院学报, 2003, 20(1): 31-34.

[2] 王瑾琦, 王红亮. 某遥测系统数据记录设备软件设计[J]. 传感器与微系统, 2011, 30(9): 128-130.

[3] 姜永增, 王崇, 姚仲敏. 实时数据曲线采集系统的设计与实现[J]. 齐齐哈尔大学学报, 2006, 22(2): 40-43.

[4] 文远熔. 使用 API 函数提高 VB 环境下绘制曲线速度[J]. 重庆工商大学学报:自然科学版, 2005, 22(2): 148-152.

[5] 焦景欣, 代亮, 胡含凯, 等. 采用双缓存的视景仿真图像显示效果改善方法[J]. 探测与控制学报, 2010, 32(5): 92-95.

[6] 叶林瓒, 雷小永, 戴树岭. 飞行仿真系统数据可视化设计与实现[J]. 计算机仿真, 2012, 29(3): 141-145.

[7] 王艳, 张礼君, 韩啸. 数字超声轮轴探伤仪软件系统的设计与实现[J]. 自动化与仪表, 2011, 7: 21-25.

[8] 郭振华, 江亚群, 杨帅雄, 等. 故障录波器后台分析软件关键问题研究[J]. 电力系统保护与控制, 2011, 39(19): 73-78.

[9] 唐海全, 邵才瑞, 李洪强. 随钻测井曲线无闪烁绘制技术[J]. 测井技术, 2010, 34(5): 479-482.

[10] 韩丽娜, 石昊苏. 使用双缓存来解决 GDI 下的闪烁问题[J]. 计算机工程与设计, 2006, 27(17): 3 258-3 260.

[11] 李河, 王祝文, 李舟波, 等. 大数据量地球物理测井绘图关键技术研究[J]. 地球物理学进展, 2005, 20(1): 71-77.

[12] 刘文怡, 甄国涌. 高速测控系统中实时数据处理软件技术研究[J]. 华北工学院学报, 2004, 25(3): 191-193.

[13] 马志刚,刘文怡,张文栋. 航天测试系统通用数据处理软件的设计[J]. 微电子学与计算机,2014,31(11):98-101.