Software Engineering and Applications 软件工程与应用, 2020, 9(1), 22-35
Published Online February 2020 in Hans. http://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2020.91004

Hans汉斯

Based on the Visualization Technology, Design and Implement a Radar Data Processing Software Platform

Zhimin Zhang, Hongmei Li, Wuping Zhan

PLA 63620 Troops, Jiuquan Gansu
Email: zzm_Fletcher@163.com

Received: Dec. 31st, 2019; accepted: Jan. 14th, 2020; published: Jan. 21st, 2020

Abstract

In order to improve the efficiency of radar data processing, a software platform for radar data processing is designed and implemented based on visualization technology. The design of two core components of the platform, namely, the waveform chart and the text window, is emphasized. The waveform chart is used for the observation of data as a whole as well as in detail. The operations imposed on the waveform curve include panning freely, scaling horizontally or vertically, viewing data point, etc. And the waveform chart data can be dynamically selected based on a proposed data selection algorithm. An alternate visualization method of viewing and processing data lies in the design of the text window where a shortcut menu is utilized to enhance the interactivity of the human-computer operation. A fast and efficient drawing algorithm is adopted to improve the drawing speeds of both the waveform chart and the text window. The application results of data processing for a certain type of monopulse radar show that using this software platform improves the data processing efficiency.

Keywords

Visualization Technology, Design and Implementation, Radar Signal Processing, Software Platform

基于可视化技术设计实现雷达数据处理软件平台

张志民，李红梅，詹武平

解放军63620部队，甘肃 酒泉
Email: zzm_Fletcher@163.com

收稿日期：2019年12月31日；录用日期：2020年1月14日；发布日期：2020年1月21日

文章引用：张志民，李红梅，詹武平．基于可视化技术设计实现雷达数据处理软件平台[J]．软件工程与应用，2020, 9(1): 22-35. DOI: 10.12677/sea.2020.91004

摘　要

为提高雷达数据处理效率，基于可视化技术设计实现雷达数据处理软件平台，重点给出波形图和文本窗两个平台核心构件的设计过程。设计波形图观测数据的整体及细节，对波形曲线进行自由平移、水平缩放、竖直缩放、查看数据点等操作，基于数据选取算法动态选取波形图数据；设计文本窗以另一种可视化方式显示并处理数据，使用快捷菜单提高人机操作的交互性。采用快速高效绘制算法提高波形图和文本窗的绘制速度。对某型号单脉冲雷达数据处理的应用结果表明，该软件平台达到了提高数据处理效率的目的。

关键词

可视化技术，设计实现，雷达数据处理，软件平台

1. 引言

所谓数据可视化，是指将大型数据集中的数据以图形图像等形式表示，利用数据分析和开发工具发现其中未知信息的过程。数据可视化利用人类感觉系统的广阔带宽，对错综复杂的过程，涉及不同学科领域的数据集，以及来源多样的大型抽象数据集进行操纵和解释。数据可视化技术是数据可视化的实现技术，它主要借助于图形化手段，来清晰、有效地传达与沟通信息，在人们目前的研究、教学和开发等领域有着广泛的应用。

雷达数据的类型一般包括目标号、绝对时、相对时、20 Hz 计数、雷达状态码、斜距 R、方位角 A、俯仰角 E、信噪比、RCS (雷达散射界面)、AGC (自动增益控制)、斜距误差、方位角误差、俯仰角误差、雷达发射功率等，雷达数据处理流程一般包括有效数据段落选取、野值剔除、方位角跳点消除、插值加密、合理性检验、时间修正、电波折射误差修正、单站求解、甄别目标等[1]。目前某单位要处理的雷达测量数据日益增多，对雷达数据处理速度的要求越来越高，甚至要求数据处理能够准实时或实时进行。出于安全保密等原因，在目前国外公开发表的文献中，尚未发现基于可视化技术实现雷达数据处理软件平台的报导。国内一些科研院所进行雷达数据处理大多基于 MATLAB 软件平台实现，处理过程人机交互性不太好，可视化性能比较差[2]。

为提高雷达数据处理速度，增强处理过程中的人机交互性，本文基于可视化技术设计、实现雷达数据处理软件平台。下面首先给出平台架构，然后详细介绍平台架构中波形图、文本窗两个核心构件的设计实现过程。在关键技术部分，以波形图为例给出图形高效绘制算法和图形选择算法。接下来以某型号单脉冲雷达数据处理为例，对平台的应用效果进行评估，最后是结束语。

2. 可视化雷达数据处理软件平台设计

基于可视化技术设计的雷达数据处理软件平台架构如图 1 所示，可以看出平台基于 Visual C++ 2010 集成环境开发，包含波形图和文本窗两个核心构件。波形图是一个以图形方式显示数据的窗口，在这个窗口中，用户可以对图形进行自由平移、水平缩放、竖直缩放、图形数据动态选择、图形数据查看等操

作；文本窗是一个以文本形式显示、编辑和处理数据的窗口，在这个窗口中，用户借助于快捷菜单操纵数据。为进一步提高处理速度，波形图和文本窗都采用快速显示算法。在可视化显示和处理数据方面，波形图和文本窗各有侧重，各有所长，两者结合起来使用，可以充分发挥各自优势，提高数据处理速度[3] [4]。

此外，图 1 中的雷达数据处理软件平台架构还包含面向对象程序设计技术、命令消息分发机制、微软基础类库(Microsoft Foundation Class, MFC)、多文档界面技术(Multiple Document Interface, MDI)模块，它们的作用是充分利用现代软件开发理念及开发工具，提高开发效率。举例来说，一批雷达数据既可以波形图显示，又可以文本窗形式显示。再考虑用户同时处理多批雷达数据的情形，不难理解为提高软件开发效率，应该基于 MFC 采用 MDI 界面设计技术，以使不同批数据采用不同的波形图和文本窗来容纳。从具体实现角度来看，所有波形图和文本窗都属于同一类对象[5] [6] [7]。

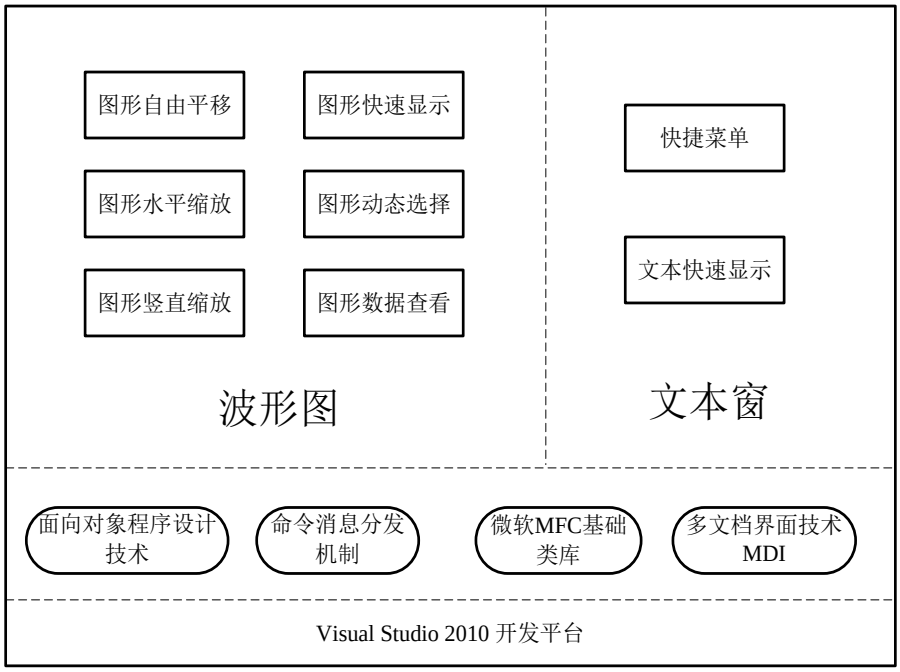

Figure 1. The software platform architecture for visual radar data processing
图 1. 可视化雷达数据处理软件平台架构

2.1. 设计实现波形图

波形图实际上包括显示波形数据或曲线的窗口及处理消息的窗口过程两部分。从面向对象程序设计的技术出发，创建波形图一般包括注册窗口类、创建窗口和实现窗口消息分发机制三个步骤。

2.1.1. 注册窗口类

每个显示雷达数据的波形图都是在特定的窗口类的基础上创建的，而每个窗口类在使用前必须在操作系统中注册。注册窗口类使用函数 RegisterClassEx 实现[8]：

ATOM RegisterClassEx(CONST WNDCLASSEX *lpwcx);

这里 lpwcx 为一指向 WNDCLASSEX 结构体的指针。若 RegisterClassEx 函数执行成功，返回值是一个唯一标识已经注册了的窗口类的原子(Atom)类；否则返回值将是 0。

WNDCLASSEX 结构体中包含了窗口类的信息，其定义如下：

```
typedef struct    _WNDCLASSEX {
    UINT              cbSize;
    UINT              style;
    WNDPROC      lpfnWndProc;
    int               cbClsExtra;
    int               cbWndExtra;
    HINSTANCE    hInstance;
    HICON          hIcon;
    HCURSOR       hCursor;
    HBRUSH         hbrBackground;
    LPCTSTR       lpszMenuName;
    LPCTSTR       lpszClassName;
    HICON          hIconSm;
} WNDCLASSEX, *PWNDCLASSEX;
```

下面分别介绍 WNDCLASSEX 结构中各个字段的含义。

cbSize：给出本结构体的以字节为单位的大小。

style：指定窗口类的风格。

lpfnWndProc：指向窗口过程的指针。

cbClsExtra、cbWndExtra：分别指定本窗口类后面、实例后面额外分配的字节数，系统会将这些字节的内容全部初始化为零。

hInstance：本类窗口实例的句柄。

hIcon：窗口类图标的句柄。

hCursor：窗口类光标的句柄。

hbrBackground：窗口类背景画刷的句柄。

lpszMenuName：指向一个以零字符结尾的字符串，该字符串代表了窗口类菜单的资源名。若 lpszMenuName 为 NULL，那么本窗口类产生的窗口将没有默认菜单。

lpszClassName：指向一个以零字符结尾的字符串，该字符串代表了窗口类的类名。

hIconSm：代表一个与窗口类关联的小图标的句柄。

2.1.2. 创建窗口

注册窗口类完成后，接下来调用 CreateWindowEx 函数创建窗口，函数原型如下所示[8]：

```
HWND CreateWindowEx(DWORD dwExStyle, LPCTSTR lpClassName,
LPCTSTR lpWindowName, DWORD dwStyle, int x, int y, int nWidth, int nHeight,
HWND hWndParent, HMENU hMenu, HINSTANCE hInstance, LPVOID lpParam);
```

CreateWindowEx 函数的参数较多，各个参数的含义简述如下。

dwExStyle：指定所创建窗口的扩展样式，具体样式参见文献[8]。

lpClassName、lpWindowName：二者均为指向以零字符结尾的字符串，两个字符串分别标识窗口类的类名和窗口的标题。

dwStyle：指定所创建窗口的风格。

x、*y*：指定窗口的最初位置。对于层叠式或弹出式窗口来说，它们分别表示屏幕坐标系中窗口左上

角的 x、y 坐标；而对于子窗口来说，它们则分别表示窗口左上角相对于其父窗口客户区的 x、y 坐标。

nWidth、nHeight：以设备单位表示的窗口的宽度和高度。

hWndParent：指定父窗口的句柄。

hMenu：其含义取决于窗口风格，即对于层叠式或弹出式窗口指的是窗口的菜单句柄，对于子窗口指的则是子窗口标识符。

hInstance：窗口关联模块的实例句柄。

lpParam：窗口创建后，窗口过程将接收到 WM_CREATE 消息，该消息的 LPARAM 类型的参数是一个指向 CREATESTRUCT 类型结构体的指针，该结构体的 LPVOID 成员即等于此 lpParam。若创建的是 MDI 的客户窗口，则 lpParam 为一指向 CLIENTCREATESTRUCT 结构体的指针。

若 CreateWindowEx 函数创建窗口能成功，则函数返回值是所创建的新窗口的句柄，否则返回值为 NULL。

2.1.3. 实现窗口消息分发机制

创建波形图的第三步是实现窗口消息分发机制。波形图显示的可以是不同类型的雷达数据，如斜距、方位角、俯仰角、目标 RCS 等，所有这些波形图都属于同一个窗口类，即它们都是基于同一个窗口类创建的不同窗口。为了使各个波形图在功能上相互区分，互不干扰，最好的方法是使用面向对象程序设计技术，使每个波形图都关联到不同的类对象，比如关联到不同的 CPlot 对象上。这样一来，就需要把同一个窗口过程接收到的消息，按对应关系分发给不同的 CPlot 对象，使得每一个 CPlot 对象只关心和处理与自己有关的数据，不影响其它 CPlot 对象的数据处理过程，即实现窗口消息分发机制。

为了实现窗口消息分发机制，定义 CPlot 类如下：

```
#include <memory>
#include <vector>
using namespace std;
class CPlot
{
public:
CPlot(HWND hwnd);
virtual ~CPlot();
protected:
HWND    m_hWnd;
static vector<shared_ptr<CPlot>>    m_svpPlots;
static LRESULT CALLBACK WndProc(HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam);
LRESULT    _WndProc(UINT message, WPARAM wParam, LPARAM lParam);
……
};
```

上面的代码定义了一个 C++类 CPlot。它有一个 HWND 类型的成员变量 m_hWnd，代表波形图窗口的句柄。每一个 CPlot 类对象都与唯一一个这样的句柄相对应，换句话说 CPlot 是波形图的封装类或包裹类。静态矢量型成员变量 m_svpPlots 用来维护所有 CPlot 对象。静态成员函数 WndProc 是波形图的窗口过程，它通过窗口消息分发机制将消息转发到成员函数_WndProc 来处理，消息分发过程如图 2 所示。

图 2 中，程序开始运行、初始化后，进入消息等待阶段。一旦检索到窗口消息，即在 m_svpPlots 中寻找消息处理窗口句柄。若能找到对应的 CPlot 对象，就将消息转交给该对象处理，之后再次进入消息等待阶段。若在 m_svpPlots 中无法找到对应的包裹类对象，说明创建的是一个新窗口，需新建一个 CPlot 关联对象，并将消息交付该对象去处理，之后同样进入消息等待阶段。当软件平台主程序运行结束时，所有波形图窗口均已销毁，所有的 CPlot 对象则由 CPlot 类静态成员 m_svpPlots 负责销毁，释放所占据的资源[9] [10]。

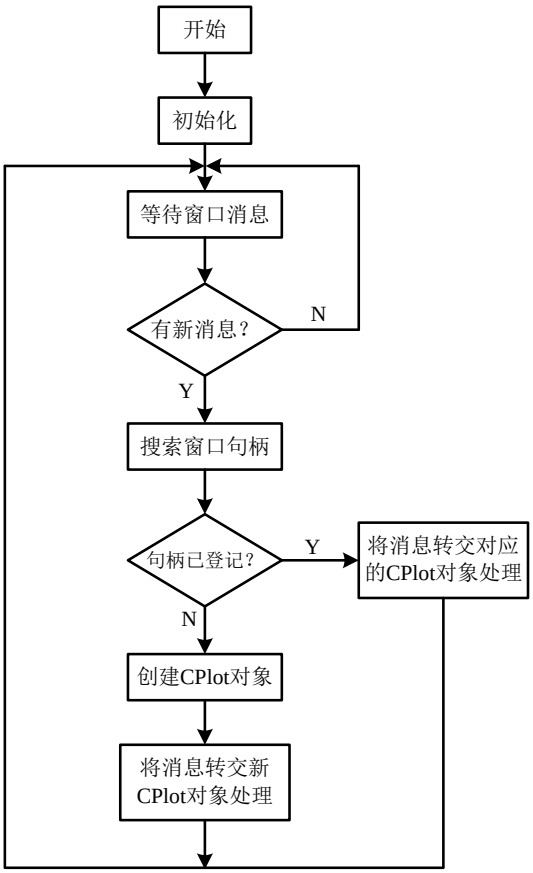

Figure 2. The message distribution mechanism of the waveform chart window
图 2. 波形图窗口的消息分发机制

2.2. 设计实现文本窗

文本窗的使用方式参见图 3 给出的例子，它以数据行的方式显示雷达数据。文本窗的设计实现过程类似于波形图，也包括注册窗口类、创建窗口和实现窗口消息分发机制三个步骤，为节省篇幅这些内容不再赘述。

为便于处理雷达数据中的目标号、绝对时、相对时、20 Hz 计数、雷达状态码、斜距 R、方位角 A、俯仰角 E、信噪比、RCS、AGC、斜距误差、方位角误差、俯仰角误差、雷达发射功率等数据，较好的作法是通过鼠标右键点击，利用弹出的快捷菜单提供的功能实现[7] [8]。使用快捷菜单时，程序执行流程涉及下述步骤：

1) 用户在文本窗的字段区即标题区按下鼠标右键；

2) 程序代码判断鼠标指针指向了哪种数据(用指针指向的数据列来区分);
3) 弹出相应的快捷菜单，例如图 3 表示用户在 *R* 数据列上按下了鼠标右键;
4) 执行相应的菜单命令，完成诸如角度弧度互化、查看数据波形、查看数据差分波形等功能。

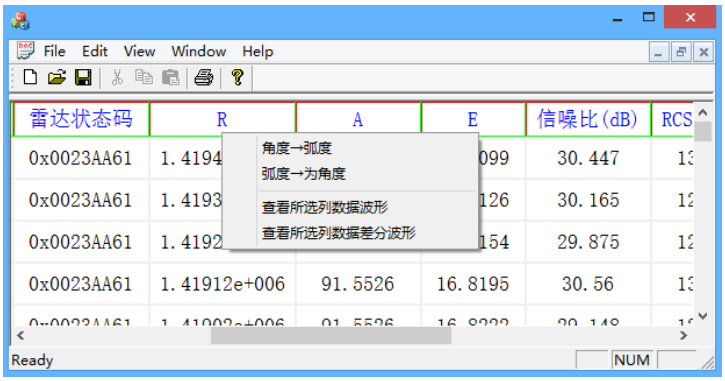

Figure 3. An example of the text window design
图 3. 文本窗设计举例

3. 关键技术

这里以波形图为例，介绍快速高效绘制算法及图形选择算法[11] [12] [13]。同样，文本窗的情形与此类似，为节省篇幅不再赘述。

3.1. 波形图快速高效绘制算法

在可视化雷达数据处理软件平台上，当对波形图进行平移、缩放等操作特别是放大操作时，整个波形图占据的显示区域较大，往往超出了显示设备如显示器的整个显示区域。绘制显示器可显示区域以外的波形图会消耗计算机处理器时间，但显然这种绘制是不必要的，因为这部分图形根本不会显示出来。再者，对于水平或竖直平移波形图来说，移动后仍显示在显示器上的图形部分也不需要重新绘制，因为这部分图形没有变化，只是显示位置发生了变化。为提高波形图绘制的速度，在此提出一种快速高效绘制算法。该算法计算波形图应该显示的部分，显示波形图时只处理这部分图形的绘制工作，从而避免大量不必要的图形重绘工作，提高雷达数据处理软件平台的实时性和响应速度[9] [10]。

3.1.1. 算法推导

在 Windows 操作系统中，当移动某个应用程序窗口，致使窗口原来被遮盖的部分显露出来，或者改变窗口大小时，Windows 都要向应用程序发送 WM_PAINT 消息，以告知应用程序有一部分客户区域已变为无效，需要重新绘制。这部分需要重绘的区域称为客户区无效区域，由于一般是矩形，故又称无效矩形。通常情况下无效矩形的大小不等于客户区，如上文所述，重绘时没有必要绘制整个客户区，只需绘制客户区中位于无效矩形中的那部分图形即可，以提高图形绘制代码的执行速度，减少绘制工作量，实现实时操作。

如图 4 所示，在客户区坐标系中，采用 MM_TEXT 映射模式，即坐标度量单位是像素，窗口客户区左上角 *O* 为坐标原点，*OX* 轴方向水平向右，*OY* 轴方向竖直向下。为便于叙述，设要显示的图形只包含一条曲线。曲线上共有 *N* 个数据点，各数据大小分别为 $Y_0, Y_1, \cdots, Y_{N-1}$，数据中的最大、最小值分别为 $Y_{\max}$、$Y_{\min}$，各数据点在客户区坐标系中的坐标分别为

$$(x_0, y_0), (x_1, y_1), \cdots, (x_I, y_I), \cdots, (x_{N-1}, y_{N-1})$$

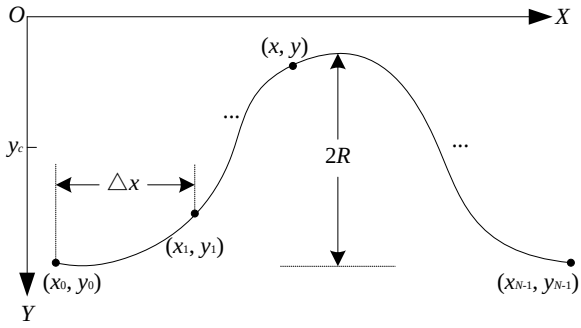

Figure 4. A drawing schematic in the application client area
图 4. 应用程序客户区图形绘制示意图

这里 I 为基于 0 的数据索引，即 $0 \le I \le N-1$。为简单起见，假设曲线上各数据点在水平方向上均匀分布，Δx 为两个相邻数据点之间的水平距离。(x, y) 为曲线上任意一点，它对应于数据 Y，y_c 为图形纵向分布中心的纵坐标，它对应于数据 Y_c，根据简单的数学推导，可得下列方程组

$$\begin{cases} I = \dfrac{x - x_0}{\Delta x} \\ Y = Y_c - \dfrac{y - y_c}{2R} ly \end{cases} \quad (\Delta x > 0, ly > 0, R > 0)$$

其中 R 为图形纵向分布半径，且

$$Y_c = (Y_{\max} + Y_{\min})/2$$
$$ly = Y_{\max} - Y_{\min}$$

3.1.2. 自由平移、水平缩放、竖直缩放关键技术

快速高效图形绘制算法的总体思想是，当对图形实施操作时，保持客户区坐标系和映射模式固定不变，通过改变 x_0、y_c、Δx、R 的值并重绘，实现图形实时变化，具体分析如下：

保持 y_c、Δx、R 不变，增大 x_0，图形整体水平右移，减小 x_0，图形整体水平左移。

保持 x_0、Δx、R 不变，增大 y_c，图形竖直向上整体平移，减小 y_c，图形竖直向下整体平移。

自由平移图形，可视作先对图形作水平平移，再对图形作竖直平移的组合。

保持 x_0、y_c、R 不变，增大 Δx，图形在水平方向上放大，减小 Δx，图形在水平方向上缩小。

保持 x_0、y_c、Δx 不变，增大 R，图形在竖直方向上放大，减小 R，图形在竖直方向上缩小。

当 x_0、y_c、Δx、R 中某一个值改变后，及时调用 InvalidateRect 函数使需要重绘的区域无效，驱使 Windows 操作系统给应用程序发送 WM_PAINT 消息。在该消息的响应处理中，利用更新的 x_0、y_c、Δx、R 重绘无效区域，达到动态、实时显示图形的目的。

3.1.3. 水平方向缩放图形

对图形的操作常用鼠标进行，当需要水平缩放图形时，为改善图形缩放视觉效果，应使位于鼠标指针处的数据在缩放前后的位置保持不变。设 (x, y) 是鼠标指针当前位置，(x', y') 是水平缩放后的位置，则应有

$$\begin{cases} \dfrac{x - x_0}{\Delta x} = \dfrac{x' - x_0'}{\Delta x'} \\ x' = x, \quad y' = y \end{cases}$$

其中 $\Delta x'$ 是已知参数，解上述方程组得到 x_0'，然后重绘客户区图形，实现水平缩放。

3.1.4. 竖直方向缩放图形

竖直缩放图形时，为了增加缩放视觉效果，同样要求鼠标指针处的图形在缩放前后的位置保持不变，设(x, y)是鼠标指针当前位置，(x', y')是竖直缩放后的位置，则有

$$\begin{cases} \dfrac{y - y_c}{R} = \dfrac{y' - y_c'}{R'} \\ x' = x, \quad y' = y \end{cases}$$

其中R'是已知参数，解上述方程组得到y_c'，然后重绘客户区图形，实现竖直缩放。

3.1.5. 自由缩放图形

自由缩放是指在按下鼠标键后不放开的情况下拖动鼠标，形成一个矩形区域，对该区域中的图形进行缩放，以观察这部分图形的详细情况。在自由缩放模式下，鼠标拖出矩形区域与客户区当前可见范围在缩放前后相对应。设(l,t)、(r,b)分别为鼠标选定矩形左上角和右下角坐标，(x_1, y_1)、(x_2, y_2)分别为客户区左上角和右下角坐标，易知

$$\begin{cases} \dfrac{l - x_0}{\Delta x} = \dfrac{x_1 - x_0'}{\Delta x'} \\ \dfrac{r - x_0}{\Delta x} = \dfrac{x_2 - x_0'}{\Delta x'} \end{cases} \qquad \begin{cases} \dfrac{t - y_c}{R} = \dfrac{y_1 - y_c'}{R'} \\ \dfrac{b - y_c}{R} = \dfrac{y_2 - y_c'}{R'} \end{cases}$$

解上述两个方程组，得到x_0'、$\Delta x'$、y_c'、R'，并重绘客户区图形，实现图形选定区域缩放。

3.1.6. 查看图形上某点数据

设鼠标指针坐标是(x, y)，将x、y代入下式

$$\begin{cases} I = \dfrac{x - x_0}{\Delta x} \\ Y = Y_c - \dfrac{y - y_c}{2R} ly \end{cases} \quad (\Delta x > 0, ly > 0, R > 0)$$

得到该点的索引I和数值Y，在此基础上，应用程序可进一步采用适当的方式显示该数据。

3.1.7. 确定无效矩形包含的数据索引

由以上分析可知，应用程序接收到 WM_PAINT 消息后要重绘客户区，为了保证波形图显示的实时性，只需绘制无效矩形区域内的图形部分。这就需要准确计算需要重绘的与无效矩形关联的数据索引范围和坐标轴刻度范围，见图 5。

设s_1、s_2分别是无效矩形左、右两边界的横坐标，处于无效矩形中待显示数据的索引范围为 iIndexBeg $\le I \le$ iIndexEnd，则计算 iIndexBeg 和 iIndexEnd 的步骤示例如下：

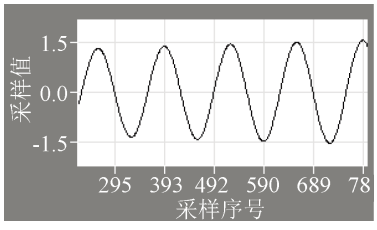

Figure 5. Determines the data range and axis scale range associated with the invalid rectangle
图 5. 确定与无效矩形关联的数据范围和坐标轴刻度范围

$$\text{iIndexBeg} = \min\left(\max\left(0, \text{int}\left(\frac{s_1 - x_0}{\Delta x}\right)\right), N-1\right);$$

$$\text{iTemp} = \text{int}\left(\frac{s_2 - x_0}{\Delta x}\right);$$

if $(s_2 > x_0$且s_2不是整数$)$

$++\text{iTemp};$

$$\text{iIndexEnd} = \max\left(\min\left(N-1, \text{iTemp}\right), 0\right);$$

其中 int 是 C++语言中的强制数据类型转换运算符，min、max 是两个宏，分别用来返回两个数中的最小值和最大值。一旦确定数据索引范围后，即可绘制图形：

if $(\text{iIndexBeg} < \text{iIndexEnd})$

{//绘制该部分图形}

3.1.8. 确定无效矩形包含的 *OX* 轴刻度

假定 x_0 处的刻度线索引为 0，对于小于 x_0 的数据，规定其刻度索引为负，大于 x_0 的数据，刻度索引为正。假定 *OX* 轴上各刻度线均匀分布且固定。设两相邻刻度线间水平像素数为 Δp_x，无效矩形包含的刻度索引范围是$[k_1, k_2]$，则确定 k_1、k_2 的示例代码如下：

$$k_1 = \text{int}\left(\frac{s_1 - x_0}{\Delta p_x}\right); k_2 = \text{int}\left(\frac{s_2 - x_0}{\Delta p_x}\right);$$

if $(s_1 > x_0$且s_1不是整数$)$

$++k_1;$

if $(s_2 < x_0$且s_2不是整数$)$

$--k_2;$

一旦确定刻度线索引范围后，即可如图 5 所示绘制 *OX* 轴刻度线：

if $(k_1 \le k_2)$

{//绘制*OX*轴刻度线}

3.1.9. 确定无效矩形包含的 *OY* 轴刻度

假定 *OY* 轴刻度线索引是整数，且 y_c 处刻度线索引为 0，小于 y_c 的刻度线索引为负，大于 y_c 的刻度线索引为正。刻度线在 *Y* 轴上均匀分布且位置固定，设两相邻刻度线竖直距离为 Δp_y，s_1、s_2 分别表示无效矩形上、下两边界的纵坐标。无效矩形包含的刻度索引范围是$[m_1, m_2]$，则确定 m_1、m_2 的示例代码如下：

$$m_1 = \text{int}\left(\frac{s_1 - y_c}{\Delta p_y}\right); m_2 = \text{int}\left(\frac{s_2 - y_c}{\Delta p_y}\right);$$

if $(s_1 > y_c$且s_1不是整数$)$

$++m_1;$

if $(s_2 < y_c$且s_2不是整数$)$

$--m_2;$

if $(m_1 \le m_2)$

{//绘制OY轴刻度线}

波形图快速高效绘制算法的应用如图 6 所示。

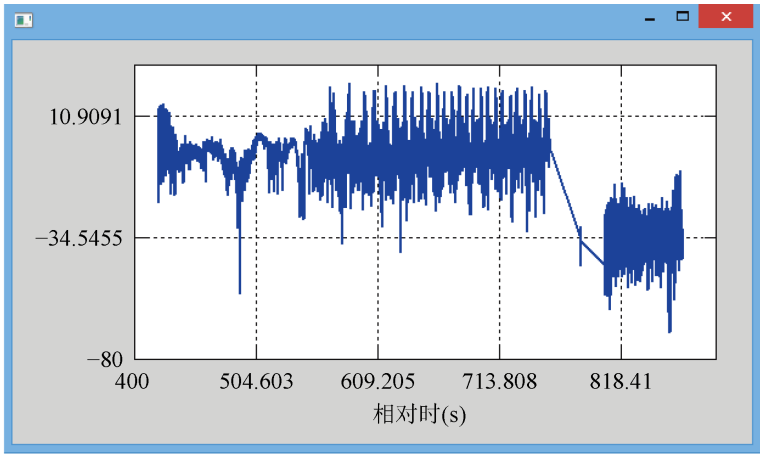

Figure 6. An example of the application of the fast and efficient drawing algorithm for a waveform chart
图 6. 波形图快速高效绘制算法应用示例

3.1.10. 快捷菜单设计的关键技术

为了设计文本窗快捷菜单，需要处理鼠标右键的 WM_CONTEXTMENU 消息，窗口过程示例如下：

```
LRESULT CALLBACK WindowProc(
    HWND hwnd,                  // 窗口句柄
    WM_CONTEXTMENU,    // 要处理的消息
    WPARAM wParam,      // 窗口句柄
    LPARAM lParam        // 鼠标指针位置
);
xPos = GET_X_LPARAM(lParam); // 获取鼠标位置的 x 坐标
yPos = GET_Y_LPARAM(lParam); // 获取鼠标位置的 x 坐标
```

获取鼠标位置后，即可调用 TrackPopupMenu 函数显示快捷菜单：

```
TrackPopupMenu(hmenuTrackPopup, TPM_LEFTALIGN | TPM_RIGHTBUTTON,
xPos, yPos, 0, hwnd, NULL);
```

用户选择快捷菜单命令后，对应功能在消息处理函数中实现。

3.2. 图形选择算法

图形选择算法原理如图 7 所示。当用户在波形图中按下鼠标左键后，程序初始化 Beg、End 两个标识选择区域左、右边界的变量，并捕获鼠标输入。因为通常情况下只有鼠标在波形图窗口内按下时，程序才会接收到输入消息，而为了选取可见区域之外的图形部分，窗口应该能够自动滚动，由实际操作过程可知这时鼠标应该位于窗口边界外面。捕获鼠标输入就是为了在这时候也能接收到鼠标输入消息。捕获鼠标后的代码由两个循环组成。

当应用程序的消息队列中没有消息时，执行流程进入第一个循环部分。这时候首先获取鼠标指针位

置，并将此位置转化到波形图窗口坐标系内。如果指针位于窗口右边界以外，说明要选择图形中尚未显示出来的右边部分，显然应让图形向左边滚动；相反，如果指针位于窗口左边界以外，说明要选择图形中尚未显示出来的左边部分，显然应让图形向右边滚动。无论是向左还是向右滚动，滚动速度应该与指针离开相应边界的距离成正比，以便符合人的感官体验。窗口滚动后，更新 End 并重绘波形图。

当应用程序的消息队列中存在要处理的消息时，执行流程进入第二个循环部分。针对的消息不同，处理方式也不同：如果是鼠标移动消息，则为了显示正确的图形选取范围，首先应擦除原有图形，再绘制更新后的图形，之后再次检查消息队列；如果是鼠标释放消息，说明图形选取过程结束，此时应检查 Beg、End 的有效性，必要时做出调整，执行流程结束；如果是鼠标移动、鼠标释放之外的消息，则调用 DiapatchMessage 函数进行默认处理。当队列中没有消息时，流程返回到第一个循环部分。

图 8 给出了波形图选取示例，选取部分用图中的黑色部分表示。

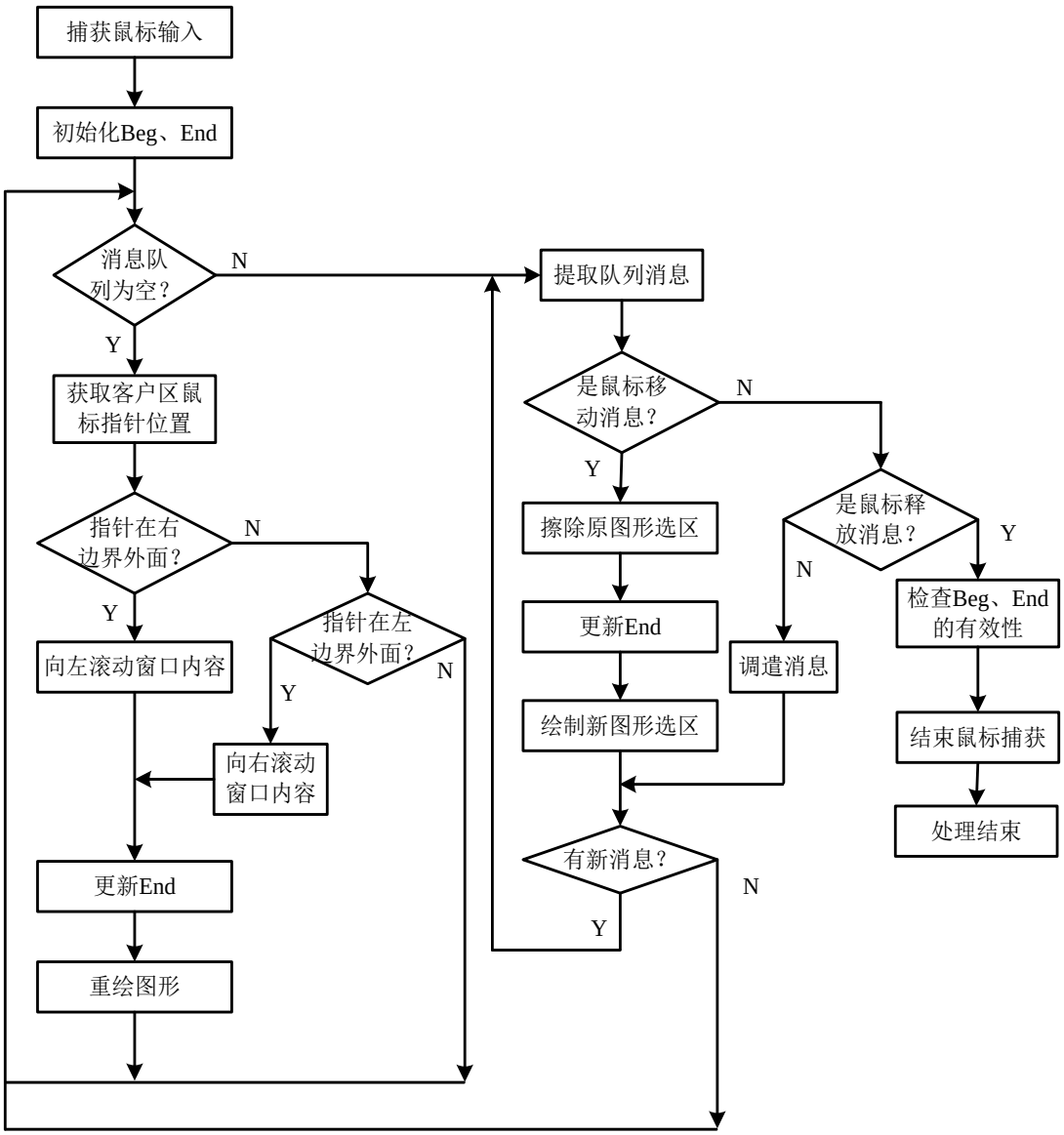

Figure 7. A schematic of the graph selection algorithm
图 7. 图形选择算法示意图

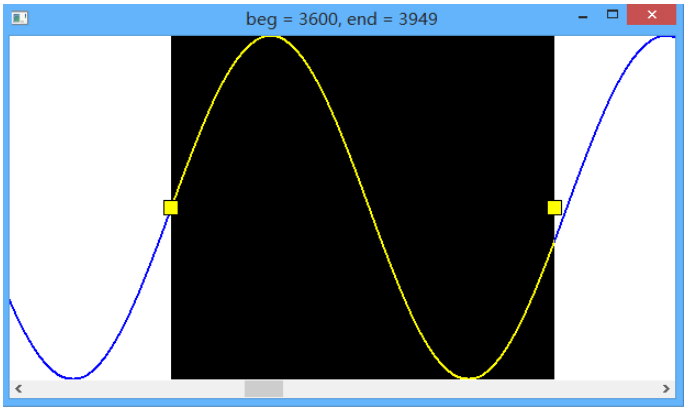

Figure 8. A graphic selection example
图 8. 图形选择示例

4. 性能评估

这里以某单脉冲雷达数据处理为例，对本文设计的可视化数据处理软件平台的性能进行评估，软件整体运行界面如图 9 所示。图 9 左半部分是文本窗，文本窗以行为单位显示数据。右边上、下是两个波形图，波形图水平轴表示时间，竖直轴表示数据值，两个波形图曲线已经处于放大状态。实际测试结果表明，与目前基于 MATLAB 的软件平台相比，利用本文基于可视化技术设计的雷达数据处理软件平台，处理相同雷达数据量所需时间大约减少了一半，证明了本文设计平台的有效性。

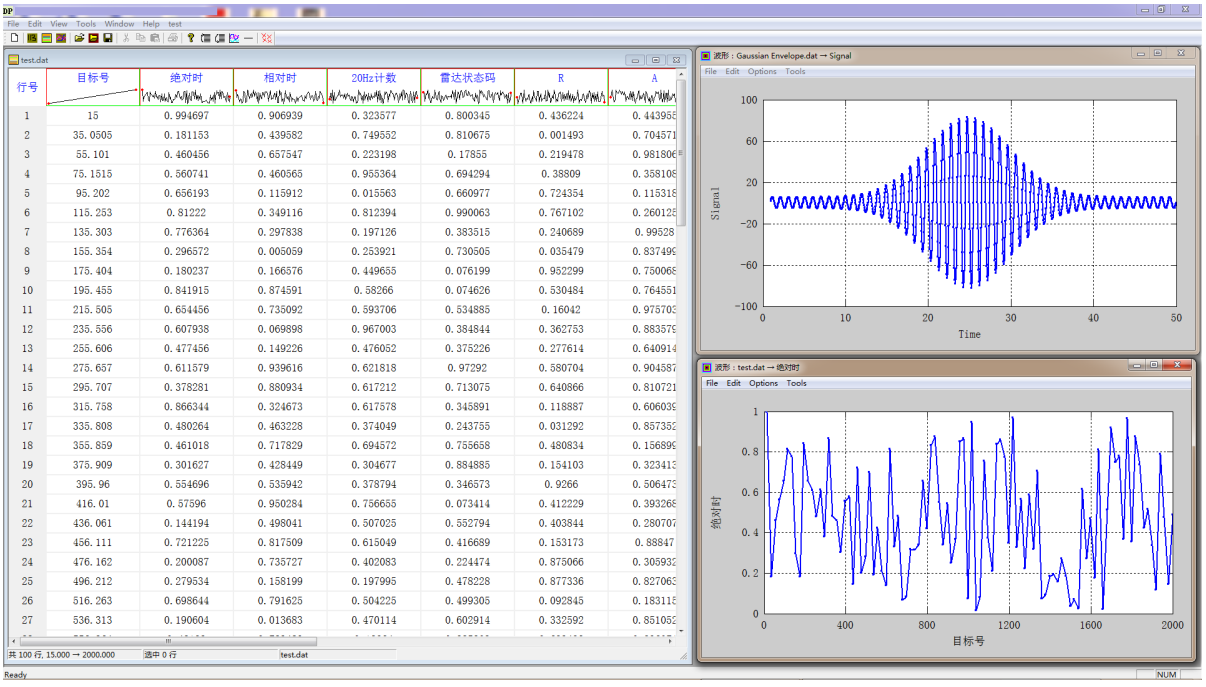

Figure 9. The visual interface of the data processing platform
图 9. 可视化数据处理平台运行界面

5. 结束语

面对日益增多的雷达数据，提高数据处理速度变得越来越迫切。本文以波形图和文本窗为重点，设

计实现了雷达数据处理软件平台。波形图用来对图形平移、缩放、查看数据点，并可利用鼠标拖动方式选择图形数据，其设计包含注册窗口类、创建窗口、实现消息分发机制。文本窗用来以行为单位对雷达数据显示、编辑，使用快捷菜单实现数据处理功能。以波形图为例介绍了快速高效绘制算法和图形选择算法。经实践检验，本文设计的可视化雷达数据处理软件平台可以减轻人们雷达数据处理的工作量，数据处理速度大约是传统方法的两倍。

参考文献

[1] Skolnik, M.I. (2007) Introduction to Radar Systems. 3rd Edition, Publishing House of Electronics Industry, Beijing, 772.

[2] 张威. MATLAB 外部接口编程[M]. 第 1 版. 西安: 西安电子科技大学出版社, 2004: 208.

[3] 霍亚飞. Qt Creator 快速入门[M]. 第 3 版. 北京: 北京航空航天大学出版社, 2017: 515.

[4] 周明扬. Visual C++界面编程技术[M]. 第 1 版. 北京: 北京希望电子出版社, 2003.

[5] Horton, I. Visual C++ 2012 入门经典[M]. 第 6 版. 北京: 清华大学出版社, 2013.

[6] Lippman, S.B., Lajoie, J. and Moo, B.E. (2007) C++ Primer. 4th Edition. Post & Telecom Press Co., LTD., Beijing, 885.

[7] Richter, J., Nasarre, C. Windows 核心编程[M]. 第 1 版. 北京: 清华大学出版社, 2008: 770.

[8] Microsoft Corporation. The October 2001 Release of the MSDN Library [DB/CD]. 2001.

[9] 李婷, 张志民, 李红梅. 火箭飞行任务动态故障树分析与管理系统设计与实现[J]. 东风航天, 2017, 29(5): 52-55.

[10] 李婷, 张志民, 贾森林. 基于“计算机+人”联合判读的火箭故障智能诊断系统设计[J]. 导弹试验技术, 2016(2): 1-7.

[11] 张志民, 欧建平, 皇甫堪. 一种 Windows 下实时高效图形绘制算法[J]. 现代电子技术, 2010(16): 43-46.

[12] 程佩青. 数字信号处理教程 MATLAB 版[M]. 第 5 版. 北京: 清华大学出版社, 2017: 675.

[13] Proakis, J. G., Manolakis, D. G. 数字信号处理: 原理、算法与应用[M]. 第 3 版. 张晓林, 译. 北京: 电子工业出版社, 2004: 827.