报告文档

1 程序优化性说明

1.1 用户交互界面说明

程序采用 C# WinForm 框架进行开发，界面风格采用标准 Windows 应用程序，界面从上到下、从左到右依次为：标题栏、菜单栏、工具栏、数据展示区、报告预览区、状态栏。整个界面布局清晰、设计操作人性化。具体如图 1 所示。

图 1 程序界面

1.2 程序运行过程说明

（1） 点击【打开】，选择原始数据文件，点击【确定】后将读取数据文件并展示到界面。

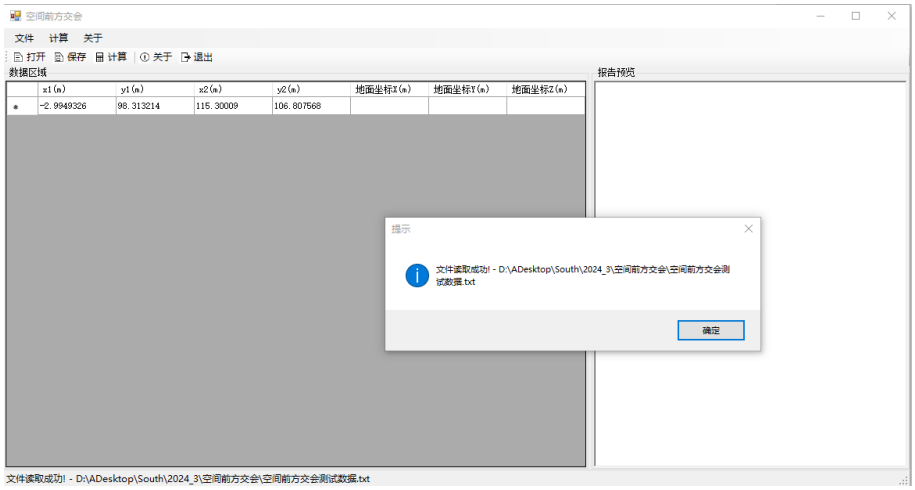

图 2 打开数据文件

（2） 点击【计算】，程序将根据读取的数据进行空间前方交会计算，并生成报告预览。

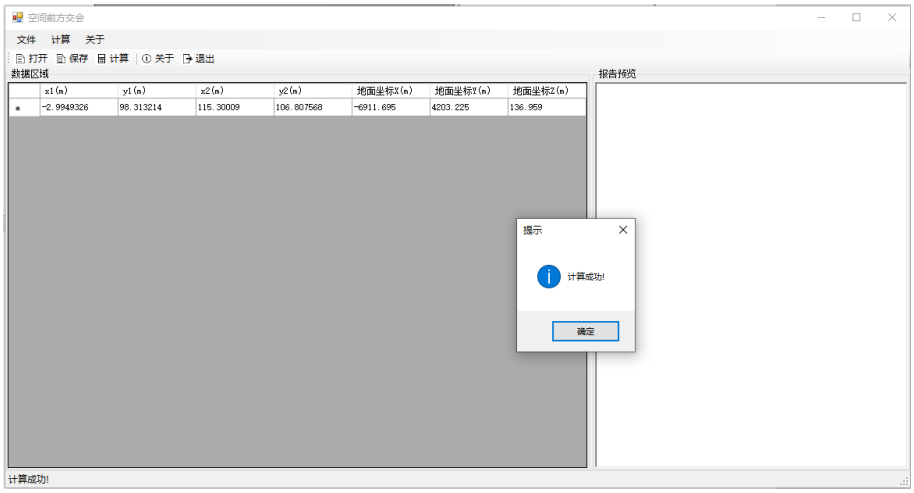

图 3 计算成功

（3） 点击【保存】，选择保存文件路径，点击【确定】后将保存【报告预览】区域内容到文件。

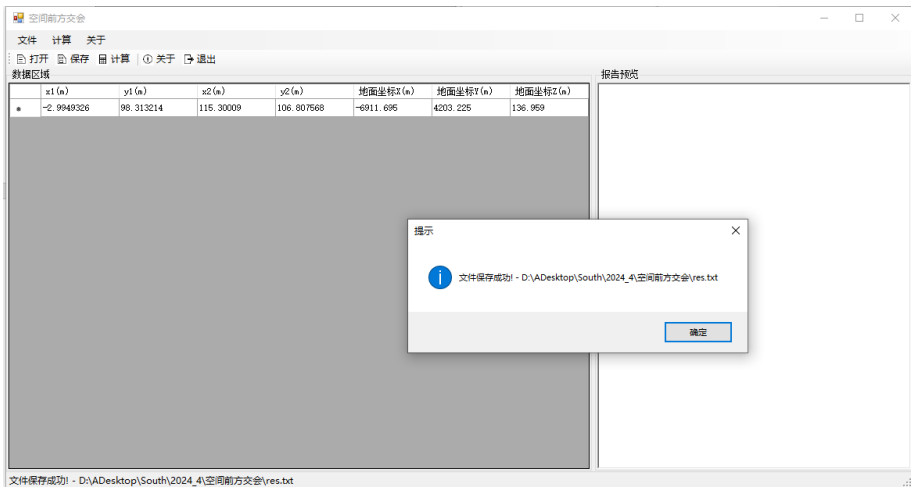

图 4 保存报告

（4） 点击【退出】，经二次确认后，退出程序。

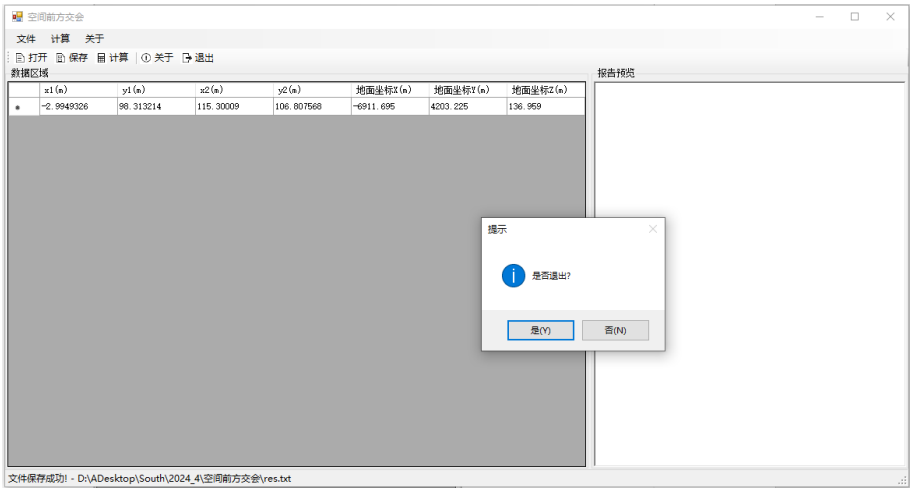

图 5 退出程序

1.3 程序运行结果

2 程序规范性说明

2.1 程序功能与结构设计说明

程序可以实现从文件读取数据到界面，进行空间前方交会计算，保存报告等功能。具体功能按菜单栏列出见表 1。

表 1 程序功能

选项	功能简介
打开	用户选择原始数据文件后，读取数据文件并展示到界面
保存	用户选择保存路径后，保存报告内容到文件
打开数据文件夹	如果用户打开过数据文件，则打开数据文件所在的文件夹，否则给出相应提示
打开报告文件夹	如果用户打开过报告文件，则打开报告文件所在的文件夹，否则给出相应提示
计算	进行空间后方交会计算并生成报告预览
关于	显示程序基本信息
帮助	打开帮助文档
退出	用户经二次确认后退出程序

程序共设计有 MyData、MyFile、MyPic、DoublePic 四个类，具体功能见表 2。

表 2 类的设计

类名	功能简介
MyData	存放维持窗体运行的必要数据，如 inFile、outFile
MyFile	读取文件和保存报告
MyPic	存储单个像片的信息，例如像片主距、像点坐标、模型基线分量、姿态元素
DoublePic	存放一对像片

2.2 核心算法源码

1. 计算像空间辅助坐标

```
/// <summary>
/// 计算像空间辅助坐标
/// </summary>
```

```
        private void Caluvw()
        {
            //计算旋转矩阵
            double sphi = Math.Sin(phi);
            double somega = Math.Sin(omega);
            double skappa = Math.Sin(kappa);

            double cphi = Math.Cos(phi);
            double comega = Math.Cos(omega);
            double ckappa = Math.Cos(kappa);

            a1 = cphi * ckappa - sphi * somega * skappa;
            a2 = -cphi * skappa - sphi * somega * skappa;
            a3 = -sphi * comega;
            b1 = comega * skappa;
            b2 = comega * ckappa;
            b3 = -somega;
            c1 = sphi * ckappa + cphi * somega * skappa;
            c2 = -somega * skappa + cphi * somega * ckappa;
            c3 = cphi * comega;
            //计算像点的像空间辅助坐标
            u = a1 * x + a2 * y + a3 * (-f);
            v = b1 * x + b2 * y + b3 * (-f);
            w = c1 * x + c2 * y + c3 * (-f);
        }
```

2. 计算投影系数

```
        /// <summary>
        /// 计算投影系数
        /// </summary>
        private void CalN()
        {
            //计算投影基线分量
            this.Bu = p2.Xs - p1.Xs;
            this.Bv = p2.Ys - p1.Ys;
            this.Bw = p2.Zs - p1.Zs;

            //计算投影系数
            double down = p1.u * p2.w - p2.u * p1.w;
            this.N1 = (Bu * p2.w - Bw * p2.u) / down;
            this.N2 = (Bu * p1.w - Bw * p1.u) / down;
        }
```

3. 计算地面坐标

```
        /// <summary>
```

```
        /// 计算地面坐标
        /// </summary>
        public void CalXYZ()
        {
            X = p1.Xs + N1 * p1.u;
            Y = 0.5 * ((p1.Ys + N1 * p1.v) + (p2.Ys + N2 * p2.v));
            Z = p1.Zs + N1 * p1.w;
        }
```