《基于子母机协同的高效铁轨检修机操作平台(V1.0)》源代码

武汉理工大学

2019 年 3 月

源程序代码：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
using System.IO.Ports;
using System.Threading;
using System.Threading.Tasks;
using System.Data.SqlClient;
using System.Globalization;
using AForge.Video.DirectShow;
using MathNet.Numerics;
using MathNet.Numerics.LinearAlgebra.Double;
using MathNet.Numerics.IntegralTransforms;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.Util;
using System.Numerics;

namespace HostComputerForRail
{
    public partial class Form1 : Form
    {
        private DateTime TimeStart = DateTime.Now;
        bool bool_start = false;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            try
            {
                //图像识别部分
                VideoCapture_ImageRecognize_Load();

                //系统信息部分
                timer_System.Start();

                //实时监控部分
                MonitorCamera_Load();

                //倾角仪传输部分
                comboBox_Inclinometer_Load();
                SerialPort_Inclinometer1.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(this.SerialPort_DataReceived1);
                SerialPort_Inclinometer2.DataReceived += new
```

```
System.IO.Ports.SerialDataReceivedEventHandler(this.SerialPort_DataReceived2);

            //底栏状态调整
            if (bool_haveCamera)
            {
                toolStripStatusLabel_Camera.Text = "摄像头已连接";
            }
            else
            {
                toolStripStatusLabel_Camera.Text = "摄像头未连接";
            }
        }
        catch (Exception ex)
        {
            toolStripStatusLabel_State.Text = "错误：" + ex.Message;
        }
    }

    private void pictureBox_Start_Click(object sender, EventArgs e)
    {
        try
        {
            bool_start = true;
            timer_Main.Start();
            timer_FFT.Start();
        }
        catch (Exception ex)
        {
            toolStripStatusLabel_State.Text = "错误：" + ex.Message;
        }
    }

    private void pictureBox_End_Click(object sender, EventArgs e)
    {
        try
        {
            bool_start = false;
            timer_Main.Stop();
            timer_FFT.Stop();
            Thread.Sleep(100);
        }
        catch (Exception ex)
        {
            toolStripStatusLabel_State.Text = "错误：" + ex.Message;
        }
    }

    private void timer_Main_Tick(object sender, EventArgs e)
    {
        try
```

```
                {
                    if (bool_start)
                    {
                        a_AfterTransform_Before = a_AfterTransform;
                        if (bool_startIntegrate)
                        {
                            DateTime_Before = DateTime.Now;
                        }
                        else
                        {
                            DateTime_Before = DateTime_Now;
                        }
                        velocity_Before = velocity;
                        DateTime_Now = DateTime.Now;
                        Transform();
                        RemoveInit();
                        Integration();
                        //TODO: 刷新数据
                        label_Inclinometer1_Ax.Text = a_AfterTransform[0, 0].ToString("F6");
                        label_Inclinometer1_Ay.Text = a_AfterTransform[0, 1].ToString("F6");
                        label_Inclinometer1_Az.Text = a_AfterTransform[0, 2].ToString("F6");
                        label_Inclinometer1_THETAx.Text = Angle[0, 0].ToString("F6");
                        label_Inclinometer1_THETAy.Text = Angle[0, 1].ToString("F6");
                        label_Inclinometer1_THETAz.Text = Angle[0, 2].ToString("F6");
                        label_Inclinometer2_Ax.Text = a_AfterTransform[1, 0].ToString("F6");
                        label_Inclinometer2_Ay.Text = a_AfterTransform[1, 1].ToString("F6");
                        label_Inclinometer2_Az.Text = a_AfterTransform[1, 2].ToString("F6");
                        label_Inclinometer2_THETAx.Text = Angle[1, 0].ToString("F6");
                        label_Inclinometer2_THETAy.Text = Angle[1, 1].ToString("F6");
                        label_Inclinometer2_THETAz.Text = Angle[1, 2].ToString("F6");
                        //TODO: 更改 label_IncrementalTime 的计算逻辑
                        label_IncrementalTime.Text = (DateTime.Now - TimeStart).TotalMilliseconds /
1000 + "";
                        if (!((a[0, 0] == 0 && a[0, 1] == 0 && a[0, 2] == 0 && Angle[0, 0] == 0 &&
Angle[0, 1] == 0 && Angle[0, 2] == 0) ||
                            (a[1, 0] == 0 && a[1, 1] == 0 && a[1, 2] == 0 && Angle[1, 0] == 0 &&
Angle[1, 1] == 0 && Angle[1, 2] == 0)))
                        {
                            chart1_Run();
                            statusStrip_Bottom.BackColor =
                                System.Drawing.Color.FromArgb(((int)(((byte)(0)))),
((int)(((byte)(122)))), ((int)(((byte)(20)))));
                            SQLconnect();
                            //TODO: 改变 toolStripStatusLabel_State.Image
                        }
                    }
                }
                catch (Exception ex)
                {
                    toolStripStatusLabel_State.Text = "错误：" + ex.Message;
```

```
            }
        }

        /*
         * ————————————————————————————————————————————————
————图像识别程序段————————————————————————————————————————————
————————
         */

        private VideoCapture VideoCapture_ImageRecognize;
        private Mat frame;
        private int index_ImageRecognize = 2;

        private void VideoCapture_ImageRecognize_Load()
        {
            try
            {
                VideoCapture_ImageRecognize = new VideoCapture(index_ImageRecognize);
                VideoCapture_ImageRecognize.ImageGrabbed += ProcessFrame;
                frame = new Mat();
                VideoCapture_ImageRecognize.Start();
            }
            catch (Exception ex)
            {
                toolStripStatusLabel_Camera.Text = "错误：" + ex.Message;
            }
        }

        private void ProcessFrame(object sender, EventArgs e)
        {
            try
            {
                if (VideoCapture_ImageRecognize != null && VideoCapture_ImageRecognize.Ptr !=
IntPtr.Zero)
                {
                    VideoCapture_ImageRecognize.Retrieve(frame, 0);
                    Image<Bgr, Byte> img = frame.ToImage<Bgr, Byte>();
                    Image<Gray, Byte> grayImage = img.Convert<Gray, Byte>();
                    Image<Gray, Byte> addImage = grayImage.ThresholdBinary(new Gray(65), new
Gray(255));
                    Image<Gray, Byte> cannyGray = grayImage.Canny(90, 150);
                    cannyGray = cannyGray.Not();
                    Image<Bgr, Byte> finalImage = img.Add(img,cannyGray);
                    imageBox1.Image = finalImage;
                    //imageBox1.Image = cannyGray;
                    //imageBox1.Image = frame;
                    //TODO: 如果 cannyGray 出现白色，将时间记录在数据库中
                }
            }
            catch (Exception ex)
```

```
                {
                    toolStripStatusLabel_Camera.Text = "错误：" + ex.Message;
                }
            }

            /*
             * ————————————————————————————————————————————————————————————————————————不平顺数据处理————————————————————————————————————————————————————————————————————————
             */
            private double[,] a_AfterTransform = new double[2, 3];
            private double[,] a_AfterTransform_Before = new double[2, 3];
            private double[,] velocity = new double[2, 3];
            private double[,] velocity_Before = new double[2, 3];
            private double[,] displacement = new double[2, 3];
            private bool bool_startIntegrate = false;
            private DateTime DateTime_Before;
            private DateTime DateTime_Now;
            private int zeroPointForVelocity;
            private double[,] error = new double[2, 3];     //零点漂移误差阈值
            private double[,] a_AfterTransform_Init = new double[2, 3]; //初始误差
            private List<double>[,] FixInit_Data = { { new List<double>(), new List<double>(), new List<double>() }, { new List<double>(), new List<double>(), new List<double>() } };

            private void Transform()
            {
                try
                {
                    for (int number = 0; number < 2; number++)
                    {
                        // 建立数组
                        MathNet.Numerics.LinearAlgebra.Matrix<double> a_Matrix = DenseMatrix.OfArray(new double[,] { { a[number, 0], a[number, 1], a[number, 2], 1 } });
                        MathNet.Numerics.LinearAlgebra.Matrix<double> Trx1 = DenseMatrix.OfArray(new double[,]
                        {
                            {1, 0, 0, 0},
                            {0, Math.Cos(Angle[number,0]*Math.PI/180), Math.Sin(Angle[number,0]*Math.PI/180), 0},
                            {0, -Math.Sin(Angle[number,0]*Math.PI/180), Math.Cos(Angle[number,0]*Math.PI/180), 0},
                            {0, 0, 0, 1},
                        });
                        MathNet.Numerics.LinearAlgebra.Matrix<double> Try1 = DenseMatrix.OfArray(new double[,]
                        {
                            {Math.Cos(Angle[number,1]*Math.PI/180), 0, -Math.Sin(Angle[number,1]*Math.PI/180), 0},
                            {0, 1, 0, 0},
                            {Math.Sin(Angle[number,1]*Math.PI/180), 0,
```

```
Math.Cos(Angle[number,1]*Math.PI/180), 0},
                                    {0, 0, 0, 1},
                                });
                                MathNet.Numerics.LinearAlgebra.Matrix<double> Trz1 =
DenseMatrix.OfArray(new double[,]
                                {
                                    {Math.Cos(Angle[number,2]*Math.PI/180),
Math.Sin(Angle[number,2]*Math.PI/180), 0, 0},
                                    {-Math.Sin(Angle[number,2]*Math.PI/180),
Math.Cos(Angle[number,2]*Math.PI/180), 0, 0},
                                    {0, 0, 1, 0},
                                    {0, 0, 0, 1},
                                });
                                MathNet.Numerics.LinearAlgebra.Matrix<double> Trx3 =
DenseMatrix.OfArray(new double[,]
                                {
                                    {1, 0, 0, 0},
                                    {0, Math.Cos(-Angle[number,0]*Math.PI/180), Math.Sin(-
Angle[number,0]*Math.PI/180), 0},
                                    {0, -Math.Sin(-Angle[number,0]*Math.PI/180), Math.Cos(-
Angle[number,0]*Math.PI/180), 0},
                                    {0, 0, 0, 1},
                                });
                                MathNet.Numerics.LinearAlgebra.Matrix<double> Try3 =
DenseMatrix.OfArray(new double[,]
                                {
                                    {Math.Cos(-Angle[number,1]*Math.PI/180), 0, -Math.Sin(-
Angle[number,1]*Math.PI/180), 0},
                                    {0, 1, 0, 0},
                                    {Math.Sin(-Angle[number,1]*Math.PI/180), 0, Math.Cos(-
Angle[number,1]*Math.PI/180), 0},
                                    {0, 0, 0, 1},
                                });
                                MathNet.Numerics.LinearAlgebra.Matrix<double> Trz3 =
DenseMatrix.OfArray(new double[,]
                                {
                                    {Math.Cos(-Angle[number,2]*Math.PI/180), Math.Sin(-
Angle[number,2]*Math.PI/180), 0, 0},
                                    {-Math.Sin(-Angle[number,2]*Math.PI/180), Math.Cos(-
Angle[number,2]*Math.PI/180), 0, 0},
                                    {0, 0, 1, 0},
                                    {0, 0, 0, 1},
                                });
                                // 三角变换
                                MathNet.Numerics.LinearAlgebra.Matrix<double> T1 = Trx1 * Try1 * Trz1;
                                MathNet.Numerics.LinearAlgebra.Matrix<double> T3 = Trz3 * Try3 * Trx3;
                                MathNet.Numerics.LinearAlgebra.Matrix<double> T2 =
DenseMatrix.OfArray(new double[,]
                                {
                                    {1, 0, 0, 0},
```

```
                    {0, 1, 0, 0},
                    {0, 0, 1, 0},
                    {0, 0, -1, 1}
                });
                MathNet.Numerics.LinearAlgebra.Matrix<double> T = T1 * T2 * T3;
                a_Matrix = a_Matrix * T;
                for (int i = 0; i < 3; i++)
                {
                    a_AfterTransform[number, i] = a_Matrix[0, i];
                }
            }
        }
        catch (Exception ex)
        {
            toolStripStatusLabel_Inclinometer.Text = "错误：" + ex.Message;
        }
    }

    private void Integration()
    {
        try
        {
            for (int i = 0; i < 2; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    // 积分求解位移和速度
                    velocity[i, j] = velocity_Before[i, j]
                        + 0.5 * (a_AfterTransform_Before[i, j] + a_AfterTransform[i, j]) *
(DateTime_Now - DateTime_Before).TotalMilliseconds / 1000;
                    displacement[i, j] = displacement[i, j]
                        + 0.5 * (a_AfterTransform_Before[i, j] + a_AfterTransform[i, j]) *
(DateTime_Now - DateTime_Before).TotalMilliseconds / 1000;
                    // 去除速度的零点漂移
                    if (a_AfterTransform[i, j] == 0)
                    {
                        zeroPointForVelocity++;
                    }
                    else
                    {
                        zeroPointForVelocity = 0;
                    }

                    if (zeroPointForVelocity >= 30)
                    {
                        velocity_Before[i, j] = 0;
                    }
                }
            }
        }
```

```
        catch (Exception ex)
        {
            toolStripStatusLabel_Inclinometer.Text = "错误：" + ex.Message;
        }
    }

    private void RemoveInit()
    {
        try
        {
            for (int i = 0; i < 2; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    //获得误差
                    a_AfterTransform[i, j] = a_AfterTransform[i, j] - a_AfterTransform_Init[i, j];
                    //去除零点漂移
                    if (Math.Abs(a_AfterTransform[i, j]) < error[i, j])
                    {
                        a_AfterTransform[i, j] = 0;
                    }
                }
            }
        }
        catch (Exception ex)
        {
            toolStripStatusLabel_Inclinometer.Text = "错误：" + ex.Message;
        }
    }

    //频域分析
    private static int Samples_num = 250;
    readonly Complex[] sample_Ay1 = new Complex[Samples_num];
    private Complex[] sample_Ay2 = new Complex[Samples_num];
    private Complex[] sample_Az1 = new Complex[Samples_num];
    private Complex[] sample_Az2 = new Complex[Samples_num];
    private Complex[] sample_θx1 = new Complex[Samples_num];
    private Complex[] sample_θx2 = new Complex[Samples_num];
    private int add_num = 0;
    private double[,] sample_data = new double[6, Samples_num];

    Thread thread_sample;
    private delegate void delegate_FFT();

    private void FFT(/*object state*/)
    {
        Fourier.Forward(sample_Ay1);
    }

    private void PlotFftAnalys()
```

```
        {
            BeginInvoke(new delegate_FFT(label3_Start));
            timer_FFT.Stop();
            for (int i = 0; i < Samples_num; i++)
            {
                sample_Ay1[i] = new Complex(sample_data[0, i], 0);
            }
            BeginInvoke(new delegate_FFT(label3_Doing));
            try
            {
                FFT();
            }
            catch (Exception ex)
            {
                MessageBox.Show("" + ex.Message);
            }
            BeginInvoke(new delegate_FFT(label3_Finish));
            BeginInvoke(new delegate_FFT(UIchange_FFT));
        }

        private void label3_Start()
        {
            label3.Text = "初始化中";
        }

        private void label3_Doing()
        {
            label3.Text = "FFT 计算中";
        }

        private void label3_Finish()
        {
            label3.Visible = false;
        }

        private void UIchange_FFT()
        {
            crtFft.Series["Frequency"].Points.Clear();

            for (int i = 0; i < sample_Ay1.Length / 4; i++)
            {
                double mag = (2.0 / Samples_num) *
(Math.Abs(Math.Sqrt(Math.Pow(sample_Ay1[i].Real, 2) +
                                                    Math.Pow(sample_Ay1[i].Imaginary, 2))));

                double hzPerSample = 20 / Samples_num;

                crtFft.Series["Frequency"].Points.AddXY(hzPerSample * i, mag);
            }
        }
```

```
private void timer_FFT_Tick(object sender, EventArgs e)
{
    if (add_num < Samples_num)
    {
        sample_data[0, add_num] = a[0, 1];
        sample_data[1, add_num] = a[1, 1];
        sample_data[2, add_num] = a[0, 2];
        sample_data[3, add_num] = a[1, 2];
        sample_data[4, add_num] = Angle[0, 0];
        sample_data[5, add_num] = Angle[1, 0];
    }
    else
    {
        for (int i = 0; i < Samples_num - 1; i++)
        {
            for (int j = 0; j < 6; j++)
            {
                sample_data[j, i] = sample_data[j, i + 1];
            }
            sample_data[0, Samples_num - 1] = a[0, 1];
            sample_data[1, Samples_num - 1] = a[1, 1];
            sample_data[2, Samples_num - 1] = a[0, 2];
            sample_data[3, Samples_num - 1] = a[1, 2];
            sample_data[4, Samples_num - 1] = Angle[0, 0];
            sample_data[5, Samples_num - 1] = Angle[1, 0];
        }
        add_num = 0;
        label3.Text = "正在处理";
        thread_sample = new Thread(PlotFftAnalys);
        thread_sample.IsBackground = true;
        thread_sample.Start();
    }
    add_num += 1;
}

private void pictureBox_fixSensor_Click(object sender, EventArgs e)
{
    try
    {
        timer_Sensor.Start();
        timer_FixInit.Start();

        label_Inclinometer1_Ax.Text = "倾角仪正在校准";
        label_Inclinometer1_Ay.Text = "倾角仪正在校准";
        label_Inclinometer1_Az.Text = "倾角仪正在校准";
        label_Inclinometer1_THETAx.Text = "倾角仪正在校准";
        label_Inclinometer1_THETAy.Text = "倾角仪正在校准";
        label_Inclinometer1_THETAz.Text = "倾角仪正在校准";
        label_Inclinometer2_Ax.Text = "倾角仪正在校准";
```

```
            label_Inclinometer2_Ay.Text = "倾角仪正在校准";
            label_Inclinometer2_Az.Text = "倾角仪正在校准";
            label_Inclinometer2_THETAx.Text = "倾角仪正在校准";
            label_Inclinometer2_THETAy.Text = "倾角仪正在校准";
            label_Inclinometer2_THETAz.Text = "倾角仪正在校准";
        }
        catch (Exception ex)
        {
            toolStripStatusLabel_Inclinometer.Text = "错误：" + ex.Message;
        }
    }

    private void timer_FixInit_Tick(object sender, EventArgs e)
    {
        try
        {
            timer_Sensor.Stop();
            for (int i = 0; i < 2; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    // 获取样本的无偏估计，及其标准偏差
                    double sum = 0;
                    double sigma = 0;
                    int k = 0;
                    a_AfterTransform_Init[i, j] = FixInit_Data[i, j].Average();

                    foreach (double x in FixInit_Data[i, j])
                    {
                        sum = sum + x;
                        sigma = sigma + Math.Pow((x - a_AfterTransform_Init[i, j]), 2);
                        k++;
                    }
                    error[i, j] = 3 * Math.Sqrt(sigma / ((k - 1) * k)) * 12;
                    FixInit_Data[i, j].Clear();
                }
            }
            timer_FixInit.Stop();
            label_Inclinometer1_Ax.Text = a_AfterTransform_Init[0, 0] + "";
            label_Inclinometer1_Ay.Text = a_AfterTransform_Init[0, 1] + "";
            label_Inclinometer1_Az.Text = a_AfterTransform_Init[0, 2] + "";
            label_Inclinometer1_THETAx.Text = "校准完成";
            label_Inclinometer1_THETAy.Text = "校准完成";
            label_Inclinometer1_THETAz.Text = "校准完成";
            label_Inclinometer2_Ax.Text = a_AfterTransform_Init[1, 0] + "";
            label_Inclinometer2_Ay.Text = a_AfterTransform_Init[1, 1] + "";
            label_Inclinometer2_Az.Text = a_AfterTransform_Init[1, 2] + "";
            label_Inclinometer2_THETAx.Text = "校准完成";
            label_Inclinometer2_THETAy.Text = "校准完成";
            label_Inclinometer2_THETAz.Text = "校准完成";
```

```
            }
            catch (Exception ex)
            {
                toolStripStatusLabel_Inclinometer.Text = "错误：" + ex.Message;
            }

        }

        private void timer_Sensor_Tick(object sender, EventArgs e)
        {
            try
            {
                Transform();
                for (int i = 0; i < 2; i++)
                {
                    for (int j = 0; j < 3; j++)
                    {
                        FixInit_Data[i, j].Add(a_AfterTransform[i, j]);
                    }
                }
            }
            catch (Exception ex)
            {
                toolStripStatusLabel_Inclinometer.Text = "错误：" + ex.Message;
            }
        }

        private void pictureBox1_Click_1(object sender, EventArgs e)
        {
            label3.Text = "正在处理";
            thread_sample = new Thread(PlotFftAnalys);
            thread_sample.IsBackground = true;
            thread_sample.Start();
            //PlotFftAnalys();
        }
        /*
         * ————————————————————————————————————————————————————
————数据库传输部分————————————————————————————————————————————
————————
         */
        private string sqlDate;

        private void SQLconnect()
        {
            // SQL server
            string connsql = "server=FU-QINGCHEN\\SQLEXPRESS;integrated
security=SSPI;database=Test";
            try
            {
                using (SqlConnection mySQL = new SqlConnection())
```

```
                {
                    mySQL.ConnectionString = connsql;
                    // 打开数据库连接
                    mySQL.Open();
                    // 向数据库中插入数据
                    var format = "yyyy-MM-dd HH:mm:ss:ffffff";
                    var stringDate = DateTime.Now.ToString(format);
                    var convertedBack = DateTime.ParseExact(stringDate, format,
CultureInfo.InvariantCulture);
                    sqlDate = "insert Inclination_OriginDate(DateTimes,"
                        + "Accelerate1_X,Accelerate1_Y,Accelerate1_Z," +
"Inclination1_X,Inclination1_Y,Inclination1_Z,"
                        + "Accelerate2_X,Accelerate2_Y,Accelerate2_Z," +
"Inclination2_X,Inclination2_Y,Inclination2_Z"
                        + ")values(SYSDATETIME(),"
                        + a_AfterTransform[0, 0] + "," + a_AfterTransform[0, 1] + "," +
a_AfterTransform[0, 2] + "," + Angle[0, 0] + "," + Angle[0, 1] + "," + Angle[0, 2] + ","
                        + a_AfterTransform[1, 0] + "," + a_AfterTransform[1, 1] + "," +
a_AfterTransform[1, 2] + "," + Angle[1, 0] + "," + Angle[1, 1] + "," + Angle[1, 2]
                        + ")";
                    // 建立一个命令
                    SqlCommand sqlCommand = new SqlCommand(sqlDate, mySQL);
                    // 执行命令
                    sqlCommand.ExecuteNonQuery();
                }
            }
            catch (Exception ex)
            {
                toolStripStatusLabel_SQL.Text = "错误：" + ex.Message;
            }
            finally
            {
                toolStripStatusLabel_SQL.Text = "数据库已连接";
            }

            // Azure SQL
            SqlConnectionStringBuilder sqlConnectionStringBuilder_Azure = new
SqlConnectionStringBuilder();
            sqlConnectionStringBuilder_Azure.DataSource =
"mysampleserver.database.chinacloudapi.cn";
            sqlConnectionStringBuilder_Azure.UserID = "WHUT";
            sqlConnectionStringBuilder_Azure.Password = "0121618380615Fqc";
            sqlConnectionStringBuilder_Azure.InitialCatalog = "RailOriginDate";
            try
            {
                using(SqlConnection sqlConnection = new
SqlConnection(sqlConnectionStringBuilder_Azure.ConnectionString))
                {
                    sqlConnection.Open();
                    var format = "yyyy-MM-dd HH:mm:ss:ffffff";
```

```
                    var stringDate = DateTime.Now.ToString(format);
                    var convertedBack = DateTime.ParseExact(stringDate, format,
CultureInfo.InvariantCulture);
                    sqlDate = "insert Inclination_OriginDate(DateTimes,"
                        + "Accelerate1_X,Accelerate1_Y,Accelerate1_Z," +
"Inclination1_X,Inclination1_Y,Inclination1_Z,"
                        + "Accelerate2_X,Accelerate2_Y,Accelerate2_Z," +
"Inclination2_X,Inclination2_Y,Inclination2_Z"
                        + ")values(SYSDATETIME(),"
                        + a_AfterTransform[0, 0] + "," + a_AfterTransform[0, 1] + "," +
a_AfterTransform[0, 2] + "," + Angle[0, 0] + "," + Angle[0, 1] + "," + Angle[0, 2] + ","
                        + a_AfterTransform[1, 0] + "," + a_AfterTransform[1, 1] + "," +
a_AfterTransform[1, 2] + "," + Angle[1, 0] + "," + Angle[1, 1] + "," + Angle[1, 2]
                        + ")";
                    SqlCommand sqlCommand = new SqlCommand(sqlDate, sqlConnection);
                    sqlCommand.ExecuteNonQuery();
                }
            }
            catch(Exception ex)
            {
                toolStripStatusLabel_SQL.Text = "错误：" + ex.Message;
            }
            finally
            {
                toolStripStatusLabel_SQL.Text = "数据库已连接";
            }
        }

        /*
         * ————————————————————————————————————————————————————————
————倾角仪传输部分————————————————————————————————————————————————
————————
         */
        private bool bool_Inclinometer1 = false, bool_Inclinometer2 = false;
        string[] serialPortName;
        SerialPort SerialPort_Inclinometer1 = new SerialPort();
        SerialPort SerialPort_Inclinometer2 = new SerialPort();
        private double[,] a = new double[2, 3], Angle = new double[2, 3];
        private int serialPortNumber;

        //查询串口并加载
        private void comboBox_Inclinometer_Load()
        {
            serialPortName = SerialPort.GetPortNames();
            if (serialPortName == null)
            {
                toolStripStatusLabel_Inclinometer.Text = "无串口连接";
            }
            else
            {
```

```
                foreach (string name in serialPortName)
                {
                    comboBox_Inclinometer1.Items.Add(name);
                    comboBox_Inclinometer1.SelectedIndex = -1;
                    comboBox_Inclinometer2.Items.Add(name);
                    comboBox_Inclinometer2.SelectedIndex = -1;
                }
                toolStripStatusLabel_Inclinometer.Text = "请选择倾角仪串口";
            }
            SerialPort_Inclinometer1.BaudRate = 115200;
            SerialPort_Inclinometer2.BaudRate = 115200;
        }

        //关闭串口，释放资源
        private void serialPort_Close(SerialPort serialPort)
        {
            if (serialPort.IsOpen == true)
            {
                serialPort.Dispose();
                serialPort.Close();
            }
        }

        private void comboBox_Inclinometer1_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (comboBox_Inclinometer1.SelectedIndex >= 0)
            {
                SerialPort_Inclinometer1.PortName =
serialPortName[comboBox_Inclinometer1.SelectedIndex];
            }

            bool_Inclinometer1 = true;
            if (bool_Inclinometer2 == true)
            {
                toolStripStatusLabel_Inclinometer.Text = "倾角仪已连接";
            }

            serialPort_Close(SerialPort_Inclinometer1);
            SerialPort_Inclinometer1.Open();

            //timer_Main.Start();
        }

        private void comboBox_Inclinometer2_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (comboBox_Inclinometer2.SelectedIndex >= 0)
            {
                SerialPort_Inclinometer2.PortName =
serialPortName[comboBox_Inclinometer2.SelectedIndex];
            }
```

```
            bool_Inclinometer2 = true;
            if (bool_Inclinometer1 == true)
            {
                toolStripStatusLabel_Inclinometer.Text = "倾角仪已连接";
            }

            serialPort_Close(SerialPort_Inclinometer2);
            SerialPort_Inclinometer2.Open();
            //timer_Main.Start();
        }

        //以下获取串口数据部分，改编于传感器厂商开源代码
        delegate void UpdateData1(byte[] byteData);//声明一个委托
        delegate void UpdateData2(byte[] byteData);//声明一个委托
        byte[] RxBuffer1 = new byte[1000];
        byte[] RxBuffer2 = new byte[1000];
        UInt16 usRxLength1 = 0;
        UInt16 usRxLength2 = 0;
        private double[] LastTime1 = new double[10];
        private double[] LastTime2 = new double[10];

        //接收数据
        private void SerialPort_DataReceived1(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
        {
            byte[] byteTemp = new byte[1000];
            UInt16 usLength = 0;
            usLength = (UInt16)SerialPort_Inclinometer1.Read(RxBuffer1, usRxLength1, 700);
            usRxLength1 += usLength;
            while (usRxLength1 >= 11)
            {
                UpdateData1 Update = new UpdateData1(DecodeData1);
                RxBuffer1.CopyTo(byteTemp, 0);
                if (!((byteTemp[0] == 0x55) & ((byteTemp[1] & 0x50) == 0x50)))
                {
                    for (int i = 1; i < usRxLength1; i++) RxBuffer1[i - 1] = RxBuffer1[i];
                    usRxLength1--;
                    continue;
                }
                if (((byteTemp[0] + byteTemp[1] + byteTemp[2] + byteTemp[3] + byteTemp[4] +
byteTemp[5] + byteTemp[6] + byteTemp[7] + byteTemp[8] + byteTemp[9]) & 0xff) == byteTemp[10])
                    this.Invoke(Update, byteTemp);
                for (int i = 11; i < usRxLength1; i++) RxBuffer1[i - 11] = RxBuffer1[i];
                usRxLength1 -= 11;
            }
            Thread.Sleep(10);
        }
        private void SerialPort_DataReceived2(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
```

```
        {
            byte[] byteTemp = new byte[1000];
            UInt16 usLength = 0;
            usLength = (UInt16)SerialPort_Inclinometer2.Read(RxBuffer2, usRxLength2, 700);
            usRxLength2 += usLength;
            while (usRxLength2 >= 11)
            {
                UpdateData2 Update2 = new UpdateData2(DecodeData2);
                RxBuffer2.CopyTo(byteTemp, 0);
                if (!((byteTemp[0] == 0x55) & ((byteTemp[1] & 0x50) == 0x50)))
                {
                    for (int i = 1; i < usRxLength2; i++) RxBuffer2[i - 1] = RxBuffer2[i];
                    usRxLength2--;
                    continue;
                }
                if (((byteTemp[0] + byteTemp[1] + byteTemp[2] + byteTemp[3] + byteTemp[4] +
byteTemp[5] + byteTemp[6] + byteTemp[7] + byteTemp[8] + byteTemp[9]) & 0xff) == byteTemp[10])
                    this.Invoke(Update2, byteTemp);
                for (int i = 11; i < usRxLength2; i++) RxBuffer2[i - 11] = RxBuffer2[i];
                usRxLength2 -= 11;
            }
            Thread.Sleep(10);
        }

        //解码数据
        private void DecodeData1(byte[] byteTemp)
        {
            serialPortNumber = 0;
            double[] Data = new double[4];
            double TimeElapse = (DateTime.Now - TimeStart).TotalMilliseconds / 1000;

            Data[0] = BitConverter.ToInt16(byteTemp, 2);
            Data[1] = BitConverter.ToInt16(byteTemp, 4);
            Data[2] = BitConverter.ToInt16(byteTemp, 6);
            Data[3] = BitConverter.ToInt16(byteTemp, 8);

            switch (byteTemp[1])
            {
                case 0x51:    //加速度输出
                    Data[0] = Data[0] / 32768.0 * 16;
                    Data[1] = Data[1] / 32768.0 * 16;
                    Data[2] = Data[2] / 32768.0 * 16;

                    a[serialPortNumber, 0] = Data[0];
                    a[serialPortNumber, 1] = Data[1];
                    a[serialPortNumber, 2] = Data[2];
                    if ((TimeElapse - LastTime1[1]) < 0.1) return;
                    LastTime1[1] = TimeElapse;
                    break;
                case 0x53:    //角度输出
```

```
                Data[0] = Data[0] / 32768.0 * 180;
                Data[1] = Data[1] / 32768.0 * 180;
                Data[2] = Data[2] / 32768.0 * 180;
                Angle[serialPortNumber, 0] = Data[0];
                Angle[serialPortNumber, 1] = Data[1];
                Angle[serialPortNumber, 2] = Data[2];
                if ((TimeElapse - LastTime1[3]) < 0.1) return;
                LastTime1[3] = TimeElapse;
                break;
            default:
                break;
        }
    }
    private void DecodeData2(byte[] byteTemp)
    {
        serialPortNumber = 1;
        double[] Data = new double[4];
        double TimeElapse = (DateTime.Now - TimeStart).TotalMilliseconds / 1000;

        Data[0] = BitConverter.ToInt16(byteTemp, 2);
        Data[1] = BitConverter.ToInt16(byteTemp, 4);
        Data[2] = BitConverter.ToInt16(byteTemp, 6);
        Data[3] = BitConverter.ToInt16(byteTemp, 8);

        switch (byteTemp[1])
        {
            case 0x51:    //加速度输出
                Data[0] = Data[0] / 32768.0 * 16;
                Data[1] = Data[1] / 32768.0 * 16;
                Data[2] = Data[2] / 32768.0 * 16;

                a[serialPortNumber, 0] = Data[0];
                a[serialPortNumber, 1] = Data[1];
                a[serialPortNumber, 2] = Data[2];
                if ((TimeElapse - LastTime2[1]) < 0.1) return;
                LastTime2[1] = TimeElapse;
                break;
            case 0x53:    //角度输出
                Data[0] = Data[0] / 32768.0 * 180;
                Data[1] = Data[1] / 32768.0 * 180;
                Data[2] = Data[2] / 32768.0 * 180;
                Angle[serialPortNumber, 0] = Data[0];
                Angle[serialPortNumber, 1] = Data[1];
                Angle[serialPortNumber, 2] = Data[2];
                if ((TimeElapse - LastTime2[3]) < 0.1) return;
                LastTime2[3] = TimeElapse;
                break;
            default:
                break;
        }
```

```
            }
            /*
         * ————————————————————————————————————————————————————
————实时监控部分————————————————————————————————————————————————
————————————
         */

            private bool bool_haveCamera;       //判断是否有可用的摄像头
            private FilterInfoCollection VideoInputDeviceCollection;       //调出所有可用设备
            private VideoCaptureDevice VideoCaptureDevice_MonitorCamera;   //视频源

            private void MonitorCamera_Load()
            {
                try
                {
                    VideoInputDeviceCollection = new
FilterInfoCollection(FilterCategory.VideoInputDevice);
                    VideoCaptureDevice_MonitorCamera = new VideoCaptureDevice
                            (VideoInputDeviceCollection[1].MonikerString);
                    videoSourcePlayer_MonitorCamera.VideoSource =
VideoCaptureDevice_MonitorCamera;
                    videoSourcePlayer_MonitorCamera.Start();
                    toolStripStatusLabel_Camera.Text = "摄像头已连接";
                }
                catch (Exception ex)
                {
                    toolStripStatusLabel_Camera.Text = "错误：" + ex.Message;
                }
                finally
                {
                    bool_haveCamera = true;
                }
            }

            /*
             * ————————————————————————————————————————————————————
————————实时图表设计————————————————————————————————————————————————
————————————————
             */

            private bool frequency_start = false;
            private void chart1_Run()
            {
                chart1.Series[0].Points.AddXY(DateTime.Now.Millisecond.ToString(), a_AfterTransform[0,
1]);
                chart1.Series[1].Points.AddXY(DateTime.Now.Millisecond.ToString(), a_AfterTransform[1,
1]);
                chart1.Series[2].Points.AddXY(DateTime.Now.Millisecond.ToString(), a_AfterTransform[0,
2]);
                chart1.Series[3].Points.AddXY(DateTime.Now.Millisecond.ToString(), a_AfterTransform[1,
```

```
2]);
                chart1.Series[4].Points.AddXY(DateTime.Now.Millisecond.ToString(), Angle[0, 0]);
                chart1.Series[5].Points.AddXY(DateTime.Now.Millisecond.ToString(), Angle[1, 0]);
                if (chart1.Series[0].Points.Count >= 500)
                {
                    chart1.Series[0].Points.RemoveAt(0);
                    chart1.Series[1].Points.RemoveAt(0);
                    chart1.Series[2].Points.RemoveAt(0);
                    chart1.Series[3].Points.RemoveAt(0);
                    chart1.Series[4].Points.RemoveAt(0);
                    chart1.Series[5].Points.RemoveAt(0);
                    frequency_start = true;
                }
                if (frequency_start)
                {

                }
            }

            /*
             * ————————————————————————————————————————————————————————UI 界面设计————————————————————————————————————————————————————————
             */

            private void toolStripStatusLabel_ControlCenter_MouseEnter(object sender, EventArgs e)
            {
                toolStripStatusLabel_ControlCenter.ForeColor =
                    System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(122)))),
((int)(((byte)(204)))));
            }

            private void toolStripStatusLabel_ControlCenter_MouseLeave(object sender, EventArgs e)
            {
                toolStripStatusLabel_ControlCenter.ForeColor =
                    System.Drawing.Color.FromArgb(((int)(((byte)(153)))), ((int)(((byte)(153)))),
((int)(((byte)(153)))));
            }

            private void toolStripStatusLabel_OriginData_MouseEnter(object sender, EventArgs e)
            {
                toolStripStatusLabel_OriginData.ForeColor =
                    System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(122)))),
((int)(((byte)(204)))));
            }

            private void toolStripStatusLabel_OriginData_MouseLeave(object sender, EventArgs e)
            {
                toolStripStatusLabel_OriginData.ForeColor =
                    System.Drawing.Color.FromArgb(((int)(((byte)(153)))), ((int)(((byte)(153)))),
```

```
((int)(((byte)(153)))));
        }

        private void toolStripStatusLabel_DataSolve_MouseEnter(object sender, EventArgs e)
        {
            toolStripStatusLabel_DataSolve.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(122)))),
((int)(((byte)(204)))));
        }

        private void toolStripStatusLabel_DataSolve_MouseLeave(object sender, EventArgs e)
        {
            toolStripStatusLabel_DataSolve.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(153)))), ((int)(((byte)(153)))),
((int)(((byte)(153)))));
        }

        private void toolStripStatusLabel_ControlCenter_Click(object sender, EventArgs e)
        {
            toolStripStatusLabel3.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(122)))),
((int)(((byte)(204)))));
        }

        private void pictureBox1_Click(object sender, EventArgs e)
        {
            System.Environment.Exit(0);
        }

        private void pictureBox2_Click(object sender, EventArgs e)
        {
            this.WindowState = FormWindowState.Minimized;
        }

        private void toolStripStatusLabel_ControlCenter_Click_1(object sender, EventArgs e)
        {
            toolStripStatusLabel3.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(122)))),
((int)(((byte)(204)))));
            toolStripStatusLabel7.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(153)))), ((int)(((byte)(153)))),
((int)(((byte)(153)))));
            toolStripStatusLabel5.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(153)))), ((int)(((byte)(153)))),
((int)(((byte)(153)))));
        }

        private void toolStripStatusLabel3_Click(object sender, EventArgs e)
        {
            toolStripStatusLabel3.ForeColor =
```

```
                System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(122)))),
((int)(((byte)(204)))));
            toolStripStatusLabel7.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(153)))), ((int)(((byte)(153)))),
((int)(((byte)(153)))));
            toolStripStatusLabel5.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(153)))), ((int)(((byte)(153)))),
((int)(((byte)(153)))));
        }

        private void toolStripStatusLabel_OriginData_Click(object sender, EventArgs e)
        {
            toolStripStatusLabel7.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(122)))),
((int)(((byte)(204)))));
            toolStripStatusLabel3.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(153)))), ((int)(((byte)(153)))),
((int)(((byte)(153)))));
            toolStripStatusLabel5.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(153)))), ((int)(((byte)(153)))),
((int)(((byte)(153)))));
        }

        private void toolStripStatusLabel7_Click(object sender, EventArgs e)
        {
            toolStripStatusLabel7.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(122)))),
((int)(((byte)(204)))));
            toolStripStatusLabel3.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(153)))), ((int)(((byte)(153)))),
((int)(((byte)(153)))));
            toolStripStatusLabel5.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(153)))), ((int)(((byte)(153)))),
((int)(((byte)(153)))));
        }

        private void timer_System_Tick(object sender, EventArgs e)
        {
            label_SystemTime.Text = DateTime.Now + "";
        }

        private void toolStripStatusLabel_DataSolve_Click(object sender, EventArgs e)
        {
            toolStripStatusLabel5.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(122)))),
((int)(((byte)(204)))));
            toolStripStatusLabel7.ForeColor =
                System.Drawing.Color.FromArgb(((int)(((byte)(153)))), ((int)(((byte)(153)))),
((int)(((byte)(153)))));
            toolStripStatusLabel3.ForeColor =
```

```
                    System.Drawing.Color.FromArgb(((int)(((byte)(153)))), ((int)(((byte)(153)))),
((int)(((byte)(153)))));
                }

                private void toolStripStatusLabel5_Click(object sender, EventArgs e)
                {
                    toolStripStatusLabel5.ForeColor =
                        System.Drawing.Color.FromArgb(((int)(((byte)(0)))), ((int)(((byte)(122)))),
((int)(((byte)(204)))));
                    toolStripStatusLabel7.ForeColor =
                        System.Drawing.Color.FromArgb(((int)(((byte)(153)))), ((int)(((byte)(153)))),
((int)(((byte)(153)))));
                    toolStripStatusLabel3.ForeColor =
                        System.Drawing.Color.FromArgb(((int)(((byte)(153)))), ((int)(((byte)(153)))),
((int)(((byte)(153)))));
                }
            }
        }
```