Driver-Assistant Software Instruction

Jingwen Mo | sherrymo1997@gmail.com

Yiming Chen | sunshine980125@gmail.com

Supervised by Prof. Quoc-Viet Dang

August 28th, 2018

1. Introduction

1.1 Purpose

The Data-Graph Software is a PC based software developed to draw the graph which depicts different sensors for the racing car and show the GPS information. This instruction is for the software users and software tester.

1.2 Background

Racing car is one of the fiercest activities. It requires people to have access to different sensors in the car in order to evaluate the performance of the driver and train the driver by the feedback from the sensors. Our software is meant to help coach to transfer the data of different sensors to intuitive graphs easily.

2. Installation

(1) visual studio installation:
Users can just open the .sln file to load it into a visual studio workspace
(2) non-visual studio installation:
...\Data-Graph\DataG\Data_Graph\Debug\ Data_Graph.msi
You can download and install the software using “Data_Graph.msi”. This will lead to a setup wizard for the software.

3. Main Menu

The main form (**Figure 1 Main Menu**) has thee buttons: Single Run, Compared Run and Exit. Single Run Button can show data graph for single driver; Compared Run Button can show data graph for two different drivers; Exit Button can be employed to exit the software.

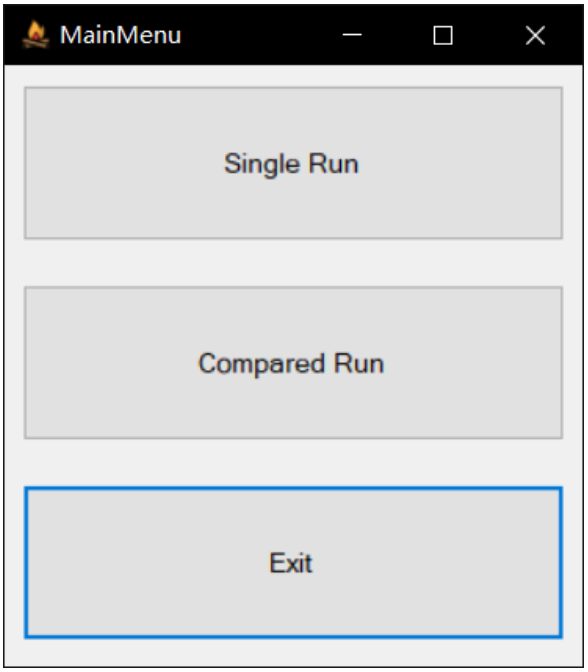

Figure 1 Main Menu

4. Single Run Form

3.1 Load CSV Files

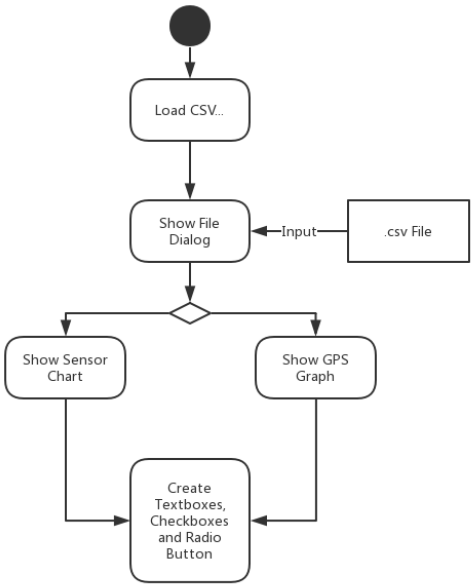

Figure 2 Flow Chart of Loading CSV Files

Call function of fileLoadingButton_Click and pop up a file dialog (**Figure 3 Opening File Dialog**) for users to choose their CSV file. After users choose their files and the

columns for latitude and longitude (**Figure 4 Choose Longitude and Latitude**), the software could draw graphs for different series of sensors and the GPS graph for the route of driving. Also, the software would dynamic create textboxes, checkboxes and radio buttons for different sensors read from the CSV file (**Figure 5 The Results of FIles Loading**).

Related Functions:

void fileLoadingButton_Click(object sender, EventArgs e)

DataTable OpenCSV(string filePath)

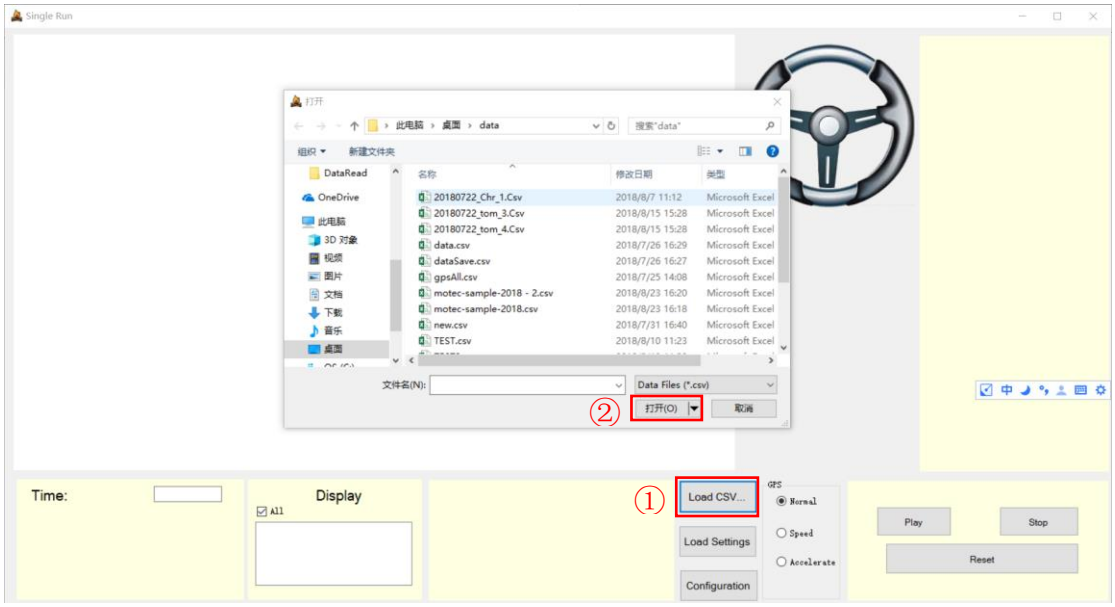

Figure 3 Opening File Dialog

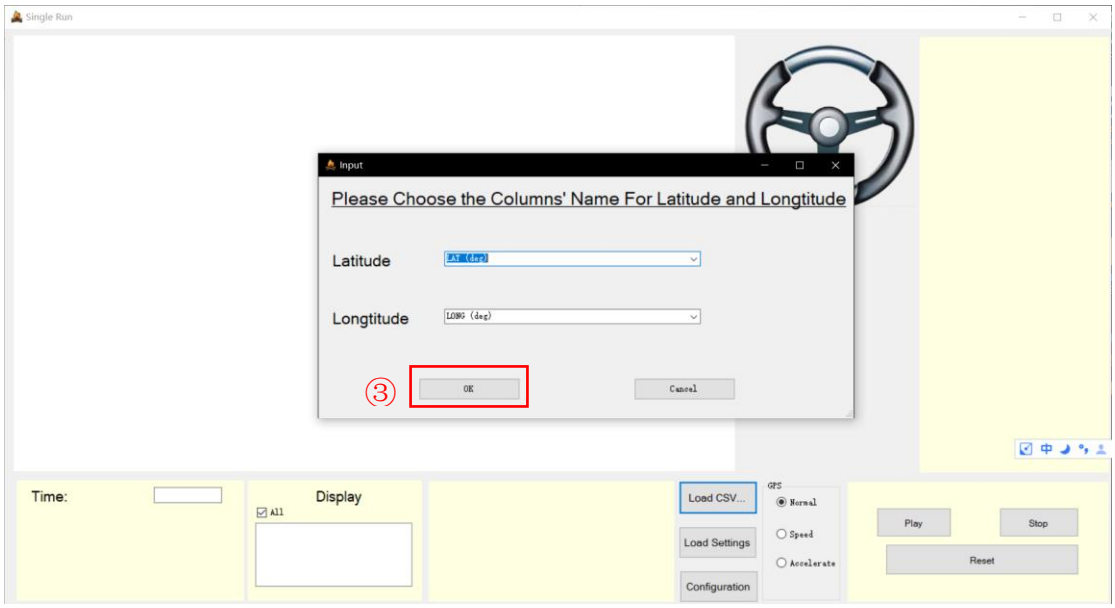

Figure 4 Choose Longitude and Latitude

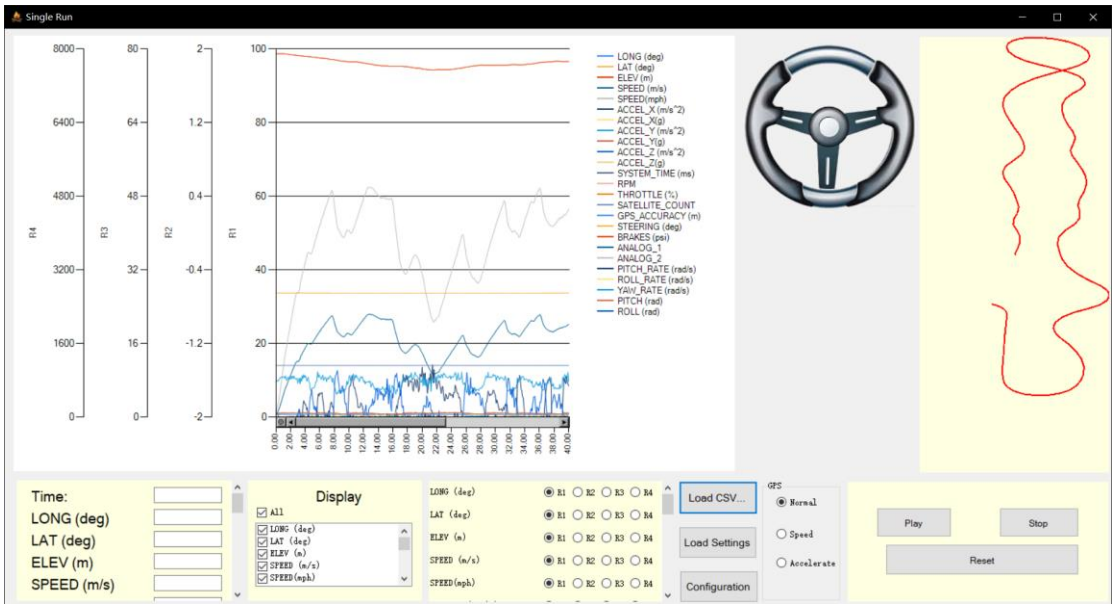

Figure 5 The Results of FIles Loading

3.2 Show Specific Value of Points

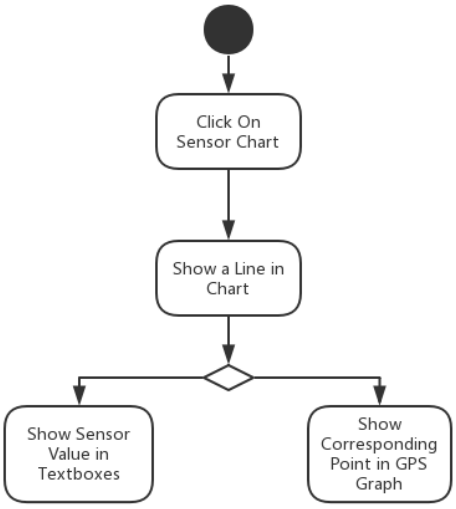

Figure 6 Flow Chart of Showing Specific Value of Points

Users could click on the graph and show a vertical line at where the mouse clicked (**Figure 7 Click on the Chart**). Then the textboxes below would show the values of different sensors. Meanwhile, there would be a black point in the GPS graph to show the same place as you click on the chart.

Related Functions:

```
void sensorChart_MouseClick(object sender, MouseEventArgs e)
int findLeftNear(double value, double[] array, int length)
```

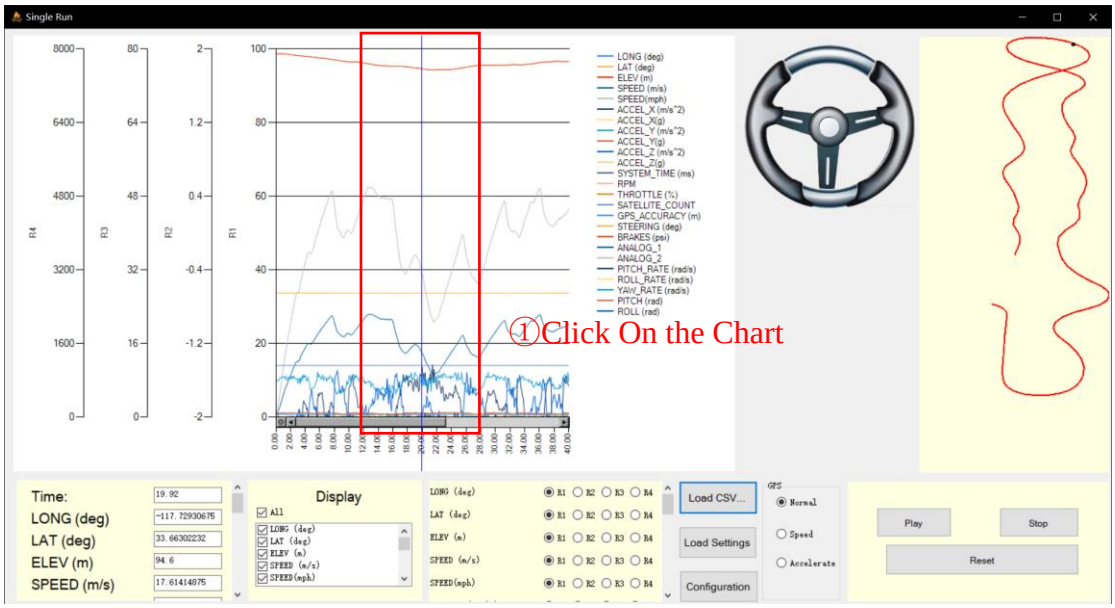

Figure 7 Click on the Chart

3.3 Replay the Data

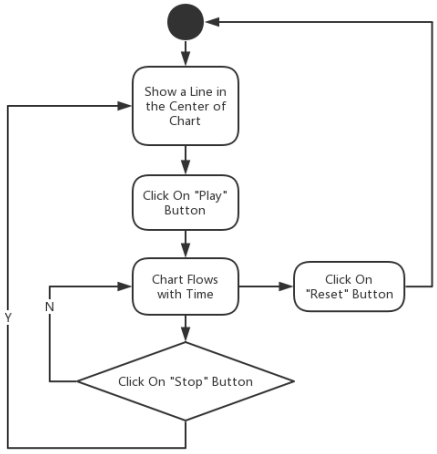

Figure 8 Flow Chart of Replaying the Data

Users could click on “Play” button (**Figure 9 Click on Play Button**) and call the function of buttonPlay_Click. This function enables chartTimer so that it could repeatedly execute the function of chartTimer_Tick at intervals. In the meantime, the textboxes below would show the values of sensors as the chart flowing. When users click on “Stop” button (**Figure 10 Click on Stop Button**), the chart would stop flowing. After “Play” button is re-clicked, the chart would begin to flow from where it stops. The “Reset” button would let users to reset the chart to the original state (**Figure 11 Click on Reset Button**).

Related Functions:

void buttonPlay_Click(object sender, EventArgs e)
void buttonStop_Click(object sender, EventArgs e)
void resetButton_Click(object sender, EventArgs e)
void chartTimer_Tick(object sender, EventArgs e)

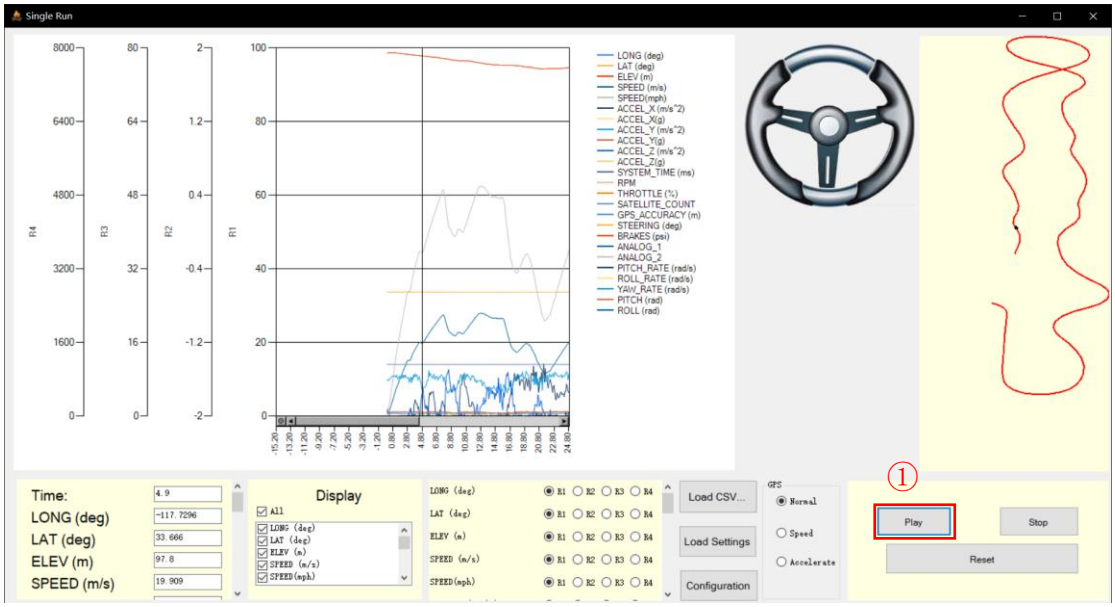

Figure 9 Click on Play Button

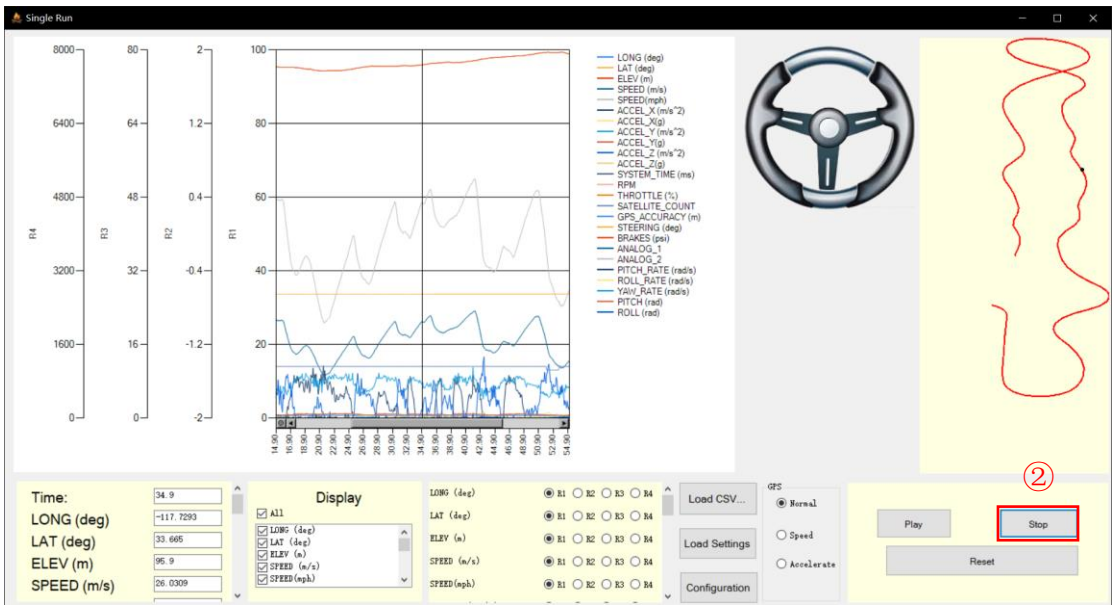

Figure 10 Click on Stop Button

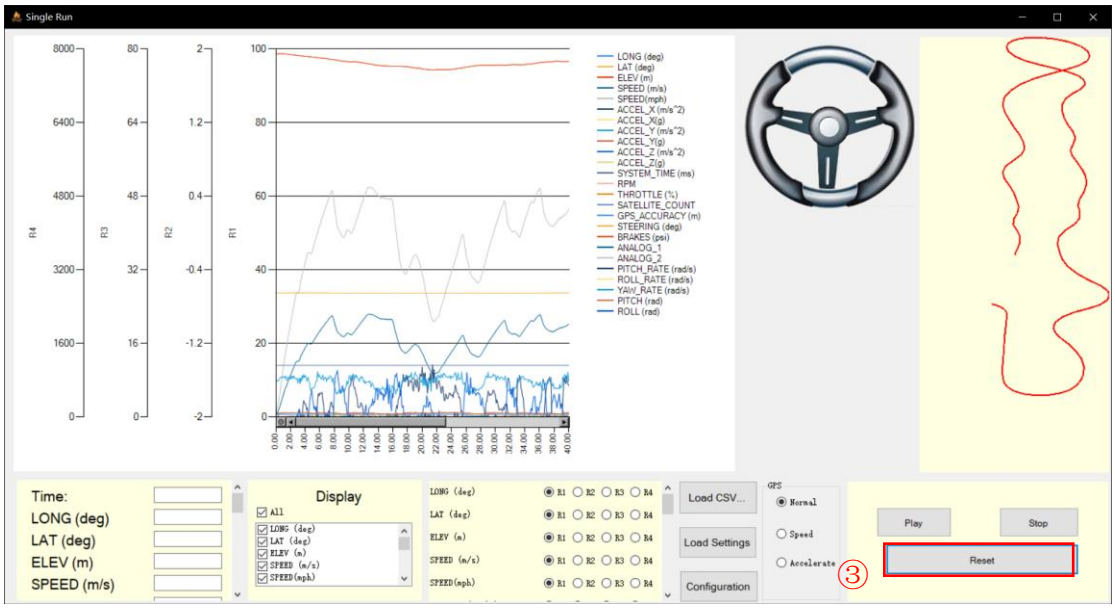

Figure 11 Click on Reset Button

3.4 Choose Different Type of Y Axis

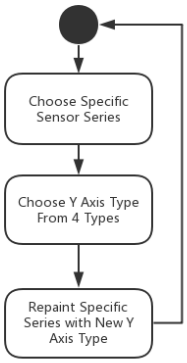

Figure 12 Flow Chart of choosing different types of Y axis

Users could change different series from one Y axis to another one. The software provides 4 different Y axes (**Figure 13 Click on Different Types of Y Axis**). We can switch between different axes by radio buttons dynamically created by fileLoadingButton_Click function.

Related Functions:

void change(int no, ChartArea caR)
void rb1_Click(object sender, EventArgs e)
void rb2_Click(object sender, EventArgs e)
void rb3_Click(object sender, EventArgs e)
void rb4_Click(object sender, EventArgs e)

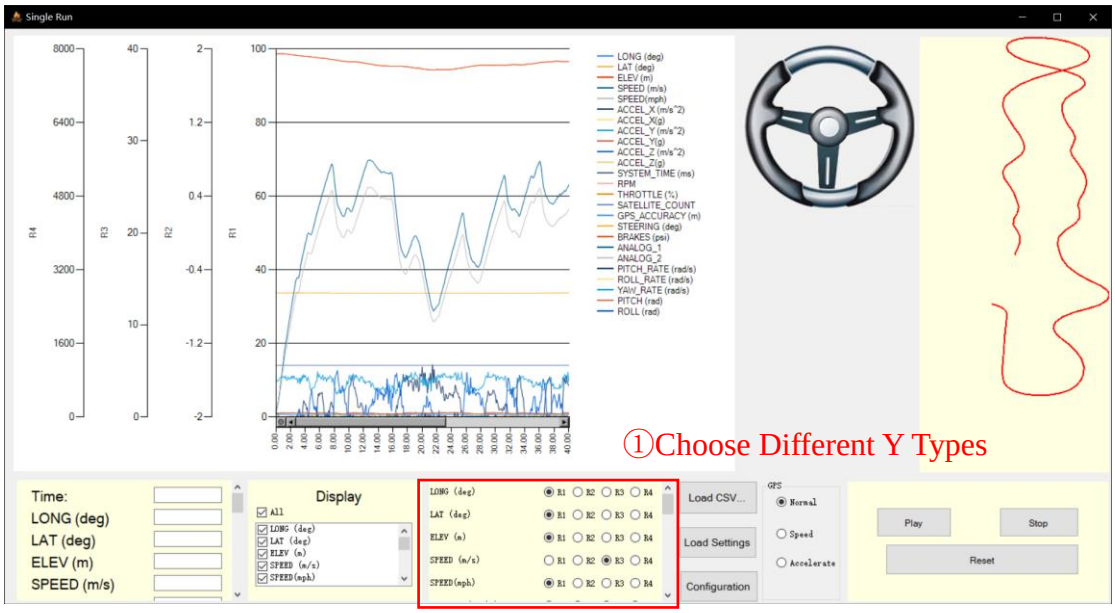

Figure 13 Click on Different Types of Y Axis

3.5 Customize X axis and Y axis

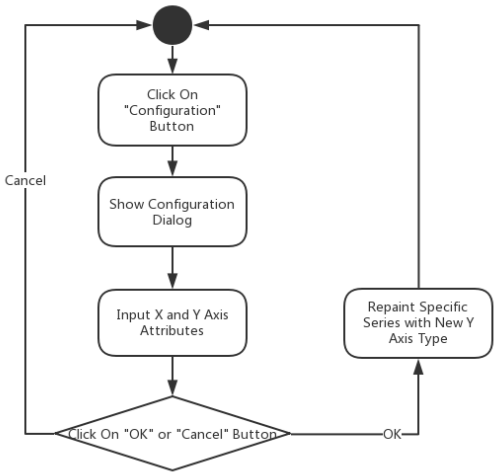

Figure 14 Flow Chart of Customizing X axis and Y axis

Users could click on “Configuration” button to pop out a dialog for X axis and Y axis customization (**Figure 15 Change X and Y Ranges, Intervals and Playing Speed**). Users could change the range of X axis range, scale and interval and Y axis range and type (**Figure 16 Results of Configuration Changing**).

Related Functions and Forms:

```
public partial class RangeForm : Form
void YRangeForm_Load(object sender, EventArgs e)
void confirmButton_Click(object sender, EventArgs e)
```

void ConfigureButton_Click(object sender, EventArgs e)

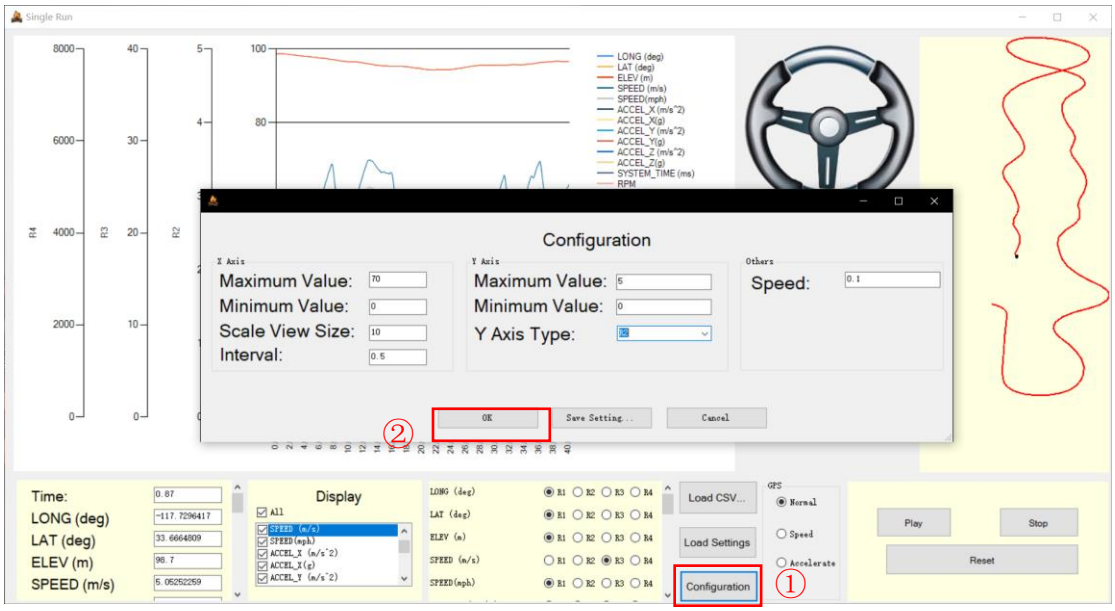

Figure 15 Change X and Y Ranges, Intervals and Playing Speed

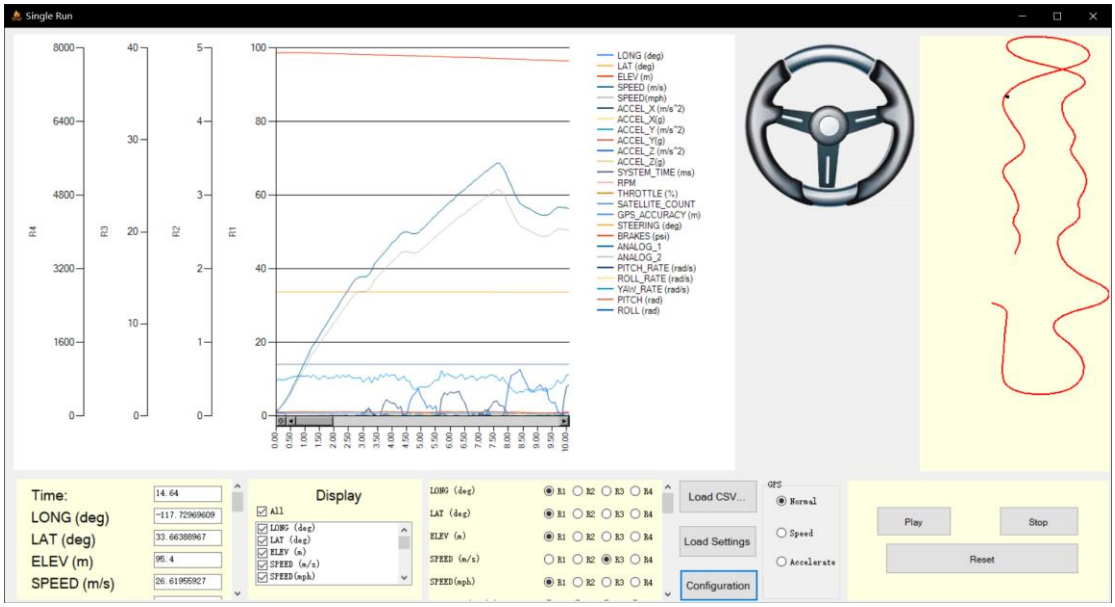

Figure 16 Results of Configuration Changing

3.6 Save and Load Setting Log Files

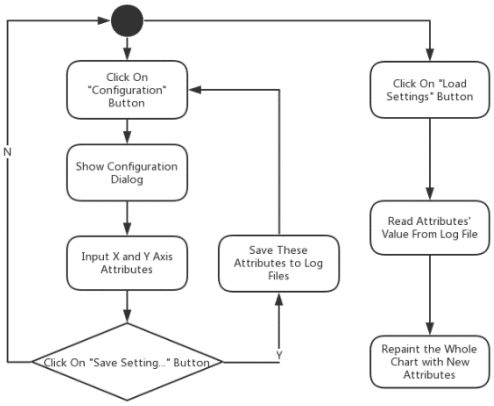

Figure 17 Flow Chart of Saving and Loading Setting Log Files

By clicking button of "Saving Setting…" in RangeForm (**Figure 18 Save Setting Log Files**), users could save log files with the X axis and Y axis configuration in specific form. Also, users could load log files created by them by clicking the button of "Loading Settings" (**Figure 19 Load Setting Log Files**).

Related Functions:

void settingSaveButton_Click(object sender, EventArgs e)

void settingButton_Click(object sender, EventArgs e)

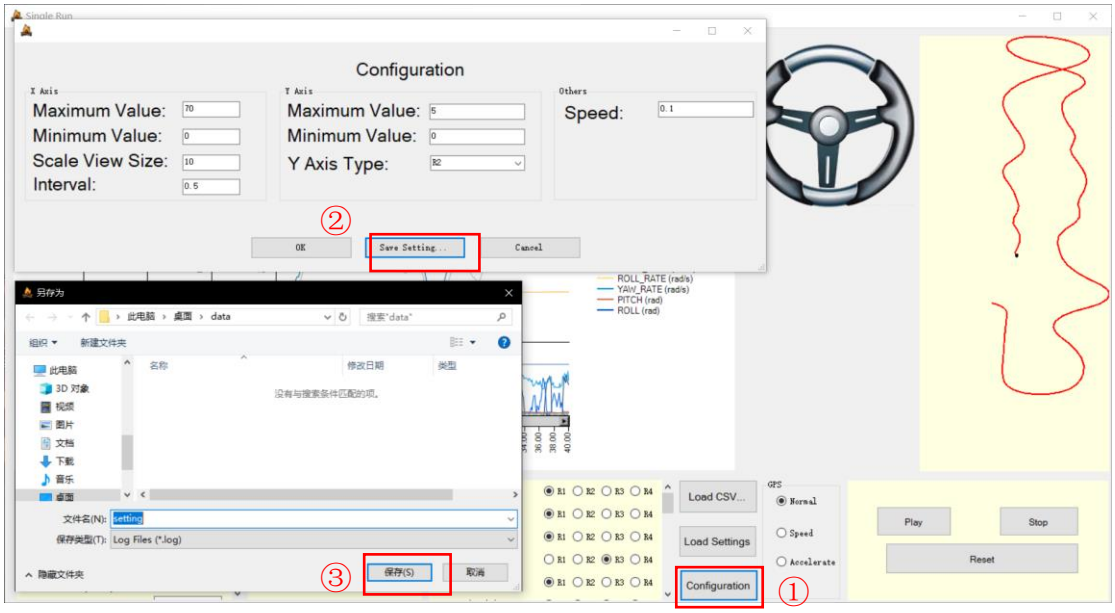

Figure 18 Save Setting Log Files

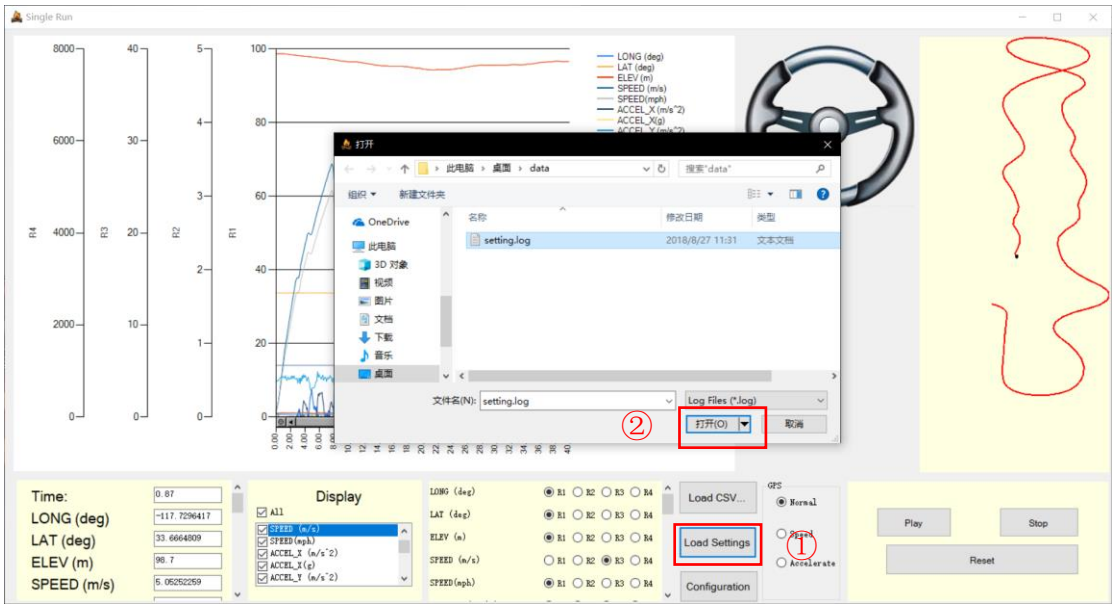

Figure 19 Load Setting Log Files

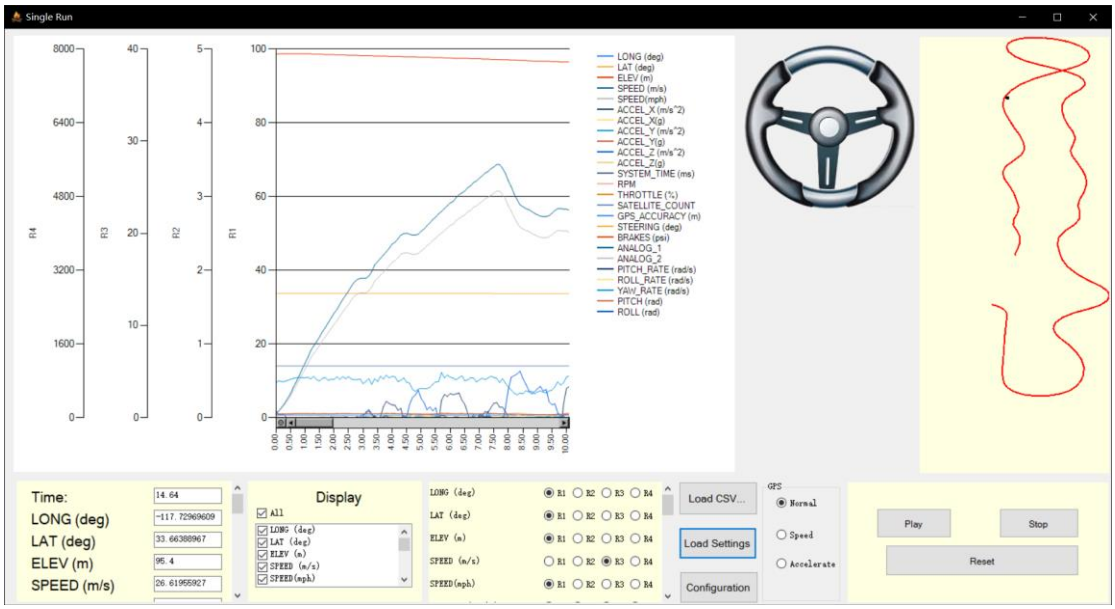

Figure 20 Results of Loading Setting Log Files

3.7 Show or Hide Specific Series

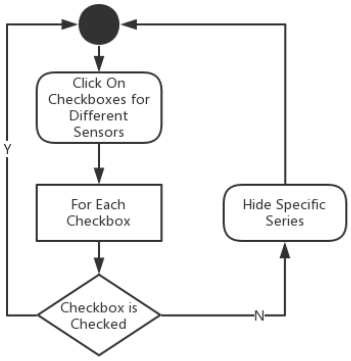

Figure 21 Flow Chart of Showing or Hiding Specific Series

Users could click on different checkboxes (**Figure 22 Click on Different CheckBoxes**) created dynamically by fileLoadingButton_Click function. In addition, there is an allSelectedCheckBox for the selection of all.

Related Functions:

void sensorCheckedListBox_ItemCheck(object sender, ItemCheckEventArgs e)

void allSelectedCheckBox_CheckedChanged(object sender, EventArgs e)

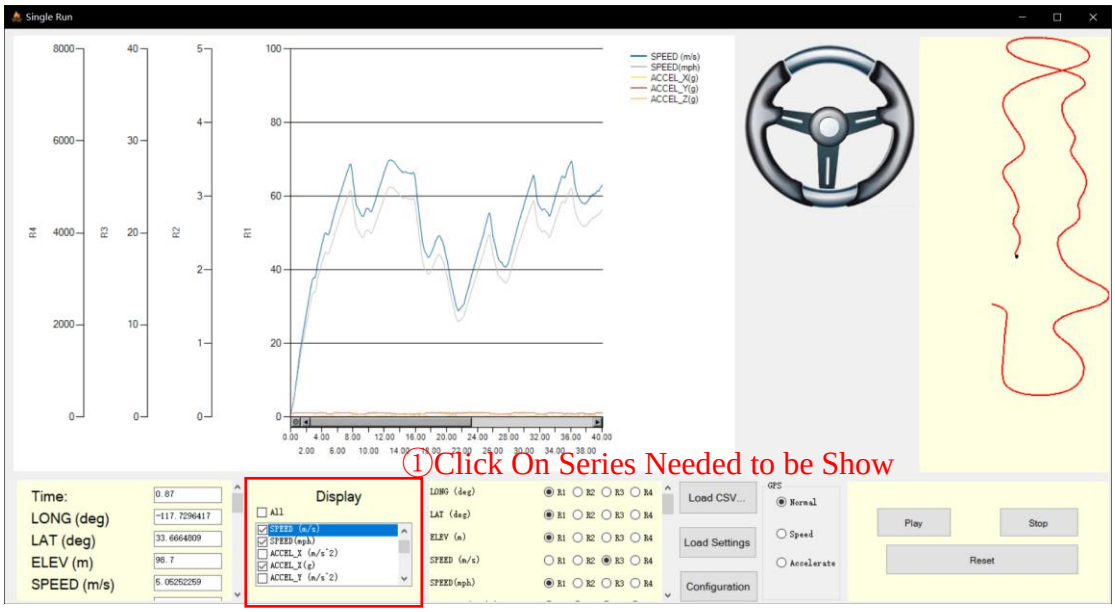

Figure 22 Click on Different CheckBoxes

3.8 Show Colored GPS Graph Changed by Speed or Acceleration

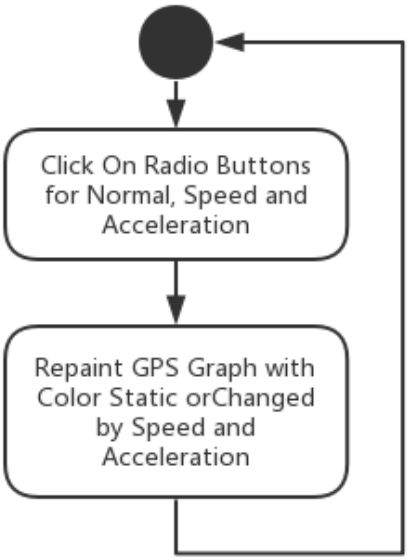

Figure 23 Flow Chart of Showing Colored GPS Graph Changed by Speed or Acceleration

Users could click on "radioButton_Normal" to show normal GPS graph, "radioButton_Speed" to show GPS graph changed by speed where green represents high speed and red represents low speed and "radioButton_Accelerate" to show GPS graph changed by acceleration (**Figure 24 Click on Different Radio Button**).

Related Functions:

```
void radioButton_Normal_CheckedChanged(object sender, EventArgs e)
void radioButton_Speed_CheckedChanged(object sender, EventArgs e)
void radioButton_Accelerate_CheckedChanged(object sender, EventArgs e)
int colorRed(double x)
int colorGreen(double x)
```

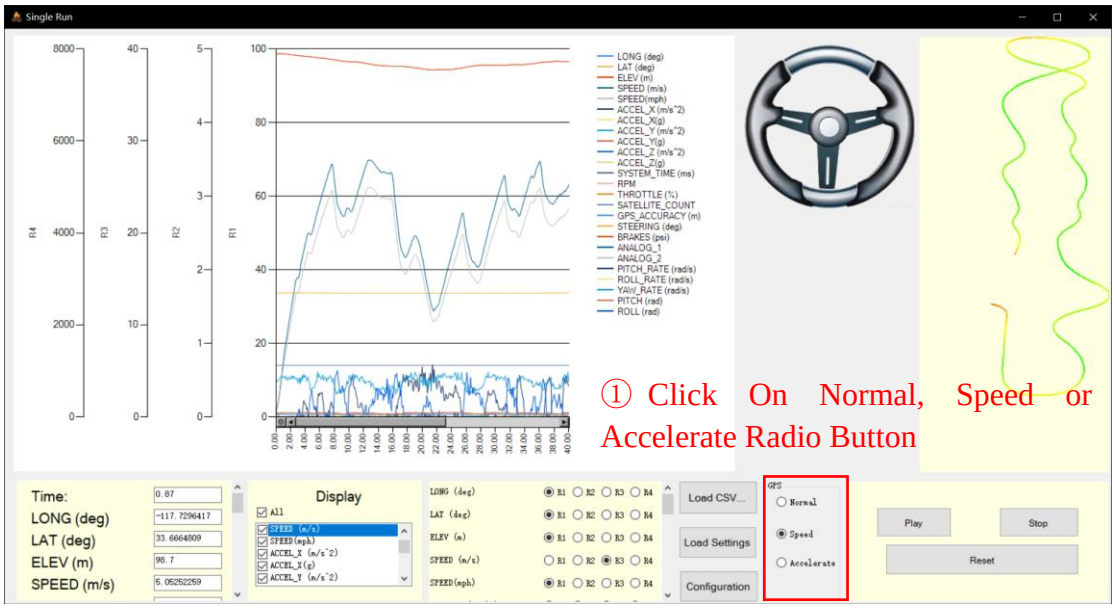

Figure 24 Click on Different Radio Button

3.9 Drag Mouse on the Chart

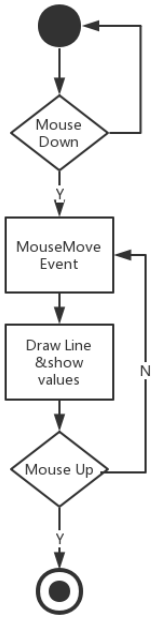

Figure 25 Flow Chart of Draging Mouse on the Chart

Users can drag the red line randomly in the chart. The chart will listen on "MouseDown" event to trigger "MouseMove" event and "MouseUp" event to end "MouseMove" event.

Related Functions:

void sensorChart_MouseMove(object sender, MouseEventArgs e)

void sensorChart_MouseDown(object sender, MouseEventArgs e)
void sensorChart_MouseUp(object sender, MouseEventArgs e)

3.10 Show Steering Position

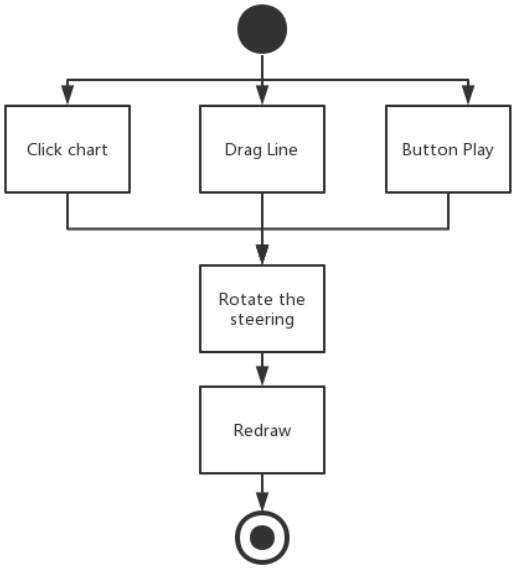

Figure 26 Flow Chart of Showing Steering Position

When user clicks on the chart, the steering wheel will be rotated a certain angle according to the CSV file. Similarly, if user clicks on Button Play, the angle of the steering wheel will be changed in real time.

Related Functions:
public static Image RotateImage(Image img, float rotationAngle);
private void sensorChart_MouseMove(object sender, MouseEventArgs e);
private void chartTimer_Tick(object sender, EventArgs e);
private void sensorChart_MouseClick(object sender, MouseEventArgs e);

5. Compared Run Form

4.1 Load CSV Files

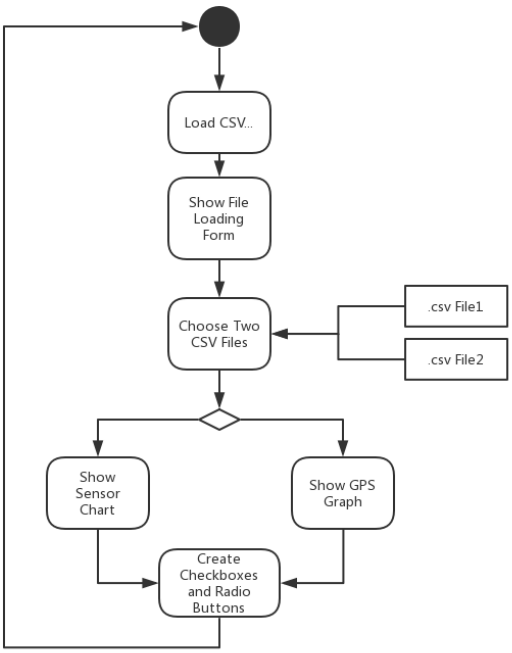

Figure 27 Flow Chart of Loading CSV Files

Call function of fileLoadingButton_Click and pop up a file dialog for users to choose their CSV files (**Figure 28 Select Two Different Files**). After users choose their files and the columns for latitude and longitude (**Figure 29 Choose Latitude and Longtitude**), the software could draw graphs for different series of sensors and the GPS graph for two routes of driving. Also, the software would dynamic create checkboxes and radio buttons for different sensors read from the CSV file (**Figure 30 Result of Files Loading**).

Related Functions:

void fileLoadingButton_Click(object sender, EventArgs e)

DataTable OpenCSV(string filePath)

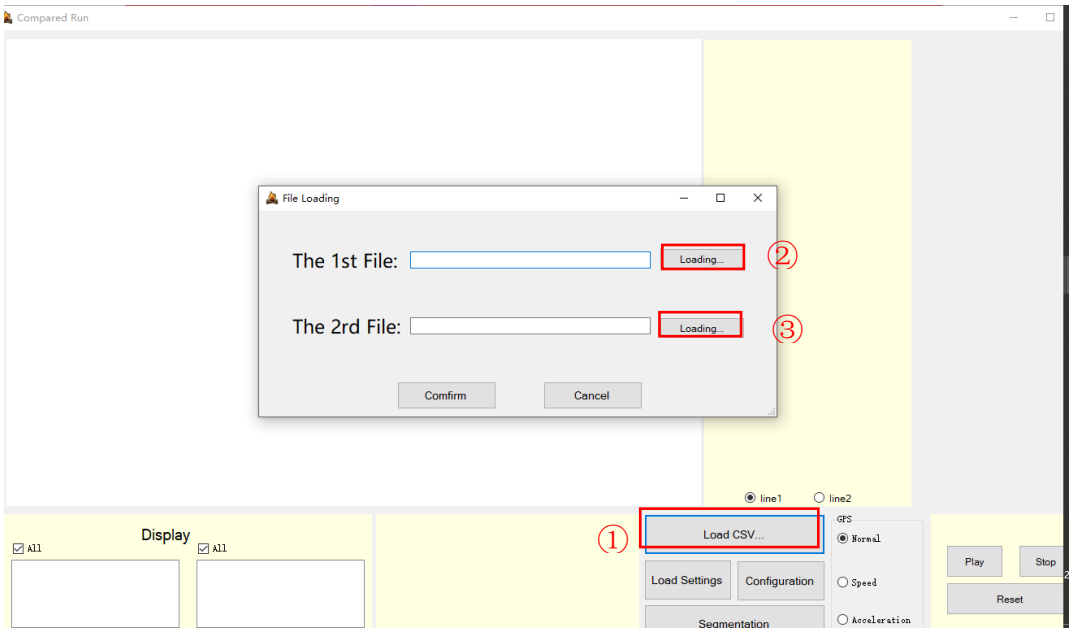

Figure 28 Select Two Different Files

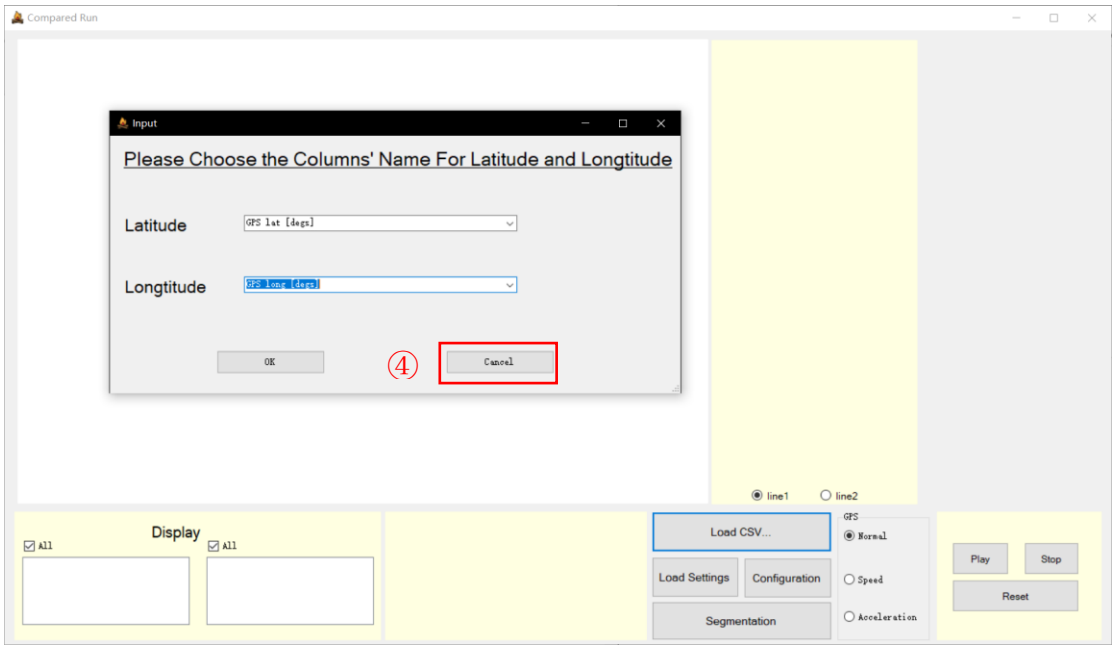

Figure 29 Choose Latitude and Longtitude

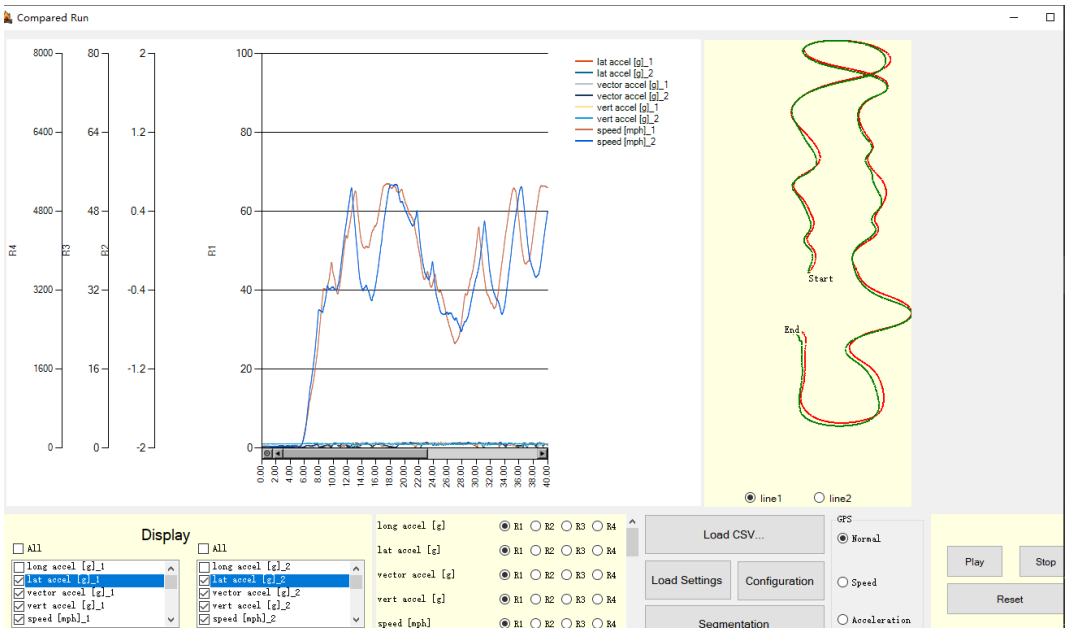

Figure 30 Result of Files Loading

4.2 Show Specific Value of Points

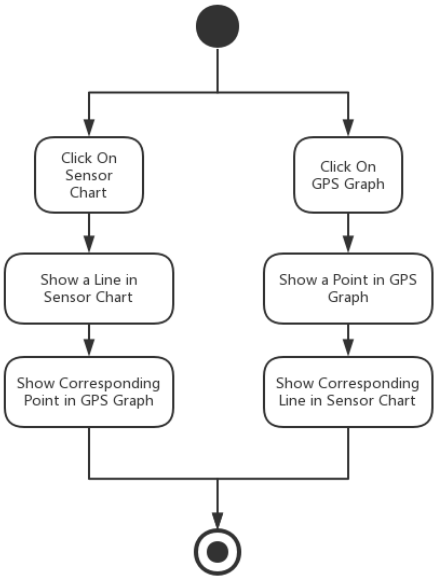

Figure 31 Flow Chart of Showing Specific Value of Points

Users could click on the graph and show a vertical line at where the mouse clicked (**Figure 32 Click on Specific Position on Chart**). Meanwhile, there would be a black point in the GPS graph to show the same place as you click on the chart. Also, users could click on the GPS graph (**Figure 33 Click on Specific Position on GPS Graph**) and show the corresponding on the sensor chart.

Related Functions:

```
void sensorChart_MouseClick(object sender, MouseEventArgs e)
private void GPSPanel_MouseClick(object sender, MouseEventArgs e)
int findLeftNear(double value, double[] array, int length)
```

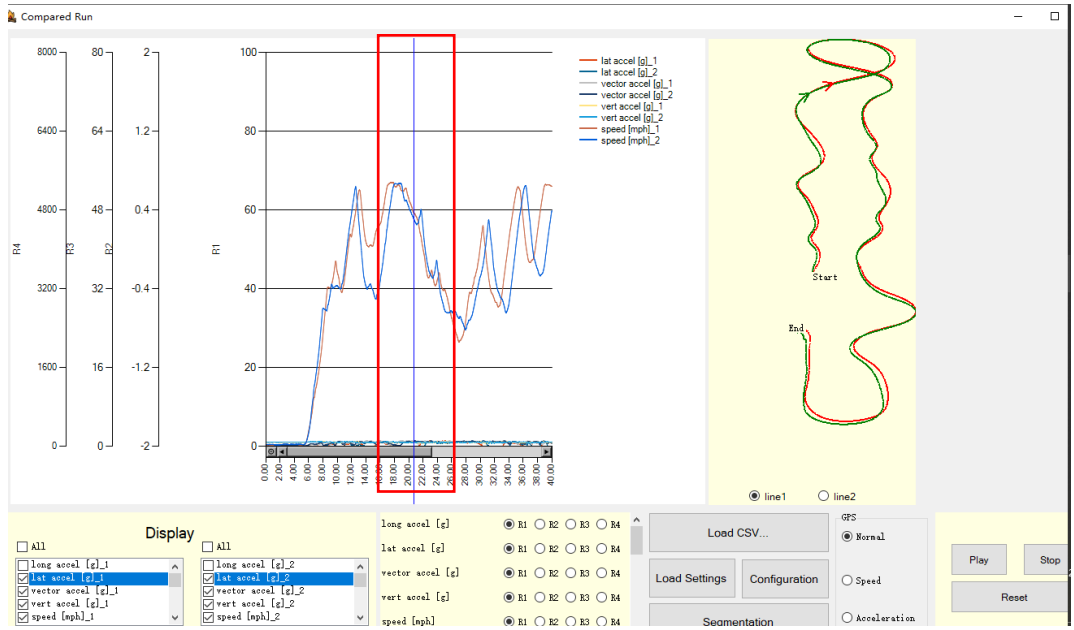

Figure 32 Click on Specific Position on Chart

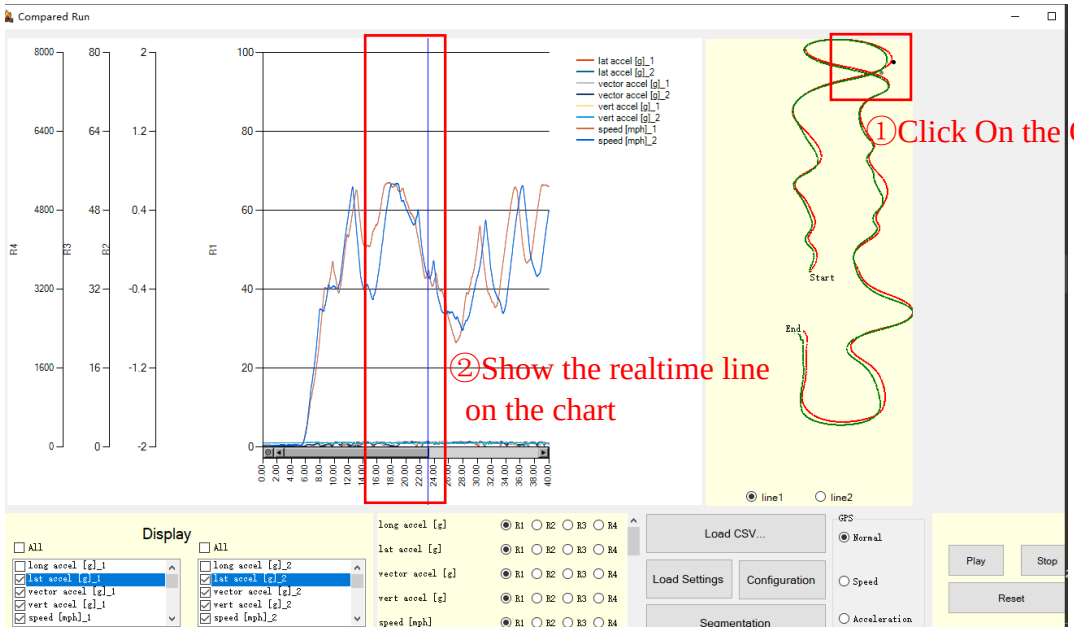

Figure 33 Click on Specific Position on GPS Graph

4.3 Replay the Data

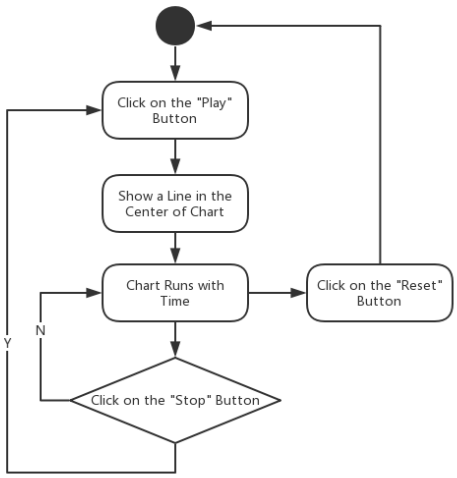

Figure 34 Flow Chart of Replaying the Data

Users could click on “Play” button and call the function of buttonPlay_Click. This function enables chartTimer so that it could repeatedly execute the function of chartTimer_Tick at intervals. In the meantime, the textboxes below would show the values of sensors as the chart flowing. When users click on “Stop” button, the chart would stop flowing. After “Play” button is re-clicked, the chart would begin to flow from where it stops. The “Reset” button would let users to reset the chart to the original state. Meanwhile, there would be two arrows on the GPS graph, each of them represents the direction of car. They will also move with the same speed of sensor chart.

Related Functions:

```
void buttonPlay_Click(object sender, EventArgs e)
void buttonStop_Click(object sender, EventArgs e)
void resetButton_Click(object sender, EventArgs e)
void chartTimer_Tick(object sender, EventArgs e)
```

4.4 Choose Different Type of Y Axis

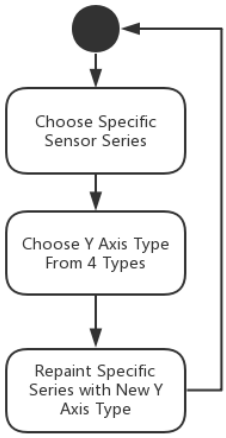

Figure 35 Flow Chart of Choosing Different Type of Y Axis

Users could change different series from one Y axis to another one to display the data in one chart more clearly. The software provides 4 different Y axes (**Figure 36 Choose Different Y Axis Types**). We can switch between different axes by radio buttons dynamically created by fileLoadingButton_Click function. Both drivers' chart will change with different Y axis type.

Related Functions:

void change(string no, ChartArea caR)
void change2(string no, ChartArea caR)
void rb1_Click(object sender, EventArgs e)
void rb2_Click(object sender, EventArgs e)
void rb3_Click(object sender, EventArgs e)
void rb4_Click(object sender, EventArgs e)

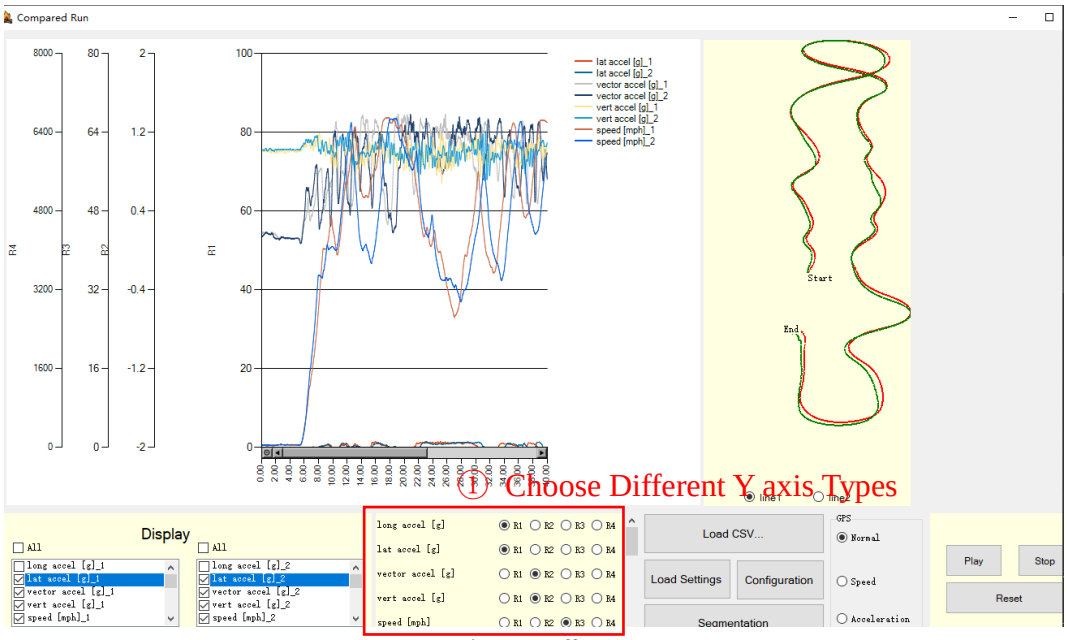

Figure 36 Choose Different Y Axis Types

4.5 Customize X axis and Y axis

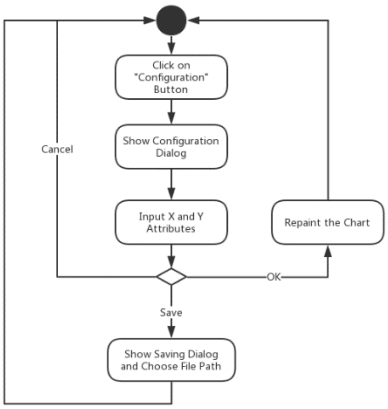

Figure 37 Flow Chart of Customizing X axis and Y axis

Users could click on “Configuration” button to pop out a dialog for X axis and Y axis customization (**Figure 38 Choose Different Axis Ranges, Intervals and Replaying Speed**). Users could change the range of X axis range, scale and interval and Y axis range and type (**Figure 39 Result of Axis Customization**).

Related Functions and Forms:

public partial class RangeForm : Form

void YRangeForm_Load(object sender, EventArgs e)

void confirmButton_Click(object sender, EventArgs e)

void ConfigureButton_Click(object sender, EventArgs e)

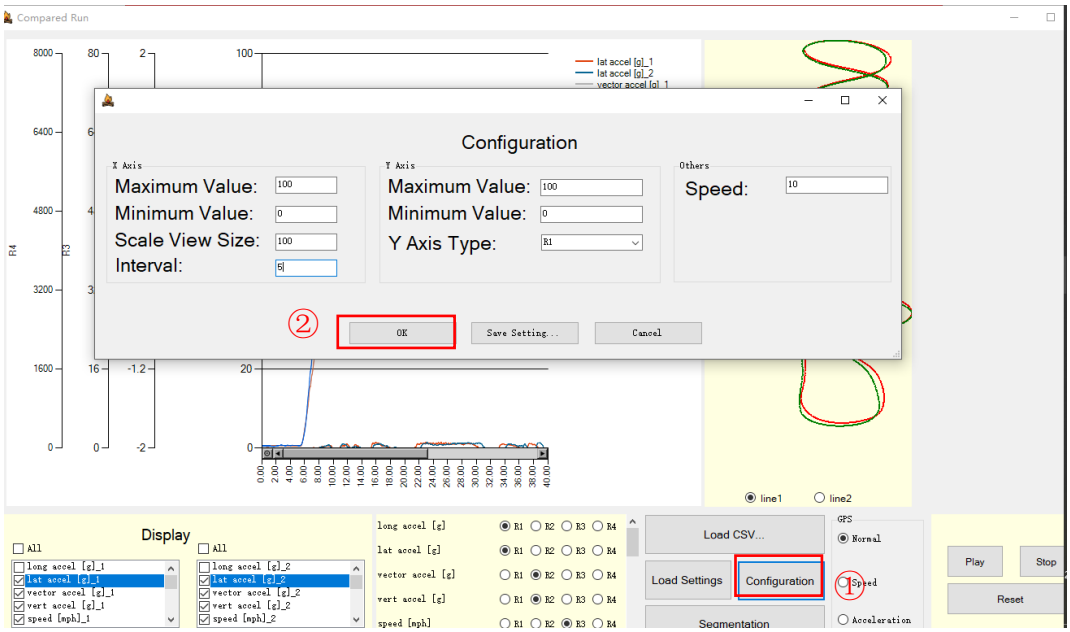

Figure 38 Choose Different Axis Ranges, Intervals and Replaying Speed

After change the max valve and scale view of X axis, we can see that the chart has changed as we expected.

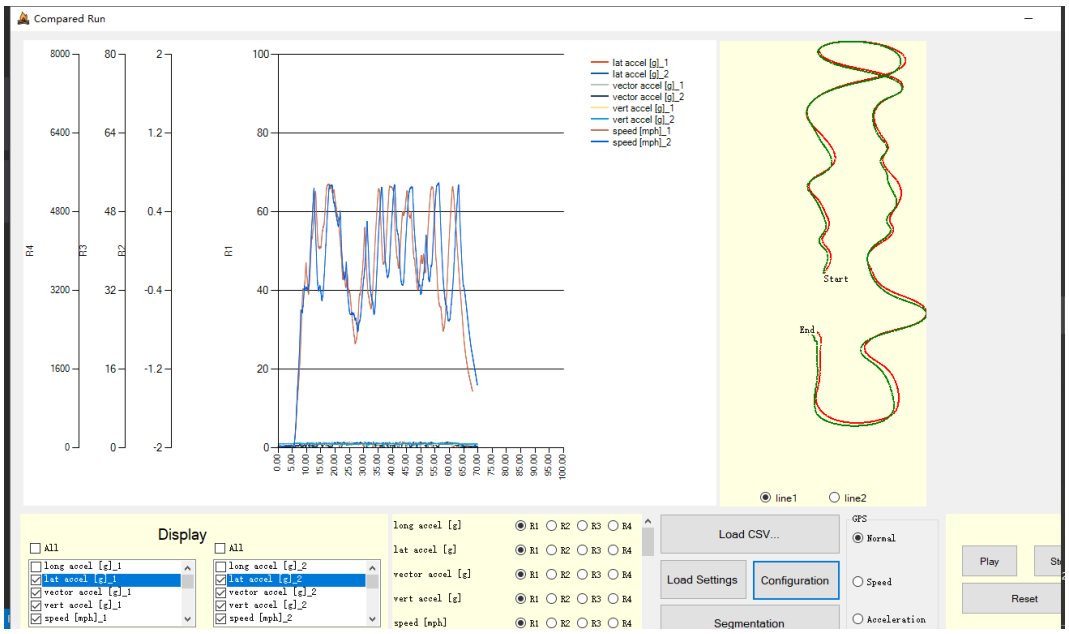

Figure 39 Result of Axis Customization

4.6 Save and Load Setting Log Files

This function is really similar to the single ride. See 2.6.

4.7 Show or Hide Specific Series

There are two checkboxeslists to show the data of two drivers. Users could click on different checkboxes created dynamically when reading the csv file (**Figure 40 Click on Different Series for displaying**). In addition, there is an allSelectedCheckBox for the selection of all.

Related Functions:
void sensorCheckedListBox_ItemCheck(object sender, ItemCheckEventArgs e)
void allSelectedCheckBox_CheckedChanged(object sender, EventArgs e)

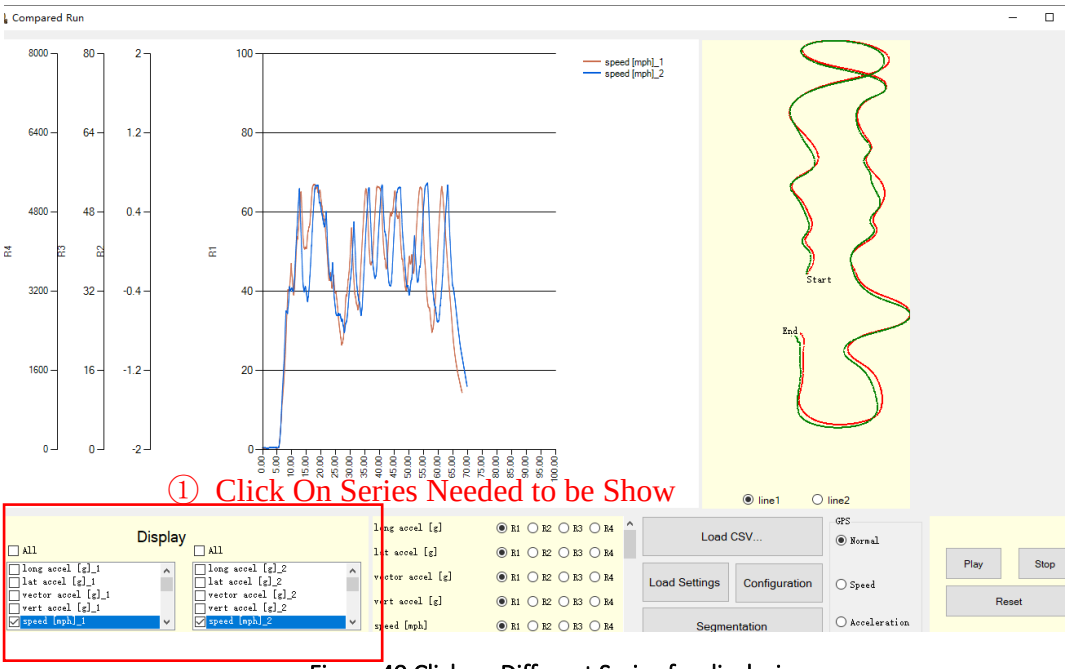

Figure 40 Click on Different Series for displaying

4.8 Click on the GPS map

When we click on one road (**Figure 41 Click on GPS Graph**), it can show where another driver is. With this, we can easily compare the performance of the two drivers.

Related Functions:
private void GPSPanel_MouseClick(object sender, MouseEventArgs e)

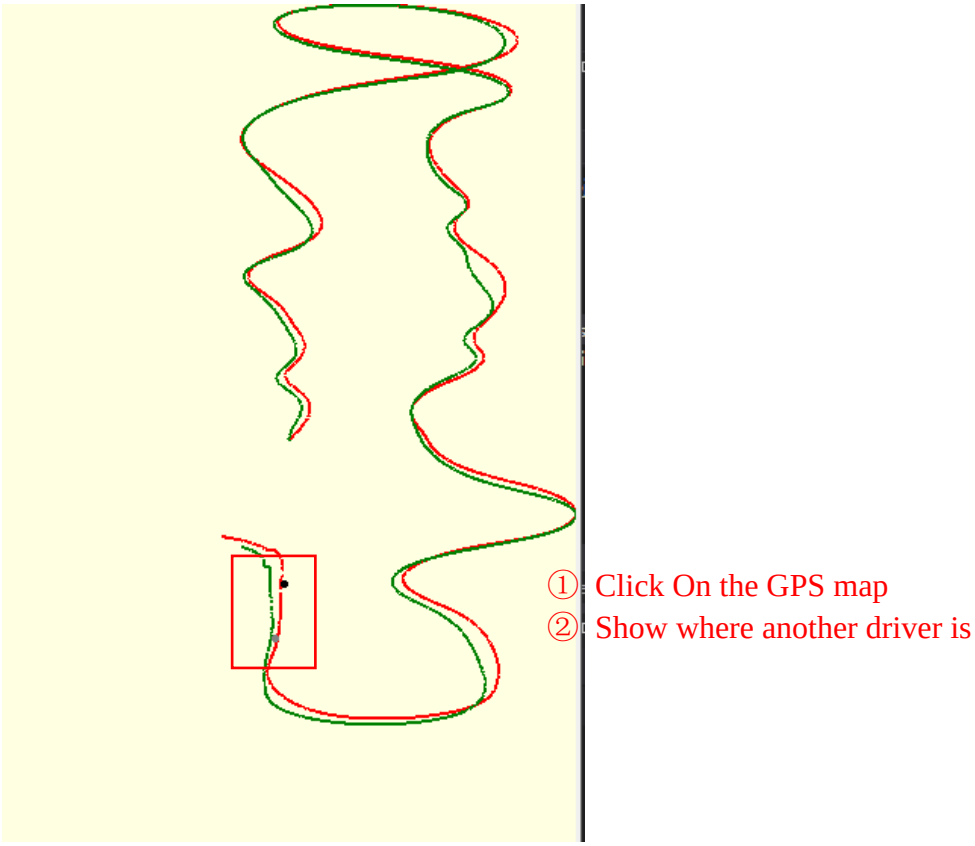

Figure 41 Click on GPS Graph

4.9 Click/Drag Mouse on the Chart

Users can drag the line randomly in the chart (**Figure 42 Click on Chart**). The chart will listen on "MouseDown" event to trigger "MouseMove" event and "MouseUp" event to end "MouseMove" event. Also, users can click on the chart. It can show the locations & heading position of the two drivers.

Related Functions:

void sensorChart_MouseMove(object sender, MouseEventArgs e)
private void sensorChart_MouseClick(object sender, MouseEventArgs e)
private void sensorChart_MouseDown(object sender, MouseEventArgs e)
private void sensorChart_MouseUp(object sender, MouseEventArgs e)

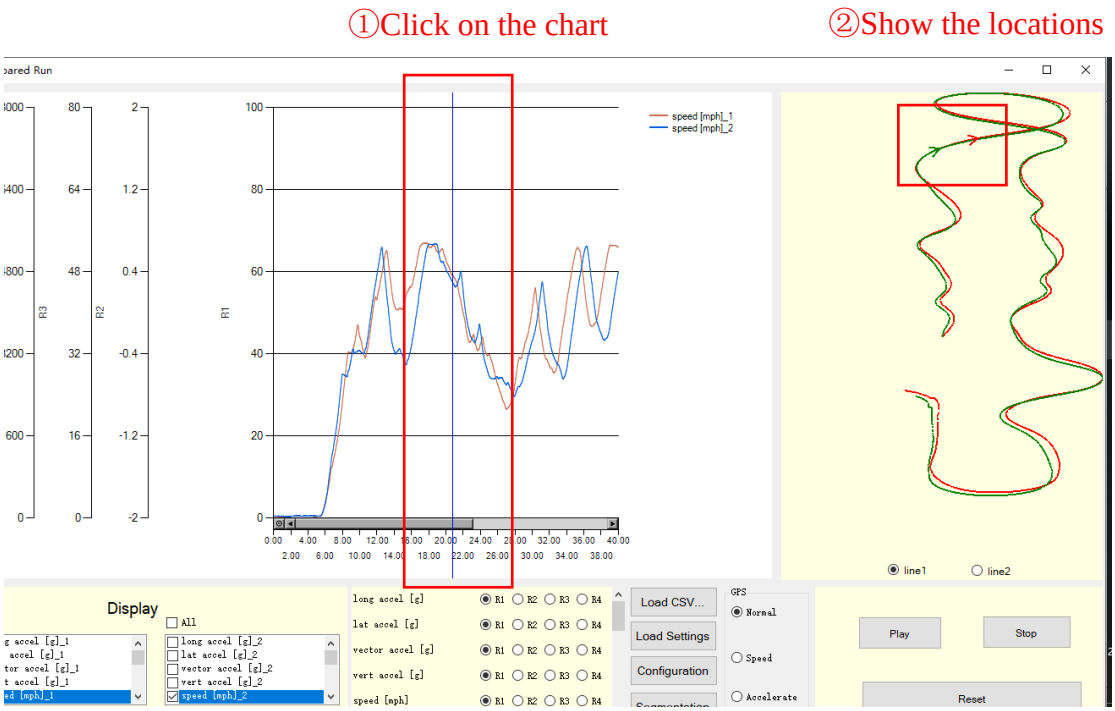

Figure 42 Click on Chart

4.10 Segmentation

When we click on “Segmentation” button, the segmentation part will come out, which shows the time, distance and average velocity it takes to complete every section for each driver (**Figure 43 Default Segmentation**). Also you can change the segmentation by sliding scroll bar (**Figure 44 Customized Segmentation**). So we can easily compare the performance of the two drivers effectively. Besides, users can change the start point and the end point because there may be many useless data in the beginning and ending.

On the map, it can show the four segmentations. The lighter color represents the first driver and the darker color represents another driver. On the scroll bar, the default segmentation is 25%, 50%, 75%.

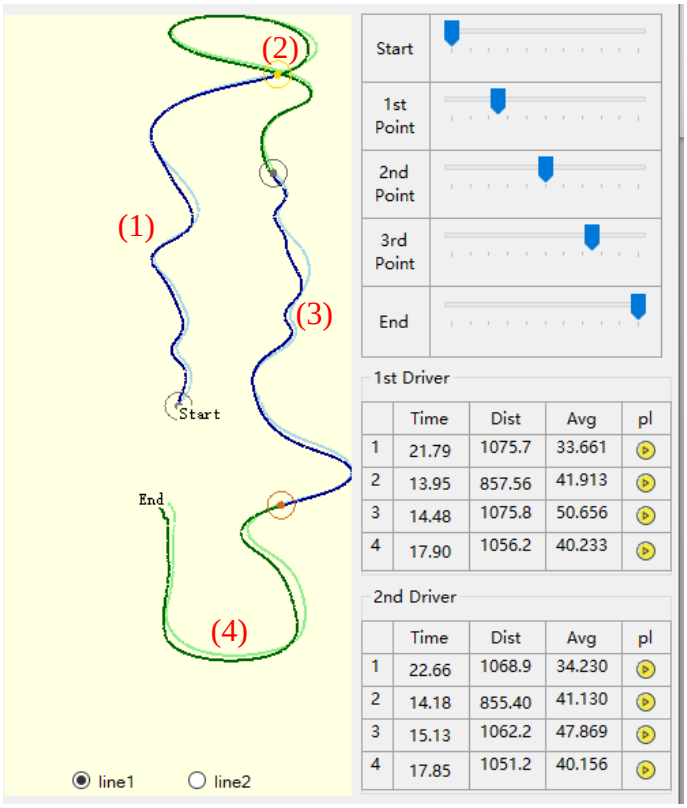

Figure 43 Default Segmentation

After sliding the scroll bar, it will be like this:

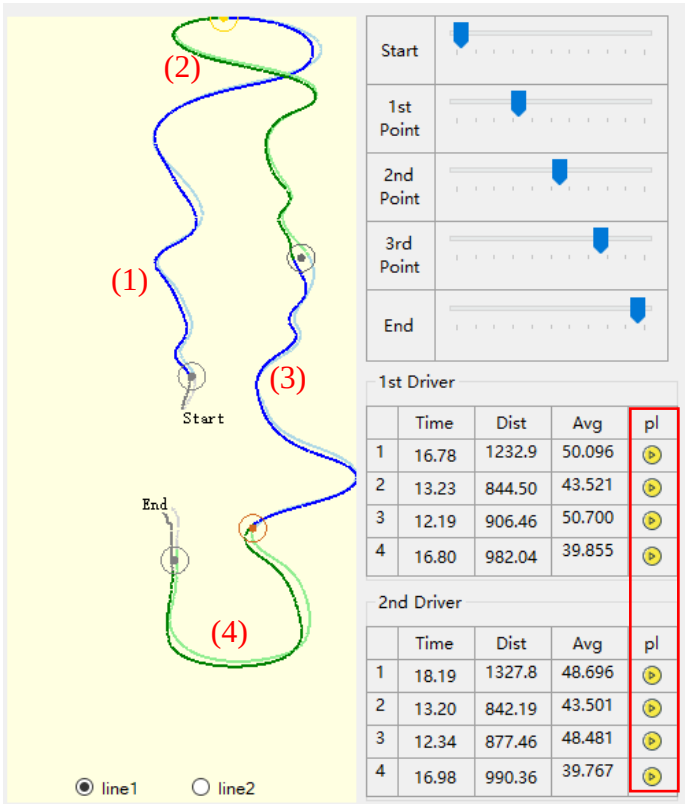

Figure 44 Customized Segmentation

Users can play every section so that users can see the movements of each driver in

each section:
You can check it out on the youtube: https://youtu.be/af4zLfwI-Bc

Related Functions:

```
private void thirdTrackBar_ValueChanged(object sender, EventArgs e)
private void secondTrackBar_ValueChanged(object sender, EventArgs e)
private void firstTrackBar_ValueChanged(object sender, EventArgs e)
private void EndTrackBar_ValueChanged(object sender, EventArgs e)
private void StartTrackBar_ValueChanged(object sender, EventArgs e)
private void segmentationButton_Click(object sender, EventArgs e)
private void Section1PictureBox_Click(object sender, EventArgs e)
private void Section2PictureBox_Click(object sender, EventArgs e)
private void Section3PictureBox_Click(object sender, EventArgs e)
private void Section4PictureBox_Click(object sender, EventArgs e)
private void section1Timer_Tick(object sender, EventArgs e)
private void section2Timer_Tick(object sender, EventArgs e)
private void section3Timer_Tick(object sender, EventArgs e)
private void section4Timer_Tick(object sender, EventArgs e)
```

2. Testing Environment

- Visual Studio 2013&2015
- Programming Language: C#