ILR #9: Progress Review 10

Harry Golash – Team A

March 23, 2017

Teammates: Amit Agarwal, Yihao Qian, Menghan Zhang, Zihao Zhang

1. Individual Progress:

1.1 Tasks:

For this progress review our team had the following tasks accomplish:

1. Filtered data from all sensors
2. Multiple-object simultaneous tracking
3. Vehicle egomotion estimation
4. Preliminary sensor fusion and system integration

The tasks I was partially responsible for were the first, third, and fourth ones in the list above. Unfortunately, we did not accomplish sensor fusion since the last progress review, and the radar data visualization and filtering is still in progress. We were however able to meaningfully visualizing the radar detection-level data in a human-understandable format via the terminal.

1.2 Implementation:

1.2.1 Visualizing the radar detection-level data:

We were able to visualize the radar data by our last progress review, however the visualization had to be done using the "netcat" command in the linux terminal. Additionally, the data could only be viewed in special characters or hex characters, which meant there was no way to determine the quality of the data we were obtaining. We are now able to visualize the radar detection-level data in real time in a human-understandable format (by displaying object parameters such as depth, angle, velocity etc. for the detected objects) (see Fig 1.).

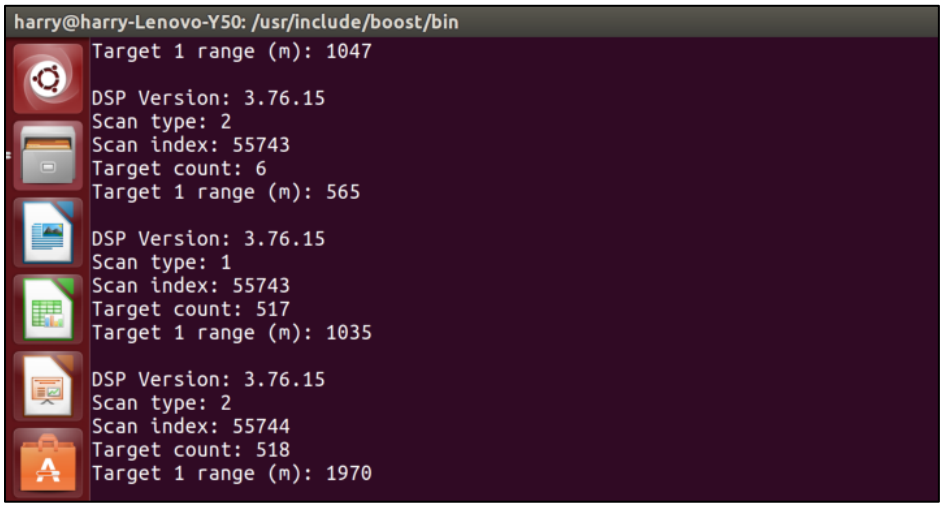

Fig. 1: Read-time radar detection-level data. The parameters and number of objects displayed can be adjusted in the TCP parser code as can be seen in Fig. 2.

To do this, we wrote our own code and GUI in C++\C# to parse the data and display it (C++ TCP parser code snippet can be seen in Fig. 2). This allows us to chose and display objects and object parameters as we choose to, as can be seen in the lines shown in Fig. 2. This code was written in C++ and was run and tested in a linux environment.

```
145     if (gotFirstPayload == true) {
146       for (unsigned int i = 0; i < rcvSize; i++) {
147         xcpMsgBuf[packetCounter + i] = msgBuf[i];
148       }
149
150       packetCounter += rcvSize;
151     }
152
153     if (rcvSize == 1268) {
154       gotFirstPayload = true;
155       //Reset everything because we got the remainder packet.
156       gotLastPacket = true;
157       //xcpMsgBuf.fill(0);
158       packetCounter = 0;
159     } else {
160       gotLastPacket = false;
161     }
162
163     if (gotLastPacket == true) {
164       //Get bytes in little-endian order.
165       unsigned long dsp1 = read_value(xcpMsgBuf, ESR_DSP_VER_1_OFFSET, ESR_DSP_VER_1_SIZE);
166       unsigned long dsp2 = read_value(xcpMsgBuf, ESR_DSP_VER_2_OFFSET, ESR_DSP_VER_2_SIZE);
167       unsigned long dsp3 = read_value(xcpMsgBuf, ESR_DSP_VER_3_OFFSET, ESR_DSP_VER_3_SIZE);
168       cout << "DSP Version: " << dsp1 << "." << dsp2 << "." << dsp3 << endl;
169       unsigned long scanType = read_value(xcpMsgBuf, ESR_SCAN_TYPE_OFFSET, ESR_SCAN_TYPE_SIZE);
170       cout << "Scan type: " << scanType << endl;
171       unsigned long scanIndex = read_value(xcpMsgBuf, ESR_SCAN_INDEX_OFFSET, ESR_SCAN_INDEX_SIZE);
172       cout << "Scan index: " << scanIndex << endl;
173       unsigned long targetCount = read_value(xcpMsgBuf, ESR_TGT_RPT_CNT_OFFSET, ESR_TGT_RPT_CNT_SIZE);
174       cout << "Target count: " << targetCount << endl;
175       uint16_t target1Rng = (uint16_t) read_value(xcpMsgBuf, ESR_TGT1_RNG_OFFSET, ESR_TGT1_RNG_SIZE);
176       cout << "Target 1 range: " << (float) target1Rng << endl << endl;
177     }
178   }
```

Fig. 2: Code snippet of the C++ TCP packets parser code we wrote for the radar.

1.2.1 Radar data visualization and GUI:

I also created a GUI in Visual C# to display the radar data in real time (see Fig. 3). I used the MSDN TcpListener Class to look for and display the socket data from the radar's IP and port address. The data was displayed in a human-understandable text format, similar to the display shown in Fig. 1. The code was written on and for Windows, but unfortunately we were unable to reliably connect to the radar on Windows (we had some initial success using Wireshark). We will need likely to change the network/firewall settings to connect the radar on Windows. Currently, we are working on visualizing all the data points simultaneously in 2D in a Linux environment using rviz in ROS. We also plan to write a GUI to overlay the radar and camera data soon, so that we can conduct real world testing and determine the accuracy and quality of all the data we get.

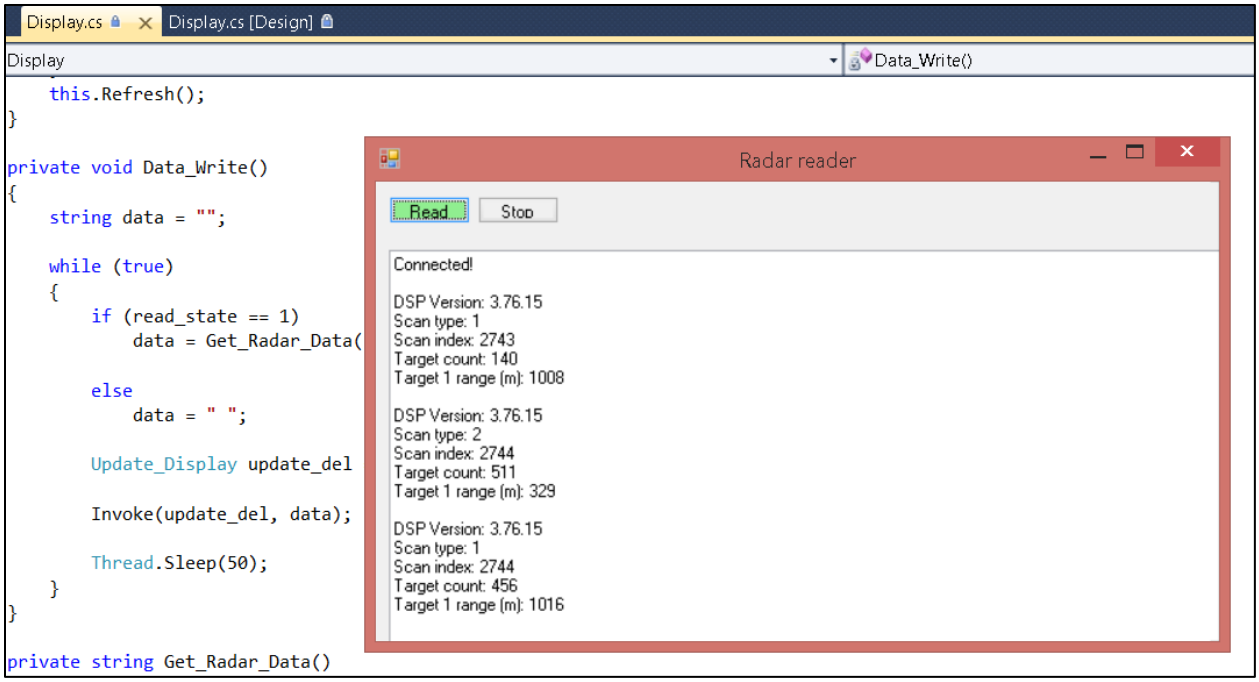

Fig. 3: Radar GUI code written in Visual C# running on Windows.

2. Challenges:

Due to personal issues and improper planning, we were unable to accomplish as much as we had hoped to since the last progress review. We underestimated the amount of time it would take us to parse the radar data using our own code. Additionally, we initially spent time on acquiring and displaying the data using a Windows GUI, which later proved to be a futile effort.

We have chronically been unable to find proper instructions and example code for the radar unit that we have. Thus, we took a long time trying various approaches to parse and meaningfully display the radar data until we found a real-time solution that worked fairly well.

Communication between teammates and between our team and our sponsor has unfortunately not been regular or effective since the last progress review, which is another possible reason why we did not end up completing as much as we planned to. We need to improve communication and work hard in the weeks ahead in order t catch up and make suitable progress on this project. We will do so by scheduling regular team meetings and meetings with our sponsor, and making sure we attend them.

3. Teamwork:

There is a large amount of work that needs to be done in the next few weeks in order to get back on schedule. We have distributed the workload among teammates and assigned tasks and internal deadlines, which we hope will allow us to make progress effectively.

Amit and I will work on the radar real-time data display, real-world testing, and data filtering. Once we have properly filtered the detection-level data, we will work on our own clustering methods to assist with vision-based tracking. Zihao and Yihao and Menghan will continue to improve the stereo vision system's real time performance. They will also work on system integration using ROS. By the next PR, we hope to achieve initial sensor fusion. We hope to visualize the radar and camera data simultaneously in a meaningful format.

4. Future Plans:

By the next progress review, we hope to accomplish the following:

1. Integrated stereo vision system
2. Data filtering and clustering for the radar
3. Meaningful integration of camera and radar data
4. Initial real-time sensor fusion and data visualization

I will be working on the second, third, and fourth items in the list above.