ILR #6: Progress Review 7

Zihao (Theo) Zhang- Team A

February 2nd, 2017

Teammates: Amit Agarwal, Harry Golash, Yihao Qian, Menghan Zhang

Individual Progress

As the only person in the team who has been working on the Delphi ESR radar since the last semester, I devoted most of my time on the radar during the past week and have made fairly satisfying progress on it.

Since the beginning of this project, the team has actively searched various solutions for processing data from the radar among very limited resources available. We finally identified using the commercial middleware called PolySync as the most effective solution in our case and have been using it for all radar-related tasks since then.

The team successfully achieved visualization of the detected targets from the radar by the Fall Validation Experiments. However, a few issues have required us to make further improvements. First of all, the number of detected "targets" were way too many compared to the number of objects of our interest. The problem was that the Delphi ESR radar would output 64 targets by default for each single scanning. As a result, the majority of the flickering points shown in the visualization were usually just noise. Moreover, the visualization was achieved using the build-in service provided by the PolySync platform, which is called the PolySync Studio. Considering our needs for further processing on the radar data as well as for sensor fusion in the future, we have to build our own software application and cannot simply rely on PolySync Studio as the only way to get access to the radar data.

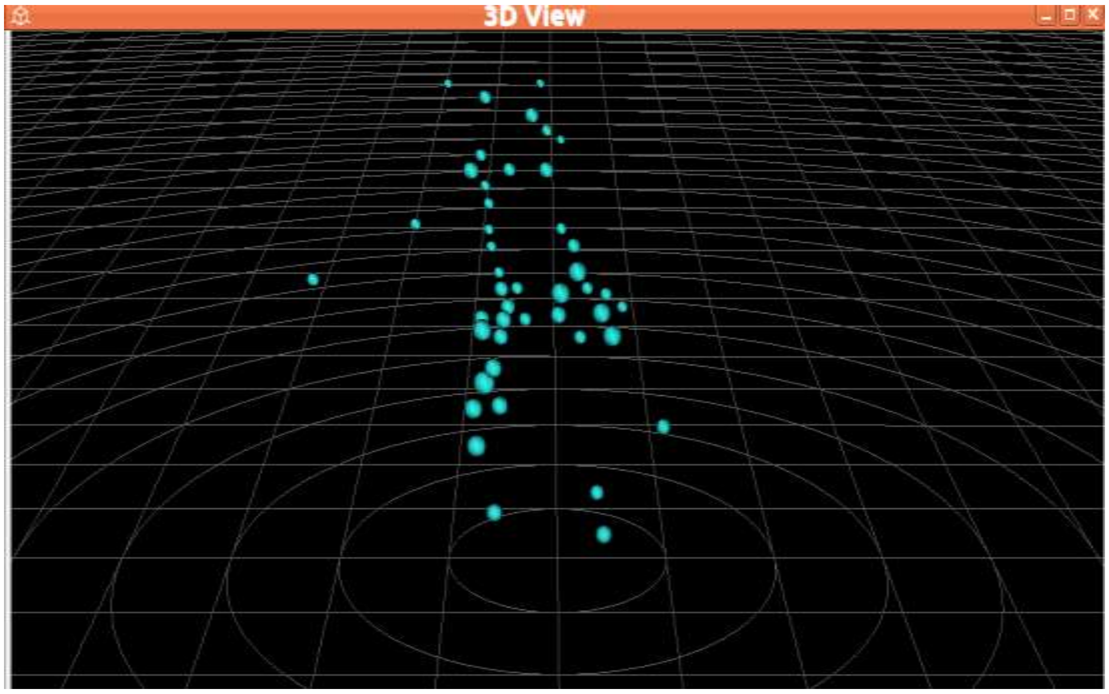
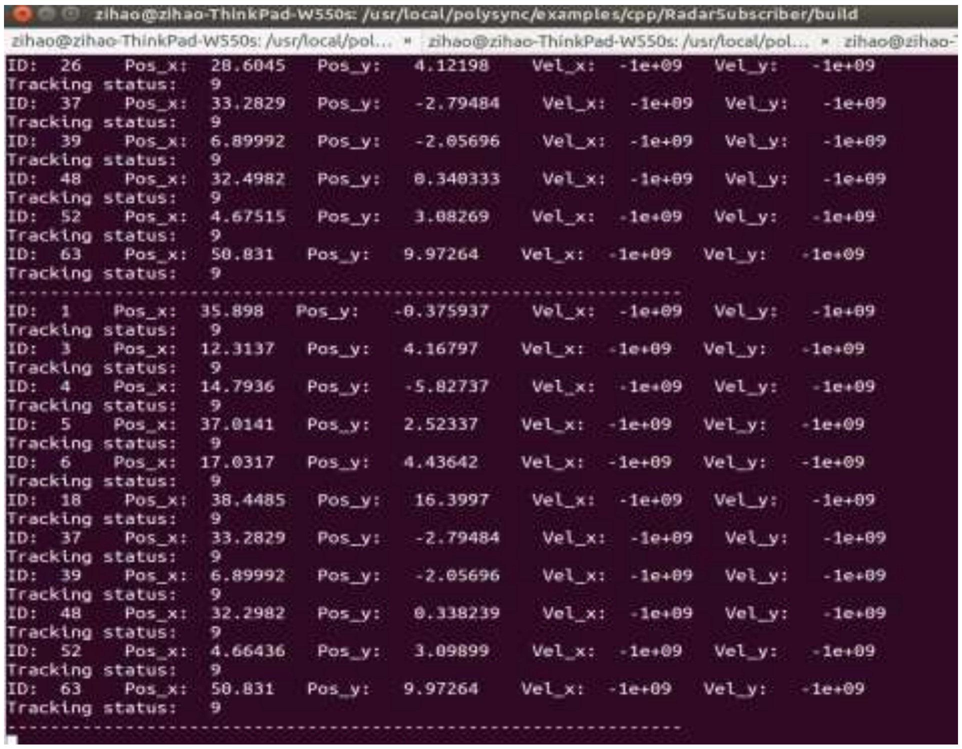


Figure 1. Ordinary visualization of radar data using PolySync Studio

Since the end of last semester, I have been learning the system architecture of the PolySync platform and the provided API in both C and C++. I realize that the system uses the so-called publisher and subscriber nodes to send data to or receive data from the message bus. Since the radar data is sent to the bus through the dynamic driver node from the sensor, a customized subscriber is therefore required to fetch desired data from the bus.

By this progress review, I have finished building a customized subscriber for collecting data from the Delphi ESR radar. I applied a simple filter on the raw message from the radar by adding constraints on both the tracking status and the range type. Specifically, any detected "target" with an invalid tracking status or range type as identified was ignored. The position and velocity information of each detected target after filtering were collected and displayed in terminal in real time. This is a big step forward on processing data from the radar, as of now we no longer have to fully depend on the PolySync studio in order to get access to the radar data.



```
zihao@zihao-ThinkPad-W550s: /usr/local/polysync/examples/cpp/RadarSubscriber/build
zihao@zihao-ThinkPad-W550s: /usr/local/pol...  zihao@zihao-ThinkPad-W550s: /usr/local/pol...  zihao@zihao-
ID:  26    Pos_x:  28.6045   Pos_y:   4.12198    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
ID:  37    Pos_x:  33.2829   Pos_y:   -2.79484    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
ID:  39    Pos_x:  6.89992   Pos_y:   -2.05696    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
ID:  48    Pos_x:  32.4982   Pos_y:   0.340333    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
ID:  52    Pos_x:  4.67515   Pos_y:   3.08269    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
ID:  63    Pos_x:  50.831   Pos_y:   9.97264    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
-------------------------------------------------------------------
ID:  1    Pos_x:  35.898   Pos_y:   -0.375937    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
ID:  3    Pos_x:  12.3137   Pos_y:   4.16797    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
ID:  4    Pos_x:  14.7936   Pos_y:   -5.82737    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
ID:  5    Pos_x:  37.0141   Pos_y:   2.52337    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
ID:  6    Pos_x:  17.0317   Pos_y:   4.43642    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
ID:  18    Pos_x:  38.4485   Pos_y:   16.3997    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
ID:  37    Pos_x:  33.2829   Pos_y:   -2.79484    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
ID:  39    Pos_x:  6.89992   Pos_y:   -2.05696    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
ID:  48    Pos_x:  32.2982   Pos_y:   0.338239    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
ID:  52    Pos_x:  4.66436   Pos_y:   3.09899    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
ID:  63    Pos_x:  50.831   Pos_y:   9.97264    Vel_x:  -1e+09   Vel_y:   -1e+09
Tracking status:   9
-------------------------------------------------------------------
```

Figure 2. Output of radar data using the customized subscriber

The figure above shows the extracted information from the input radar data using the customized subscriber. By simply adding constraints on tracking status and range type to the output as the most basic filter, the number of displayed targets was reduced from 64 to around 10. This satisfying result serves as a great start and give me more confidence on making further improvement on radar data processing.

Challenges

The main challenge that I encountered for my individual work during the past week came from building the software application (subscriber). The progress was initially very slow due to my limited practice in C++ programming and lack of experience on developing applications using API. With help from Alex, I was finally guided to the right track and established right understanding on the general structure as well as specific functions from the provided API in order to make the right implementation.

Another challenge was that the only radar cable that the team had was already mounted on our testing vehicle (as we attached all sensors on the testing vehicle for FVE), and it required lots of effort to detach it from the vehicle. Therefore, testing on the radar could not be performed without the presence of the testing vehicle. As a result, the team has requested our sponsor for another radar cable so that future work on the radar can be performed even when the testing vehicle is absent.

Teamwork

Amit Agarwal:

Amit has been building knowledge on object tracking through reading some research papers, as he plans to help with work on stereo vision this semester.

Harry Golash:

Harry made some repairs on the testing vehicle. He is working on getting familiar with the PolySync, as he plans to help with work on the radar this semester.

Yihao Qian:

Yihao has been working on transplanting the existing work on stereo vision from Matlab to C++ using the OpenCV. He started the work during the past week with limited progress.

Menghan Zhang:

There was no much substantial work/contribution from Menghan that the team was aware of during the past week, partly due to the lack of communication between her and the rest of the team.

Plans

Before the next ILR/ progress review, our team would like to achieve the following goals on our project of developing the perception system using Stereo Vision and radar:

1. Achieve calibrated stereo vision using OpenCV in C++

2. Acquire depth information from detected object

3. Store extracted information from radar to file

4. Further research on filtering techniques for radar

5. (optional) visualization of extracted information from the radar

In term of individual work, I will be mainly involved in tasks related to the radar (Task 3-5).