

/High-performance-radar branch Analysis and results

Observations

1. Only Raw data is being logged into csv in the format :

```
timestamp,radar_id,frame_number,angle,distance,velocity,x,y,z,peak_val,range_idx,doppler_idx
```

Output:

Console O/P

```
python3 radar_system.py --live /dev/ttyACM1 --csv live_data.csv  
Added live radar 0: /dev/ttyACM1
```

Starting radar system with 1 sources...

Logging to CSV: live_data.csv

[SYSTEM] Started with 1 radar sources

[LIVE-0] Connected to /dev/ttyACM1

R0: 0.0° 468.8cm vel:562.2 peak:65530

[PERF] 0.0 Hz (1 frames)

R0: 0.0° 468.8cm vel:561.2 peak:1

R0: 0.0° 468.8cm vel:576.8 peak:65529

R0: 0.0° 468.8cm vel:576.6 peak:65512

R0: 0.0° 468.8cm vel:579.5 peak:65521

R0: 0.0° 468.8cm vel:581.2 peak:65512

R0: NOISE avg=16522.0 samples=8

R0: 0.0° 468.8cm vel:576.8 peak:65519

R0: NOISE avg=16522.2 samples=8

R0: 0.0° 468.8cm vel:572.6 peak:65521

R0: NOISE avg=8330.5 samples=8

R0: NOISE avg=8330.5 samples=8

R0: NOISE avg=8330.5 samples=8

R0: 0.0° 468.8cm vel:576.5 peak:65518

[PERF] 0.4 Hz (14 frames)

R0: 0.0° 468.0cm vel:576.4 peak:65500

R0: 0.0° 468.0cm vel:575.5 peak:65504

R0: 0.0° 468.0cm vel:576.2 peak:65507

R0: 0.0° 468.8cm vel:576.2 peak:65514

R0: 0.0° 468.0cm vel:576.3 peak:65507

R0: 0.0° 468.0cm vel:576.9 peak:65507

R0: 0.0° 468.0cm vel:580.9 peak:65495

R0: NOISE avg=16521.0 samples=8

R0: NOISE avg=16520.9 samples=8

R0: 0.0° 468.0cm vel:579.4 peak:65500

R0: NOISE avg=16520.9 samples=8

R0: 0.0° 468.0cm vel:580.7 peak:65497

R0: NOISE avg=16520.6 samples=8

R0: NOISE avg=16520.4 samples=8

R0: 0.0° 468.0cm vel:581.0 peak:65500

R0: NOISE avg=16520.4 samples=8

R0: 0.0° 468.0cm vel:580.2 peak:65498
 R0: NOISE avg=16520.0 samples=8
 R0: 0.0° 468.0cm vel:579.4 peak:65499
 R0: NOISE avg=16520.0 samples=8
 R0: NOISE avg=16519.9 samples=8
 R0: NOISE avg=16520.2 samples=8
 R0: 0.0° 468.0cm vel:581.7 peak:65499
 R0: NOISE avg=16520.1 samples=8
 R0: 0.0° 468.0cm vel:579.1 peak:65499
 R0: NOISE avg=16520.0 samples=8
 R0: NOISE avg=16519.8 samples=8
 R0: 0.0° 468.8cm vel:575.8 peak:65517
 R0: NOISE avg=8328.9 samples=8
 R0: NOISE avg=8329.4 samples=8
 R0: 0.0° 468.0cm vel:575.5 peak:65510
 R0: NOISE avg=8329.6 samples=8
 R0: NOISE avg=8329.8 samples=8
 R0: NOISE avg=8329.8 samples=8
 R0: 0.0° 468.0cm vel:571.6 peak:32
 R0: NOISE avg=9669.4 samples=14
 R0: 0.0° 468.0cm vel:575.1 peak:65508
 R0: NOISE avg=9670.2 samples=14
 R0: 0.0° 468.0cm vel:568.8 peak:35
 R0: 0.0° 468.0cm vel:575.5 peak:39
 R0: NOISE avg=145.5 samples=8
 [PERF] 1.5 Hz (55 frames)
 R0: NOISE avg=8337.4 samples=8
 R0: NOISE avg=8337.1 samples=8
 R0: NOISE avg=8337.1 samples=8
 R0: NOISE avg=8337.0 samples=8
 R0: NOISE avg=8337.1 samples=8
 R0: NOISE avg=8337.0 samples=8

CSV O/P:

```

timestamp , radar_id, frame_number, angle, distance , velocity , x , y , z , peak_val, range_idx, doppler_idx
1762589751.1152923, 0, 2561, 0.0, 468.75 , 562.2 , 4.6875 , 0.0, 0.0, 65530, 0, 5622
1762589751.1653802, 0, 2563, 0.0, 468.75 , 561.2 , 4.6875 , 0.0, 0.0, 1, 0, 5612
1762589753.2187352, 0, 2631, 0.0, 468.75 , 576.8000000000001, 4.6875 , 0.0, 0.0, 65529, 0, 5768
1762589753.4190316, 0, 2637, 0.0, 468.75 , 576.6 , 4.6875 , 0.0, 0.0, 65512, 0, 5766
1762589753.6193957, 0, 2645, 0.0, 468.75 , 579.5 , 4.6875 , 0.0, 0.0, 65521, 0, 5795
1762589753.7195728, 0, 2647, 0.0, 468.75 , 581.2 , 4.6875 , 0.0, 0.0, 65512, 0, 5812
1762589753.7195728, 0, 0, 0, 0, 16522.0 , 0, 0, 0, 0, 0, 0
1762589753.769654 , 0, 2649, 0.0, 468.75 , 576.8000000000001, 4.6875 , 0.0, 0.0, 65519, 0, 5768
1762589753.769654 , 0, 0, 0, 0, 16522.25 , 0, 0, 0, 0, 0, 0
1762589753.8197427, 0, 2651, 0.0, 468.75 , 572.6 , 4.6875 , 0.0, 0.0, 65521, 0, 5726
1762589753.8197427, 0, 0, 0, 0, 8330.5 , 0, 0, 0, 0, 0, 0
1762589753.8698337, 0, 0, 0, 0, 8330.5 , 0, 0, 0, 0, 0, 0
1762589753.9199147, 0, 0, 0, 0, 8330.5 , 0, 0, 0, 0, 0, 0
1762589755.0217664, 0, 2691, 0.0, 468.75 , 576.5 , 4.6875 , 0.0, 0.0, 65518, 0, 5765
  
```

```
1762589755.0718522, 0, 2693, 0.0, 467.96875, 576.4, 4.6796875, 0.0, 0.0, 65500, 0, 5764
1762589755.1219423, 0, 2695, 0.0, 467.96875, 575.5, 4.6796875, 0.0, 0.0, 65504, 0, 5755
1762589755.1720254, 0, 2697, 0.0, 467.96875, 576.2, 4.6796875, 0.0, 0.0, 65507, 0, 5762
1762589755.2722244, 0, 2699, 0.0, 468.75, 576.2, 4.6875, 0.0, 0.0, 65514, 0, 5762
```

2. Package is using an external Python parser and not CPP based drwig_parser

Logger should be using optimized CPP based drwig_parser with better resource utilization and performance.

3. I am assuming, the [PERF] log we are getting in console is the FPS for the logger. The max value we could obtain while running on live radar data was 16.7 Hz (657 frames). I have also attached the console output as well:

```
R0: NOISE avg=24875.4 samples=8
R0: NOISE avg=24876.8 samples=8
R0: NOISE avg=24877.9 samples=8
[PERF] 16.7 Hz (657 frames)
R0: NOISE avg=24879.2 samples=8
R0: NOISE avg=24879.8 samples=8
R0: NOISE avg=24879.1 samples=8
```

Updates needed

- I. The CSV logger should be completely CPP based . Only the processed data is to be stored in CSV files. Data should be stored in CSV after applying required scaling and formatting.

[Note : Refer drwig_parser.cpp for formatting and scaling examples.]

Seperate CSV for tracked and detected in the following format:

Note :All the below mentioned sample CSV format gives real world values.

Sample CSV for tracked_objects:

timestamp	frame	idx	y	vx	vy	relative_velocity	relative_velocity y (kmph)	width	length
2025-11-05 17:22:27.741	1	0	0.273438	10.1875	0.21875	0	0.00586	2.0390	2.0390
						924		6	6
2025-11-05 17:22:27.761	2	0	0.015625	10.1953	0.89062	0.00781	0.00917	2.0390	2.0390
					5	25	743	6	6
2025-11-05 17:22:27.761	2	1	0.5625	10.7656	-	0	-	2.0390	2.0390
					0.02343	0.00122		6	6
					75	293			

Units:

timestamp : YYYY-MM-DD HH:MM:SS:MS

x,y: Meters

vx,vy,relative_velocity(m/s): Meters per second

relative_velocity(kmph): Kilometers per hour
width,length: Meters

Position (x and y) , width and length in meters .Velocity along axes (vx ,vy) and relative velocity in meter per second and kmph . The relative velocity can be calculated from position and velocity variables.

$$\text{relative_velocity} = (x*vx + y*vy) / \sqrt{x*x + y*y}$$

where $\sqrt{x*x + y*y}$ will be your range.

Sample CSV format for detected_objects:

timestamp	frame	id	x	y	z	doppler_velocity	peak_value
2025-11-07							
13:56:55.755	1	0	-0.429688	4.66406	0	0	5080
2025-11-07							
13:56:55.755	1	1	0.796875	9.42969	0	0	4400
2025-11-07							
13:56:55.775	2	0	-0.375	4.67188	0	0	5160

Units:

timestamp : YYYY-MM-DD HH:MM:SS:MS

x,y: Meters

doppler_velocity(m/s): Meters per second

II. CPP based drwig_parser should be optimised to get better performance. Better thread and buffer handling and parallel threading for Dual radars parsing. We should be able to parse and store dual radar data at the same time.

III. Assuming the [PERF] log we are getting in console is the FPS for the logger, it needs to be increased. Max value I could notice after running for some time was 16.7 Hz (657 frames).

Application:

The end use case is that once a flag or function for activating the radar is called , the radar should start and store the real time tracked and detected objects data in a shared memory. The client's AEBS and BSIS systems will be accessing this shared memory to run their warning algorithms.Parallely once the radar is activated , the tracked and detected data should be stored in a csv file for verification and logging purposes. The logging should be done in atleast 20 Hz or more for tracked and detected objects. The more the fps the better. The end application will be completely CPP based. The criteria for activating and deactivating the radar will be the ego vehicle speed. Say if ego speed is between 20-30 kmph activate radar for BSIS and if ego speed is greater than 30 Kmph activate

speed : 0-20 kmph → Activate Rear Radar → BSIS Active

speed : 20-30 kmph → Activate Rear Radar + Front Radar → BSIS & AEBS Active

speed : >30 kmph → Activate Front Radar → AEBS Active

Conclusions:

1. CSV Logger

I. CPP logger should be used instead of Python.

ii. Store realtime processed, formatted and scaled data in separate CSV files for detected and tracked objects.

iii. Use optimized drwig_parser instead of external parsers

iv. Logger FPS should be at least 20Hz or more.

2. drwig_parser.cpp optimization

I. Better thread and buffer handling for better performance.

ii. Parallel threading for Dual radar processing and parsing.

iii. Radar should be able to take the ego vehicle speed as an input from CAN and use it for filtering out static objects.

3. Firmware

I. Activation and deactivation of radar by sending a message or command to the firmware. More like a request – response system. Even while doing so, what are the hardware issues that can happen?

ii. Radar data available over both UART & CAN