1. Steps to reproduce the results

1.1. Need docker?

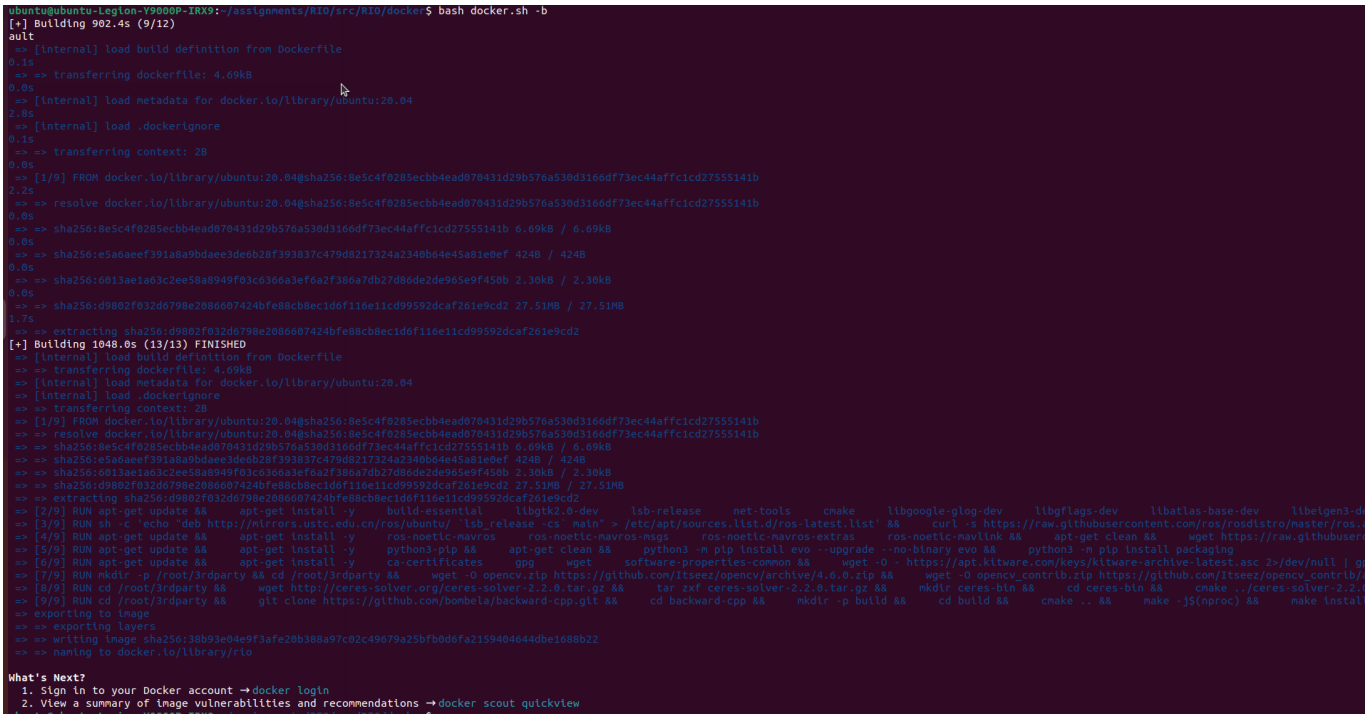
1.1.1. Download the docker image by

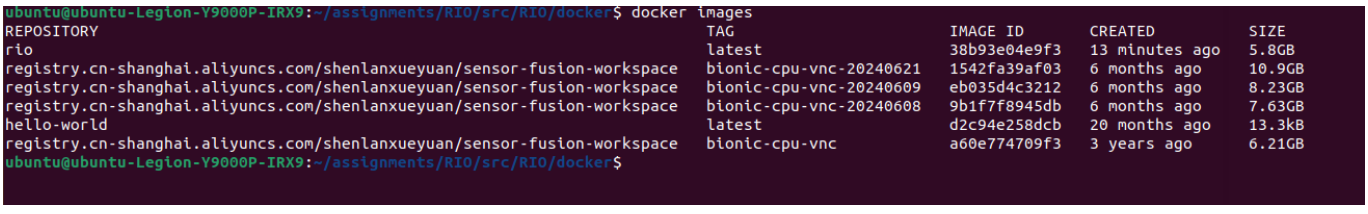
```
bash docker.sh -b
```

(then, it start download the file and build

```
function build() {
    docker build \
    -t rio \
    -f $SCRIPT_DIR/Dockerfile \
    $SCRIPT_DIR/..
}
```

)





1.1.2. Run the docker

```
bash docker.sh -r
```

This starts the Docker container and attaches your terminal to it. Once the container starts successfully, the terminal will switch to being inside the Docker container.

(mapping the current folder to docker

```
function run() {
    docker run -it --rm \
    --network host \
    --privileged \
    -v /dev:/dev \
    -v $SCRIPT_DIR/../:/ws/src \
    -e DISPLAY=$DISPLAY \
    -v /tmp/.X11-unix:/tmp/.X11-unix \
    rio \
    /bin/bash
}
```

)

1.2. ROS

bash 1

```
roscore &
rviz -d rio/config/RIO.rviz
```

bash 2

```
python3 /ws/src/docker/run.py -a -n rio -c /ws/src/rio/config/ars548.yaml -
d /ws/src/dataset/exp/Sequence_3.bag -r 1 -p 1
```

```
python3 /ws/src/docker/run.py -a -n rio -c /ws/src/rio/config/ars548.yaml -
d /ws/src/dataset/exp/Sequence_3_modified_gt_horizontal.bag -r 1 -p 1
```

1.2.1. coloradar

python3 /ws/src/docker/run.py -a -n rio -c /ws/src/rio/config/coloradar.yaml -d dataset/coloradar_trim/colo_trim_outdoors_run_0_modified_gt2.bag -r 1 -p 1

python3 /ws/src/docker/run.py -a -n rio -c /ws/src/rio/config/coloradar.yaml -d dataset/coloradar_trim/colo_trim_outdoors_run_0_modified_gt2.bag -r 1 -p 1

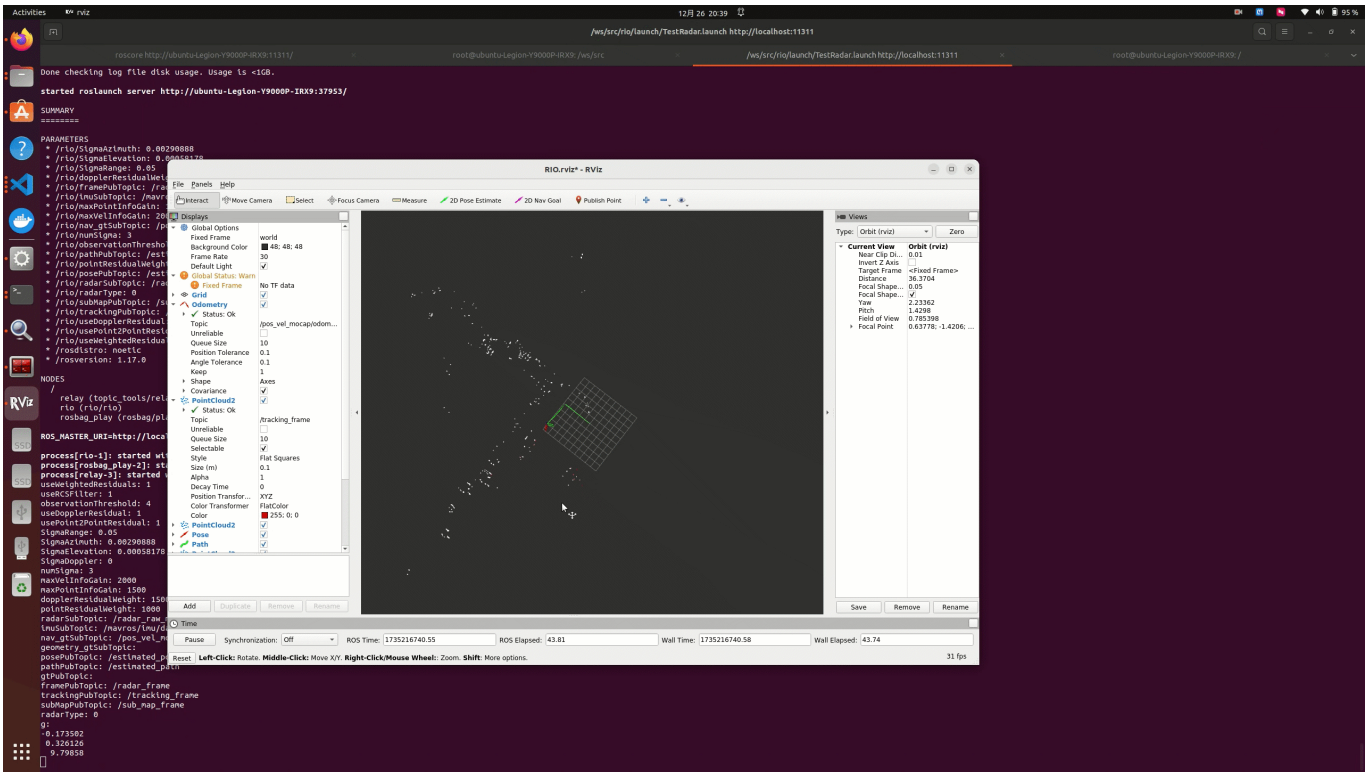
python3 /ws/src/docker/run.py -a -n rio -c /ws/src/rio/config/coloradar.yaml -d dataset/coloradar_trim/classroom3.bag -r 1 -p 1

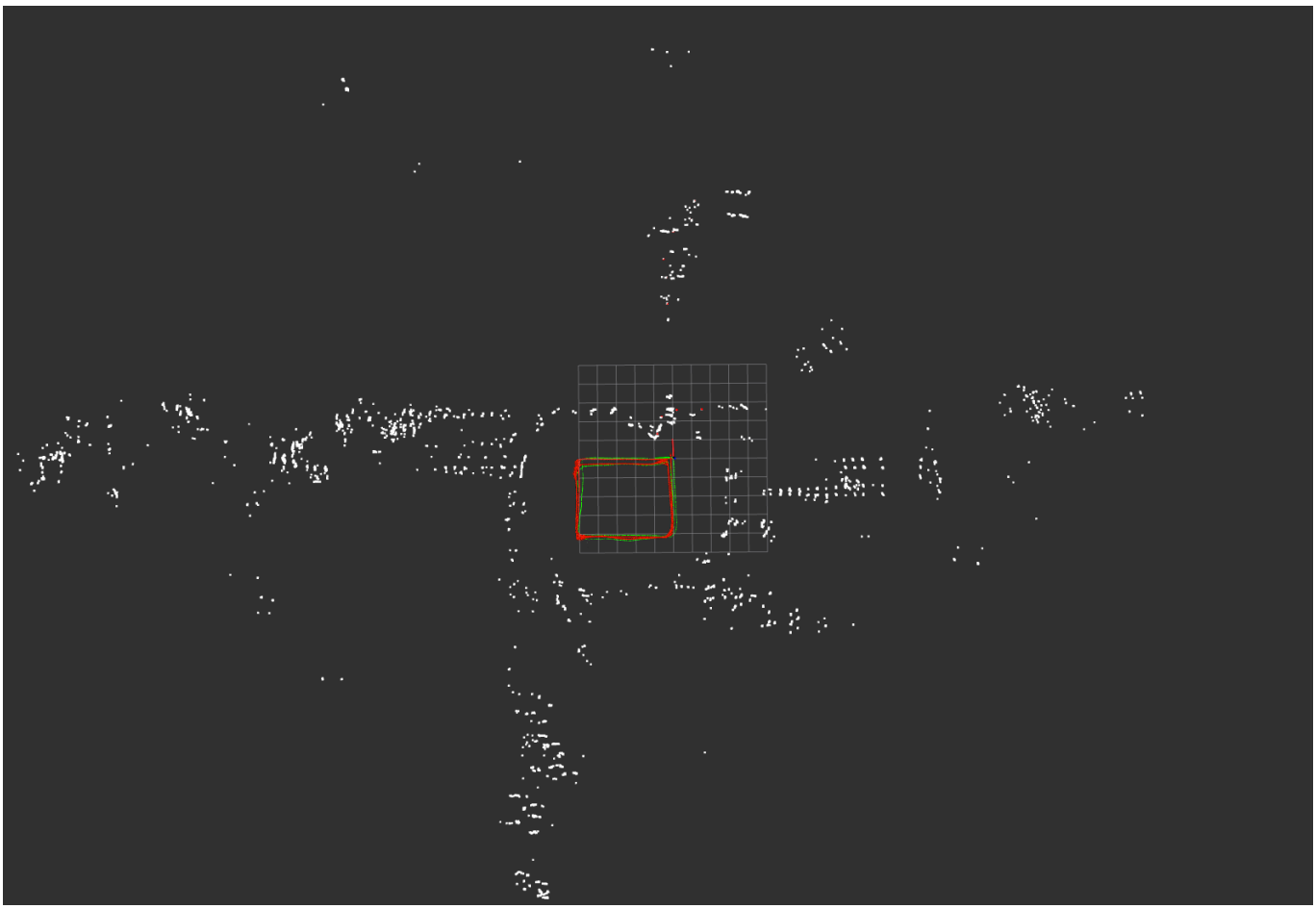
python3 /ws/src/docker/run.py -a -n rio -c /ws/src/rio/config/coloradar.yaml -d dataset/coloradar_trim/classroom3_modified_gt_horizontal.bag -r 1 -p 1

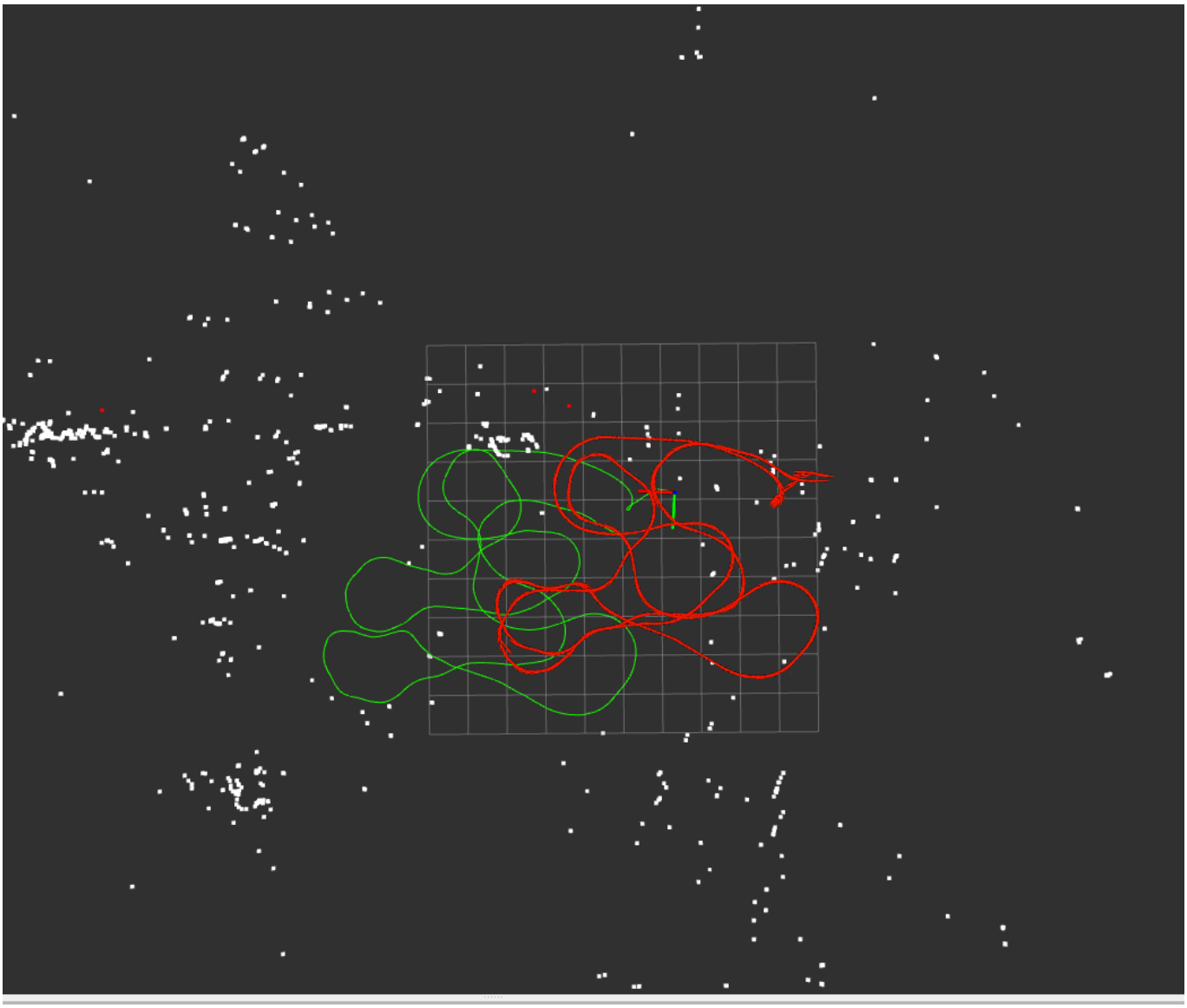
1.2.2. reproduce the groundtruth

evo_ape bag ./rio_output_seq3_2025-02-09-13-43-21_0.2toend.bag /estimated_pose /lidar_ground_truth -va --plot_mode xy --plot --t_max_diff 0.05

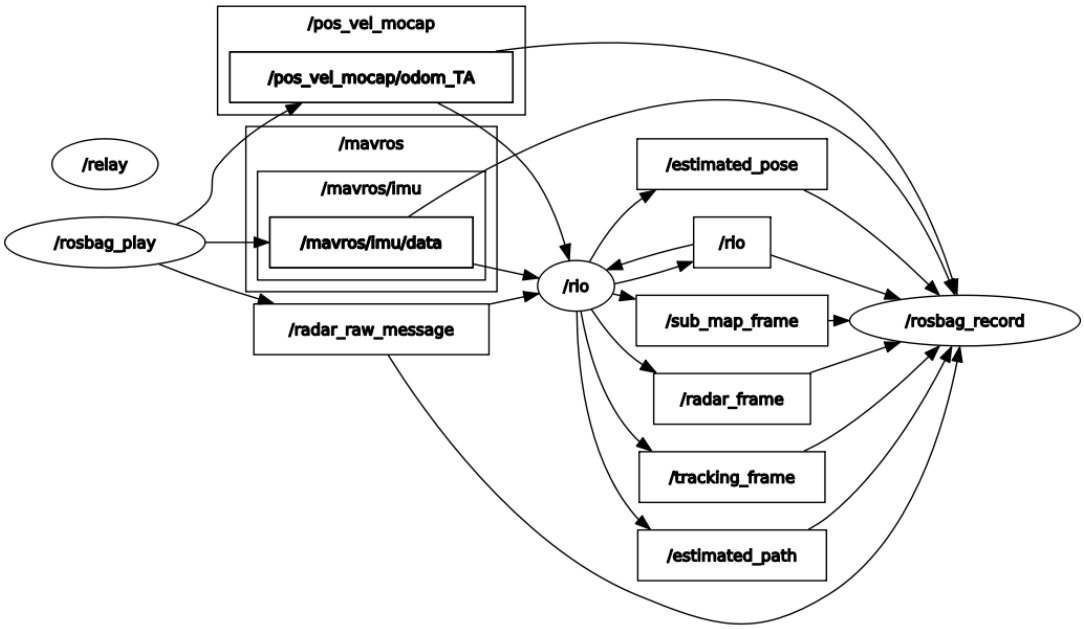
1.2.3. demo







1.2.4. rqt_graph



2. Code interpretation

2.1. what is frame

```
frame = scan2scanTracker.trackPoints(frameRadarData, timeStamp);

radarData.data.emplace_back(frame);
```

3. inherit from vins-mono

3.1. reducevector

3.2.

4. Sensor Charactoristic/Front end

4.1. Radar

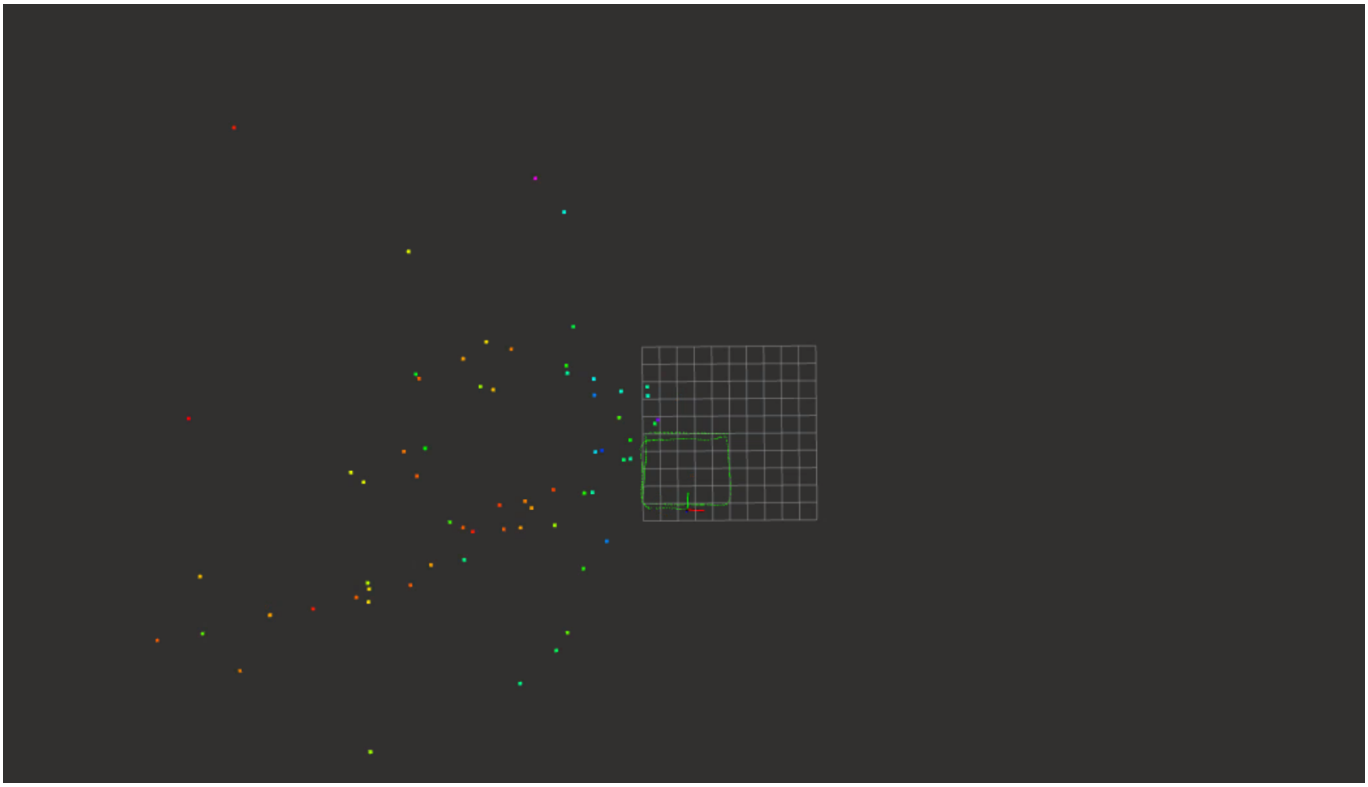
2. Structure of Binary Data:

The binary data likely looks like this (for a single point):

Field Name	Size (Bytes)	Type	Offset
azimuth	4	float	`offsetAzimuth`
azimuthSTD	4	float	`offsetAzimuthSTD`
elevation	4	float	`offsetElevation`
elevationSTD	4	float	`offsetElevationSTD`
range	4	float	`offsetRange`
rangeSTD	4	float	`offsetRangeSTD`
velocity	4	float	`offsetVelocity`
velocitySTD	4	float	`offsetVelocitySTD`
rcs	1	int8_t	`offsetRCS`

The offsets for each field correspond to the `fset` specified in the `fields` metadata. For instance:

4.1.1. /radar_frame



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
	%time	field.header.seq	field.header.stamp	field.header.frame_id	field.height	field.width	field.fields0.name	field.fields0.offset	field.fields0.datatype	field.fields0.count	field.fields1.name	field.fields1.offset	field.fields1.datatype	field.fields1.count	field.fields2.name	field.fields2.offset	field.fields2.datatype	field.fields2.count
1	1.7352256829073E+018	18	0	world	1	84	x	0	7	1	y	4	7	1	z	8	7	1
2	1.735225682863 1E+018	19	0	world	1	84	x	0	7	1	y	4	7	1	z	8	7	1
3	1.7352256828941 2E+018	20	0	world	1	79	x	0	7	1	y	4	7	1	z	8	7	1
4	1.7352256829360 1E+018	21	0	world	1	82	x	0	7	1	y	4	7	1	z	8	7	1
5	1.7352256829827 9E+018	22	0	world	1	75	x	0	7	1	y	4	7	1	z	8	7	1
6	1.7352256830407 4E+018	23	0	world	1	74	x	0	7	1	y	4	7	1	z	8	7	1
7	1.7352256830901 9E+018	24	0	world	1	85	x	0	7	1	y	4	7	1	z	8	7	1
8	1.7352256831360 6E+018	25	0	world	1	84	x	0	7	1	y	4	7	1	z	8	7	1
9	1.7352256831960 8E+018	26	0	world	1	80	x	0	7	1	y	4	7	1	z	8	7	1
10	1.7352256832345 6E+018	27	0	world	1	86	x	0	7	1	y	4	7	1	z	8	7	1
11	1.7352256832871 2E+018	28	0	world	1	87	x	0	7	1	y	4	7	1	z	8	7	1
12	1.7352256833293 8E+018	29	0	world	1	86	x	0	7	1	y	4	7	1	z	8	7	1
13	1.735225683383 5E+018	30	0	world	1	72	x	0	7	1	y	4	7	1	z	8	7	1
14	1.7352256834271 3E+018	31	0	world	1	86	x	0	7	1	y	4	7	1	z	8	7	1
15	1.7352256834796 9E+018	32	0	world	1	84	x	0	7	1	y	4	7	1	z	8	7	1
16	1.7352256835231 3E+018	33	0	world	1	79	x	0	7	1	y	4	7	1	z	8	7	1
17	1.7352256836011 1E+018	34	0	world	1	81	x	0	7	1	y	4	7	1	z	8	7	1

angle seems to be [-60, 60]x[-20, 20]

4.2. what is row_step

it is how many points there are in a single frame.

4.3.

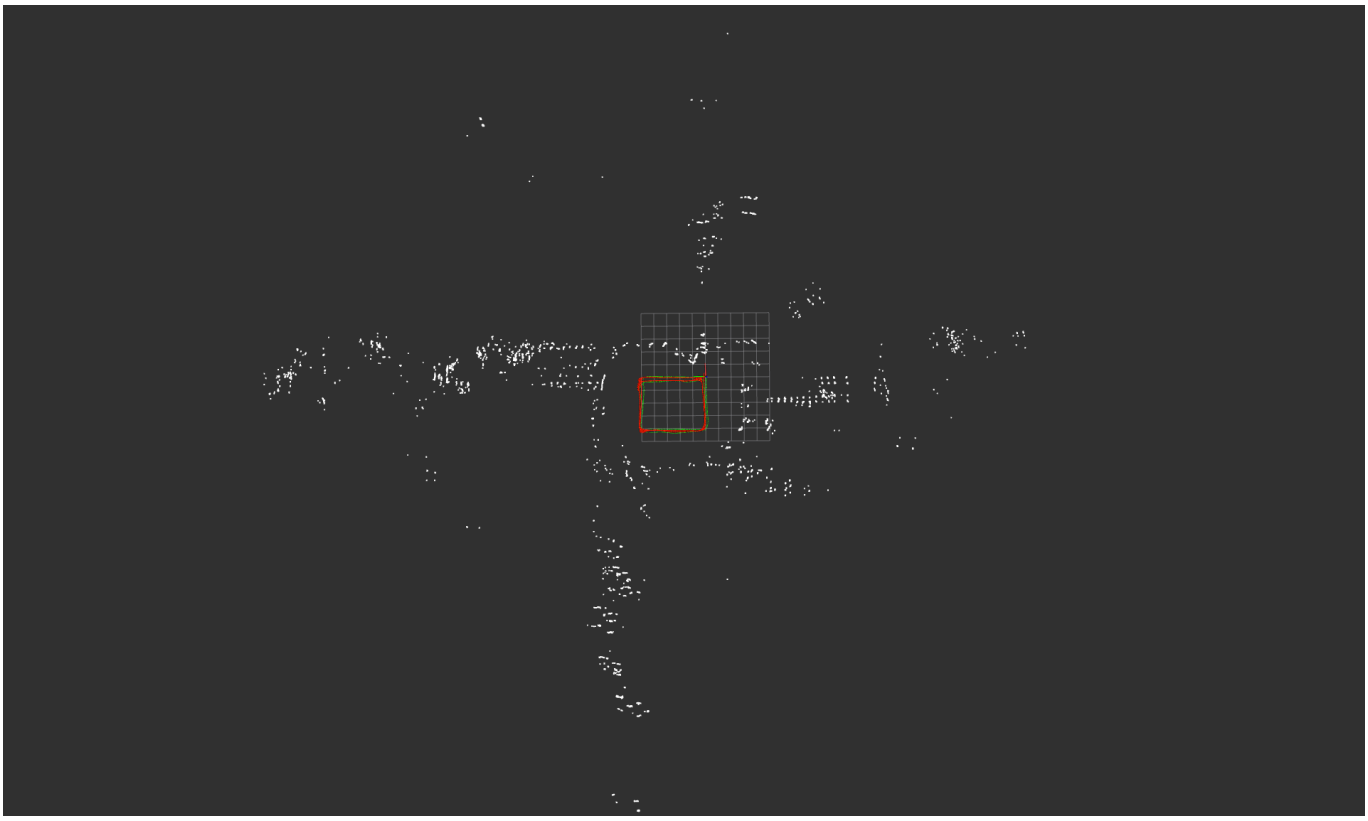
4.4. IMU

5. Back end/Optimization

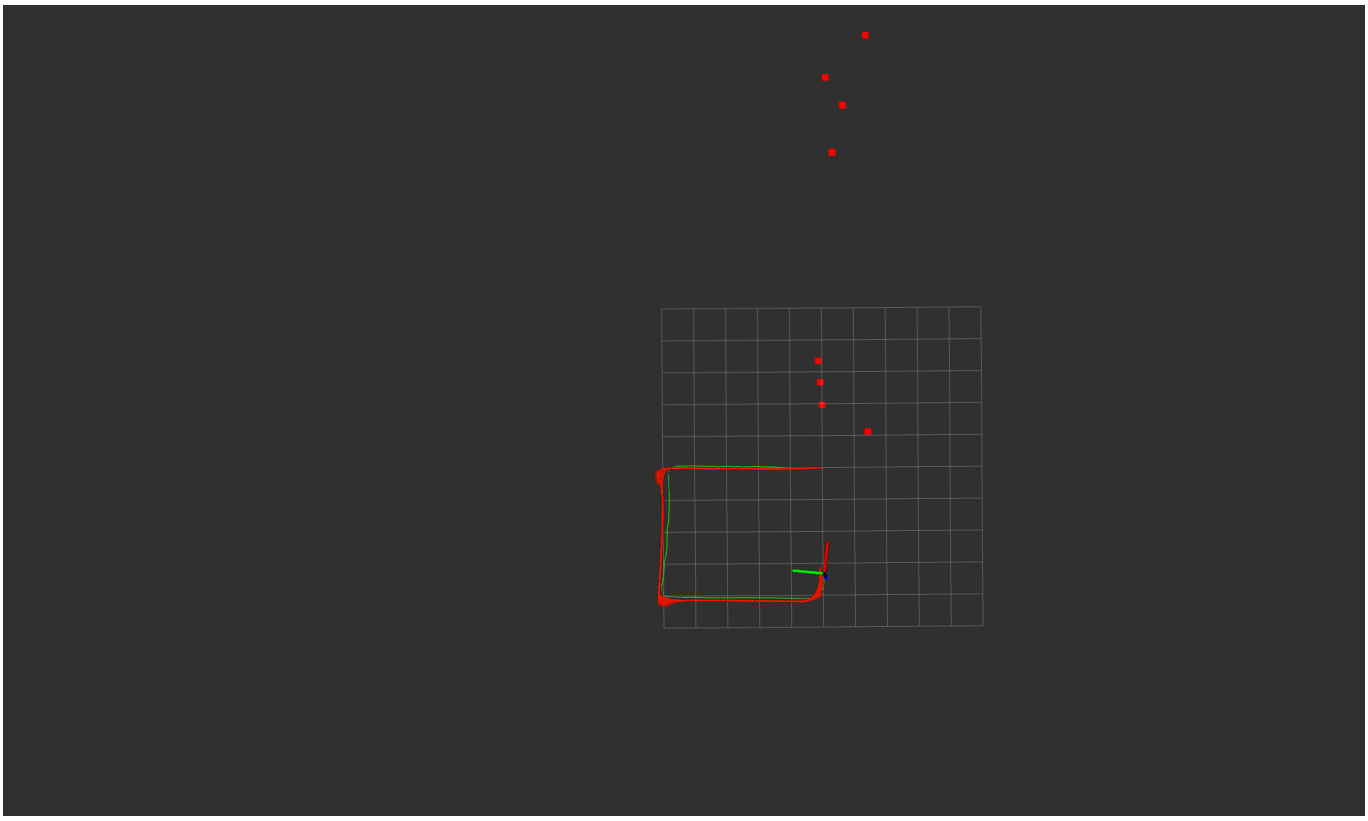
6. Output Data Extraction

6.1. map

6.1.1. /sub_map_frame



6.1.2. /tracking_frame



6.1.2.1. Is /tracking_frame the overlap of the current frame with sub_map?

Yes

6.1.2.2. Is radarFeatureFactor.pointRelation contains each point from the current frame, where the point without match turns out to be zeros-match?

Likely

6.2. Pose

6.2.1. /estimated_pose (world to inertial frame)

6.2.2. rio

seems to be groundtruth, the same name as node name

7. RIO frame management

all of the states uses inertial frames as body frames

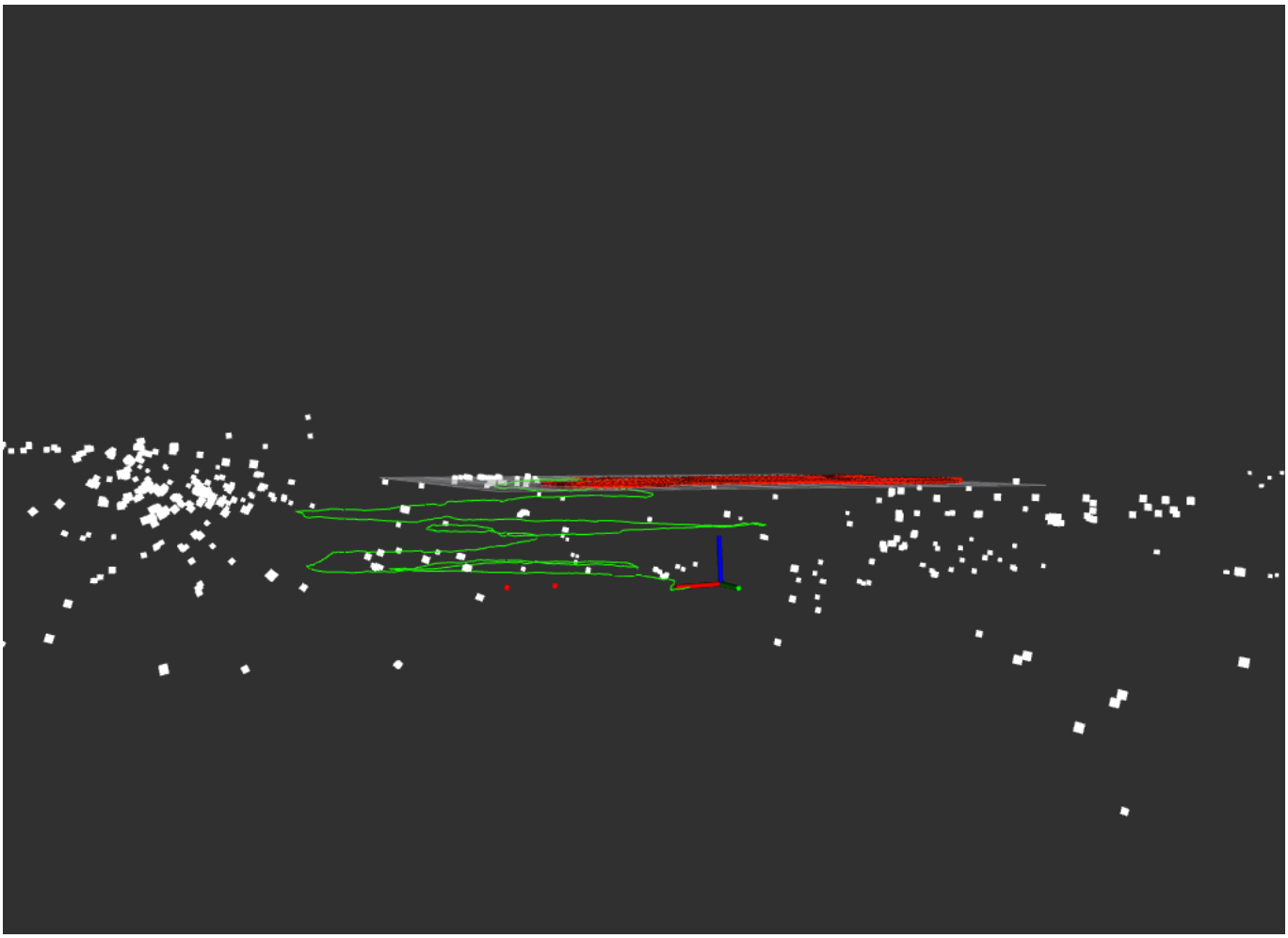
8. Issue

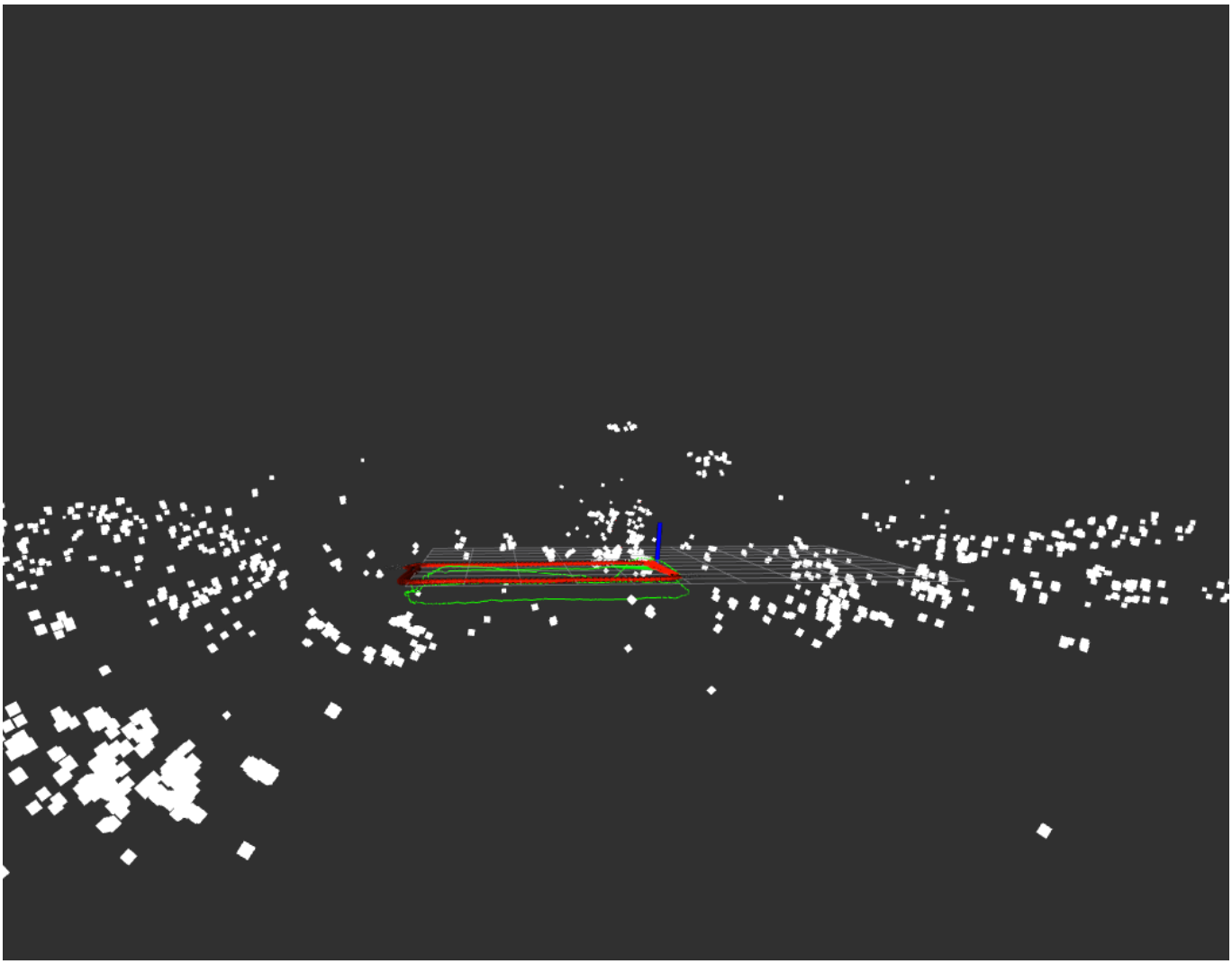
8.1. ERROR: Unable to start XML-RPC server, port 11311 is already in use (just occasionally occurs)

```
roscore -p 11312
```

```
export ROS_MASTER_URI=http://localhost:11312
```

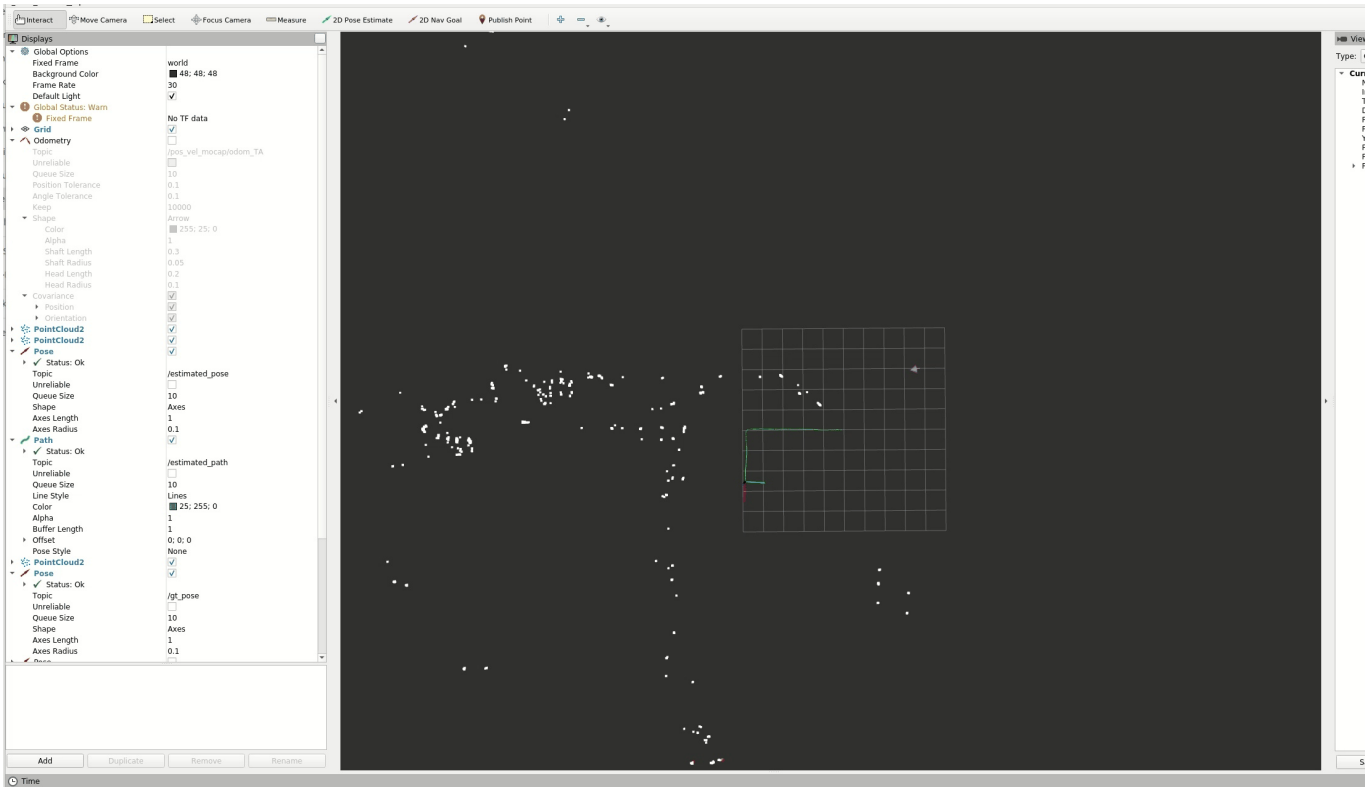
8.2. z error is very large





8.3. Bug

8.3.1. the rio msg is strange, always being jumping



8.3.2. the header timestamp of radar_frame is wrong

now it is zero, not very good. also the frame is world, which is not appropriate for visualization.

8.3.3. the rviz is dark

we can adjust window size in the rviz file, makeing it smaller.

sometimes retry many times can help alleviate the problem.

other times waiting and retrying are okay.

9. Xu Yang questions