<1>command for Nice online visualization of everything(not collect data)

1. rosnode kill -a

   # kill all the nodes

2. systemctl restart roverrobotics.service

   # open controller node, camera node and initialize the rover

3. (can be skipped)systemctl status roverrobotics.service

   # check if there are errors for the nodes

4. rosrun rslidar_sdk rslidar_sdk_node

   #open lidar

5. rosrun tf static_transform_publisher -0.02 -0.17 0.1 0 0 0 1 zed2_left_camera_frame rslidar 100

   # publish the transform between cam and lidar; the values can be adjusted wrt the calibration results

6. rviz #open rviz

   in the rviz interface, click files->open recent file-> click rvizmansetjinjingnodelet.rviz (this config file is saved under dir: /home/tud-jxavier/catkin_ws/src/handy_rviz)

   you can manually choose the sensor data you want to visualize.BY default, it displays the bounding box, zed point cloud, lidar point cloud.

<2>A minimal ros bag recording for visualise the same view **in rviz** in an offline manner: OD_rslidar.bag (will introduce more in detail in following offline visulisation section)

0.rosnode kill -a

1.rviz (setting the rviz file with rvizmansetjinjingnodelet.rviz)

2.rosbag play OD_rslidar.bag

<3>Data collcetion:<via BAG file generate full rate pointcloud pcd files>

In brief, simply run shell file:
./media/tud-jxavier/SSD/data_collection_shellsdata_collect.sh , and follow the console information press ctrl+c for several times, we will finish one round of data collection named with the collecting time.

(1)about the shell:

details about data_collect.sh file:

Preparation: cleaning data dir:

```
rosnode kill -a
systemctl restart roverrobotics.service
rosrun rslidar_pointcloud pointcloud_dir_cleaner_rslidar.py
rosrun zed_wrapper pointcloud_dir_cleaner_zed.py
```

Launch data collection and generate data:
(wait a sec, shut down the .sh file)

roslaunch rslidar_sdk **data_collection.launch** (CTRL+C when we want to stop the recording)
roslaunch rslidar_sdk pcd_tsp_generator.launch (the reason not putting the two launch files together is because there might be incorrect tracking of tsp before finishing writing all the pcd data; **details about this launch file is introduced later**)

Rename the newly generate dataset with time:

```
NOW=`date '+%F_%H:%M:%S'`
cp -R /media/tud-jxavier/SSD/data  /media/tud-jxavier/SSD/data_$NOW
```

(2)about ros launch file data_collection.launch:

Data collection and processing are the three stages below:
<via BAG file generate full rate pointcloud pcd files>

0. Initialize all the sensors:

   systemctl restart roverrobotics.service

1. all the data collection!: now we have 2 launch files, the data collection launch file will get all the required data and its tsp information except .pcd format point cloud, it will generate three bag files, one is the OD visualization bag and the other two are the full point cloud data from zed(30Hz) and rslidar(10Hz). We don't consider directly generating the pcd files because it has the problem of

dropping pcd frames if we use pointcloud_to_pcd node. For simplicity, we will collect .pcd files directly.

for the lidar zed auto calibration reason, we edit two sites of the file: /home/tud-jxavier/catkin_ws/src/rslidar_sdk/config/config.yaml , the two sites are the frame_id and the topic_name. When doing data_collection, we need to temporally change the yaml param file back. (refer to config_ori.yaml)

some changes you may want to do for the data collection.launch file:

1. the node “distance scan publisher”: you may want to change the rows you want to have the scan: M N ! (M<N, and in 0~125); by dedault: M N are both 80, means they get the 80 rows distance data.
2. the calibration of cam and lidar: by default : -0.04 -0.63 0.05 0 0 0 1 zed2_left_camera_frame rslidar

roslaunch rslidar_sdk data_collection.launch

2. **(this step 2 can be ignored** if we don't want a dense pcd files)manually converts the bag file to the corresponding pcd files **(this way can’t make pcd file be read with vim properly, so we decided to remain the sparse frame dropping manner point cloud not the pose proceesing bag file methods, so this step can be ignored. if it is needed, beforehand may need to run dummy_rslidar_pcd_cleaner dummy_zed_pcd_cleaner to clean directory)**

   rosrun rslidar_pointcloud pointcloud_dir_cleaner_rslidar.py
   rosrun zed_wrapper pointcloud_dir_cleaner_zed.py

   rosrun my_pcl_nodes jj_bag2pcd /media/tud-jxavier/SSD/data/bagdata/ZED_PCD.bag /zed2/zed_nodelet/point_cloud/cloud_registered /media/tud-jxavier/SSD/data/zed/point_cloud/

   rosrun my_pcl_nodes jj_bag2pcd /media/tud-jxavier/SSD/data/bagdata/rslidar_PCD.bag /rslidar_points /media/tud-jxavier/SSD/data/rslidar/Pointcloud/

3. Finally run a second launch file which generates the related tsp in a text file based on the properly named pcd files of zed and rslidar.

   roslaunch rslidar_sdk pcd_tsp_generator.launch

<some other information you may want to know if you want to collect a more densely pcd files for zed point cloud>

(can ve ignored if we are happpy with some dropping frames of pcd files)To increase the frame rate of pcd file, (for zed it can be up to 30 Hz, for rslidar it can be up to 10 Hz), we considered writing the source code for pointcloud2pcd from scratch.

tud-jxavier@tudjxavier-desktop:/media/tud-jxavier/SSD/data/zed/point_cloud$ rbag record /zed2/zed_nodelet/point_cloud/cloud_registered

tud-jxavier@tudjxavier-desktop:/media/tud-jxavier/SSD/data/zed/point_cloud$ rbag info 2021-07-09-17-45-48.bag

tud-jxavier@tudjxavier-desktop:/media/tud-jxavier/SSD/data/zed/point_cloud$ rrun pcl_ros bag_to_pcd 2021-07-09-17-45-48.bag /zed2/zed_nodelet/point_cloud/cloud_registered ./

altho above scheme won't drop messages, but have below flaws:
1> get every frame, take too much memory. Eg, zed pcd is 30Hz, will only generate with full rate.
2> when convert into the pcd file when from bag file, **the name of the pcd file is not formatted as we did before, we have to process again.**

so what other way round? change the rate of publication! manually via some script to change the name format of pcd files so meet the requirement : 16 unit int

now i embeded t**he name changing cod**e in the time stamp generator node, so maybe in the future the process should be:
1> auto data collection script (include OD bag and 2 full pcd bag)
2> play 2 full pcd bag and use plc_ros bag_to_pcd convert into non-corrected named pcd files in the corresponding directory.
3> Manually run the processing auto node: editor , generator.
4> vis:

```
for f in os.listdir('/media/tud-jxavier/SSD/data/zed/point_cloud/'):
    if  f.endswith('.pcd'):
        new_name=''.join(x for x in f if x.isdigit())[:16]+'.pcd'
        print('new name is :', new_name)
        os.rename(f,new_name)
```

<4>command for Nice offiline visualization of everything(with and without rviz)

(1)with rviz:

1. vis the OD,lidar pointcloud ,zed pointcloud in rviz.

rosnode kill -a
rosrun rviz rviz -d /media/tud-jxavier/SSD/data/Rviz_config/vis_bagdata.rviz
rosbag play /media/tud-jxavier/SSD/data/bagdata/OD.bag

rosnode kill -a

rviz (setting the rviz file with rvizmansetjinjingnodelet.rviz)

rosbag play OD_rslidar_zedpcd.bag

(2)without rviz:

2. vis 2D aligned frame stream with BB and label:

align timestamp data of OD tsp and left tsp.
python /media/tud-jxavier/SSD/data/data_helper/assosiate.py
/media/tud-jxavier/SSD/data/zed/timestamps/timestamp_OD.txt
/media/tud-jxavier/SSD/data/zed/timestamps/timestamp_IMG.txt 0.10001
/media/tud-jxavier/SSD/data/zed/timestamps/alighed_timestamp.txt
vis 2d stream:
python /media/tud-jxavier/SSD/data/zed/visulisation/vis2d.py
/media/tud-jxavier/SSD/data/zed/timestamps/alighed_timestamp.txt

3. vis certain aligned left frame with BB and label:

pick one reasonable(i.e. there are corresponding OD info) left image tsp 1626352358974896 for example.

usage: python /media/tud-jxavier/SSD/data/zed/visulisation/vis2d.py 1637659459417035

4. vis 3D aligned pointcloud stream with BB:(not recomended!)

aligh tsp of OD and zed pcd:

python /media/tud-jxavier/SSD/data/data_helper/assosiate.py
/media/tud-jxavier/SSD/data/zed/timestamps/timestamp_OD.txt
/media/tud-jxavier/SSD/data/zed/timestamps/timestamp_pcd_zed.txt 0.10001
/media/tud-jxavier/SSD/data/zed/timestamps/alighed_timestamp.txt
vis slow stream:
/media/tud-jxavier/SSD/data/zed/visulisation/vis3d_proj/build/vis3d
/media/tud-jxavier/SSD/data/zed/timestamps/alighed_timestamp.txt

5. vis certain aligned pcd with BB and label:
pick one reasonable(i.e. there are corresponding OD info)pcd tsp
1626352359374947 for example.
/media/tud-jxavier/SSD/data/zed/visulisation/vis3d_proj/build/vis3d
1637659449616185

6, visualize single frame via **pcl_viewer**

pcl_viewer /media/tud-jxavier/SSD/data/zed/point_cloud/1635699662568589.pcd
-use_point_picking

then press H,u,g,5, you can have different visualization toolkit

trash doc:

systemstl restart rover
rosrun rviz rviz -d /media/tud-jxavier/SSD/ta/Rviz_config/vis_rslidar.rviz
rosrun rviz rviz -d /media/tud-jxavier/SSD/ta/Rviz_config/vis_bagdata.rviz // change to rslidar

roslaunch rslidar_sdk data_collection_backup.launch

rosrun rviz rviz -d /media/tud-jxavier/SSD/ta/Rviz_config/vis_rslidar.rviz

rosrun rviz rviz -d /media/tud-jxavier/SSD/ta/Rviz_config/vis_bagdata.rviz

data_collection_ori

SYSTEMCTL

EVERYTHING

systemctl restart roverrobotics.service

<!-- rviz , this is for really carrying out OD-->

```
  <node pkg="rviz" name="rviz" type="rviz" args="-d
/media/tud-jsxavier/SSD/data/Rviz_config/vis_bagdata.rviz" >
  </node>
  <node pkg="rviz" name="rviz2" type="rviz" args="-d
/media/tud-jxavier/SSD/data/Rviz_config/vis_rslidar.rviz" >
  </node>

rosrun strata_radar  dummy.py 30
rosrun strata_radar strata_RangeDoppler_vis.y
rosrun rslidar_sdk rslidar_sdk_node
```

```
 systemctl restart roverrobotics.service
rosrun strata_radar  dummy.py 30
rosrun strata_radar strata_RangeDoppler_vis.y
roslaunch rslidar_sdk everything.launch
```