Deep Learning for Autonomous Driving Spring 2021

Project 1: Understanding Multimodal Driving Data

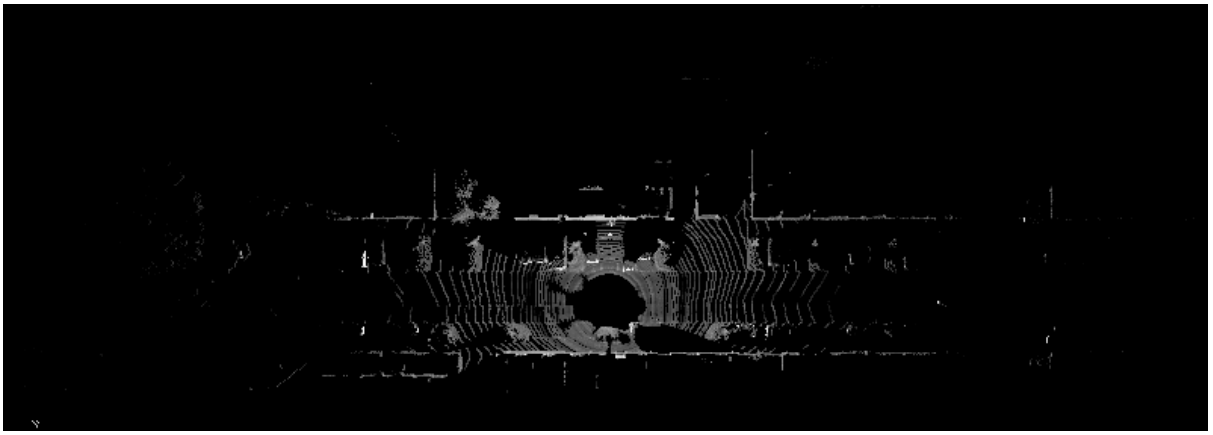
Group members: Xiao'ao Song(18-747-949) Qi Ma(20-960-225)

Problem 1: Bird's eye view

Problem description: Display the BEV image of the given scene with pixel intensities corresponding to the points' respective reflectance values.

The resolution of BEV image is (0.2, 0.2)m in x, y respectively, therefore, we will first the x,y coordinates of the laser 3D points by 0.2m.
After that, if multiple points lie within the same bin, we sample the highest intensity point.
The coordinate systems are defined the following way, where directions are informally given from the drivers view, when looking forward onto the road: Velodyne: x: forward, y: left, z: up

Result image:



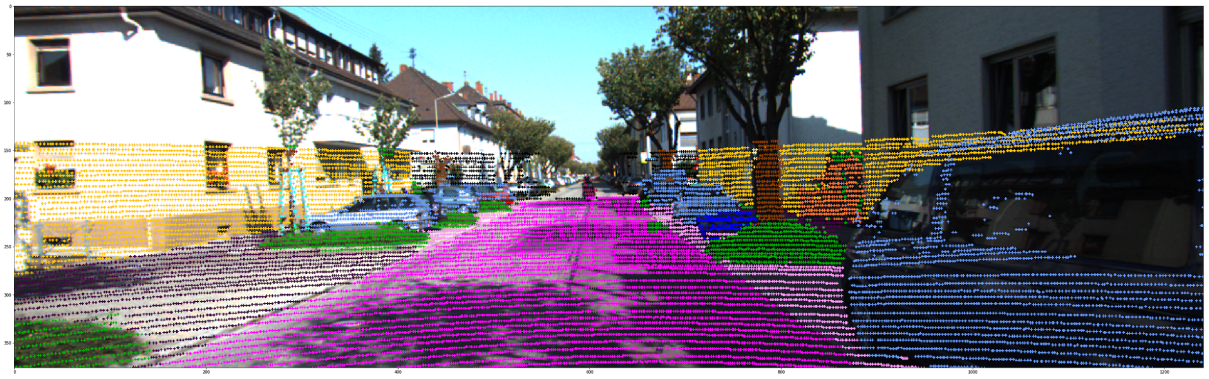
For visualization purposes the figure is rotated by 90°. The x-axis of the image is the negative y-direction of the Lidar coordinate system and the y-axis of the image is the x-direction of the Lidar coordinate system.

Problem 2: Visualization

2.1 Semantic Segmentation: Displaying Semantic Labels
Problem description: Given the intrinsic and extrinsic projection matrices, project the point cloud onto the image of Cam 2. Color each point projected according to their respective semantic label. The color map for each label is provided with the data.

To do this, we first need to define the transformation from Velodyne system to camera2 system. We get this transformation by multiplying the 'P_rect_20' matrix with the 'T_cam2_velo' matrix. Next, we project the 3D Velodyne points onto the 2D image frame by applying this transformation matrix and then normalize the 2D points. With the known FOV info, we also have to filter out the 2D points by the height and width of the image because only the points that are inside of the range of image height and width can be displayed. The last step is we visualize each point according to its colormap. The final result can be seen in the below image.

Result image:



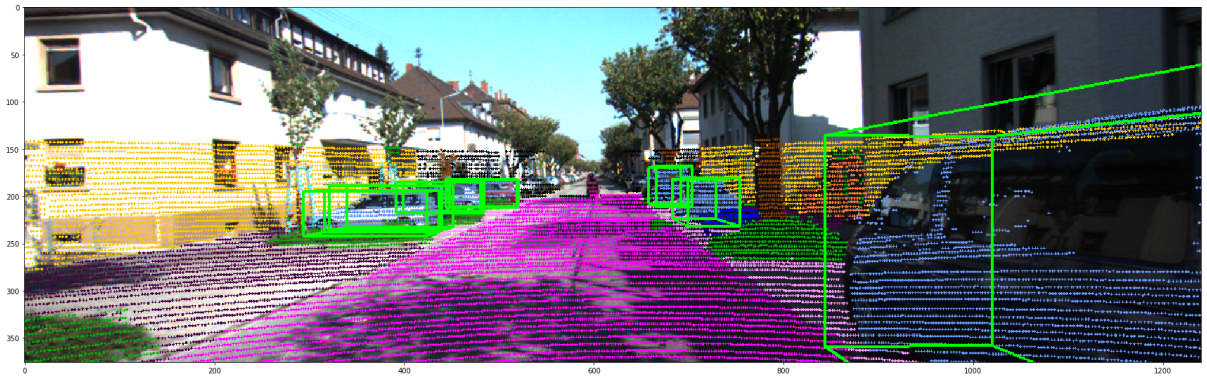
In the above image, the point cloud have been projected onto image and colored according to semantic labels

2.2 Object Detection: Drawing 3D Bounding Boxes

Problem description: In addition to the points with semantic labels, project the 3D bounding boxes of all given vehicles onto the Cam 2 image.

We need to first compute the center of the box which is the center of the bottom face, get the corners of the bounding box according to its height, width and length information, and get its rotation around Y-axis in Cam 0 coordinates. Then we project these box corners onto Camera 2 coordinates by applying the transformation matrix 'P_rect_20'. Next, we draw the lines to visualize the bounding boxes for given objects. Last but not the least, we also need to clip some values if any line is out of the FOV range of image. The final result can be seen in the below image.

Result image:

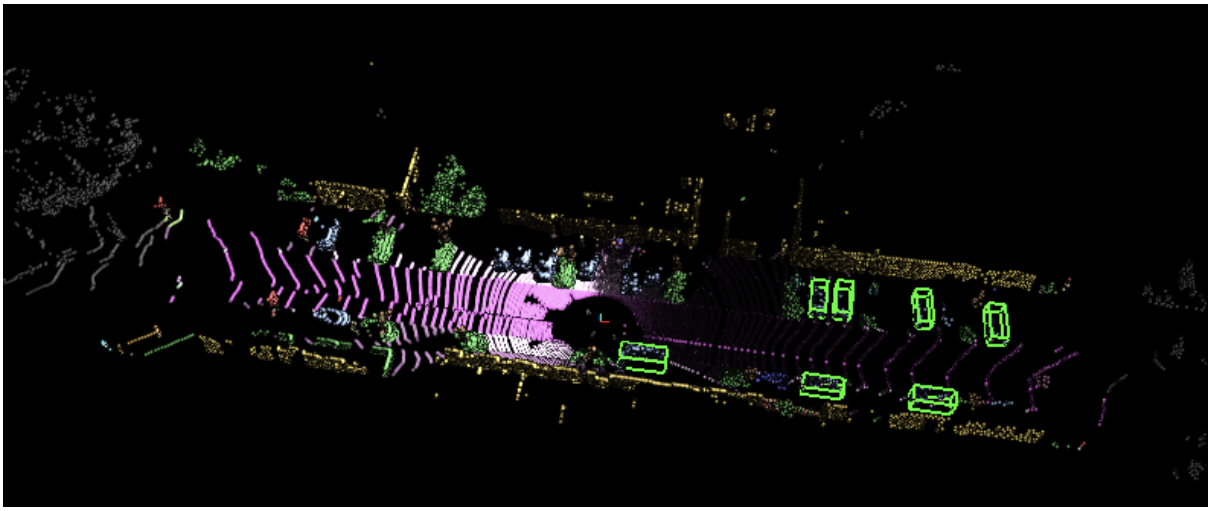


In the above image, we projected the 3D bounding boxes of all given vehicles onto the Cam 2 image and colored the box in green.

2.3 Modify LiDAR point cloud visualization in 3D

After modifying the 3dvis.py, we can visualize the scene in 3D. We do not need to filter out points because the point clouds are visualized in 360 degree. From this 3D plot, we can see 7 cars clearly, however, these 7 cars cannot be recognized clearly on the 2d image. On the left side, we can see that 2 cars overlap together. Therefore, we might count 6 cars most likely from this 2d image.

Result image:



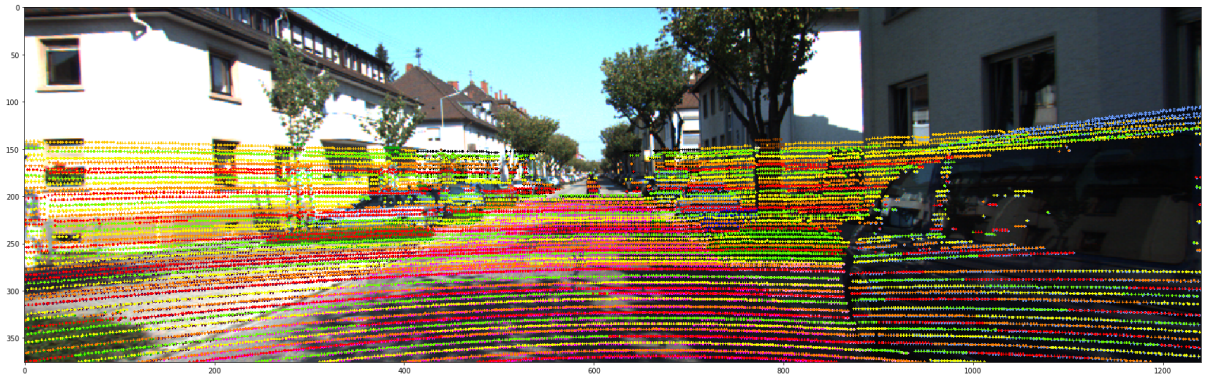
Problem 3: ID Laser ID

The laser IDs can be 'roughly' (due to noise in the data) figured out from the point cloud directly by the elevation. Therefore, we first calculate the elevation of each point cloud by computing the tangent.

$$tan(elevation) = \frac{z}{\sqrt{x^2 + y^2}}$$

The standard vertical Lidar Field of View is given by +2.0° to -24.9° (range: 26.9°). However, we might get some values beyond this range, e.g. -25.79°, 2.87° and etc. This might be caused by measurement noise. Therefore, we filtered out these points that are beyond the vertical measurement scope of Lidar. Then, we divided the standard scope into 64 bins and assigned each point into a bin according to its vertical angle we just computed. Finally, we use four alternating colors to indicate the identified IDs. This coloring function was directly taken from data_utils.py file. The final result can be seen in the below image.

Result image:



Problem 4:

Problem description:

Motion distortion comes from the motion of a mobile base which includes rotation and translation. Since the lidar detects points in its own frame, it is important to transform all the depth information to the world frame unless we have no idea where the pedestrian other vehicles are in real time. Therefore the first task is take the vehicle motion into consideration.

If we could get continuous information about the environment like speed, camera image, the above question is nothing but some rigid transformation. However, consider the real world implementation: the sensors could only get discrete information also with noise. Different sensors also have different frequency and bandwidth. In short, sensor fusion is also an important task in this problem.

In conclusion, we divide the problem into 2 tasks, namely sensor fusion and rigid transformation. Which also divide into sub-task

Task 1: Sensor fusion

1. Sensor synchronization (done)
2. data filter (done)
3. Data reorder

Task 2: Rigid transformation

1. relative rigid transformation calculation
2. data transform
3. Data visualization

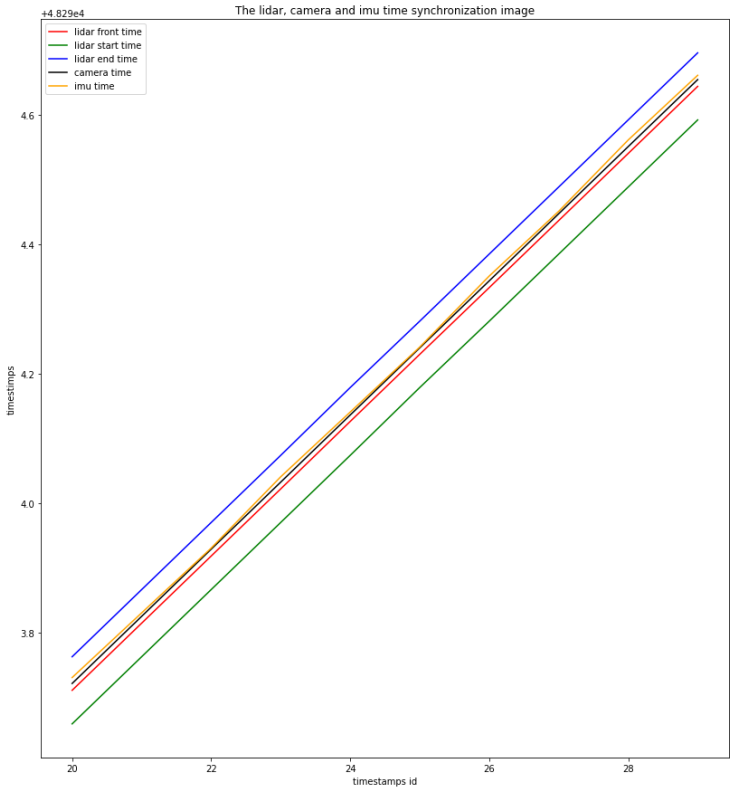
Assumption:

In order to tackle this problem more easily and avoid complicated and unnecessary calculations the hint shows that we can simplify our questions and make following assumptions:

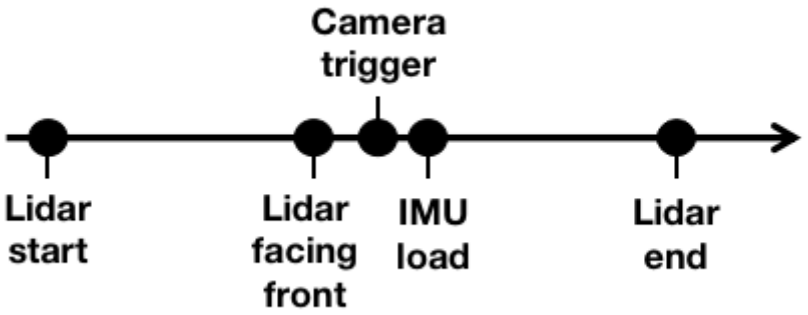
1. The event emerged in such order: 'Lidar start -> Lidar face front -> Camera triggered -> Lidar ends ...'
2. The vehicle only rotate in z axis and it move in plan
3. Translation and rotation are independent and can be solved separately
4. We assumed the rotation and translation speed was constant during one period of lidar.
5. We assumed the lidar point is fire in sequence and when we ignore the overlap area problem
6. We assume the velocity and rotation speed sensed by IMU is same as the lidar frame

Assumption evaluation:

1. We show the timestamps in frame 20-30 and the following image shows the event happened in such a sequence.

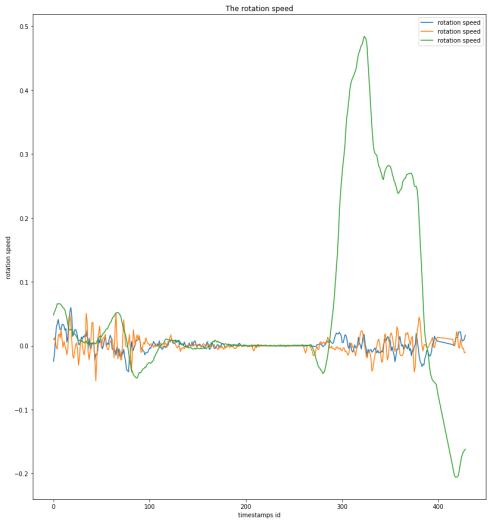


timestamps of different event in frame 20-30

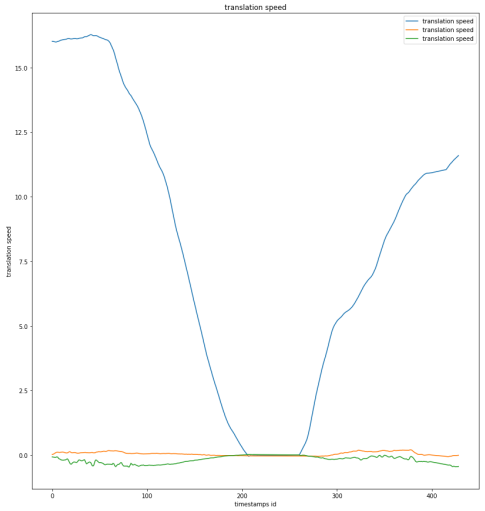


event sequence in one lidar period

2. We plot the x,y,z direction rotation speed in the total frame: we can see that the rotation speed in x, y. keep no more than 0.5 rad/s. In a period of lidar scan (about 0.1 second). The maximum error is 0.05 rad and far lesser than angular resolution. We can assume only rotate in yaw.

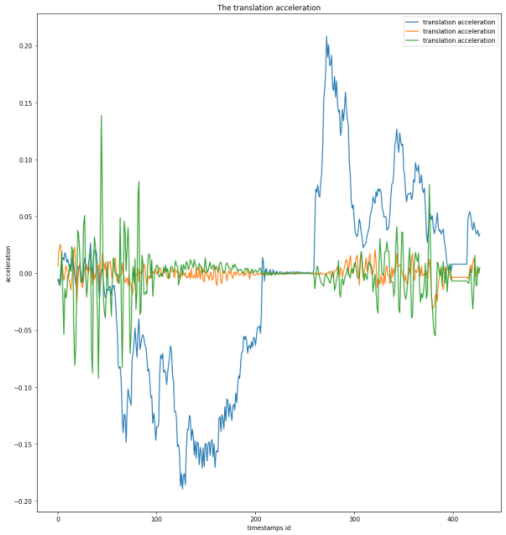


speed in x,y,z direction

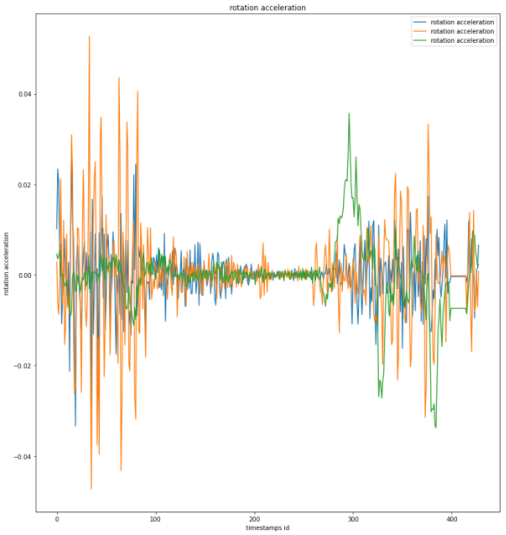


angular speed in x,y,z direction

3. In the car dynamics the translation is highly coupled with rotation. Since the vehicle could not purely rotate. However, in such a short time period we analyze the data the maximum speed is 16.27 m/s, maximum rotation is 0.484 rad/s. Whereas in 0.1 second the translation is 1.67 m and 0.0484 rad, In this case we can assume these two terms decoupled and independent.
4. We assumed the rotation and translation speed was constant during one period of lidar. The maximum acceleration (we define the speed and rotation difference between continuous timestamps is our acceleration) the maximum is 0.208 m^2/s and 0.052 rad^2/s. **This bring some errors when car have a large acceleration**
5. We assumed the lidar point is fired in sequence and when we ignore the overlap area problem: When the lidar rotates the same direction with the car rotation, it brings an overlap point cloud problem. Some points in the overlap area come from the lidar start while some of them come from the lidar end. In our problem the car rotation speed is far smaller than lidar rotation speed. So the overlapping area is so small that we can ignore this problem.



acceleration in x,y,z direction



angular acceleration in x,y,z direction

6. We assume the velocity and rotation speed sensed by IMU is same as the lidar frame

We could import the IMU transform matrix:

[R, T]

```
[[ 9.999976e-01  7.553071e-04 -2.035826e-03]  [[-0.8086759]
 [-7.854027e-04  9.998898e-01 -1.482298e-02]   [ 0.3195559]
 [ 2.024406e-03  1.482454e-02  9.998881e-01]]  [-0.7997231]]
```

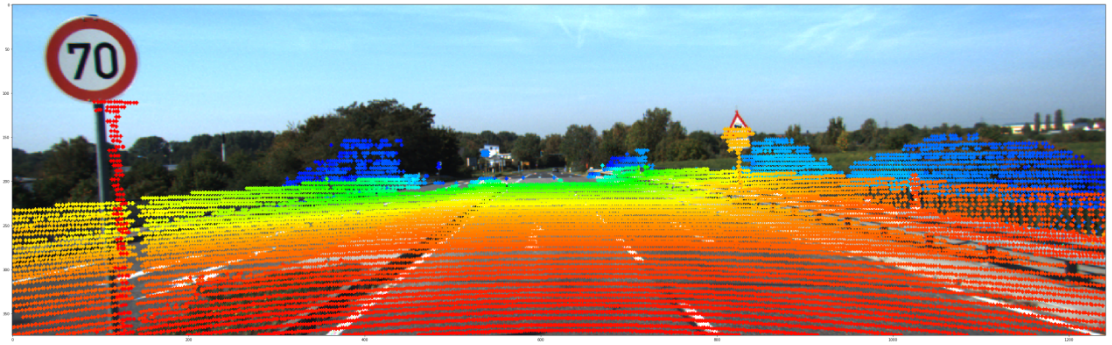
In the end we will make a comparison with the outcome with or without imu transformation.

Question solving

Motion distortion visualization:

In this part we first show how big motion distortion influences by projecting the lidar point in the camera frame without motion distortion.

As solved in question 2 we first project the point in the lidar frame into the camera image frame and then use direct linear transformation and project 3D point to 2D image plane. Moreover, we filter out the points in the back and out of the image.



frame 37 with motion distortion

We can see very clearly in the left of the image the speed limit sign has an obvious shift. We can make a brief analysis here:

1. Why is there a shift and why is the shift in the right direction?
   We can set the time when the camera takes the image as 0, and since lidar rotates clockwise and triggers the camera when facing front, the speed limit sign is sensed by lidar before the camera triggers. However, when we project all the points into the image we assume these events happened at the same time which is definitely wrong since the car moves forward for some distance. So we actually add this distance to the sign so it move in front where it should be (right direction)
2. Why is there a gap near the sign limit?
   The gap in fact is some environmental information which can not be seen by the lidar because of the occlusion. However, the gap is not smooth and narrower than the sign because when cars move forward we reveal some environment information which could not be seen first.
   Except from still objects the moving object also causes the occlusion problem, even if the car is not moving, the different position of camera and lidar will also result in these gaps.

In this part we will solve the motion distortion task by task

Start angle compute:

Because in the provided LiDAR point cloud, the order of the points is lost – no information about which point is taken before which other points. The task with highest priority is to reorder the data. We order the point cloud by the angle relative to the lidar coordinate system. But firstly, we need to figure out what angle the lidar starts.

Start angle compute:

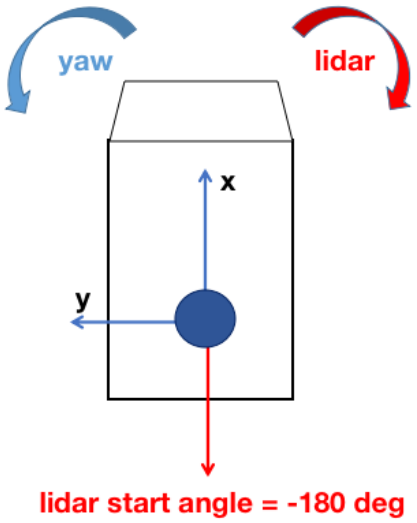
$$start\ angle\ =\ 0deg\ -\ 360deg\ *\ (forward\ time\ -\ start\ time)/(end\ time\ -\ start\ time)$$

1)

From this equation we could approximately get the start angle -180 degree. The minus here can be understood as we set the front angle as 0 degree. So the whole period of lidar start from -180 degree to 180 degree with 0 degree facing forward.

Horizontal angle compute:

Since the lidar rotates clockwise which is in opposite the yaw angle of vehicle rotation. We construct the *get_horizon_angle* function to calculate all the lidar 3D points horizontal angle. From lidar start angle to forward: -180deg - 0 deg and from forward angle to end angle: 0deg - 180 deg.



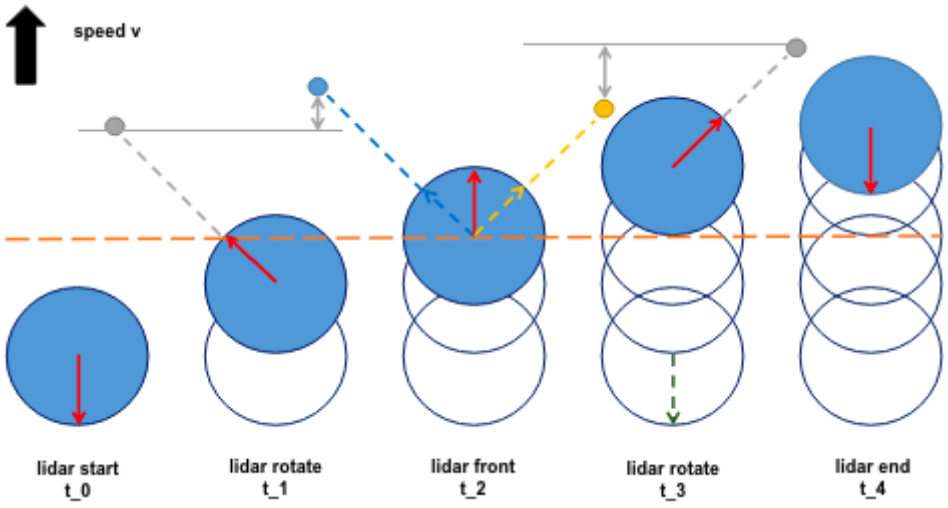
define lidar front angle be zero and start angle -180 as example

Data reorder:

We append the angle information with the 3D points (x,y,z,confidence) and order all the points by the horizontal angle. So far we have a 3D points data in order from lidar start to lidar end.

Rigid transformation:

In this part we first introduce how motion distortion influences the depth estimation and then calculate the motion translation distance and rotation degree we dismiss.



distortion by translation(moving front)

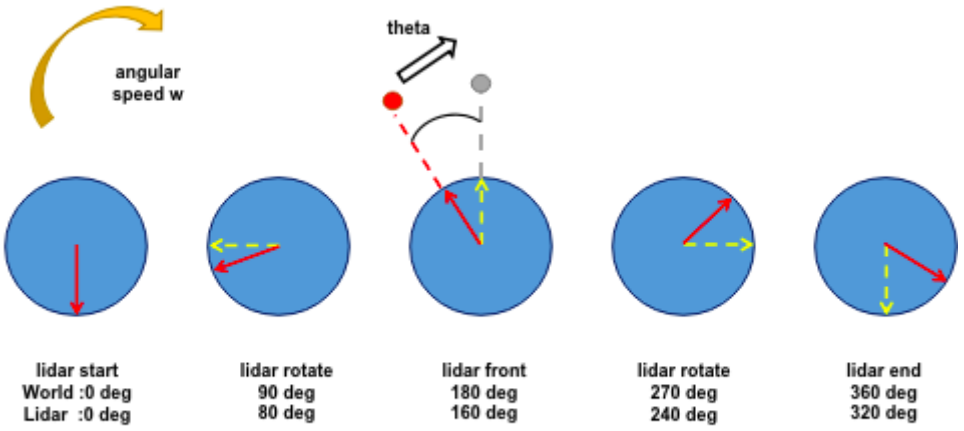
In the above image, the blue circle symbolify the lidar and red arrow symbolify the ray direction in the lidar coordinate system. The grey point is the real 3D point detected by the lidar and blue point is the false assumed point in the front image. We can easily conclude that the point detected before the image taken should move back while the point detected after the image taken should move forward the moving direction.

We set the translation distortion a one period as:

$$distance\ error\ =\ speed\ *\ (t_{end} - t_{start})\quad 2)$$

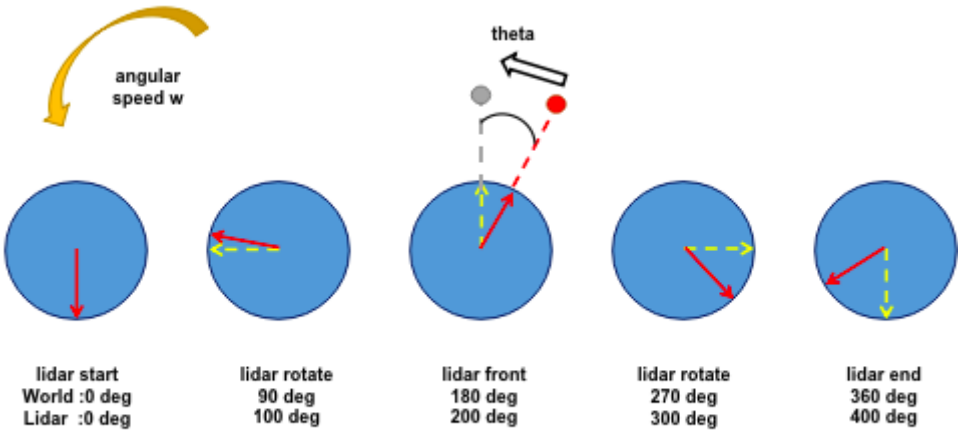
So if we have the time difference between the detect time t_1 and front view time t_2, the shift distance t is:

$$t\ =\ -\ distance\ error\ *\ (t_2 - t_1)/(t_{end} - t_{start})\quad 3)$$



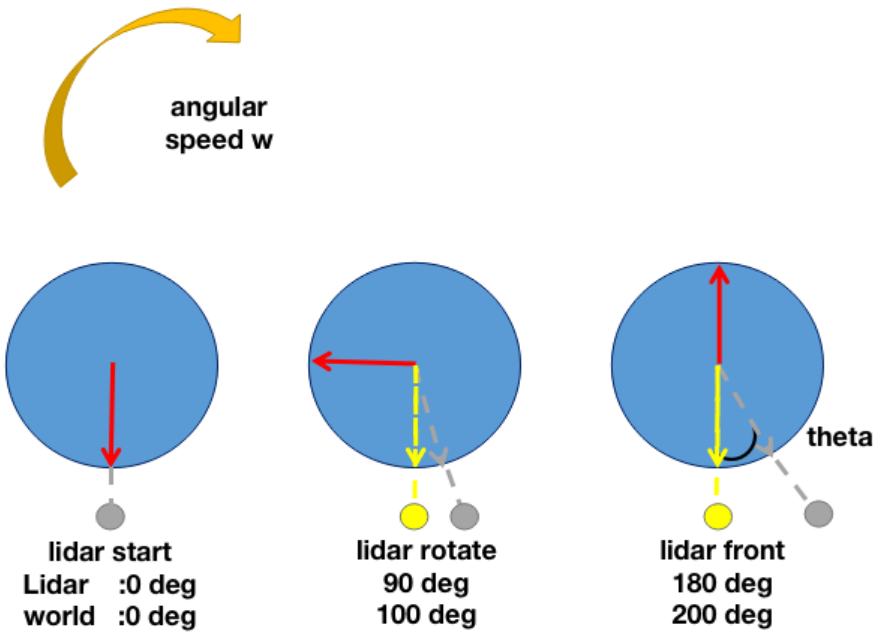
distortion by rotation and comparison of lidar and world coordinate(moving clockwise)

In the rotation case we have two situations, when the car rotates in the same direction as the lidar, namely the clockwise, The red arrow symbolify the ray direction in the lidar coordinate system whereas the yellow dash arrow means the direction in real word scenario. From the image we can conclude that the lidar rotates in the real word coordinate system 180 deg, in the lidar coordinate system it rotates only 160 deg. In this case, the red point detected by the lidar is actually in the position of the grey point. So we have to rotate the point in the lidar coordinate system an extra theta in the direction of angular speed. Below the situation can be analyzed in the same way. However, in the situation above, when the lidar rotate 180 degree and stop, it actually detects some point more than once.



distortion by rotation and comparison of lidar and world coordinate(moving counterclockwise)

This is the relationship between the actual 3D point and the rotation of the vehicle. However, if we want to project all the points detected in one period to the front camera image, we have to consider the vehicle coordinate system. When the lidar rotates towards the front, the point we detect before is no longer the position it used to be. The follow image explain this fact:



distortion by rotation when project on camera image (moving clockwise)

For instance the rotate in the same direction with lidar, when the lidar rotates 180 degree and face front, the first point we detect when scanning start is actually 200 degree away(the grey dot) but we assume the point is behind as yellow dot. In order to project the point in the front image, all the points detected before the front image have to rotate in the different direction of angular speed. In contrast, all points detected after the front image will move in the same direction of angular speed.

We set the rotation distortion a one period as:

$$rotation\ error\ =\ angular\ speed\ *\ (t_{end} - t_{start})\ 4)$$

So if we have the time difference between the detect time t_1 and front view time t_2

$$theta\ =\ -\ rotation\ error\ *\ (t_2 - t_1)/(t_{end} - t_{start})\ 5)$$

The point to project in camera coordinate have position_2:

$$R_{err} = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos((\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}\ 6)$$

So the rigid transformation T is:

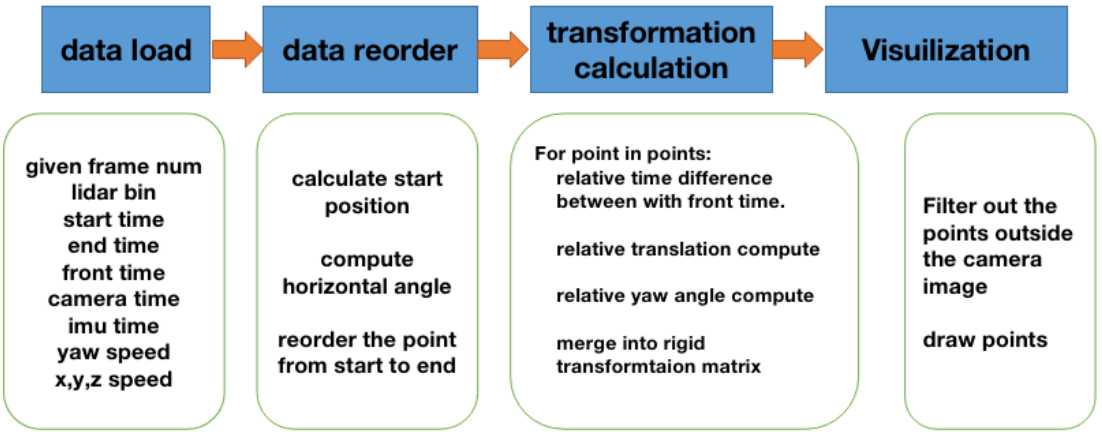
$$T\ =\ \begin{bmatrix} R_{err}, & t \\ 0\ , & 1 \end{bmatrix}\ 7)$$

In short, the rigid transformation T is a function of which can move the points position from where it is detected by the lidar to the new position when the lidar faces forward.

We add one element in the point position to build the homogeneous coordinate:

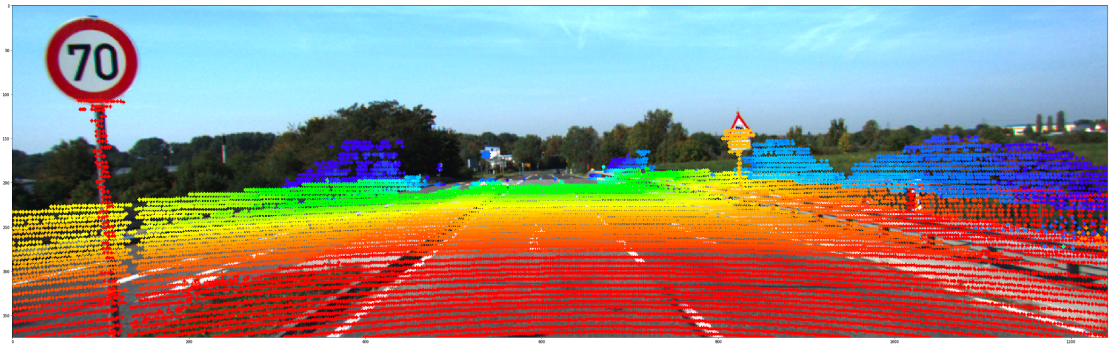
$$position'\ =\ [position,\ 1]$$
$$position\ to\ project\ =\ T\ *\ position'\ \ 8)$$

In conclusion, the whole pipeline of our work is below:



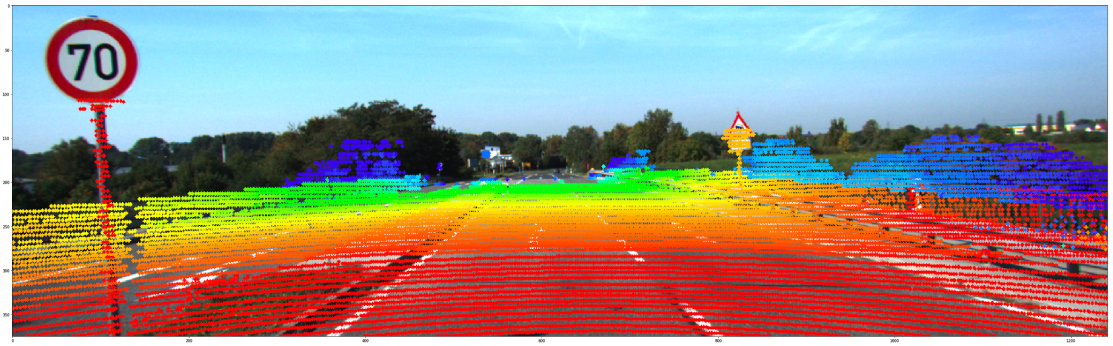
Workflow of our methods

We input the frame 0000000037 and get the image below:



frame 37 without motion distortion

If we consider the transformation from imu to lidar. We will get the follow image:



frame 37 without motion distortion and use imu_lidar transformation

Conclusion:

In this problem 4 we learn that motion distortion has a huge influence in the autonomous driving scenario and should never be ignored. We use the linear interpolate to distribute the error in rotation and translation, and finally solve this problem using rigid transformation.

However, in order to solve this question we made a lot of assumptions, all of these only hold in some cases and we have to always remember by heart that re-evaluate the assumption by looking at specific data. For example, in this problem we have a maximum 0.208 m/s difference between neighbor frames. So it will be more precise if we do not treat the speed as constant in the lidar period.

After finishing this project I learned that when autonomous reach high speed scenarios, many tasks will be much harder like the overlapping issue and motion distortion. We desperately need more computation resources as well as better sensors.

Problem 5: Bonus Questions
Problem description:

1. Eye safety: It is not safe for the human eye to look at a spinning LiDAR when it is too close. Why is the risk higher when we are closer to the sensor?

Electromagnetic waves lose energy during transmission. The closer to the emitter, the stronger the energy. Therefore, the risk is higher when people are closer to the sensor.

2. Wet roads pose challenges for both cameras and LiDAR. What are these challenges and why?

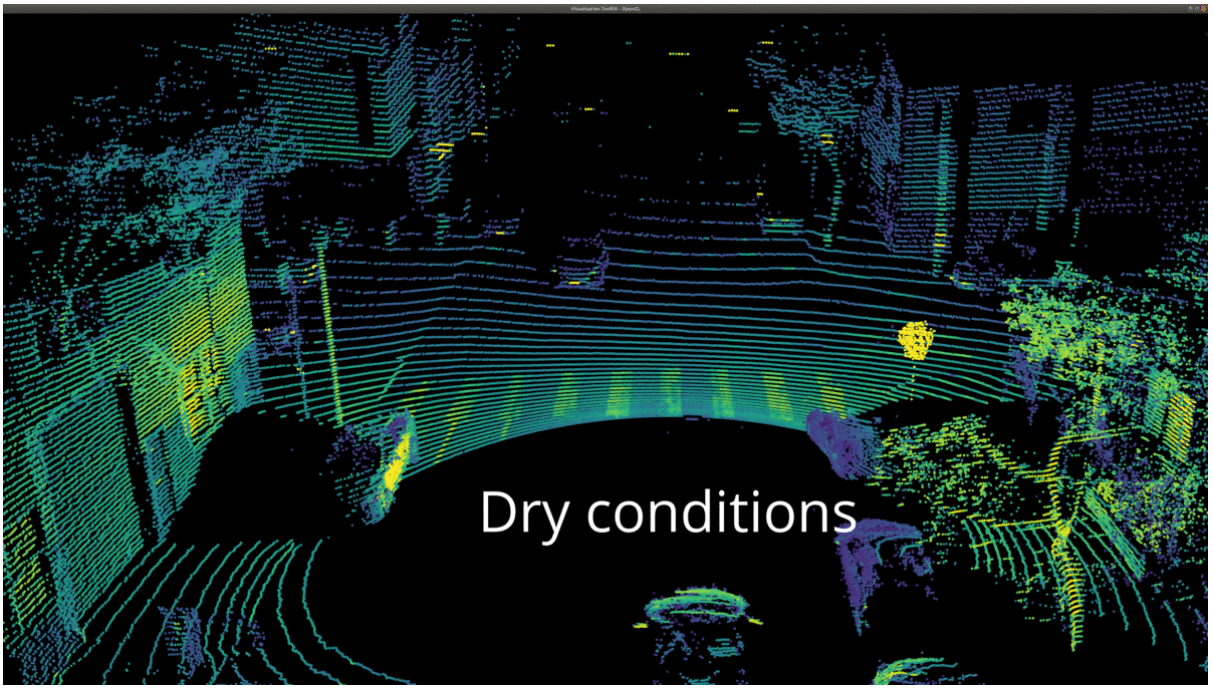
On a wet road, the water acts as a specular, mirror-like surface that reflects light in a very different pattern than in dry conditions. This reflection creates challenges for both sensors, creating confusing reflections for the camera and reducing the range of the lidar sensor. On a dry road, the rough asphalt of the road will diffuse light, sending laser light bouncing in all directions. But on a wet road, the water turns the road into an imperfect specular surface, acting like a mirror reflecting a portion of the light.

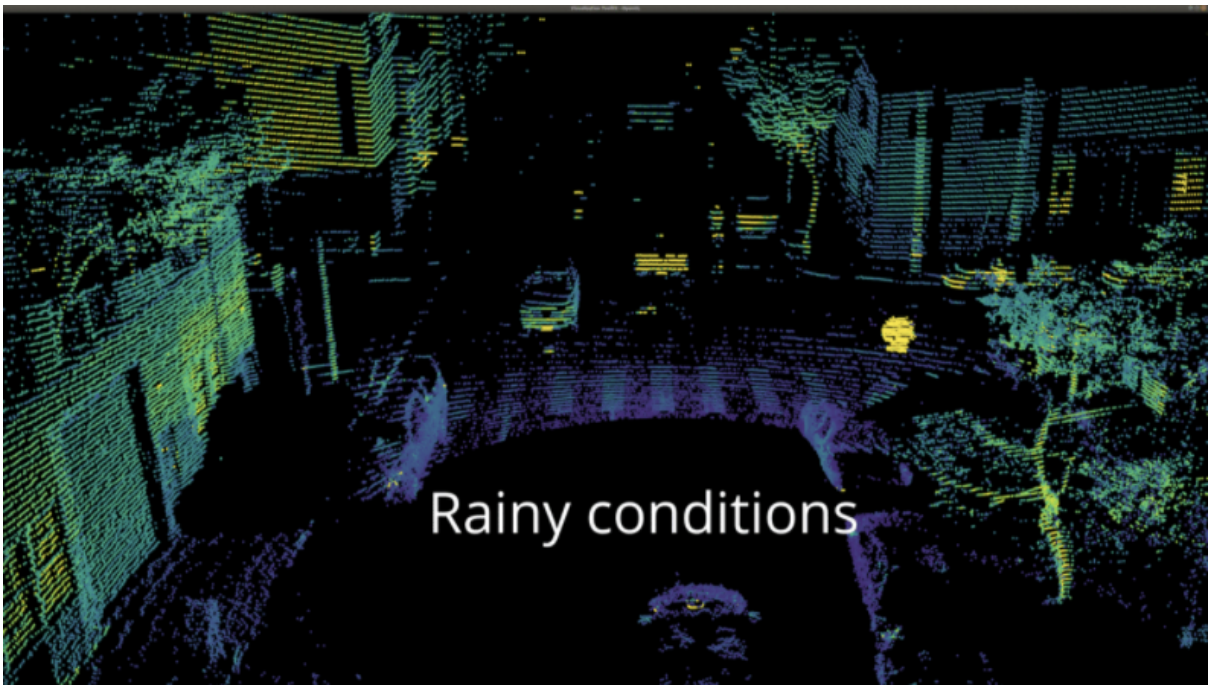
For a camera, the challenge comes from reflections that appear on the road that may create confusing mirror images of objects. In the image below, you can see the headlights of a car reflecting off the road surface. For a camera with object recognition software, this can potentially confuse the system into thinking that a second car may be present, or that the car may be closer than it actually is.

For the lidar sensor, the adverse impact was that the range of the sensor was reduced on the surface of the road. A portion of the laser light emitted by the lidar sensor reflects off the water on the road and away from the sensor. This means that the sensor is less able to see the road surface at long ranges. That said, the range of the sensor is unaffected on all other objects (cars, buildings, trees, etc.).

We can see this effect in the below images that compare conditions on a rainy day vs. a dry day.

The buildings show little change in visibility in the rain, but the range on the road surface is reduced in the rain due to the specular reflection of the road.





3. In this exercise, you have projected LiDAR points onto images. In the setup in Fig.1, the LiDAR sensor and the cameras are non-cocentered – it can never be exactly co-centered. What problem can this cause for the data projection between the two sensors (LiDAR and Cam2 for instance)? Do you think this problem will be more severe or less severe when the two sensors are more distant from each other?

It may cause projection error.
In order to project Lidar data onto the image captured by camera. We need certain matrix. This involves calibration. The system calibration can be divided into a few basic steps:

- Lever arm calibration: finding the linear offsets between each measurement unit
- Boresight calibration: determining the angular offsets between the IMU and the sensors due to mounting
- Interior calibration: finding or re-fining parameters related to sensors' interior orientation (Lidar, camera)

Installing the camera close to Lidar is recommended because it can minimize the length of the lever arm between the two sensors. Therefore, the problem will be more severe when the two sensors are more distant from each other.