Strojnícky časopis – *Journal of* MECHANICAL ENGINEERING,
VOL 75 (2025), NO 3, 47 - 52

# EFFICIENT FUSION OF DUAL LIDAR DATA STREAMS WITH MAINTAINED ORGANIZED STRUCTURE

**MILESICH Tomáš[1*], KEVICKÝ Igor[1], BUCHA Jozef[1]**

*[1]Slovak University of Technology in Bratislava, Faculty of Mechanical Engineering, Institute of Automotive Engineering and Design, Nám. slobody 17,812 31 Bratislava, Slovakia, e – mail: tomas.milesich@stuba.sk*

**Abstract:** This paper presents a method for synchronizing and merging point cloud data from two LiDAR sensors mounted on a vehicle to create a dense, machine-learning-compatible dataset. Two identical Ouster OS1 sensors (32×1024, 20 Hz) were tested in real traffic, and a synchronization algorithm was developed to align frames based on timestamp differences. The merged point clouds were organized into a 64×1024 matrix, with effective handling of missing data to ensure compatibility with neural networks. Computational performance was analyzed across processing steps, highlighting trade-offs between accuracy and speed. Results show that proper synchronization enables efficient sensor fusion, improving perception and robustness in autonomous driving.

**KEYWORDS:** LiDAR, point cloud, sensor fusion, synchronization, data processing

## 1 Introduction

With the rapid advancements in autonomous vehicle technologies and the increasing complexity and demands of tasks that autonomous vehicles must handle, it is crucial to find ways to ensure their safe and reliable operation in real-world road conditions. To achieve this, we need to understand how sensors perceive the surrounding environment, especially sensors that allow us to perceive the surroundings and determine our location within the environment. Obstacle detection is a fundamental and significant problem in autonomous driving safety. [1-3].

This article focuses on the processing and evaluation of point cloud data acquired from two separate LiDAR sensors. Accurate obstacles perception in the environment is a key problem for autopilot. LiDAR is a popular sensor in perception task as it can accurately measure the distance from surrounding environment surface. [4] Point clouds, as 3D visualizations of the surrounding environment, are a crucial element in the perception of autonomous vehicles, providing vital information for safe navigation and decision-making on the roads. Virtual simulations enable us to create realistic 3D models of road scenarios and environments, where we can test algorithms and systems of autonomous vehicles in a safe, controlled, and repeatable environment.

By inserting real-world LiDAR data into the virtual environment, we can simulate various road situations, changes in weather conditions, and other challenging scenarios that would be expensive, dangerous, or practically infeasible to execute in the physical world. In the article, we will explore methods to process and evaluate point cloud data obtained from two different LiDAR sensors in a real-world environment.

To enhance safety, reliability, and efficiency in autonomous vehicles, virtual simulations have become an indispensable tool in their development. Through these simulations, we can anticipate and address various situations that may arise on the roads, enabling autonomous vehicles to achieve a new level of autonomy and contribute to a safer future of transportation. When a vehicle is autonomously driving on the road, it not only needs to know where the drivable region is, but also basic road shape and topological relations. [5]

DOI: 10.2478/scjme-2025-0037, Print ISSN 0039-2472, On-line ISSN 2450-5471

For data collection, we used a rental vehicle as shown in Figure 1 and several sensors, while for the purpose of this article we only need to focus on two LiDARs. Data collecting took place in real traffic around our faculty.

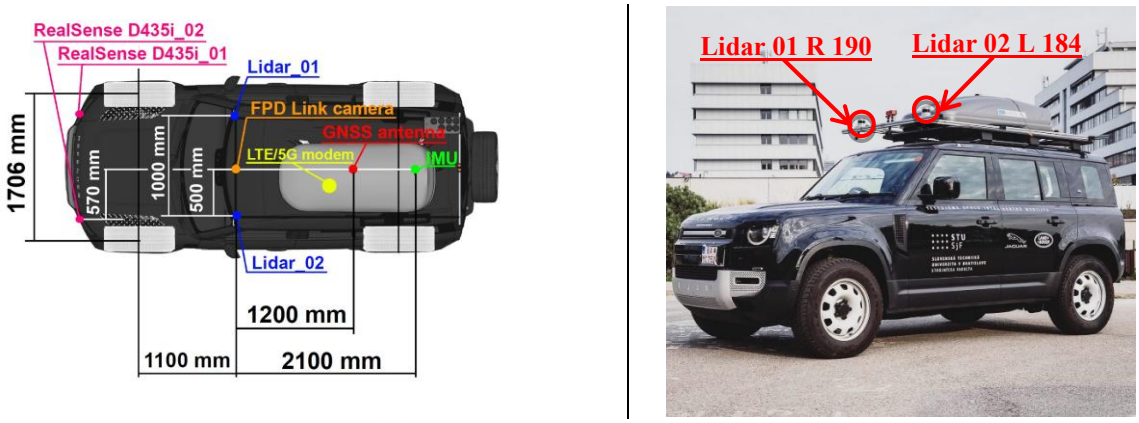


Fig. 1 Vehicle designed for data collection

We utilized two Ouster OS1 sensors. The mid-range OS1 lidar sensor is a powerful tool for autonomous vehicles. It has a 360-degree field of view, precise angular accuracy, and a high rotation rate of 20 Hz. With a horizontal resolution of 1024, it captures detailed and accurate point cloud data. It connects via UDP over gigabit Ethernet, ensuring fast data transfer. It can handle 1,310,720 points per second, making it for real-time decision-making and perception tasks in autonomous driving.

## 2 Data collecting

During playback of point clouds from two sensors, there may be time discrepancies between the point clouds. Temporal mismatch refers to the situation where the data captured by each sensor is not synchronized in time, resulting in a time shift between the two point clouds from the two sensors. As a result, one sensor is ahead of the other when playing individual point clouds.

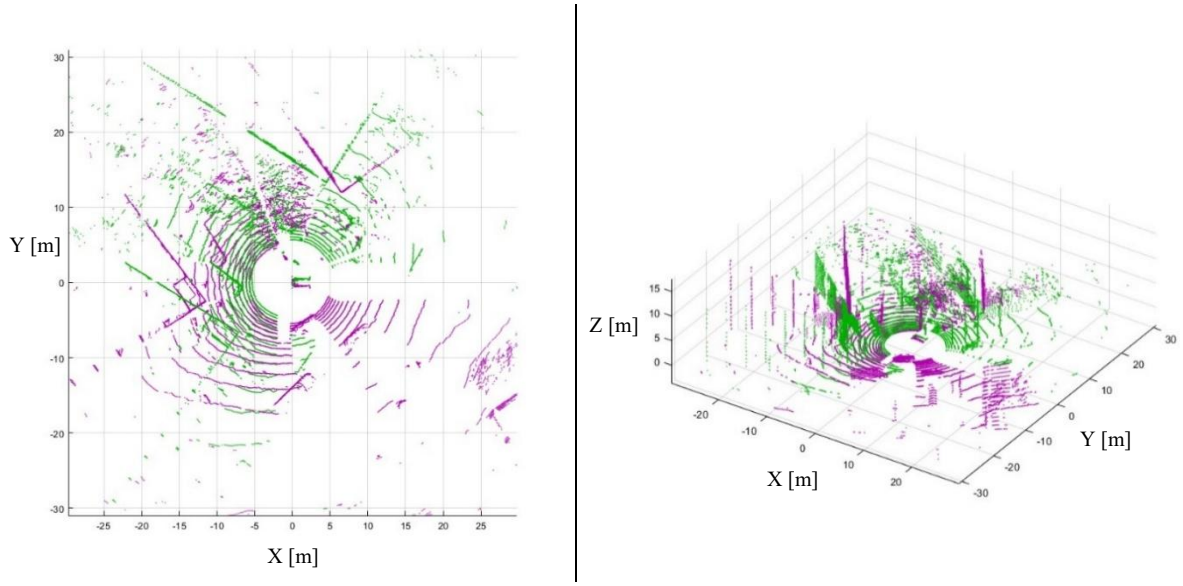


Fig. 2 Two unsynchronized separate point clouds

The problem arises mainly because individual images from each sensor are continuously loaded separately. Figure 2 shows how one cloud of points (green - Lidar 01) is ahead of the other (purple - Lidar 02) in time.

Acquiring data at a frequency of 20Hz (which was the frequency at which we were acquiring data from the LiDAR), the spacing between frame recordings from the stream may not be precisely 0.05 seconds. This can be due to various factors affecting the data acquisition process. The data transmission and storage processes can also contribute to the observed discrepancies. Network latency, data buffering, or storage delays can introduce slight time differences between frames as they are processed, transmitted, or saved. For utilizing a denser point cloud obtained from two sensors, it is crucial to merge these two point clouds into one. To achieve this, it is necessary to synchronize them first. In each point cloud within the stream, there is information about the stream's start time and a timestamp that indicates the current time increment. Two unsynchronized streams from two LiDARs and their differences in each frame can be seen in Figure 3. I can look at the individual streams as they follow each other and compare them between two LiDARs, from that I can get the time difference in each read frame. If I know the current value of the time difference for a particular frame within the stream, I can compare it with the maximum theoretical time difference corresponding to the data acquisition frequency from the sensor. In our case, that is 20Hz, or every 0.05 seconds. If the time difference between the frames from the left and right sensors is smaller than this given time, I can read the point cloud from the frame. If either the left or right sensor is lagging in time, I will add the value of the current time difference to the lagging running stream. This way, I avoid accumulating reading errors based on different current times.

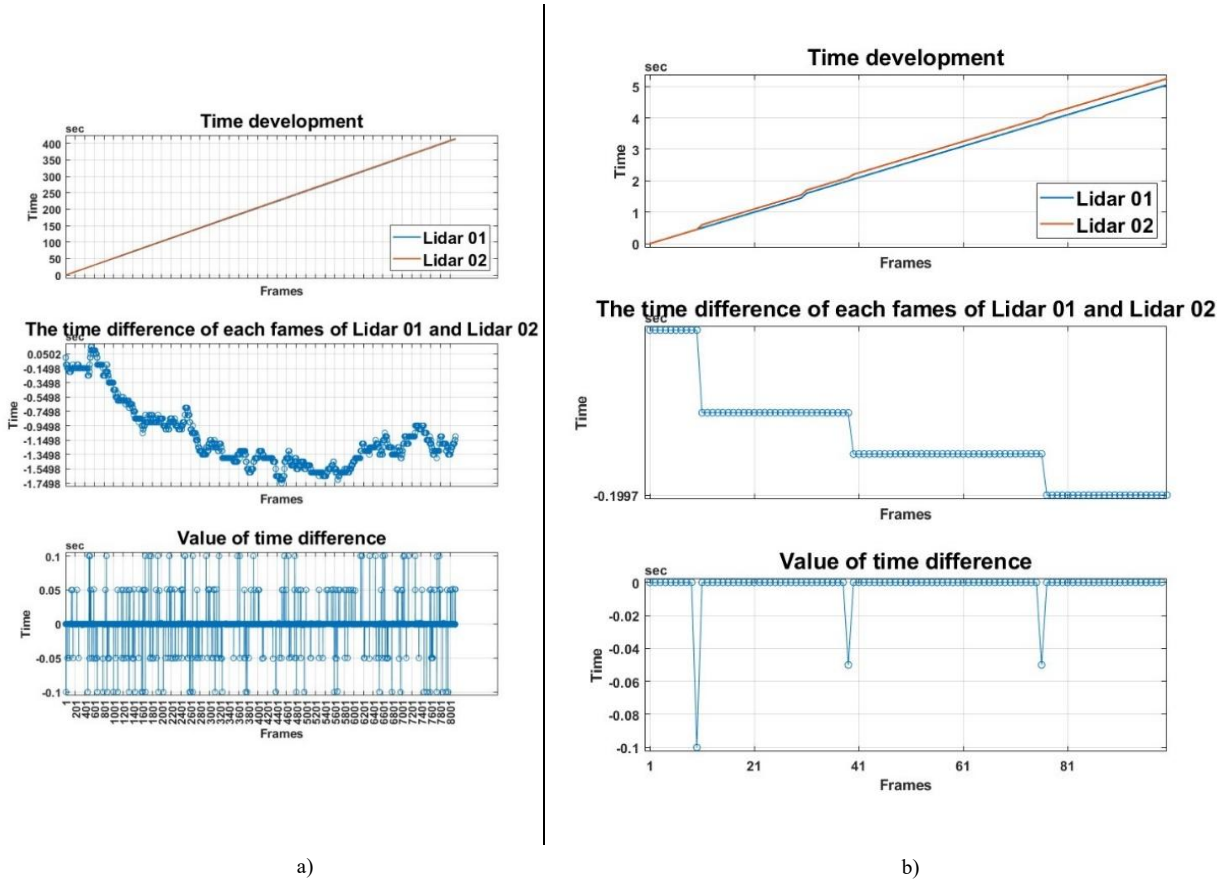


Fig. 3 Visualization of the time development during the playback of two LiDARs streams: a) full record, b) record detail

For a better understanding of the synchronization, I am attaching a script for aligning the time development during the stream.

```
CurrentTime_difference = CurrentTime_Lidar_02 - CurrentTime_Lidar_01
  if abs(CurrentTime_ difference) < milliseconds(50)
    ptCloud_Lidar_02 = read_Frame(Lidar_02)
    ptCloud_Lidar_01 = read_Frame(Lidar_01)
  else
    if CurrentTime_diff < milliseconds(0)
        Lidar_02.CurrentTime = Lidar_02.CurrentTime + abs(CurrentTime_difference)
        ptCloud_Lidar_02 = readFrame(Lidar_02)
        ptCloud_Lidar_01 = readFrame(Lidar_01)
    else
        Lidar_01.CurrentTime = Lidar_01.CurrentTime + CurrentTime_difference
        ptCloud_Lidar_02 = readFrame(Lidar_02)
        ptCloud_Lidar_01 = readFrame(Lidar_01)
    end
  end
```

Visualization of two point clouds that are aligned using synchronization is in Figure 4, it is the same frame as in Figure 2. Synchronization enabled the exact overlapping of the point clouds, which guarantees a continuous and complete view of the scene. Aligned point clouds can be used to recognize objects, or environment mapping.

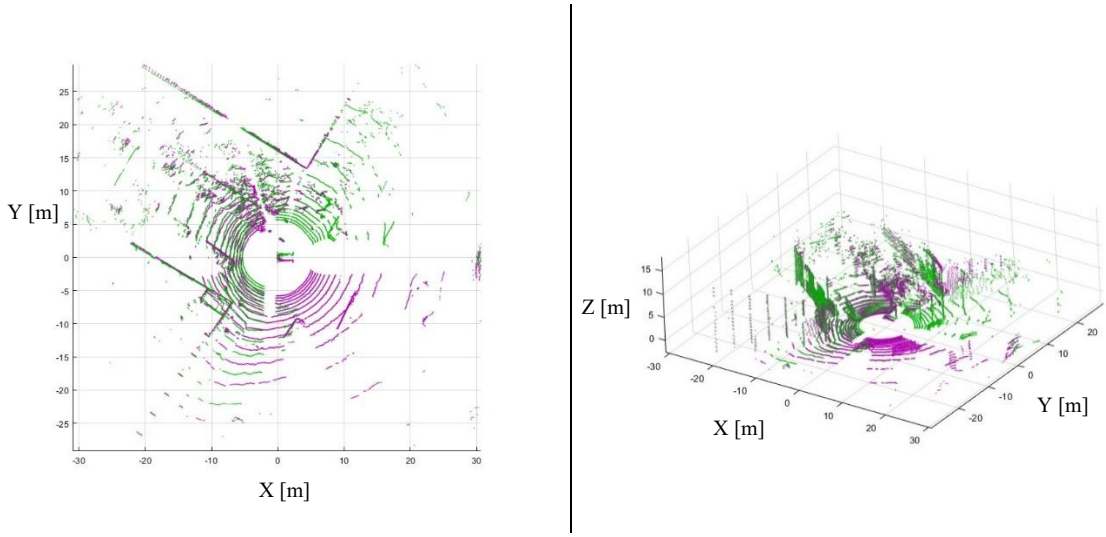


Fig. 4 Two synchronized separate point clouds

## 3 Creating a pointcloud from two separate frames

The positions of points in 3D coordinate space can be organized or unorganized. In the case of unorganized point clouds, the point cloud represents a matrix M-x-3, where M is the number of points, and the 3 columns represent the X, Y, and Z coordinates. For organized point clouds, the point cloud represents an M-by-N-x-3 array, where M is the number of horizontal layers, and N is the number of points in one horizontal layer. This M-x-N array contains information about the position coordinates X, Y, Z for each point. If we want to use the point cloud for segmentation of objects, or for object detection and boxing, it is necessary to keep the organized structure of each point cloud from frames from the entire stream. A trained neural network requires the input layer to contain information about the X, Y, Z coordinates, distances of individual points from the sensor, and intensity values.

We have two identical LiDARs at our disposal, which have the same resolution of 32x1024, which means that the combined pointcloud will have a resolution of 64x1024

points. First, we need to create an M-x-N-3 array with X, Y, Z values. For faster program execution, it is important to first create an array with NaN values and enter values into it according to Figure 5.

| | 1 | 2 | 3 | | N |
|---|---|---|---|---|---|
| 1 | *Lidar_01.Location 1x1* | *Lidar_01.Location 1x2* | *Lidar_01.Location 1x3* | . . . | *Lidar_01.Location 1xN* |
| 2 | *Lidar_02.Location 1x1* | *Lidar_02.Location 1x2* | *Lidar_02.Location 1x3* | . . . | *Lidar_02.Location 1xN* |
| 3 | *Lidar_01.Location 2x1* | *Lidar_01.Location 2x2* | *Lidar_01.Location 2x3* | . . . | *Lidar_01.Location 2xN* |
| 4 | *Lidar_02.Location 2x1* | *Lidar_02.Location 2x2* | *Lidar_02.Location 2x3* | . . . | *Lidar_02.Location 2xN* |
| 5 | *Lidar_01.Location 3x1* | *Lidar_01.Location 3x2* | *Lidar_01.Location 3x3* | . . . | *Lidar_01.Location 3xN* |
| 6 | *Lidar_02.Location 3x1* | *Lidar_02.Location 3x2* | *Lidar_02.Location 3x3* | . . . | *Lidar_02.Location 3xN* |
| 7 | *Lidar_01.Location 4x1* | *Lidar_01.Location 4x2* | *Lidar_01.Location 4x3* | . . . | *Lidar_01.Location 4xN* |
| | . . . | . . . | . . . | . . . | . . . |
| M-1 | *Lidar_01.Location Mx1* | *Lidar_01.Location Mx2* | *Lidar_01.Location Mx3* | . . . | *Lidar_01.Location MxN* |
| M | *Lidar_02.Location Mx1* | *Lidar_02.Location Mx2* | *Lidar_02.Location Mx3* | . . . | *Lidar_02.Location MxN* |

Fig. 5 Data array structure with X, Y, Z coordinate values

Naturally, every real point cloud contains NaN data in the data field. Replacing NaN values with the average values from neighbouring columns and rows of a matrix can be a quick way to fill in data in a point cloud matrix. However, from a time perspective, this method has its limitations and may not be suitable in certain situations. Computing the average from neighbouring columns and rows in large matrices is a computationally intensive process that significantly slows down operations with the given matrix. This can lead to a substantial increase in computational time. If the matrix is relatively sparse and contains many NaN values, using averages may result in decreased accuracy. In some applications, replacing NaN values with averages from neighbouring columns and rows may be sufficient and adequately efficient. However, before employing this method, it is crucial to carefully consider the size and structure of the matrix, as well as the specific needs and objectives of its use.

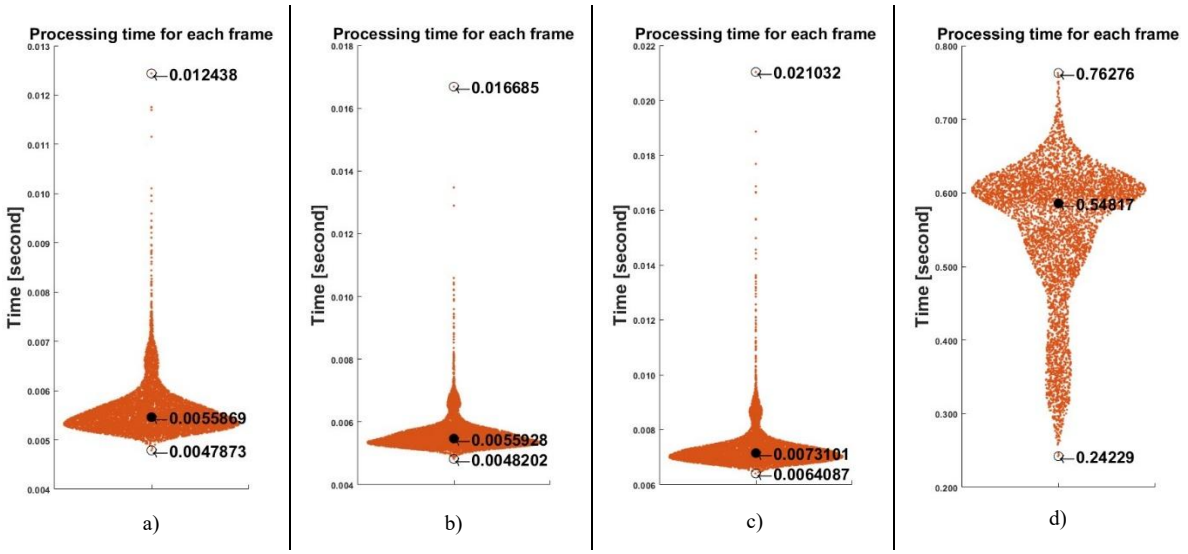


Fig. 6 The loading time of each frame

The display of the loading duration of each frame can be seen in Figure 6, where a) it represents the loading of individual frames from the stream, b) the loading of individual frames from the stream together with time synchronization, c) the addition of the merge of two pointclouds into one, d) the addition of replacing NaN with averaged neighbors in the matrix. We can see that the large matrix processing superstructure has a significant impact on computing performance.

Table 1. Value of loading time of frame

| | Mean duration [s] | Max duration [s] | Min duration [s] |
|---|---|---|---|
| **Simple read frame of two poincloud** | 0.0056 | 0.0124 | 0.0048 |
| **Simple read frame of two poincloud with time synchronization** | 0.0056 | 0.0167 | 0.0048 |
| **Read frame of two poincloud with time synchronization and merge** | 0.0073 | 0.021 | 0.0064 |
| **Read frame of two poincloud with time synchronization, merge and replace NaN** | 0.5482 | 0.7628 | 0.2423 |

## CONCLUSION

In this work, we addressed the issue of processing data from two LiDARs and solving the issue of the speed of this processing on a computer. With the development of autonomous vehicles and similar applications, the processing of LiDAR point clouds becomes a critical factor for successful recording and interpretation of the surrounding environment. Our method of processing data from two LiDARs will ensure the maintenance of information in individual frames for the possibility of using the resulting point cloud for machine learning applications. By this, we have contributed to improving the computer's ability to process point clouds from multiple sensors and achieve more reliable and faster decision-making in various real scenarios. Based on this article, we see a perspective for further improvements and optimization of the method of combining two streams from LiDAR in order to achieve even better processing speed and to extend its use to other applications in the field of autonomous vehicles. Overall, we consider this research to be beneficial and promising for further development in this area.


### Acknowledgement

This research was supported by the Slovak Research and Development Agency under Contract No. APVV: SK-SRB-23-0024, APVV-24-0526, APVV-23-0650.


## REFERENCES


[1] Wei, Y., Yang, J., Gong, C. et al. "Obstacle Detection by Fusing Point Clouds and Monocular Image", Neural Process Lett 49, pp. 1007 – 1019, **2019**. DOI: https: 10.1007

[2] Lehoczký, P., Janeba, M., Galinski, M., Šoltés L. "Testing the Readiness of Slovak Road Infrastructure for the Deployment of Intelligent Transportation", Strojnícky časopis – Journal of Mechanical Engineering 72 (2), pp. 103 – 112, **2022**. DOI: 10.2478/scjme-2022-0020

[3] Nemčík, J., Šoltés, L., Galinski M., Kotuliak I. "Edge to Cloud Task Offloading Optimization in Internet of Vehicles Networks", Strojnícky časopis – Journal of Mechanical Engineering 75(1), pp. 123 – 134, **2022**. DOI: 10.2478/scjme-2025-0013

[4] Li, H., Zhang, S. "An Obstacles Detection Method for Sparse PointCloud Based on Hybrid Model and Multi-dimensional Segmentation", In: Zhang, Z. (eds) 2021 6th International Conference on Intelligent Transportation Engineering (ICITE 2021). ICITE 2021. Lecture Notes in Electrical Engineering 901, Springer, Singapore, **2022**. DOI: 10.1007/978-981-19-2259-6_106

[5] Zhu, Q., Chen, L., Li, Q., Li, M., Nüchter, A., Wang, J. "3D LIDAR point cloud based intersection recognition for autonomous driving", 2012 IEEE Intelligent Vehicles Symposium, Madrid, Spain, pp. 456 – 461, **2012**. DOI: 10.1109/IVS.2012.6232219