HydraView: A Synchronized 360°-View of Multiple Sensors for Autonomous Vehicles

Luodai Yang
College of Engineering and Technology
Eastern Michigan University
lyang15@emich.edu

Qian Jia
College of Arts and Science
Eastern Michigan University
qjia@emich.edu

Ruijun Wang
Dept.of Computer Science
Wayne State University
ruijun@wayne.edu

Jie Cao
School of Information Security and Applied Computing
Eastern Michigan University
jcao3@emich.edu

Weisong Shi
Dept.of Computer Science
Wayne State University
weisong@wayne.edu

***Abstract*—Today's autonomous vehicles will deploy multiple sensors to achieve safe and reliable navigation and precise perception of the environment. Although multiple sensors can be advantageous in terms of providing a robust and complete description of the surrounding area, the synchronization of multi-sensors in real-time processing is extremely important. When data is synchronized, primary functional systems such as localization, perception, planning, and control, will all benefit. In this paper, we proposed a synchronized data illustration and collection method to assist the data processing applications for autonomous driving. Our proposed solution among different sensors can be directly deployed on autonomous vehicles for data integration and environment analysis to support the driving model construction. The experimental results validate that our proposed method can present a 360° synchronized view while providing the capability of real-time scanning with up to 80% reduced latency.**

***Index Terms*—autonomous vehicles, multiple sensor, synchronization, camera, LiDAR, transformation matrix**

I. Introduction

Autonomous driving is a complex task that relies on precise localization, navigation, planning, and system control. In order to maximize the safety and reliability of the vehicle, multiple sensors have been installed for collecting data and performing computing tasks accordingly. This process requires the utilization of several types of sensors for different computing purposes. Since many sensors are applied for data acquisition, data synchronization, and data fusion are extremely important. In fact, cameras and 3D LiDARs are widely used sensors for supporting autonomous driving [1], [2], [3]. When incorporating data of various sensors into a single description: the field of view is enlarged, the precision of the estimation is increased, and consequently, the system is economically efficient as different applications share a set of sensors. The most challenging part of such a system is data association, which requires a synchronization of the sensor data and the actual corresponding object state. The main advantage of synchronization is that it provides a completed view for the incoming sources to support the ADAS (Advanced Driver Assistance Systems), In-vehicle infotainment, real-time diagnostics, as well as some third-party applications. The synchronized data will improves the performance and accuracy for all the above applications because there is no gap or overlap for the different data sources to create a reliable and consistent middle-ware for the top-level applications. Some researchers conducted an investigation into exploring hardware-triggering synchronization, network synchronization, clock synchronization, and software synchronization among different type of sensors [4]–[6]. For hardware synchronization, camera supported-hardware synchronization is rarely deployed in applications due to their high cost and high data transfer requirement among multi-cameras. In addition, Network synchronization strictly relies on network reliability and the design of algorithms. In the event of network failure happens, the catastrophe caused by network delay is tremendously dangerous for time-critical autonomous driving applications. Moreover, the current solution for software synchronization using the time stamp method can solve the asynchronous problem. However the processing time overhead makes the solution unsatisfiable for the real-time performance requirement. Thus, there is a need to generate an effective synchronization method for multiple sensors on autonomous vehicles with extremely low latency. In this paper, we proposed a synchronized data illustration and collection method of multiple sensors for autonomous vehicles in real-time environment. This solution can provide an efficient and accurate platform for applications running on top of these data. Our significant contributions of this work can be summarized as follows:

- Propose an effective algorithm to reduce computing time for multiple sensors data fusion in real-time environment.
- Propose a robust synchronization algorithm to locate the corresponding image data of each camera and LiDAR.
- Implement the proposed method on a real mobile platform with six web cameras and one Velodyne 3D LiDAR to collect data and test the real-time performances.
- Verify our design using two different sensors and then collect a set of data on our test-bed to prove the synchronization result.

- Conduct intensive experiments on a HydraOne [7] platform to test the efficiency and effectiveness of our proposed solution.

The rest of the paper is organized in the following structure. In Section II, we analyze the previous research works regarding multi-sensors synchronization and calibration methods. Section III focuses on the system design and implementation of our proposed methods. We present the experimental setup and evaluation results with a presentation of the future works in Section V and conclude our work in Section VI.

II. Related Works

To precisely perceive surrounding environments, multi-sensors are needed to support autonomous driving. Sensors frequently used in autonomous driving include LiDAR, cameras, Radar, GPS, and inertial measurement unit (IMU). Many datasets for autonomous driving have been released on different mobile platforms with various combinations of these sensors [6], [8]–[12]. As we can see from these datasets, 3D LiDAR and cameras are commonly adopted for perceiving surrounding environment in autonomous driving. LiDAR can provides higher accuracy and reduce the computation cost in comparison to cameras. [13]. Therefore, sensor fusion has been applied in many aspects regarding autonomous driving [4], [5], [14]–[18]. To ensure high accuracy of the estimation of sensor fusion, Kaempchen *et al*. pointed out that the critical part of sensor fusion is synchronization among multiple sensors [5]. Since different sensors have different work frequency aligning multiple data in corresponding order becomes a very critical task. Most approaches found for synchronization among sensors are hardware-triggering synchronization, network synchronization, clock synchronization, software synchronization. For hardware synchronization, the approaches to implementation are vary. External synchronization control unit or internal camera synchronization has been adopted for synchronization [8], [19]. And another way is to use reed contact to trigger cameras capturing pictures [6]. However, cameras supporting hardware synchronization are rarely employed in applications due to their high cost and camera connection buses can not satisfy the requirement of high data transfer among multi-cameras. For network synchronization, Precision Time Protocol (PTP) and Network Time Protocol(NTP) are proposed to perform synchronization in distributed systems [20], [21]. Network synchronization highly depends on reliable network and design of the algorithm. In the event of network failure, the catastrophe caused by network delay is tremendously dangerous for time-critical autonomous driving. For clock synchronization, proposed algorithms for synchronization among clocks enable that one clock can correct it's time to match with another clock [22], [23]. However, these algorithms are not applicable to sensors utilized in autonomous driving. For software synchronization, Wu *et al*. proposed a soft time synchronization framework for synchronizing multi-sensors which was implemented under ROS [24]. Shin *et al*. also used the seemingly same method for synchronization between LiDAR and camera [25].

III. Design and Implementation

In this section, we will present the overall design of our HydraView system and its implementation, limitation, and challenges.

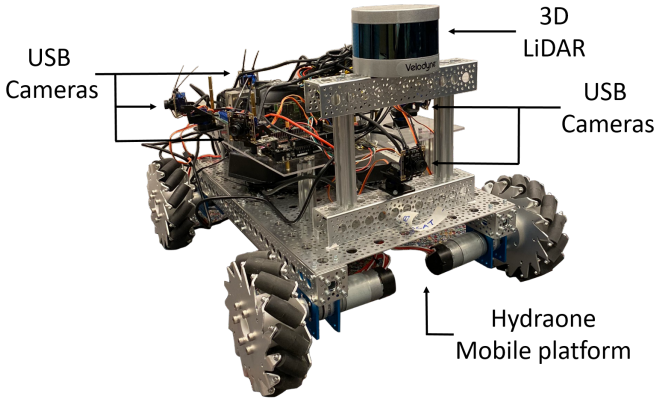


Fig. 1: Overview of HydraView with HydraOne Mobile platform.

A. HydraView System Design

In this paper, we propose HydraView, a Synchronized 360°-View of Multiple Sensors for Autonomous Vehicles. As shown in Figure 1, HydraView includes one VLP-16 LiDAR and six USB cameras. The mobile platform and process running environment are based on HydraOne [26].

HydraView includes five main processing parts as shown in Figure 2. In the processing of HydraView, the first part is ROI (region of interest), which is used to find the corresponding relationship between spatial-point cloud data and the different views that are taken from each camera. Additionally, to address the part of data synchronization, HydraView uses LiDAR data as the main thread. Once HydraView gets one frame of LiDAR data and the timestamp of finishing one cycle of scanning, TOI (time of interest) that converts from ROI will be utilized to find the period of time that can represent the moment of scanning spatial area that correspond to the view of each camera. Then, HydraView can determines which image that have been saved can represents the moment based on its timestamp. After that, ROI is used to re-project 2D pixel view into 3D space as a filter to extract the point data that can represent the view that each camera can see. Following distortion corrections in image data, the transformation matrix received from the calibration between LiDAR and cameras are used to project 3D point cloud into 2D pixel space. Lastly, the fusion-processing part will create an N-layer matrix as a container to store integrated data. The first 3 layers store traditional RGB image data while the rest of the layers store distance, reflect intensity, etc.

B. ROI and TOI

In this paper, we propose a location layout between LiDAR and six cameras, as shown in Figure 1. The LiDAR has a 360° view of the surrounding environment, but one camera only can see a part of it , as represented by the black sector. In this location layout, the location relationship between LiDAR and each camera is fixed. The corresponding point data represents

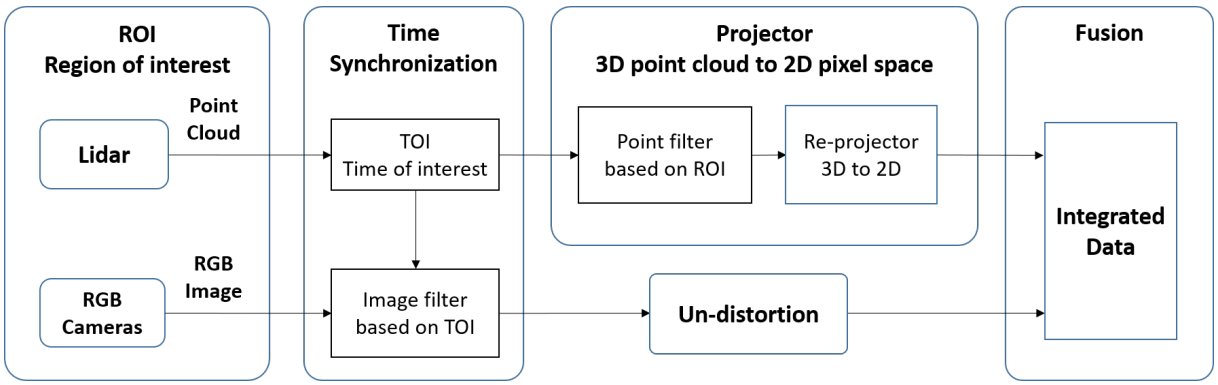


Fig. 2: FlowChart of the data syncrhinzation in HydraView.

different camera views in time space and spatial space is also fixed.

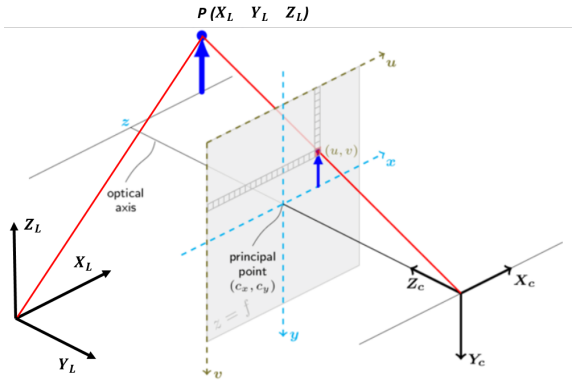


Fig. 3: Pinhole camera model.

1) ROI: When it comes to addressing ROI, the goal is to determine the fixed scanning range to represent the image view for each camera. It can be used as a filter to find the corresponding point cloud data for one camera view. Figure 3 illustrates the basic pinhole camera model. Here, P(X_L, Y_L, Z_L) represent object points in the LiDAR coordination system and [X_C Y_C Z_C] represent the camera coordination system. Plane [x y] is a camera image coordination system and (u, v) is the corresponding location in pixel space. Based on this model, the corresponding relationship between LiDAR and camera coordination systems can be described using the basic transformation matrix, defined by Formula 1:

$$[X_c \quad Y_c \quad Z_c] = [R \mid T][X_L \quad Y_L \quad Z_L \quad 1]^T \tag{1}$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2}$$

$$T = [t_x \quad t_y \quad t_z]^T \tag{3}$$

In Formula 1, (X_C, Y_C, Z_C) denote the location of the point in the camera coordination system. Moreover, [R | T] is the extrinsic matrix that includes rotation and translation matrix. r_{ij} are nine rotation parameters. t_x, t_y, and t_z indicate the translation in a different direction along x, y, and z axis. (X_L, Y_L, Z_L, 1) is the homogeneous coordinate in LiDAR coordination system. Then, Formula 4 can be used to convert camera coordinates into pixel space: Where K denotes the intrinsic matrix of the camera.

$$Z_C[u \quad v \quad 1]^T = K[X_C \quad Y_C \quad Z_C] \tag{4}$$

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

f_x and f_y represent the focal lengths while c_x and c_y are the principal points that are usually located at the center of an image. Z_C shows the distance from the center of camera coordination to object point, and in this paper, Z_C has been assumed as 10 meters. In this case, Formula 6 can be used to find the corresponding range from the 2D pixel space to a 3D LiDAR coordination system, and Formula 7 can be utilized to determine whether one 3D point belongs to this 2D pixel space based on its angle in the LiDAR coordination system.

$$[X_L \quad Y_L \quad Z_L \quad 1] = Z_C * K^{-1}[R \mid T]^{-1}[u \quad v \quad 1] \tag{6}$$

$$\begin{cases} Angle_{max} = max(arctan2(Y_L, X_L)) \\ Angle_{min} = min(arctan2(Y_L, X_L)) \end{cases} \tag{7}$$

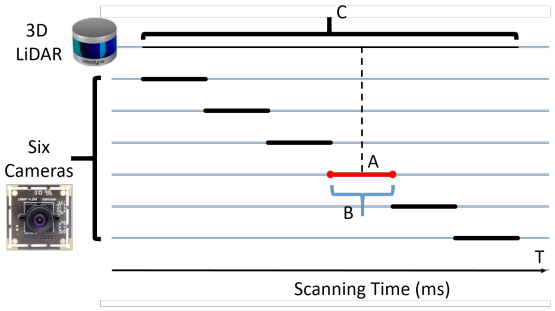


Fig. 4: Corresponding time period for each camera in one scanning cycle.

2) TOI: TOI is aimed at finding the time in one scanning cycle that can represents the image view for each camera. The rotation speed for LiDAR is 600RPM, and it needs 55.296s for each 0.2° view, which means the LiDAR needs 100ms to scan one 360° view frame. As seen in Figure 4, A represents the moment when the camera takes the image, B represents the time that LiDAR takes to scan the corresponding area for the view of the camera, and C represents the time that LiDAR needs to scan one frame of the 360° view. Therefore, TOI can be calculated by (one cycle time * ROI / 360° view) for each camera.

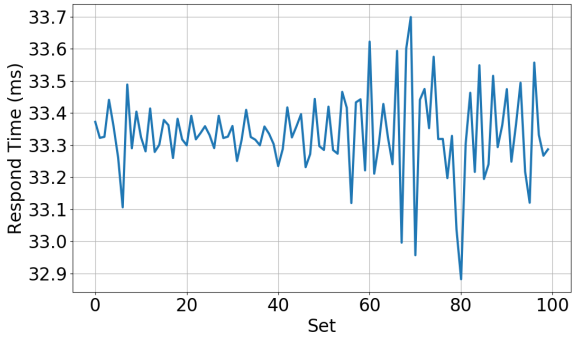
C. Synchronization

In this paper, we propose a synchronization algorithm to find the corresponding image data of each camera that is synchronized with LiDAR. LiDAR data has been defined as the main thread, and Algorithm1 has been applied to each camera:

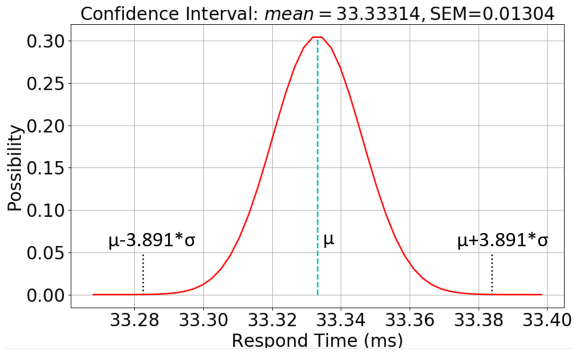
Algorithm 1 Synchronization between LiDAR and one camera.

Require: LiDAR message; Image message; Scanning time T_s; TOI; Threshold;
Ensure: Synchronized LiDAR and image data;
1: **for** each $Timestamp \in LiDARmessage$ **do**
2: Time period of scanning the area camera can see = Timestamp of LiDAR message – Scanning time + TOI;
3: **while** $|Timestampofimage–Timestampperiod| > Threshold$ **do**
Update image message;
4: **end while**
5: **return** image message
6: **end for**=0

Once the HydraView receives one LiDAR message, it will extract the timestamp from the ROS PointCloud2 message that represents the completion time of one scanning cycle. Then, TOI will be initiated for a different camera to find the period of time that can represent the image view. This period of time can be compared with the image timestamp from the last image stored in the message queue. If the threshold is less than the pre-set time gap, then the stored image will be taken as the corresponding image. Otherwise, the next image that will be taken after a period of time that is less than the pre-set threshold will be used as the corresponding image. Here, the time gap is used as the threshold, which is determined by the working frequency of the camera. In this paper, all six cameras have a similar work frequency around 30Hz. Figure 5 shows statistic results and confidence levels of response time for cameras based on the distribution of the sample. To achieve 99.99% confidence level, we decided to use $\mu + 3.891 * \sigma$ to represent the upper limit for responding time. In the case of this study, μ represents mean value and σ represents SEM (standard error of the mean). Half of the upper limit was used as the threshold. Therefore, either the image that is already stored in the message queue or the next image that the camera will take is the closest image that can represent the period of time in scanning.



(a)



(b)

Fig. 5: (a) Distribution of responding time of camera; (b) Confidence intervals of responding time of camera

D. Transformation Matrix

1) Point filter: In this section, HydraView uses a transformation matrix to project the pixel plane into the LiDAR coordination system in order to find ROI for each camera. Then, for each point data, x and y coordinate in the LiDAR coordination system are used to calculate the angle in LiDAR space, as Formula 8 described.

$$D = arctan2(y, x) \tag{8}$$

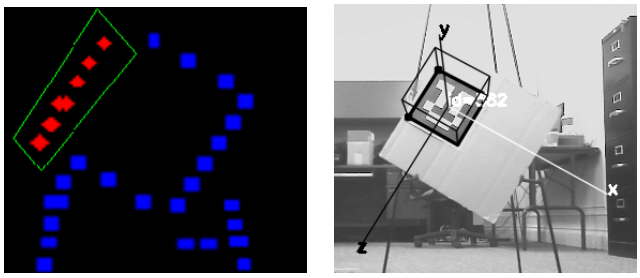
Based on the ROI of each camera, D is used as the limitation to determine which camera can see this point.

2) Project 3D spatial point into 2D pixel space: In this part, HydraView uses a transformation matrix, including rotation and translation matrix, to project 3D points cloud in the LiDAR coordination system into 2D pixel space by Formula 9.

$$Z_C[u \quad v \quad 1]^T = K[R \mid T][X_L \quad Y_L \quad Z_L \quad 1]^T \tag{9}$$

Ankit [27] proposed a way to calculate the rotation and translation parameters, R and T, for a single camera. In Figure 6 (a), the blue points that extracted from points cloud data indicates the edges of this rectangle object while the red points represents one edge after manually draw a green circle to chose them. Once four edges of the marker from LiDAR data are captured, calculations can be performed to determine the coordinates of four corner points in the LiDAR

coordination system. In camera view (b), ArUco marker [28] [29] can offer the position transformation between marker and camera. After manually measuring the actual dimension of the marker, the result can be used to calculate the 3D coordinates of four corners in the camera coordination system. [R | T] between two set 3D corresponding points can be estimated by the Iterative Closest Point (ICP) algorithm [30]. Formula 10 describes the way ICP is used to minimize the error in 3D space.



(a) (b)

Fig. 6: (a) 3D points represent the edge of marker captured by LiDAR; (b) The position of ArUco marker captured by camera

$$\frac{arg\ min}{R \in SO(3), t \in R^3} ||(RP + t) - Q||^2 \tag{10}$$

P represents the points captured by the LiDAR and Q represents the points captured by the camera. This method needs an initial translation matrix, which is determined by manually-measured spatial rotation and translation. Spatial rotation can be divided into three different parts: rotation around x, y, and z axis, represented by the roll, pitch, and yaw. The following formula converts these three angles into a single rotation matrix:

$$R_X(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\beta & -sin\beta \\ 0 & sin\beta & cos\beta \end{bmatrix} \tag{11}$$

$$R_y(\alpha) = \begin{bmatrix} cos\alpha & 0 & sin\alpha \\ 0 & 1 & 0 \\ -sin\alpha & 0 & cos\alpha \end{bmatrix} \tag{12}$$

$$R_z(\gamma) = \begin{bmatrix} cos\gamma & -sin\gamma & 0 \\ sin\gamma & con\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{13}$$

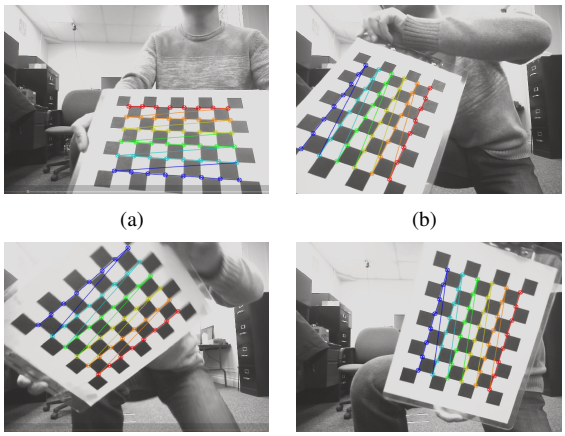
$$R = R_Z(\alpha)R_Y(\beta)R_X(\gamma) \tag{14}$$

3) Un-distortion: The image data captured by camera have distortion caused by camera lens. There are two types of distortion, radial distortion and tangential distortion. Radial distortion is called the barrel effect or fish eye view, caused by the physic shape of the lens. There are two types of radial distortion: barrel distortion and pincushion distortion. Tangential distortion is caused by the image view plane which is not parallel to the camera lens. The distortion effect can be represented by Formula [31]:

$$D_{radial} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} x_p \\ y_p \end{bmatrix} \tag{15}$$

$$D_{tangential} = \begin{bmatrix} 2p_1 x_p y_p + p_2(r^2 + 2x_p^2) \\ p_1(r^2 + 2y_p^2) + 2p_2 x_p y_p \end{bmatrix} \tag{16}$$

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = D_{radial} + D_{tangential} \tag{17}$$



(a) (b) (c) (d)

Fig. 7: Checkerboard on camera view in different orientation.

For a point (x_p, y_p) in the distorted image, the corresponding point in an undistorted image is (x_c, y_c). Use [k_1 k_2 p_1 p_2 k_3] to describe this distortion model, while r can represent the distance from a point to the center of radial distortion. Thus this distortion model can be represented by five distortion coefficients [k_1 k_2 p_1 p_2 k_3]. Tsai [32] and Zhang [33] proposed the most common resolution for this problem. Here, an 8x6 white and black checkerboard has been used to generate multiple sets of coordinate data, as Figure 7 shows. For each image, the relationship between 8x6 corner points in pixels space and camera coordination system can be detected after manually measuring the dimension of the physical checkerboard. Then, the following formula is used to calculate the focus of lens f and the center (u_0, v_0) of image. Here, (u, v) represent pixel coordinates and (x_m, y_m, z_m) represent the coordinates in camera space.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{d_x} & 0 & u_o \\ 0 & \frac{1}{d_y} & v_o \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ z_m \\ 1 \end{bmatrix} \tag{18}$$

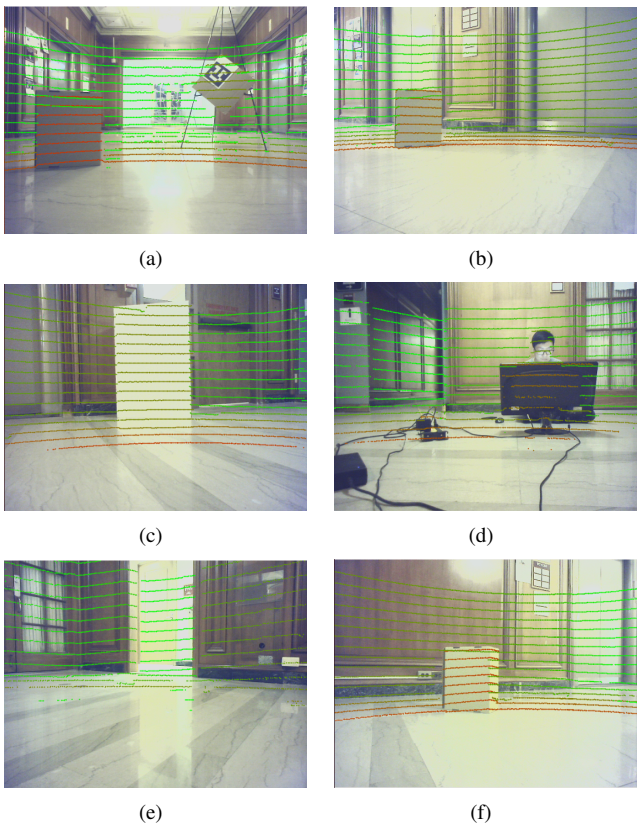
Proceedinly, we can use Formula[6] to build the relationship between (x_m, y_m, z_m) and (u, v). Thus, distortion coefficients can be calculated.

E. Fusion

In this paper, the target of data fusion is to integrate two different sets of sensor data. This integrated data can offer information including color, shape and spatial location at the same time for analysis. Such data can be used to build a

single object or a model which includes different features from different sensors. To achieve this target, the LiDAR data, 3D point clouds, and RGB images received from the camera should be synchronized in both time order and spatial location. Here, HydraView uses LiDAR data as the main thread and separately implements project and un-distortion to each image capture from different cameras. After that, the integrated data format for each frame is stored in an N-layer matrix, which is used as a container to store each fusion image. In this paper, HydraView uses the first three layers to store RGB image data, and the rest of the layers store distance reflects intensity, etc. Moreover, for visible results, it is capable of drawing projected cloud points on RGB image for each camera. For 360° view, HydraView simply extends the matrix and stores six different views by clockwise order.

F. Results



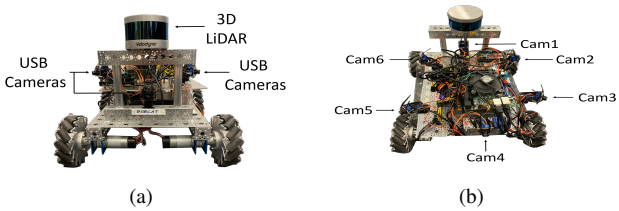
(a) (b) (c) (d) (e) (f)

Fig. 8: Fusion image in six different views.

Figure 8 shows the images of six camera views in different directions with LiDAR scanning lines. Green lines mean that the object is far from LiDAR and red lines mean that the object is close to LiDAR.

IV. Experiment and Evaluation

In this section, we present the test environment for HydraView and parameters for multiple sensors. The test environment setting is a clear space without dynamic moving or any other noise. All devices are assumed to work properly. Since all six cameras are same model, they are assumed to display similar performance under the same parameters. For the evaluation part, this paper focus on the improvement in both computing and synchronizing to demonstrate the ability to achieve real-time processing.



(a) (b)

Fig. 9: (a) Front View; (b) Top View

A. Experiment Setup

The layout for HydraView includes six USB cameras and one VLP-16 LiDAR. The working frequency of each camera is 30fps, which means that the camera is capable of getting a new frame of the image around every 33ms. Moreover, since the work frequency for LiDAR is 10Hz, it can get a new scanning frame around every 100ms. All programs are running in the Ubuntu operation system, and the driver for LiDAR and the camera are from the ROS package. The union time stamp and raw data also take from the ROS service. Achievement for synchronization, ROI, TOI, and project are programmed in Python.

1) System Layout: As Figure 9 (a) shows, the LiDAR is mounted on the front-top of the mobile platform, and one camera is mounted under the LiDAR for the front view. Figure 9 (b) shows the location relationship between six cameras. Each camera can cover around a 60° field of view.

(a) (b)

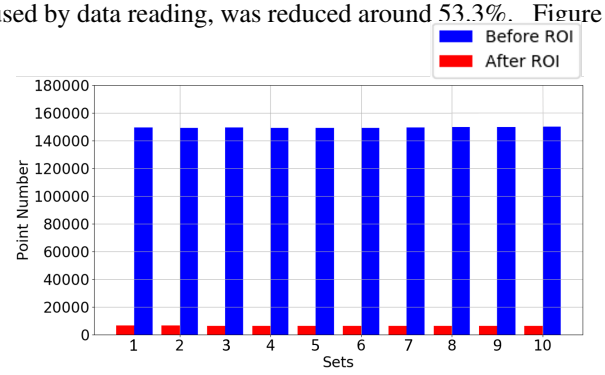
Fig. 10: (a) Back View; (b) Test Environment

2) Environment Setup: The test field location is in the elevator hall, as Figure 10 shows. In here, multiple static objects have been placed in front of HydraView to test visible distance information and the performance of calibration results.

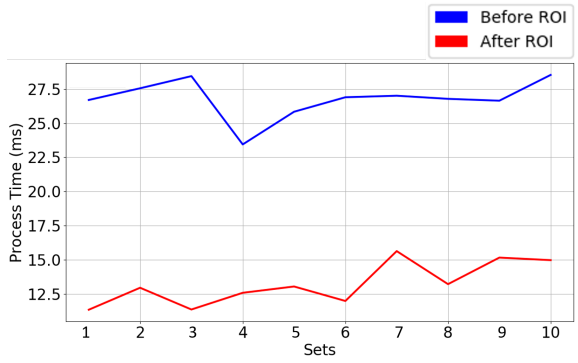
B. Performance Evaluation

Performance evaluation mainly focused on the improvement in computing time reduction in synchronization and projection to achieve real-time scanning. Concerning synchronization, this paper shows the comparison results in processing time and point number of synchronization by taking the time gap between image and corresponding point cloud as a synchronized result. For processing time of projection, this paper shows the result of processing time along with the increased point number for the projected 3D point cloud into 2D pixel space. The last part shows the ability in real-time processing.

1) Performance of Synchronization: In this part, the ROI was implemented with TOI, meaning that the data has been synchronized both in time and spatial location. This synchronization can filter out the point cloud that can not be seen by the camera, which includes the side view that is out of view range and back view. Figure 11 shows the comparison result for synchronization. Here, the blue bar and curve represent performance before applying ROI, which processes all point cloud data without any filter. Furthermore, the red bar and curve represent the performance of the algorithm, which only processes the region of interest point cloud data. The number of point cloud needs to be processed were reduced around 80.8%, and the time for synchronization, which is mainly caused by data reading, was reduced around 53.3%. Figure 12



(a)



(b)

Fig. 11: Synchronization of six cameras and one LiDAR: (a) Point number reduce; (b) Process time reduce

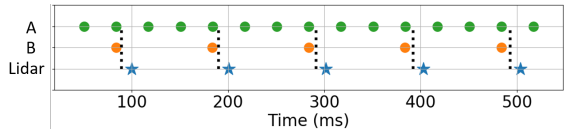


Fig. 12: Synchronization for one camera.

shows a comparison between synchronized image data and raw image data. Point set A represents the time when the camera takes one image and work frequency for one camera. Point sets B represent the post-synchronizing image data. Here, the vertically dashed lines represent the scanning moment for the corresponding image.

There is little time shift for both LiDAR and the camera data in union time order, which is caused by the reason that the working frequency of LiDAR and camera is not precisely 100ms and 33ms. Moreover, there also exists a time shift between LiDAR data and the camera data caused by the same reason. Figure 13 shows the 100 sets time gap data between the image and the corresponding LiDAR data. The maximum

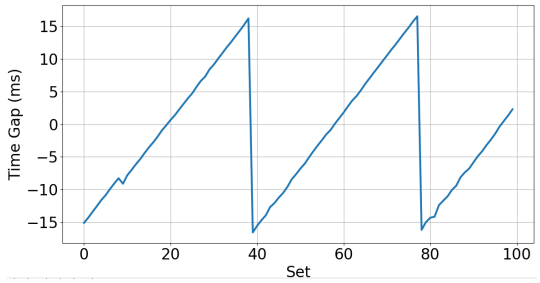


Fig. 13: Time gap between synchronized image and corresponding LiDAR data.

time gap, in other words, the waiting time in each camera for corresponding point data is around 16ms based on the work frequency of cameras. The maximum time gap is determined by the working frequency of the camera, which is shown in Figure 5. Figure 14 shows the synchronized results between six cameras and LiDAR. Each colored circle represents the image moment, which is determined by HydraView from storage, the blue star represents the moment of finishing one cycle of LiDAR scanning, and the straight dashed line represents the ideal synchronized result. The time shift between ideal results and HydraView results is mainly caused by different starting time of each camera and the unstable work frequency.

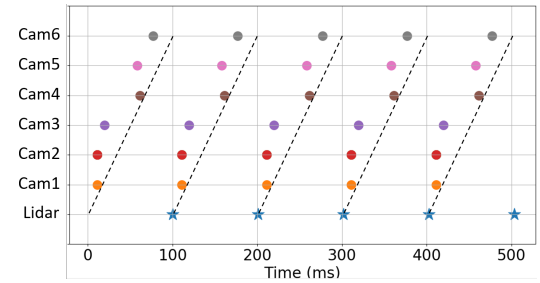


Fig. 14: Synchronized six cameras data and LiDAR data.

2) Performance of Projection: In this paper, the target of projection is to integrate two different data, image and points cloud, as fusion data. HydraView has six cameras and one LiDAR, therefore, the project part needs to process the projection task six times for one scanning cycle. Right now, HydraView only uses one single thread to process six projection task one by one. However, multiple threads should be a better option in future work. Figure 16 shows the processing time for different point numbers. The black dashed line represents the mean value of processing time for project 3D points cloud into 2D pixel space, the red line represents the maximum value, and the blue line represents the minimum value in collected sample. The vertical bar A represents the distribution of point number and processing time for synchronized data, and B

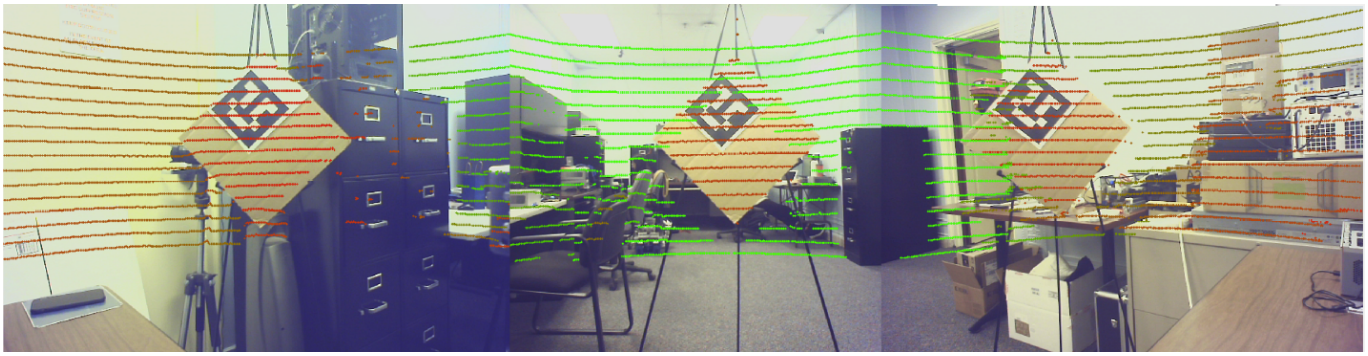


Fig. 15: 180° fusion view.

represents the data that without the synchronization algorithm as a comparison result.

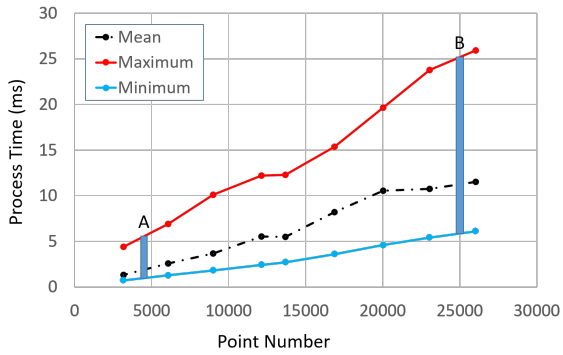


Fig. 16: Processing time for different point number.

3) Performance of Real-time Processing: To implement HydraView in autonomous vehicles, the real-time processing needs to be achieved. In this paper, the maximum waiting time is determined by the work frequency of LiDAR. Therefore, the two tasks, synchronization and projection, need to be finished in one cycle scanning time for real-time processing. According to the statistics results from our sample in Figure 11 and Figure 16 shown, the maximum time of synchronization and projection for raw data are around 28ms and 25 * 6ms. For HydraView, the maximum time of synchronization and projection for raw data are around 16ms and 4 * 6ms. Processing time has been reduced by around 77.53%. Figure 17 shows the comparison results for the ability to achieve real-time processing in autonomous vehicles. Here, A represents data processing tasks of a and b, which is projection and synchronization, without any filter. And B represents the result from HydraView. 100ms is determined by one scanning cycle time of LiDAR.

V. Limitation and Future Work

As Figure 15 shows, HydraView simply combined different views by clockwise order for wider view purpose. Based on the current results, two problems should be considered in future works. First, we identify that the black oil paint has massive affection for LiDAR scanning results, which shown in Figure 15. Secondly, we should eliminate the distortion and color change on the junctions between two images. We

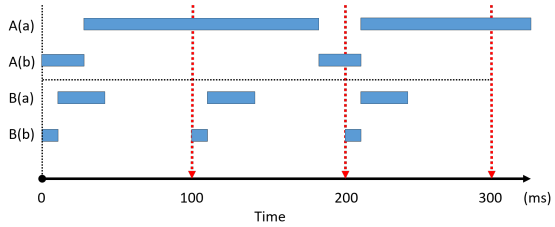


Fig. 17: Comparison results for real-time processing ability.

will also focus on more evaluation in different platforms to compare computing performances. From our experiments, the dynamic test environment with moving objects and motion in mobile platform have affected performance, and distance must also need to be considered as another factor. Moreover, motion compensation should be considered to reduce point location shifts caused by motion in a dynamic environment. Furthermore, the results show that there is plenty of rest time in one cycle processing time. Therefore, more tasks like object detection could also be considered in future work. Also, the additional images which are not used in fusion, that also had been stored during the LiDAR scanning period, can be used to form compensate information.

VI. Conclusion

In this paper, we propose a 360°-view synchronization and fusion platform for multiple sensors that can be deployed in the autonomous vehicles for real-time processing. It offers integrated and synchronized data of the surrounding environment that can be extremely helpful to build a driving model for further analysis. The results show that HydraView has an excellent ability for multiple sensor fusion and synchronization in the 360° field of view with significantly reduced processing times. The successful implementation of ROI and TOI largely reduced the computing time, which is capable to achieve real-time processing that is comparable to state of art approaches. Also, the results show that HydraView has a good robustness in synchronization for time shift caused by sensors. Additionally, since the synchronization task can be finished earlier than one scanning cycle of the LiDAR, We consider to make

full usage of this period by introducing other tasks such as object detection and more related works to improve the system performance in the future.

REFERENCES

[1] S. Liu, J. Tang, Z. Zhang, and J. Gaudiot, "Computer architectures for autonomous driving," *Computer*, vol. 50, no. 8, pp. 18–25, 2017.

[2] C. Ilas, "Electronic sensing technologies for autonomous ground vehicles: A review," in *2013 8TH INTERNATIONAL SYMPOSIUM ON ADVANCED TOPICS IN ELECTRICAL ENGINEERING (ATEE)*, May 2013, pp. 1–6.

[3] W. Shi, M. B. Alawieh, X. Li, and H. Yu, "Algorithm and hardware implementation for visual perception system in autonomous vehicle: A survey," *Integration*, vol. 59, pp. 148 – 156, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167926017303218

[4] A. Asvadi, L. Garrote, C. Premebida, P. Peixoto, and U. J. Nunes, "Multimodal vehicle detection: fusing 3d-lidar and color camera data," *Pattern Recognition Letters*, vol. 115, pp. 20–29, 2018.

[5] N. Kaempchen and K. Dietmayer, "Data synchronization strategies for multi-sensor fusion," in *Proceedings of the IEEE Conference on Intelligent Transportation Systems*, vol. 85, no. 1, 2003, pp. 1–9.

[6] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[7] Y. Wang, L. Liu, X. Zhang, and W. Shi, "Hydraone: An indoor experimental research and education platform for cavs," Jul. 2019. [Online]. Available: https://www.usenix.org/conference/hotedge19/presentation/wang

[8] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.

[9] J.-L. Blanco-Claraco, F.-Á. Moreno-Dueñas, and J. González-Jiménez, "The málaga urban dataset: High-rate stereo and lidar in a realistic urban scenario," *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 207–214, 2014.

[10] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang, "The apolloscape dataset for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 954–960.

[11] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[12] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[13] Autopilot Review, "Lidar vs. cameras for self driving cars – what's best?" https://www.autopilotreview.com/lidar-vs-cameras-self-driving-cars/,.

[14] X. Gu, A. Zang, X. Huang, A. Tokuta, and X. Chen, "Fusion of color images and lidar data for lane classification," in *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2015, p. 69.

[15] Q. Li, L. Chen, M. Li, S.-L. Shaw, and A. Nüchter, "A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 2, pp. 540–555, 2013.

[16] H. Cho, Y.-W. Seo, B. V. Kumar, and R. R. Rajkumar, "A multi-sensor fusion system for moving object detection and tracking in urban driving environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1836–1843.

[17] V. Subramanian, T. Burks, and W. Dixon, "Sensor fusion using fuzzy logic enhanced kalman filter for autonomous vehicle guidance in citrus groves," *Transactions of the ASABE*, vol. 52, no. 5, pp. 1411–1422, 2009.

[18] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[19] K. Cai, R. Yang, H. Chen, Y. Huang, X. Wen, W. Huang, and S. Ou, "Synchronization design and error analysis of near-infrared cameras in surgical navigation," *Journal of medical systems*, vol. 40, no. 1, p. 7, 2016.

[20] A. Noda, Y. Yamakawa, and M. Ishikawa, "Frame synchronization for networked high-speed vision systems," in *SENSORS, 2014 IEEE*. IEEE, 2014, pp. 269–272.

[21] D. L. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on communications*, vol. 39, no. 10, pp. 1482–1493, 1991.

[22] M. D. Lemmon, J. Ganguly, and L. Xia, "Model-based clock synchronization in networks with drifting clocks," in *Proceedings. 2000 Pacific Rim International Symposium on Dependable Computing*. IEEE, 2000, pp. 177–184.

[23] M. L. Sichitiu and C. Veerarittiphan, "Simple, accurate time synchronization for wireless sensor networks," in *2003 IEEE Wireless Communications and Networking, 2003. WCNC 2003.*, vol. 2. IEEE, 2003, pp. 1266–1273.

[24] J. Wu, Z. Xiong *et al.*, "A soft time synchronization framework for multi-sensors in autonomous localization and navigation," in *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 2018, pp. 694–699.

[25] Y.-S. Shin, Y. S. Park, and A. Kim, "Direct visual slam using sparse depth for camera-lidar system," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–8.

[26] Y. Wang, L. Liu, X. Zhang, and W. Shi, "Hydraone: An indoor experimental research and education platform for cavs," in *2nd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 19)*, 2019.

[27] A. Dhall, K. Chelani, V. Radhakrishnan, and K. M. Krishna, "Lidar-camera calibration using 3d-3d point correspondences," *arXiv preprint arXiv:1705.09785*, 2017.

[28] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and vision Computing*, vol. 76, pp. 38–47, 2018.

[29] S. Garrido-Jurado, R. Munoz-Salinas, F. J. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using mixed integer linear programming," *Pattern Recognition*, vol. 51, pp. 481–491, 2016.

[30] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International journal of computer vision*, vol. 13, no. 2, pp. 119–152, 1994.

[31] J. Heikkila, O. Silven *et al.*, "A four-step camera calibration procedure with implicit image correction," in *cvpr*, vol. 97. Citeseer, 1997, p. 1106.

[32] R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[33] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, 2000.