Part of Synchronization

Wan Haochuan

***Abstract*—This part of paper presents an approach to synchronizing different types of sensors which are installed on mapping robot.**

***Index Terms*—Sensor Synchronization**

I. Introduction

For autonomous robots, the ability of getting from one place to another place is the most important capability. In order to realize this capability, Simulaneous Localization and Mapping(SLAM) system is often used and the result of SLAM is positively correlated with the quality of input data. To get better data, more sensors are used to collect information from environment. However, as the number of sensors increases, the problem about synchronization between sensors is rised because unsynchronized system can lead sensors collect data at different time which results in errors in the generated map.

To solve the problem above, we design a system to synchronize different tpye of sensors.

II. State of the Art

In some previous works, most of approaches of synchronization just stayed on the level of software. Hang Hu [4] mainly used ROS nodes to synchronize on software level. They realized the synchronization between cameras, LIDARs and GPS. However, the approach they provided focus on calibrating data rather synchronizing when sensors generating data. In [6], a solution based on math calculation is introducted. It is a passive method to synchronize by eliminating transmission error between sensor data, which is a low-cost way to realize synchronzation. A system [1] is designed for heterogeneous sensor networks is used to synchronize time. In this work, authors designed both software and hardware for synchronization. Software guatantees sensors receive signal at the same time and hardware like phase-locked loop is to correct time when errors occur.

By Georgios Litos's work [5], a software-based system is provided. This method concerns cameras which are installed on different computers, and these computers are connected with NTP to get correct time. In order to trigger cameras at the same time, an estimate L of how much time is required for cameras data is broadcast to all computers. Though time error will occur when the message propagate, yet each computer can calculate the time to wait and trigger the camera at the right time. In this case, the system can be used in a long distance sensor synchronization with NTP. But disadvantages are also existed if this system is applyed on mapping robot directly. First, the distance between cameras on mapping robot is much closer so that to trigger different cameras with different copmuters is a waste. Second, this method can just synchronize trigger sensors, like camera and IMU. But there is another type

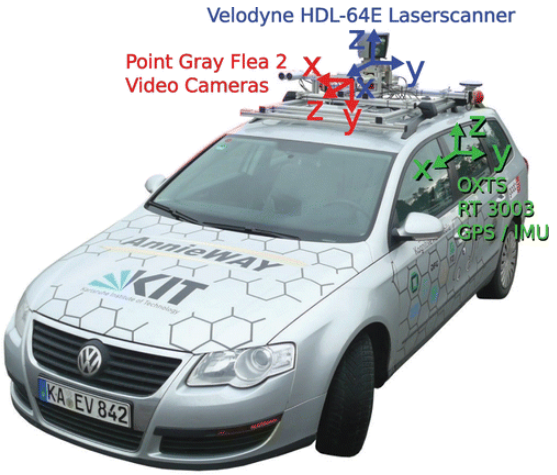

Fig. 1: KITTI car

of sensor by timestamp like pps and NMEA data which cannot be triggered, such as LIDRA and GPS. Third, the NTP can get a millisecond accuracy, which cannot meet the requirement of mapping robot in that some sensors like IMU should synchronized by a 200Hz signal. However, this work [5] provides a experiment which can evaluate the synchronization. They use a LED array controlled by a MircoChip. This array has 4 rows of 10 LEDs that can correspond to 1000ms, 100ms 10ms and 1ms granuilarities. Compared a 100Hz CRT screen which just provides 10ms accuracy, the LED array has an edge. By photographing the LED array, we can get the time of shot and judge the result of synchronization. This experiment method can evaluate some photosensitive sensor like cameras, but sensors such as IMU and LIDRA cannot be evaluated in this experiment.

KITTI dataset [3] is one of the most famous opensource dataset regarding on autonomous driving. The main function of KITTI is collecting data from real world to build map for autonomous. Though the raw data is not in formal of ROS bag, yet we can convert it to rosbag easiliy by "kitti2bag" in python. And here we can use a ROS package which is "kitti_to_rosbag". By this ros package, KITTI dataset raw data can be converted to a ROS bag and allows a library for direct access to poses, velodyne scans and images. The car used by KITTI is equipped with a HDL-64E 3D Lidar, grayscale and RGB stereo cameras, which is show in Fig 1. All the Lidar and the cameras are synchronized and generates data at 10Hz. Global positioning system (GPS) and IMU data are also included in this dataset with 100Hz. The synchronization between GPS and IMU is not hardware-level synchronized with the camera or the Lidar. They use the time stamp to synchronize among them. There are six categories in KITTI and they are city, residential, road, campus, person

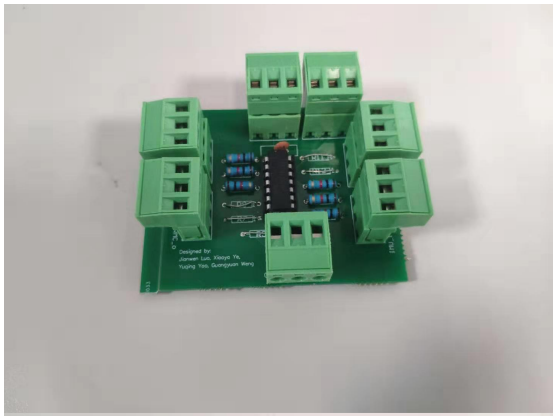

Fig. 2: PCB for singal amplification

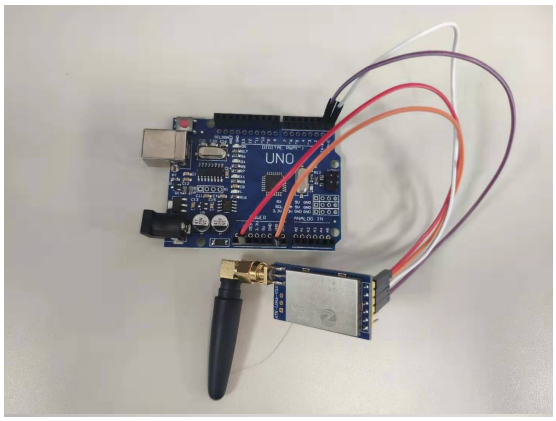

Fig. 3: Wireless singal transmitter for tracking system

and calibration. KITTI dataset is also used to evaluate the performance of computer vision technology such as stereo, optical flow, visual odometry, 3D object detection and 3D tracking in the vehicle environment. KITTI contains real-world image data from scenes such as urban, rural, and highways, with up to 15 vehicles and 30 pedestrians per image, as well as varying degrees of occlusion and truncation. The entire data set consists of 389 pairs of stereo images and optical flow maps, 39.2km visual ranging sequences and images of 3D labeled objects, sampled and synchronized at 10Hz. For 3D object detection, the label is subdivided into car, van, truck, pedestrian, pedestrian (sitting), cyclist, tram and misc.

In Xu Ding's work [2], they introduced a method based on LED arrays to measure errors between different cameras. They use a FPGA to control LED matrix to get a high precision method to detect the synchronization accuracy. The time of cameras capturing an image can be represented by the state of LED matrix. According to the capture time, the relative time interval between the cameras can be calculated. However, in their work, they only realized 8ms accuracy, which is not enough to measure our system where the highest frequence is up to 200Hz.

III. System Description

On our machine, there are two different classes deviced to be synchronized. One is triggered devices and GPS-stamp

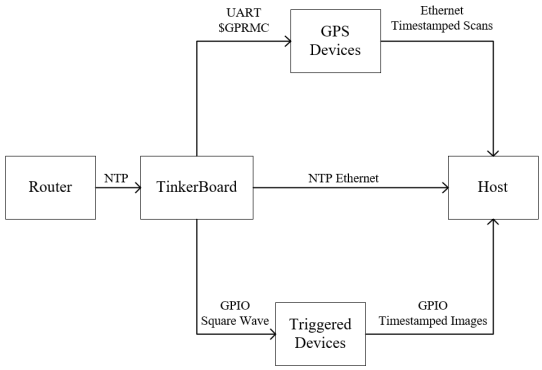

Fig. 4: Trigger & Timestamp Spread Topology

devices. For triggered devices (e.g. cameras and IMUs), we need to generate a square wave (its initial phase is aligned with the second boundary) and pass the wave to the device as a trigger signal. For another GPS timestamp device (e.g. Velodyne Lidars), we use a 1Hz square wave, whose duty cycle is adapted to the specified target device as PPS through GPIO (General Purpose Input Output), and the timestamp is encoded as GPRMC (NMEA 2.0) Sentence) format via UART (Universal Asynchronous Receiver / Transmitter). The main problem of synchronization is to generate two different types of signals and ensure the delay between each signals are acceptable.

The core of the synchronization is TinkerBoard which broadcast triggering signals and NMEA message for two different kinds of sensors —— Triggered devices and GPS devices. Triggered devices includes cameras at 10Hz, tracking system at 30Hz and IMU at 200Hz; GPS devices are LIDRAs, Velodyne HDL-32E, which is synchronized by a PPS and message of $GPRMC every second. The clock of TinkerBorad is synchronized by NTP with the host. To control different cameras with same signal, a PCB in Fig 2 is designed to broadcase signal from one signal and convert 3.3V to 5V. Considering synchronization with Optitrack tracking system, a wireless signal transmitter is designed based on Arduino in Fig 3. The whole system is a hardware-level synchronization and showed in Fig 4.

a) Details for synchronization: There are 2 programs in */src* of the synchronization ROS package, which are *"camera_sync.cc"* and *"velodyne64_sync.c"*. **"camera_sync.cc"** is for cameras synchronization. It achieves synchronization by obtaining the cpu time, calculating the time that the trigger appears at different frequencies, and manipulating the GPIO port. The program will output the high / low level through the GPIO port at a specific time, and send the current ros timestamp at the same time through the ROS message. It supports the output of up to 5 different camera frequencies, and the output frequency can be within $1 \sim 200$Hz, which can be changed in *"/launch/sync.launch"*. **"velodyne64_sync.c"**'s workflow is similar to *camera_sync.cc*, but the difference is that it will start sending a 60ms high-level PPS every second to synchronize velodyne, and then send a simulated GPS sentence ($GPRMC) through the serial port.

Fig. 5: Photo at about 00:00:09.674

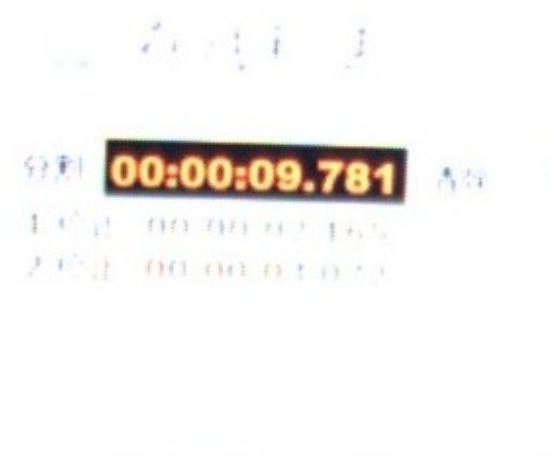

Fig. 6: Photo at about 00:00:09.781

IV. System Evaluation

In order to evaluate the sychronization system, experiment are designed. Refer to the [5], we will make a LED array controlled by MircoChip to represent time. Let cameras to photograph the LED array and compare pictures from different cameras to visualize delay between cameras. For part of camera synchronization, we define it is a "good synchronization" as the relavite time differnece between two cameras is smaller than 1ms.

For other sensors, such as IMU and LIDRA, cannot detect their time error by the approach from [5] in that their own sensor properties. So, we should update the experiment system. Because IMU detects the movement of object and LIDRA uses light beams to scan objects around, so besides the LED array, we will build a rotating strick like a windmill with IMU installed. Put different LIDRAs in front of the rotating strick to scan and achieve the status of the system together with IMU. By comparing the different statuses from LIDRAs and IMU at the same time, we can evalue the synchronization system.

However, there are some remaining problem of the experiment system. To compare system statuses from LIDRAs and IMU, system calibration is essential. But for now it is difficult to calibrate LIDRA and IMU.

So far, we have done a simple test for camera synchronization. Fig 5 and 6 are two adjacent images about an online timer captured by the same camera. It shows that the time interval bewteen two images is about 107ms, which is close to 10Hz. Due to the limitations of the screen refresh rate and the accuracy of the network timer, we could not get the exact time difference. We also observed a blurred image from the image, which indicates that this measurement is not accurate. So next we will perform more accurate measurement according to the methods of [5] and [2].

V. Conclusions

For synchronization part, the synchronization system has been completed and worked well and we will perform a more accurate measurement in next stage.

References

[1] Isaac Amundson, Manish Kushwaha, Branislav Kusy, Peter Volgyesi, Gyula Simon, Xenofon Koutsoukos, and Akos Ledeczi. Time synchronization for multi-modal target tracking in heterogeneous sensor networks. *Social Science Electronic Publishing*, 11(2):152–4, 2006.

[2] X. Ding and P. Tao. Synchronization detection of multicamera system based on led matrix. In *2016 13th International Conference on Embedded Software and Systems (ICESS)*, pages 18–23, Aug 2016.

[3] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[4] h. Hu, J. Wu, and Z. Xiong. A soft time synchronization framework for multi-sensors in autonomous localization and navigation. In *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 694–699, July 2018.

[5] Georgios Litos, Xenophon Zabulis, and Georgios Triantafyllidis. Synchronous image acquisition based on network synchronization. In *Conference on Computer Vision & Pattern Recognition Workshop*, 2006.

[6] E. Olson. A passive solution to the sensor synchronization problem. 2010.