Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

IEEE 1588-Based General and Precise Time Synchronization Method for Multiple Sensors*

Limeng Lu[1,2,#], Chaofan Zhang[1,#], Yong Liu[1], Wen Zhang[1], Yingwei Xia[1,§]

[1] *Opto-Electronics Applied Technology Research Centre, Institute of Applied Technology, and Hefei Institutes of Physical Science, Chinese Academy of Sciences Department of Intelligent Robotics.*

[2] *Science Island Branch of Graduate School, University of Science and Technology of China.*

Hefei 230026, P. R. China.

xiayw@aiofm.ac.cn, zcf0413@mail.ustc.edu.cn

***Abstract* - An increasing number of robotic systems contain multiple sensors, and the accurate timestamps of sensors are crucial for fusing multiple sensors information in robot applications. However, the accuracy and stability of traditional timestamp synchronization algorithms are degraded when synchronizing the timestamps of multiple sensors, due to the clock inconsistency, asynchronous trigger delay, and variable transmission delay. This paper presents a general and precise time synchronization method for multiple sensors with respect to a local clock (i.e. typically a computer's internal clock). First, to reduce the software scheduling delay, an IEEE 1588-based method, which combines hardware and software to obtain timestamps, is formulated to modify the master–slave timestamp acquisition location. Second, a hardware synchronization trigger method is proposed to reduce the runtime of trigger threads in the software, by introducing the microprocessor to trigger multiple sensors synchronously. Finally, a timestamp matching method is proposed to reduce the unreliability of variable communication delays by matching the trigger timestamp and the timestamp of data reception. To the best of our knowledge, this approach is the first synchronization method to combine hardware and software to obtain the timestamps of multiple sensors. Results show that the ability of our method can estimate time offset to microsecond accuracy.**

Index Terms* - *multiple sensors, IEEE 1588, precise timestamp, hardware trigger, matching method.

I. INTRODUCTION

Recently, an increasing number of robotic systems use multiple sensors for improved perception of environment information [1-4], where sensor fusion, which involves combining data from several sensors, is the basic step. Generally, the performance of sensor fusion algorithms is significantly affected by the timestamp offsets of multiple sensors [5-6]. For example, the sensor fusion algorithm will produce inaccurate or even erroneous results when the timestamp offset is too large [7]. Thus, it's critical to generate accurate synchronized timestamps of multiple sensors.

Usually, it's challenge to obtain the accurate synchronization timestamps for multiple sensors with respect to a local clock (i.e. typically a computer's internal clock), due to the inconsistency of clock frequencies, variable transmission delays, and operating system scheduling delays [8]. Numerous synchronization methods have been proposed in recent years [9-16]. A hardware synchronization method was proposed in Reference [9], and Nikolic et al. used a FPGA-specialized platform to achieve accurate timestamp synchronization. A synchronized method that uses global positioning system was proposed in Reference [11], and this method can synchronize the clock with the coordinated universal time. The abovementioned approaches [9], [11] are hardware synchronization methods and are not always practical because it is difficult to integrate commercial sensors into hardware devices. Recently, some software synchronization methods that are suitable for multiple sensors have been proposed [12-16]. Furgale et al. proposed an optimization-based software synchronization method to calibrate time offset in temporal and spatial domains [12]. Li et al. presented an online filtering-based software synchronization method to reduce the time offset [13]. These methods [12], [13] are generally sensor-specific, moreover, they assume a static offset and cannot account for variable transport delays. Harrison et al. used the TICSync based on Network Time Protocol (NTP) to synchronize timestamp [14]. In 2015, English et al. presented a TriggerSync one-way synchronization algorithm [15] and used a sensor's minimum delay as the trigger timestamp. Unlike the abovementioned approaches of NTP-based synchronization methods [14], [15], Kovácsházy et al. [16] implemented an IEEE 1588 Precision Time Protocol (PTP)-based method to obtain the timestamp of packets from different locations. The IEEE 1588 PTP-based method [16] can achieve higher synchronization precision than the NTP-based method [14], [15]. However, on the one hand, the accuracy of the IEEE 1588-based full-software synchronization algorithm is difficult to meet the requirement of practical applications. On the other hand, the full-hardware method cannot be used in ordinary computers, where the network card does not support the IEEE 1588 PTP protocol. Thus, a general and precise time

§ Yingwei Xia is the corresponding author (email: xiayw@aiofm.ac.cn).

These authors contributed equally to this work.

* The work was supported by Science and Technology Service Network Initiative(KFJ-STS-QYZD-097), The key research and development program of Anhui province of china(201904a05020060), Special Foundation of President of the Hefei Institutes of Physical Science(YZJJ2019QN22) and The Key Research Program of the Hefei Institutes of Physical Science during the 13th Five-Year Plan Period(Y97Z031892).

synchronization method for multiple sensors is a challenging problem.

In the present study, motivated by the abovementioned two kinds of software and hardware synchronization methods, an IEEE 1588-based general and precise time synchronization method with respect to a local clock is proposed to obtain accurate timestamps for multiple trigger sensors. We initially modify the master–slave timestamp acquisition location of the IEEE 1588 PTP protocol to reduce software scheduling delays. The modification aims to propose a hardware and software combination method to obtain timestamps directly from the driver and physical layers. The trigger delay problem caused by task scheduling is addressed by adopting a hardware-triggered method. A microprocessor is adopted as the slave clock node to hardware trigger multiple sensors, thereby ensuring that multiple sensors are triggered simultaneously. Furthermore, a timestamp matching method is proposed by matching the trigger timestamp and the timestamp of data reception to eliminate variable communication delays. The main contributions of this paper are presented as follows:

1) To improve synchronization accuracy, a novel IEEE1588-based time synchronization method for multiple sensors is formulated. This method is proposed to combine hardware and software for the first time to obtain timestamps.

2) To solve the trigger delay problem caused by task scheduling, a microprocessor is introduced as the slave clock node to hardware trigger multiple sensors simultaneously.

3) To improve the accuracy of data matching, a timestamp matching method is proposed to match the trigger timestamp and the timestamp of reception.

The remainder of this paper is organized as follows. We briefly introduce the PTP protocol in Section II. Then, we introduce the timestamp synchronization mechanism in Section III. Section IV discusses the experiment details and results. Finally, the conclusions drawn from this paper are presented in Section V.

II. SYNCHRONIZATION MECHANISM

PTP is a time synchronization protocol that aims at local area networks and supports software and hardware timestamping of packets in the network interface controller (NIC). The PTP is standardized as IEEE 1588 [17]. Although the protocol does not limit the medium to Ethernet, nearly all implementations use the Ethernet. Then, we provide a brief overview of the theoretical knowledge and algorithmic mechanisms for the PTP protocol.

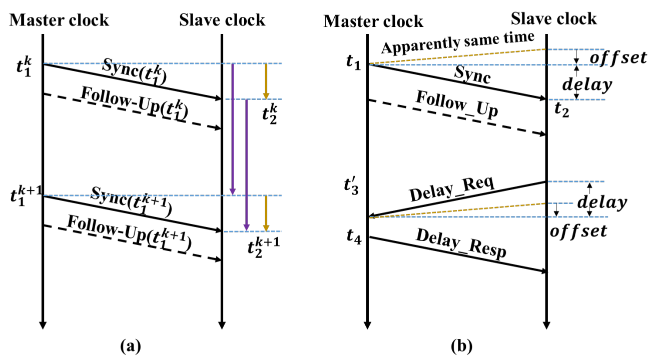


Fig. 1 PTP clock synchronization principle.
The (a) shows that the frequency of the master and slave clocks is adjusted.
The (b) shows that the one-way delay and offset are calculated.

PTP is basically a master–slave architecture. The master provides the clock to synchronize all slaves. If the master has an inaccurate clock, then all the slaves will adjust the inaccuracy of the master. Therefore, the optimal clock in the system is voted as the master by the best master clock (BMC) algorithm. Each node announces its clock capabilities to choose the best one as the master clock. The slaves can start synchronizing once the master is defined. The whole synchronization process is divided into two procedures.

A. *Principle of synchronization*

The first procedure is responsible for adjusting the slave clock to make it run at the exact same frequency as the master clock (Fig. 1(a)). Therefore, Sync messages are continuously sent by the master and processed by the slaves. The calculation of the frequency deviation requires the transmit and the receive timestamps. After sending the Sync message, the master clock sends a Follow-up message containing the transmit timestamp of the Sync message. The timestamp of reception is produced by the slaves themselves. With the knowledge of the sending and receiving times, the slaves can calculate the interval between two Sync messages and the transmit delay using the following formulas:

$$\text{Interval:} \quad t_1^{k+1} - t_1^k = t_2^{k+1} - t_2^k \,. \tag{1}$$

$$\text{DownlinkDealy:} \quad t_2^{k+1} - t_1^{k+1} = t_2^k - t_1^k \,. \tag{2}$$

B. *Principle of synchronization*

The second procedure is responsible for time synchronization. After the first procedure, the clocks of the master and the slaves will run at the same speed but likely with an offset in time because they were not started at exactly the same time (Fig. 1(b)). Therefore, the round trip time between the master and slave clocks is measured. The downlink is already known from the Sync message before, and the uplink is measured with a Delay Request message sent from the slaves. For simplicity, assume that the uplink and downlink are symmetric. The Delay Response is used to return the timestamp of reception of the Delay Request back to the slaves. The calculation of the one-way delay and the offset are therefore performed as follows:

$$\text{Delay} = \frac{[(t_2-t_1)+(t_4-t_3)]}{2} \,. \tag{3}$$

$$\text{Offset} = \frac{[(t_2-t_1)-(t_4-t_3)]}{2} \,. \tag{4}$$

III. TIMESTAMP SYNCHRONIZATION ALGORITHM

In this section, we describe the basic structure (Fig. 2) of the proposed timestamp synchronization protocol. The process of implementation is divided into two steps, that is, timestamp

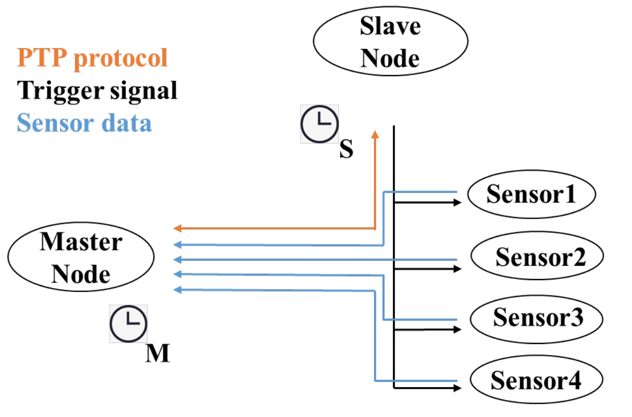


Fig. 2 The procedure of timestamp synchronization. There are three aspects: the PTP protocol was used to synchronize master-slave timestamp, the slave node was used to synchronization trigger multiple sensors, and the matching method correctly matches sensor data and the corresponding timestamp.

acquisition location (Section A) and hardware synchronization trigger and matching method (Section B).

A. Timestamp acquisition location

The PTP protocol uses a timestamp to indicate the transmission and reception times of the messages. The acquisition position of timestamp directly affects the synchronization precision. Three kinds of acquisition location are identified. These locations are the application, the NIC driver, and the physical layers (Fig. 3). Location A represents the most common way of obtaining timestamps directly from the application layer. Location B denotes the timestamp obtained from the NIC driver layer. Location C signifies the timestamp obtained from the physical layer.

The PTP message is transmitted over user datagram protocol in IEEE802.3 Ethernet. The PTP message packet is encapsulated at the application layer and then passes through the transport, the network, and the data link layers to reach the physical layer. The transmission time from the application layer to the physical layer is uncontrollable because operating

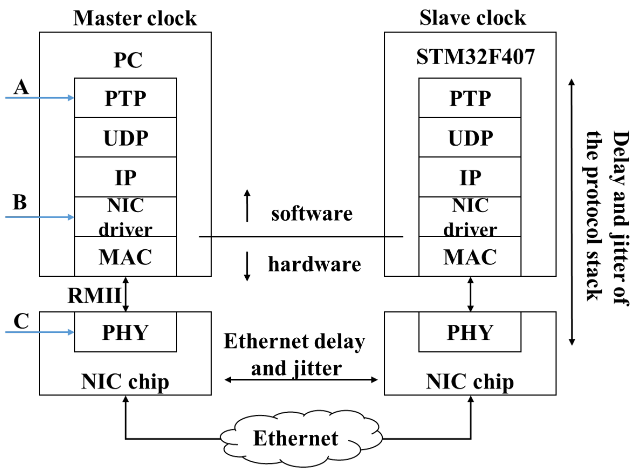


Fig. 3 Timestamp acquisition location of the PTP protocol. The location A represents the application layer, the location B represents the NIC driver layer and the location C represents the physical layer.

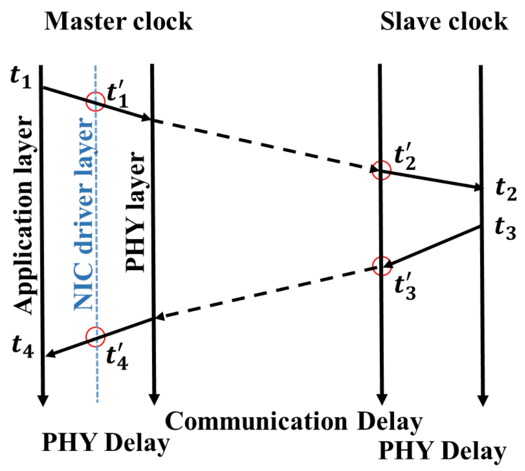


Fig. 4 Get the timestamp by selecting the location closest to the physical layer. In our platform, the master clock gets the timestamp at the NIC driver layer, and the slave clock gets the timestamp at the physical layer.

system scheduling and other random sources can drastically influence timestamp accuracy. Thus, the timestamp obtained from the application layer will generate a large offset. The timestamp can be obtained from the NIC driver layer. In this case, the performance is primarily affected by the interrupt latency and the execution time of the interrupt handler code. The timestamps generated from the NIC driver layer have more accuracy than the timestamps generated from the application layer because the NIC driver layer is closer to the physical layer. In addition, the NIC driver layer is the most accurate location for software methods to gather timestamps. To obtain the most accurate timestamps with nearly no jitter, it is necessary to generate the timestamps as close as possible to the communication wire [16]. In the present study, we propose to generate timestamps in the physical-layer chip. This approach can accurately record the moment when the message leaves the device. However, obtaining timestamps at the physical layer requires hardware devices.

We choose STM32F407 as our slave clock node. To improve the timestamp accuracy, the physical-layer chip DP83640 is selected to obtain the timestamp of the packet. The chip can also eliminate the delay and jitter of the protocol stack. Thus, the calculated delay of the message on the transmission path and the deviation of the master–slave clock is increasingly accurate (Fig. 4, the timestamp of $t_2^{'}$ and $t_3^{'}$ rather than t_2 and t_3 is selected). Simultaneously, we select a normal computer as the main clock node. We can only use the software method to obtain a timestamp because most network cards do not support the IEEE 1588 PTP. Therefore, the timestamp of the NIC driver layer is preferred.

B. Hardware synchronization trigger and matching algorithm

The microprocessor STM32F407 is used to trigger multiple sensors simultaneously. We assume that all sensors are triggered simultaneously when the delay of the trigger line is ignored [18]. The difference in the size of sensor data can

Algorithm 1. The timestamps matching method

```
Input :  $\delta_{t\min}[c]$, $\delta_{t\max}[c]$, numsensors.
Output:  time, data[c].
Process:
c ← 0
  for c < numsensors do
    if receivetime - triggertime > - $\delta_{t\min}[c]$
    && receivetime -triggertime < $\delta_{t\max}[c]$ then
      time ← triggertime
      data[c] ← data
    endif
    c ← c+1
 endfor
end
```

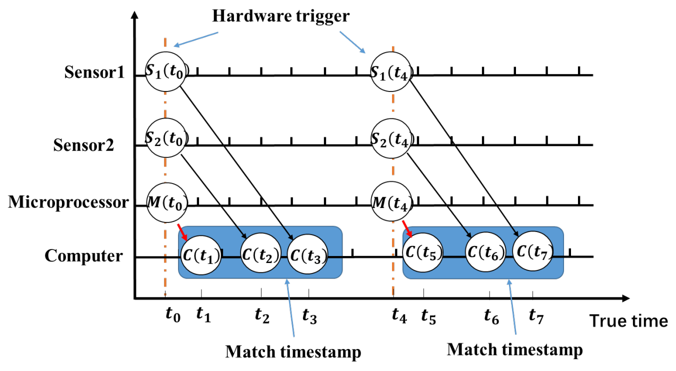


Fig. 5 The multiple sensors are triggered simultaneously. We can match messages together according to the delay of sensors and can use the trigger timestamps as the clock of the multiple sensors.

result in the difference in transmission delays. For example, in a visual inertial navigation system, cameras often take tens of milliseconds to transmit images of several megabytes in size, whereas an Inertial Measurement Unit (IMU) may take less than a millisecond to transmit a sensor measurement.

In Fig. 5, multiple sensors are synchronously triggered at t_0. After a certain period, the computer receives the sensor data and records the current time. Therefore, the calculation of the transmission delay is performed as follows:

$$\delta_t = C(t) - S(t)\,. \tag{5}$$

The transmission delay of each sensor is within a threshold range, and a range of trigger times can be calculated. We can correctly match the timestamp and sensor data by judging whether the trigger time is within the current range. The proposed matching algorithm is demonstrated in Algorithm 1.

Although the messages have the same size, the different transmission delays will cause a different arrival time. In accordance with the transmission delay, we can match the timestamp and the corresponding data, which can significantly reduce the matching error. The proposed method is also suitable for other commercial sensors.

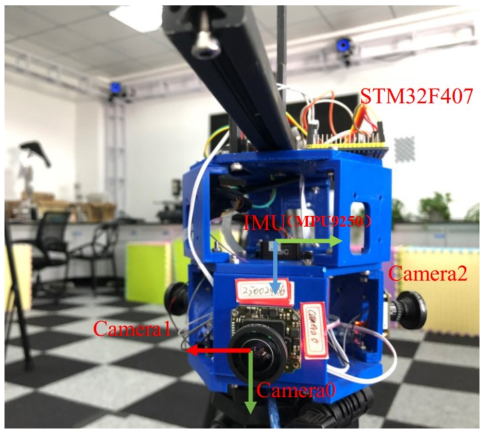


Fig. 6 Our multiple sensors platform, equipped with 3 cameras and an IMU and triggered by STM32F407.

IV. EXPERIMENTS

We performed various experiments to evaluate the performance of the proposed synchronization method for multiple sensors. In Experiment 1, the performance of various parts of the proposed synchronized method, including the accuracy of the IEEE 1588–based timestamp synchronization and the correctness of the timestamp matching method for multiple sensors, was tested. Then, Experiment 2 validated the accuracy of the proposed time synchronization method for multiple sensors in comparison with other typical algorithms.

Implementation Details: These experiments were performed using our home-made platform (Fig. 6), which was equipped with three global shutter cameras (mvBlueFOX-MLC200w) and a consumer-grade IMU (MPU9250). These devices are connected to a computer through a USB. During the experiments, the microprocessor (STM32F407) was used as the slave clock node to generate a trigger signal, and each sensor has a different trigger frequency.

Experiment 1. Performance of the IEEE 1588-based timestamp synchronization and data matching

In the experiment of the IEEE 1588-based timestamp synchronization, we tested the time offset of the master–slave

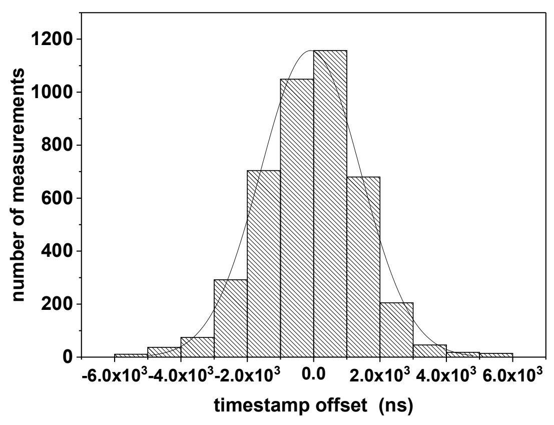


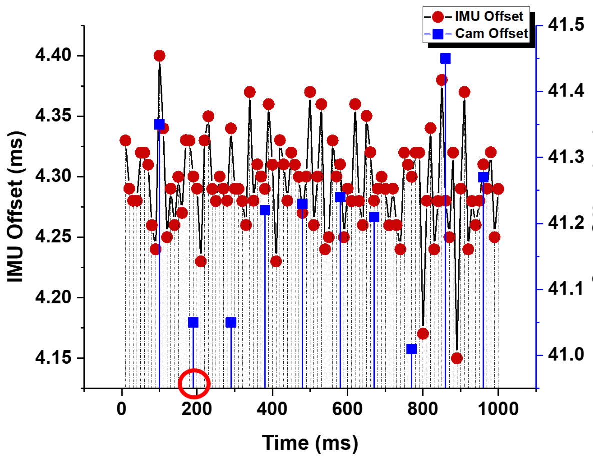
Fig. 7 The histogram of Timestamp synchronization

TABLE I MEASUREMENT RESULTS - TIMESTAMP SYNCHRONIZATION

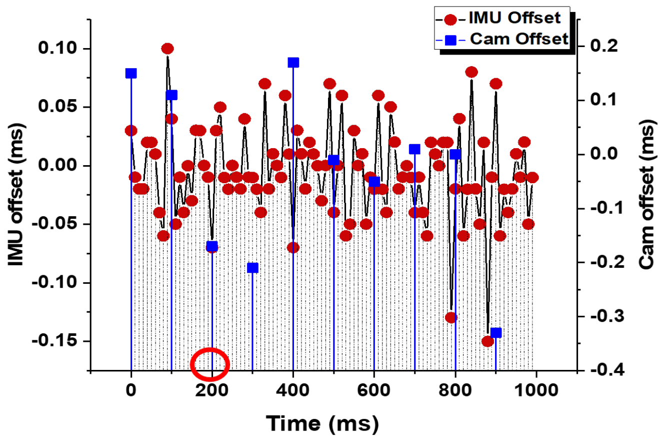
Method	Synchronization accuracy (us)				Sample size
	Mean	Std. deviation	Min	Max	
software and hardware IEEE1588	-0.091	1.51	-5.935	5.988	4287
full-software IEEE 1588	-0.048	22.24	-225.1	164.5	3239

TABLE II THE DETAILED RESULTS ON SENSOR DELAY RANGE.

Sensor	Max(ms)	Min(ms)	Mean(ms)	Threshold Range(ms)
IMU	4.42	4.15	4.30	0.5
Cam	41.80	40.77	41.20	2



(a)



(b)

Fig. 8 Multiple sensors matching at different trigger frequencies. The matching result is mapped to the time axis. If the timestamp is matched accurately, a set of camera data (blue line) should correspond to 10 sets of IMU data (black line). The (a) shows that a set of camera data and 10 sets of IMU data cannot be matched each time. The (b) shows that IMU and Cam can be correctly matched when using the matching algorithm.

node of IEEE 1588. We compared the full-software IEEE 1588 and the proposed IEEE1588-based that combines software and hardware. We use mean and standard deviation of the timestamp offset as the evaluation metrics to demonstrate the results quantitatively. Table I summarizes the detailed results, and Fig. 7 depicts the histograms of Table I. The comparison presented in Table I and Fig. 7 indicated that the method of the proposed IEEE 1588-based combining software and hardware showed a smaller time offset and a lower standard deviation than the full-software IEEE 1588. This result was due to the proposed IEEE 1588-based timestamp synchronization method that obtains the timestamp directly from the NIC driver and the physical layers, thus it can significantly reduce the software scheduling delays.

The timestamp matching method for multiple sensors was mainly proposed to match the trigger timestamp and the timestamp of reception. Before we tested the correctness of the proposed timestamp matching method, we should input the threshold range of sensors. The detailed parameters of the sensors are listed in Table II. The maximum and minimum values of the IMU were 4.55 ms and 4.05 ms, respectively, and the maximum and minimum values of the camera were 42.2 ms and 40.2 ms, correspondingly.

In the experiment of the proposed timestamp matching method, the trigger frequency of camera is 10 Hz and the trigger frequency of IMU is 100 Hz. The principle of the experiment is that, usually, if the timestamp is matched accurately, a set of camera data should correspond to 10 sets of IMU data. Therefore, we randomly selected the data during 1 s and record the camera data and the IMU data. The detailed matching results are illustrated in Fig. 8. The comparison presented in Figs. 8 (a) and (b) implied that the matching algorithms has better accuracy. Therefore, it can be found that the proposed data matching method can improve the correctness of matching. This result was due to we consider the transmission delay of the sensor and use the hardware trigger to reduce the trigger delay.

Experiment 2. Accuracy of the proposed synchronization method for multiple sensors

This experiment aimed to demonstrate the accuracy of the proposed synchronization method. We compared our method with other typical methods, including the TriggerSync [15] and the full-software IEEE 1588[16]. TriggerSync is a typical method for obtaining timestamps based on the NTP protocol, and the full-software IEEE 1588 is a high-precision timestamp synchronization that mainly based on the PTP protocol. During the experiments, these three methods were used to trigger the same multiple sensors separately, and the result of

TABLE III TIMESTAMP SYNCHRONIZATION OFFSET

Timestamp synchronization	Time offset
software and hardware IEEE1588	<10us
full-software IEEE 1588	1ms
TriggerSync	<1ms

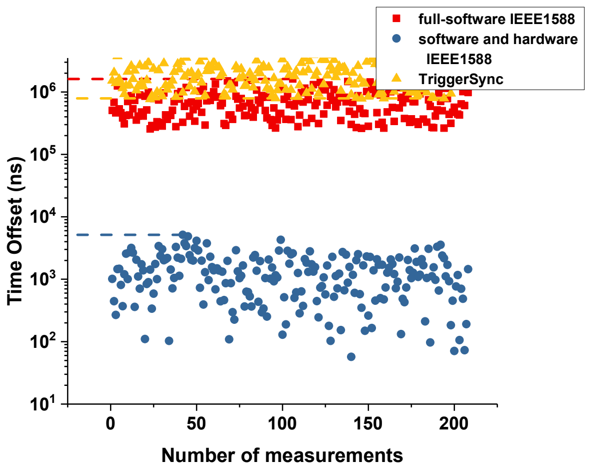


Fig. 9 Timestamp synchronization offset. The full-software IEEE 1588 and the software and hardware IEEE 1588 select the maximum offset as the synchronization accuracy. The TriggerSync selects the smallest delay as the synchronization precision.

the accuracy of the time synchronization for multiple sensors were record, as Table III and Fig. 9 showed. The result showed that the proposed time synchronization method has higher precision than TriggerSync and the full-software IEEE 1588. This result was due to the accuracy of the full-software IEEE 1588 is affected by the software transmission delay and the trigger delay. TriggerSync uses a one-way synchronization algorithm and ignores the effects of delay of the data transmission. The accuracy of the proposed synchronization method was benefited from the optimized timestamp acquisition location, the data matching methods and the triggering methods, so that the timestamp precision was determined at the microsecond level. The above experimental results prove the accuracy of the proposed time synchronization method for multiple sensors.

V. CONCLUSION

In this paper, a synchronization method for multiple triggered sensors with respect to a local clock was presented, thereby ensuring precisely synchronized timestamps in multiple sensors. The accuracy of the timestamp synchronization benefited from the proposed combination of hardware and software to obtain timestamps. Furthermore, the hardware synchronization trigger method addressed the problem of the trigger delay. The proposed data matching method significantly improved the accuracy of data matching. The precision of the proposed synchronization method was verified through various comparison experiments. The proposed precise time synchronization method can be applied in many fields, including robots, VR, and unmanned driving. However, the accuracy of the proposed synchronization method of timestamps is limited by the software latency, and the commercial computers as the master node cannot be used in mobile scenarios. To avoid this problem, we plan to use high-performance embedded processors as the master clock node to obtain a physical-layer timestamp in the future.

REFERENCES

[1] Y. Liu, F. Wang, W. Zhang, C. Zhang and Y. Xia, "Online Self-Calibration Initialization for Multi-Camera Visual-Inertial SLAM," 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), Kuala Lumpur, Malaysia, 2018, pp. 192-199.

[2] Zhang C, Liu Y, Wang F, Xia Y, Zhang W. VINS-MKF：A Tightly-Coupled Multi-Keyframe Visual-Inertial Odometry for Accurate and Robust State Estimation. Sensors (Basel). 2018 Nov 19;18(11):4036.

[3] Heng, L.; Lee, G.H.; Pollefeys, M. Self-calibration and visual SLAM with a multi-camera system on a micro aerial vehicle. Auton. Robots 2015, 39, 259–277.

[4] Z. Zhang, S. Liu, G. Tsai, H. Hu, C. Chu and F. Zheng, "PIRVS: An Advanced Visual-Inertial SLAM System with Flexible Sensor Fusion and Hardware Co-Design," 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, 2018, pp. 1-7.

[5] B. Chandrasekaran, S. Gangadhar and J. M. Conrad, "A survey of multisensor fusion techniques, architectures and methodologies," SoutheastCon 2017, Charlotte, NC, 2017, pp. 1-8.

[6] P. Cadell and B. Upcroft, "Distributed data acquisition unit with microsecond-accurate wireless clock synchronisation," 2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Melbourne, VIC, 2013, pp. 117-122.

[7] Ping Song, Xiaodong Shan, Kejie Li and Guangping Qi, "Highly precise time Synchronization Protocol for ZigBee networks," 2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Singapore, 2009, pp. 1254-1258.

[8] Mingjie Jia, Zhigang Sun, Qi Su and Hui Cheng, "Precise delay measurement based on NetMagic," 2011 International Conference on Multimedia Technology, Hangzhou, 2011, pp. 5017-5020.

[9] J. Nikolic et al., "A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM," 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, 2014, pp. 431-437.

[10] Z. Zhang, S. Liu, G. Tsai, H. Hu, C. Chu and F. Zheng, "PIRVS: An Advanced Visual-Inertial SLAM System with Flexible Sensor Fusion and Hardware Co-Design," 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, 2018, pp. 1-7.

[11] Y. Zhang, W. Xia, C. Li and Z. He, "Research and realization of high-precision GPS time transfer system," 2013 International Conference on Computational Problem-Solving (ICCP), Jiuzhai, 2013, pp. 334-337.

[12] P. Furgale, J. Rehder, and R. Siegwart, “Unified temporal and spatial calibration for multi-sensor systems,” 2013 IEEE/RSJ Int.Conf. Intell. Robot. Syst., pp. 1280–1286, Nov. 2013.

[13] M. Li and A. I. Mourikis, "3-D motion estimation and online temporal calibration for camera-IMU systems," 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, 2013, pp. 5709-5716.

[14] A. Harrison and P. Newman, "TICSync: Knowing when things happened," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011, pp. 356-363.

[15] A. English, P. Ross, D. Ball, B. Upcroft and P. Corke, "TriggerSync: A time synchronisation tool," 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, 2015, pp. 6220-6226.

[16] T. Kovácsházy and B. Ferencz, "Performance evaluation of PTPd, a IEEE 1588 implementation, on the x86 Linux platform for typical application scenarios," 2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings, Graz, 2012, pp. 2548-2552.

[17] IEEE Draft Standard Profile for Use of IEEE 1588 Precision Time Protocol in Power System Applications," in IEEE PC37.238/D6, June 2014, vol., no., pp.1-38, 1 Jan. 2014.

[18] Schmid K, Lutz P, Teodor Tomić, et al. Autonomous Vision-based Micro Air Vehicle for Indoor and Outdoor Navigation[J]. Journal of Field Robotics, 2014, 31(4):537-570.