2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Clock-based time synchronization for an event-based camera dataset acquisition platform*

Vitalijs Osadcuks[1], Mihails Pudzs[2], Andrejs Zujevs[3], Aldis Pecka[4] and Arturs Ardavs[5]

Abstract— **The Dynamic Visual Sensor is considered to be a next-generation vision sensor. Since event-based vision is in its early stage of development, a small number of datasets has been created during the last decade. Dataset creation is motivated by the need for real data from one or many sensors. Temporal accuracy of data in such datasets is crucially important since the events have high temporal resolution measured in microseconds and, during an algorithm evaluation task, such type of visual data is usually fused with data from other types of sensors. The main aim of our research is to achieve the most accurate possible time synchronization between an event camera, LIDAR, and ambient environment sensors during a session of data acquisition. All the mentioned sensors as well as a stereo and a monocular camera were installed on a mobile robotic platform. In this work, a time synchronization architecture and algorithm are proposed for time synchronization with an implementation example on a PIC32 microcontroller. The overall time synchronization approach is scalable for other sensors where there is a need for accurate time synchronization between many nodes. The evaluation results of the proposed solution are reported and discussed in the paper.**

I. Introduction

A. Event-based vision sensors and datasets

The Dynamic Visual Sensor (DVS) is a next-generation vision sensor, called event camera or "silicon retinas". The DVS is a biologically inspired alternative to conventional cameras, designed to overcome their limitations. The DVS imitates the operating principle of the retina. Instead of transmitting all the pixels from the image sensor, the DVS transmits only the pixels that undergo significant brightness changes (both for positive and negative magnitudes), called "events" [1], [2], [3]. DVSs are power-efficient, low-bandwidth devices that provide real-time performance. Events are time-stamped with μs resolution and transmitted asynchronously. DVSs have high temporal resolution and low latency, both expressed in μs. DVSs also provide high dynamic range - 140 dB, compared to 60 dB of conventional cameras [1].

The DVS, however, is not good at perceiving static scenes, which restricts its use in static scene analysis. This restriction is overcome by the asynchronous time-based image sensor (ATIS, [4]), where the DVS sensor is combined with an event-triggered PWM intensity readout of all the DVS pixels [2]. ATIS is the first visual sensor that combines the event-based concept and the frame-based concept, which is carried out by pulse-width-modulation (PWM) exposure measurement for imaging [4]. The second attempt to combine both concepts is presented in [2] and is called apsDVS (active pixel sensor combined with DVS) and later, in [5], named DAVIS (dynamic and active pixel sensor). This sensor allows to simultaneously output asynchronous events and synchronous image frames. The sensors mentioned above use Address-Event-Representation (AER) data encoding format, which has become the standard for communication between neuromorphic chips [6].

Dataset creation is motivated by the need for real data from one or many sensors or synthetic data for algorithm design and evaluation. The availability of datasets boosts research, provides a baseline, reduces time and cost of a project and allows researchers to focus on the main goals of their projects. Dataset exploitation, different benchmarks and quantifiable metrics in Computer Vision are the most used techniques in designing and benchmarking of a new algorithm, conducting a comparative study or providing quantitative evaluations [7]. Many requirements should be met to provide a dataset as a publicly available resource. A dataset has to be validated and well documented; accuracy and error level description as well as different technical and intrinsic and/or extrinsic parameters have to be reported [8].

Only few event-based vision datasets have been created during the last decade. These datasets provide synthetic and real data combined with data from other frame-based sensors. For example, the dataset described in [7] includes synthetic and real data from DAVIS240B and RGB-D (Microsoft Kinect Sensor) sensor mounted on a mobile robot (Pioner 3DX Mobile Robot); an indoor with ground-truth and outdoor without ground-truth DAVIS sensor datasets created in case of 6-DOF in [9]; indoor and outdoor datasets created with synchronized stereo DAVIS cameras and ground-truth provided by LIDAR, additionally used stereo-camera and data recorded on a hexacopter, on a motorcycle and on a car [10]; the first driving dataset has over 12 hours of records under various driving conditions, includes DAVIS and a car

*This research is funded by the Latvian Council of Science, Funding Number: lzp-2018/1-0482 and has been supported by the European Regional Development Fund within the Activity 1.1.1.2 "Post-doctoral Research Aid" of the Specific Aid Objective 1.1.1 (No.1.1.1.2/VIAA/2/18/334).

[1]Vitalijs Osadcuks, Ph.D., Asoc.Prof. at Latvia University of Life Sciences and Technologies, Latvia, vitalijs.osadcuks@llu.lv

[2]Mihails Pudzs, Ph.D. student and researcher at Riga Technical University, Latvia, mihails.pudzs@rtu.lv

[3]Andrejs Zujevs, Ph.D., postdoctoral researcher at Riga Technical University, Latvia, andrejs.zujevs@rtu.lv

[4]Aldis Pecka, Ph.D. student at Latvia University of Life Sciences and Technologies, Latvia, aldis.pecka@llu.lv

[5]Arturs Ardavs, M.Sc. student at Riga Technical University, Latvia, arturs.ardavs@rtu.lv

data [11]; other datasets were created and described in [12], [13]. Here it should also be noted that there have been attempts to create a DVS simulator, for example, in [14], [15].

B. Time synchronization issue

When acquiring a dataset with multiple sensors, time synchronization is an important issue. This issue becomes especially critical in the case of a high temporal resolution vision sensor, in contrast to other sensors, which are used for data fusion and providing ground-truth. Since there is currently no commercial and ready-to-use product to perform time synchronization between different types of sensors, custom solutions are often employed, ranging from not synchronizing sensors at all in [9], to performing synchronization just with the host computer's internal clock in [10] and [11]. These approaches are justified mainly because the accumulated time drift is small within the recording time (e.g. 2 ms/min for ≈ 1 minute recording in [9]) or because introduced timestamp errors are insignificant compared to the readout rate of the sensors (e.g. milliseconds versus ≈ 10 Hz readout in [11]). More precise timestamps are acquired in [12], because the data is provided only by the DAVIS camera with built-in IMU, which is hardware synchronized with microsecond precision. Papers [7] and [13] do not mention how synchronization was performed and what are the timestamp errors.

Typically, time alignment for data from different sensors is part of the dataset creation process (offline processing), where the alignment is done manually or semi-manually. The manual alignment is not useful or may take a lot of effort for long time period datasets and for datasets where more than two or three unsynchronized sensors are involved.

Without synchronization during data acquisition, timestamps can be aligned by means of post-processing. For example, in the paper [16], authors define and use a timestamp alignment method that is based on cross-correlation between the image total intensity of changes and the DVS event rate to find the time offset between different data streams. In a more recent paper [17], IMU and laser tracker movement measurements are aligned in the post-processing. However, these methods may fail in case of longer recordings where sensor clock speeds may vary significantly over time.

Our current research focuses on dataset creation from different sensors, including an event camera, LIDAR and ambient environment sensors, with high precision time alignment. The stereo and monocular cameras are also integrated and used. However, they are not hardware synchronized due to the unavailability of the synchronization inputs.

In this paper, we propose an architecture for the data acquisition sensor bundle that employs clock-based time synchronization and time offset estimation. It is a promising approach since it ensures that timestamps are orders of magnitude more precise than with existing methods. It also greatly simplifies or eliminates the post-processing timestamp alignment.

The description of the architecture is provided in Section II, the time synchronization algorithm is described in Section III, and the methodology and evaluation results are provided in Section IV.

II. Description of Architecture

The proposed sensor bundle comprises the following components (see Figure 1): an event camera (DAVIS; DAVIS240C), a LIDAR (Ouster OS-1), a custom designed Sensor Board (SB) for ambient environment sensing and time synchronization, Depth and RGB Camera (Stereo Camera, SC; Intel® RealSense™ D435i), and a Computing Module (CM; IB811F series computer). The SB is the essential time synchronization unit for the entire system. The CM is the data recording device. All other sensors and devices are connected to the CM for control and data transfer: the LIDAR connects via the 1 Gbit Ethernet port and uses TCP and UDP; the SB connects via the 100 Mbit Ethernet and uses UDP; the DAVIS connects via USB 2.0, whereas SC requires the higher bandwidth and uses the USB 3.0 port. CM and LIDAR connection is also used for time synchronization using Precision Time Protocol (PTP). This choice was inspired by the method from [18], where authors synchronized a high-speed vision network equipped with 500-fps cameras.

The internal (counted) time of the CM is the main source of time for the entire system. The CM and the LIDAR are synchronized using PTP. PTP-synchronized LIDAR outputs 1 Hz clock for the SB. Using the algorithm from Section III, the SB adjusts its internal time and converts the input 1 Hz clock from the LIDAR to 10 kHz clock, which is used to synchronize the DAVIS with the remaining components. Using PTP and clock signals all the nodes (excluding the SC) are synchronized with the CM.

However, clock-based synchronization only ensures that the time is counted by the CM, LIDAR, SB and DAVIS with the same speed. There is still an offset between counted times, because the modules were initialized differently. We estimate timestamp offsets by recording CM, SB and DAVIS timestamps of the GPIO trigger event, generated by the CM.

The SB is based on PIC32MX series 32-bit microcontroller running at 80 MHz both on the CPU and the periphery. The clock is formed using an 8 MHz external crystal (20 ppm tolerance, 30 ppm stability) and an internal oscillator with PLL. A 32-bit system counter with 25 ns tick is used as a continuous timer and a 16-bit hardware timer TMR2 with 12.5 ns tick - as a period timer with associated interrupts. Additionally, the SB runs a serial console and TCP stack with a UDP server and a client for monitoring and debugging purposes. All this functionality is operated with lower priority and does not affect the synchronization process. The CM (IB811F-420) is based on Intel® Pentium® N4200@2.5GHz CPU and has 2x Intel® I210IT PCI-E Gigabit LAN network device, which supports PTP time synchronization.

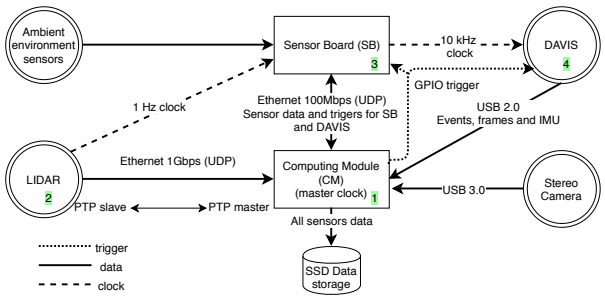


Fig. 1: The overall system architecture.

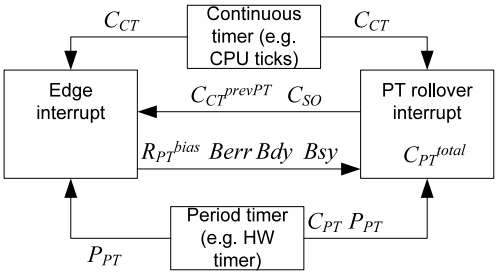


Fig. 2: Synchronization algorithm data flow

III. Time synchronization algorithm

The proposed algorithm allows to generate continuous output pulse signal (SO) and maintain the tick counter in sync with the input signal (SI). Synchronization is performed with respect to the selected edge of the SI. It is assumed that SI frequency is lower than that of the SO. Variations of SI period should be smaller than that of the SO period for synchronization to be successful. Otherwise, SO pulses will not fit into SI period and synchronization will be lost. Nominal frequency values for SI and SO must be divisible without remainder, i.e. output signal periods should fit into the input period. The tick counter value can be used for precise event timestamping. In the present implementation, the input frequency is lower than the output frequency but the algorithm can be redesigned to work conversely. The algorithm is implemented in C language, uses only 32-bit integer math (64 bit for timestamp) and one external library for the PID algorithm[1]. Therefore, the algorithm can be performed by low-cost microcontrollers with interrupt support. The nomenclature of the variables is given in Table I.

TABLE I: Description of algorithm variables

Symbol	Description
SI	Synchronization Input singal
SO	Synchronized Output signal
CT	Continuous Timer
PT	Period Timer
T_{sync}	Synchronization period i.e. time between a given number of SI edges
nom	Nominal value
$total$	Total count during entire sync process
D	Delay between given signal edges
E	Error
C	Count of given signal edges or timer ticks
R	Correction value
Scl	Scale between two values (1st divided by 2nd)
$Berr, Be2, Bdy, Bsy$	Variables of Bresenham's algorithm

Two hardware timers are used for synchronization purposes: a continuously running timer (CT) and a periodic timer with an adjustable period (PT), see Fig. 2. The algorithm also implies the use of two hardware interrupts: one external for SI and one internal for PT rollover. CT is used to count ticks between two SI edges and calculate synchronization error, whereas PT serves as the base for SO period and, when adjusted, helps to maintain SO synchronization to SI edges. PT also accumulates the total tick count of the synchronized time C_{PT}^{total}. SO pulses can be generated either by a hardware module associated with PT or in software during its rollover event. The pseudo code of the synchronization algorithm for both interrupts is given in Pseudo code listings 1 and 2.

Algorithm 1 PT rollover interrupt

INPUT: C_{CT}
$C_{PT}^{total} \leftarrow C_{PT}^{total} + P_{PT} + 1$
$C_{CT}^{prevPT} \leftarrow C_{CT}$
$R \leftarrow P_{PT}^{nom}$
$Be2 \leftarrow Berr$
if $Be2 > -C_{SO}^{nom}$ **then**
  $Berr \leftarrow Berr - Bdy$
end if
if $Be2 < Bdy$ **then**
  $Berr \leftarrow Berr + C_{SO}^{nom}$
  $R \leftarrow R + Bsy$
end if
$P_{PT} \leftarrow R + R_{PT}^{bias}$
$C_{SO} \leftarrow C_{SO} + 1$

The main novelty of the proposed algorithm is the use of Bresenham's line drawing algorithm, which was originally intended to efficiently distribute pixels between two points in a 2D space [19]. In our adaptation, both coordinates of the first point are zeroes, but for the second point, X coordinate is the total count of SO pulses during $Tsync$, and Y coordinate is a correction value to compensate for the time delay between SI and SO. The correction value is calculated from CT tick counter during the previous T_{sync} period. The idea is shown in a more general way in Fig. 3 (a), (b) and with an example (c), (d).

Also, the algorithm makes use of PID regulator for fine adjustment of time delay; regulator output is added to the calculated correction value. SO pulses are generated by

[1] https://github.com/mike-matera/FastPID

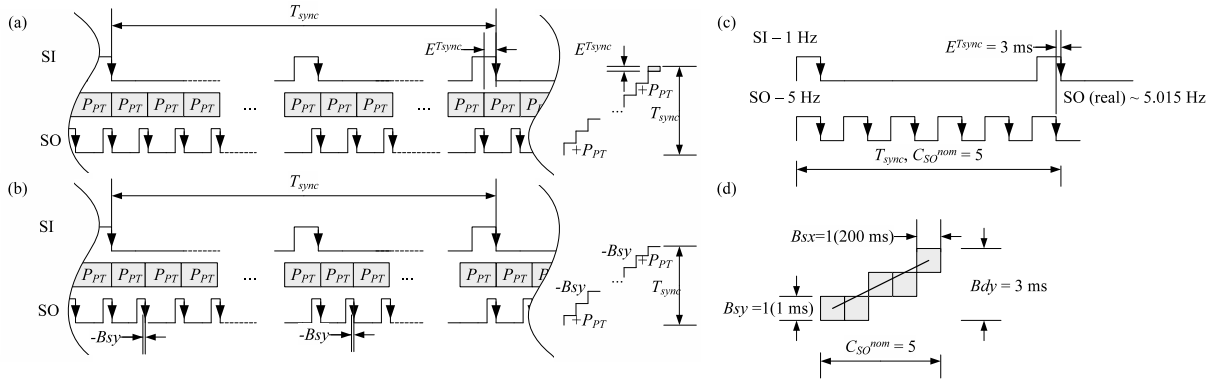


Fig. 3: Timer speed adjustment operation: (a) unadjusted timer operation with CPU faster than the nominal; (b) operation after PT period adjustment; (c) an example with 1 Hz synchronization input signal and output signal, CPU is faster and 5 output pulses are generated in 4997ms (d) PT adjustment according to Bresenham's line drawing algorithm - 3ms error is evenly distributed among all SO periods.

MCU's hardware Output Compare module with PT as its timebase. Total tick count C_{PT}^{total} is accumulated from PT and can be used as precision timestamp. The time starts along with the algorithm and it can be considered valid and running at the same speed as the input synchronization signal. When using the described synchronization approach, one should ensure that the input clock period stability is better than that of the output clock period. The algorithm can correct the amount of error less than one PT period and, in case of such failure, the necessary PT period count between SI edges will change and synchronization will be lost.

IV. Evaluation

A. Methodology

The evaluation of the synchronization approach was conducted in two steps. First, lab function generators for sync and test event signals were used to test how the synchronization algorithm works with calibrated signals. Then, the CM, SB, LIDAR, and the DAVIS were connected pairwise to evaluate how the proposed synchronization approach performs with real signals.

An experimental setup was created for the evaluation of the SB syncing and timestamping performance with calibrated signals. The SB adjusts its 10 kHz output clock to 1 Hz (50 ppm precision) coming from MSO-X2024A wave generator output. The scope also measures both signals. The additional AFG-2005 function generator sends rising edge events at 10 Hz; these are timestamped and the time values are sent via UDP connection to PC.

To evaluate how the proposed time synchronization approach performs with real signals, we compare internal times of all modules that count time and assign timestamps to the recorded data and trigger events. The τ symbol is used for internal counted times. There are four counted times: τ_{DAVIS} for the DAVIS, τ_{SB} for the SB, τ_{CM} for the CM and τ_{LIDAR} for the LIDAR.

We compare counted times pairwise. Equation (1) demonstrates how two counted times τ_1 and τ_2 are related mathematically with τ_1 taken as the baseline and τ_2 evaluated against it.

$$\tau_2(\tau_1) = \int_0^{\tau_1} s_{2,1}(\tau_1) \cdot d\tau_1 + \tau_2(0). \quad (1)$$

The value of $s_{2,1}(\tau_1)$ tells how fast τ_2 is counted with the passage of τ_1 and is measured in τ_2 seconds per τ_1 second, and $\tau_2(0)$ shows what was the value of τ_2 when τ_1 started counting. Any deviation of $s_{2,1}(\tau_1)$ from the value of 1 means time drift and, therefore, the value of

$$\Delta s_{2,1} = s_{2,1}(\tau_1) - 1 \quad (2)$$

shows extra or missing τ_2 seconds per τ_1 second.

Figure 4(a) demonstrates simplified time counting model for two counted times — τ_1 (baseline) and τ_2. To align data recorded during the data acquisition interval (here — between τ_S and τ_E), $s_{2,1}(\tau_1) - 1$ should be as close as possible to 0, and a trigger event must be captured by both modules (i.e. τ_T and $\tau_2(\tau_T)$ recorded).

In our system, there are four counted times. Because the main data is provided by the DAVIS sensor, and other sensors provide auxiliary data for comparison/validation, the τ_{DAVIS} is the baseline, and time drifts are estimated for other counted times — τ_{SB}, τ_{LIDAR} and τ_{CM}. Additionally, trigger event is recorded by $\tau_{DAVIS-T}$, and the other sensors — $\tau_{SB}(\tau_{DAVIS-T})$ and $\tau_{CM}(\tau_{DAVIS-T})$.

Time drifts. To estimate time drifts experimentally, we use the FPGA-based time counter shown in Fig. 4(b). The FPGA is clocked by the 50 MHz master clock. There are two inputs for low frequency and high-frequency clocks – LO and HI. The drift of the master clock is not important as long as its frequency is considerably higher than the frequency of LO and HI clocks and it is only used for sampling LO and HI signal, internal logic and communication (upload of acquired data). Figure 4(c) illustrates the counting principle — for every period of the LO clock the number of HI clock cycles is recorded (a) for LO being high as *A*, (b) for LO being low

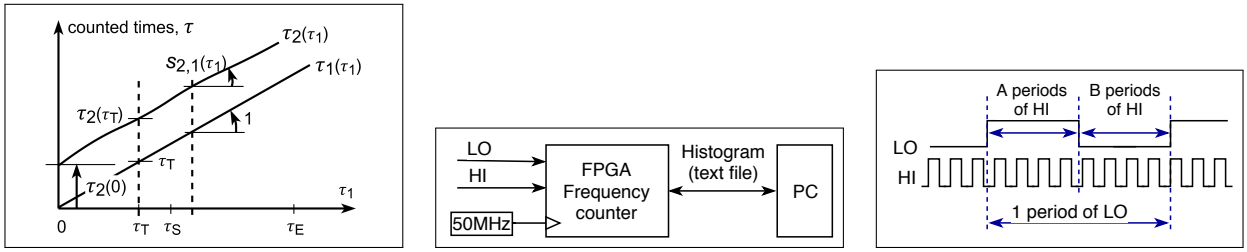


Fig. 4: Evaluation of time counting: (a) simplified time counting model, (b) clock evaluation setup, (c) counted clock cycles

Algorithm 2 SI edge interrupt

INPUT: C_{CT}
  $C_{CT}^{elapsed} \leftarrow C_{CT} + C_{CT}^{MAX} - C_{CT}^{prev} + 1$
  $C_{CT}^{prev} \leftarrow C_{CT}$
  $C_{CT}^{T_{sync}} \leftarrow C_{CT}^{T_{sync}} + C_{CT}^{elapsed}$
  $C_{SI} \leftarrow C_{SI} + 1$
  if $C_{SI} = C_{SI}^{nom}$ **then**
    $D_{SI..SO} \leftarrow C_{CT} + C_{CT}^{MAX} - C_{CT}^{prevPT} + 1$
    $D_{SI..SO} \leftarrow D_{SI..SO} - P_{PT}/Scl_{PT..CT}$
    $E^{T_{sync}} \leftarrow (C_{SI}^{nom} \times C_{SO}^{nom} - C_{SO}) \times P_{PT}^{nom}/Scl_{PT..CT}$
    $E^{T_{sync}} \leftarrow \text{abs}(E^{T_{sync}}) - D_{SI..SO}$
    if *StartSync* = true **then**
      $StartSync \leftarrow$ false
      $P_{PT} \leftarrow P_{PT}^{nom}$
      $C_{PT} \leftarrow C_{PT}^{total} \leftarrow 0$
      InitPID()
    end if
    $R_{PT}^{total} \leftarrow C_{CT}^{T_{sync}} \times Scl_{PT..CT} - C_{SI}^{T_{sync}} \times Scl_{PT..SI} - C_{SO}^{nom}$
    if $C_{SO} = C_{SO}^{nom}$ **then**
      $RegOut \leftarrow \text{StepPID}(E^{T_{sync}})$
    else
      ClearPID()
      $RegOut \leftarrow 0$
    end if
    $R_{PT}^{total} \leftarrow R_{PT}^{total} + RegOut$
    $R_{PT}^{bias} \leftarrow R_{PT}^{total}/C_{SO}^{nom}$
    $R_{PT} \leftarrow \text{mod}(R_{PT}^{total}, C_{SO}^{nom})$
    $Bdy \leftarrow \text{abs}(R_{PT})$
    $Bsy \leftarrow \text{sign}(R_{PT})$
    if $C_{SO}^{nom} > Bdy$ **then**
      $Berr \leftarrow C_{SO}^{nom}/2$
    else
      $Berr \leftarrow -Bdy/2$
    end if
    $C_{SI} \leftarrow C_{SO} \leftarrow C_{CT}^{T_{sync}} \leftarrow 0$
  end if

as B, and (c) for the whole period of LO as $A+B$. The data are accumulated as three separate histograms.

When the LO clock is compared to the baseline HI clock, the time drift $\Delta s_{LO,HI}$ for each bin of the histogram is calculated using

$$\Delta s_{LO,HI} = \frac{f_{HI}}{f_{LO} \cdot (A+B)} - 1 \tag{3}$$

In our system, LO is the LIDAR 1 Hz clock and baseline HI is the SB 10 kHz clock, so the evaluated amount is $\Delta s_{LIDAR,SB}$. It is the number of extra or missing seconds that LIDAR counts per second of DAVIS time. $\Delta s_{LIDAR,SB}$ is evaluated within every period of LIDAR clock which is 1 second. Therefore, the standard deviation of the histogram shows the local (for every second) time drift of the LIDAR clock against the SB clock. Please note that the local time drift is not the same as the accumulated or maximum time drift. The expected value of the histogram shows the mean time drift of LIDAR clock against the SB clock.

DAVIS clock. We found that the synchronization of the DAVIS sensor works well (no resets of its internal clock) if the synchronization clock of 10kHz has an error no greater than 1%. To be sure of that, we evaluate the clock generated by SB against the calibrated 4 MHz clock with an accuracy of 50 ppm provided by the Keysight DSO-X 3034A oscilloscope generator[2].

When comparing 10 kHz LO clock of the SB against the calibrated 4 MHz HI clock, we also evaluate errors of duty ratio and period using:

$$d_A = \frac{A}{f_{HI}} \qquad d_B = \frac{B}{f_{HI}} \qquad d = \frac{A+B}{f_{HI}}. \tag{4}$$

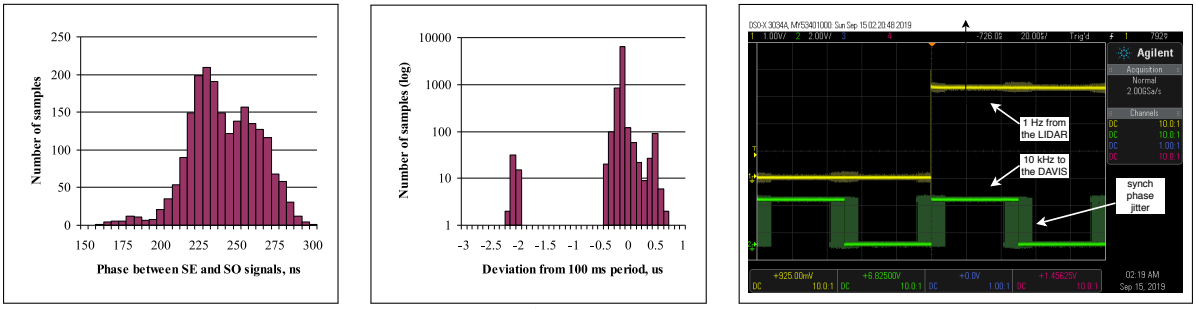
It is expected for the mean values $\hat{d_A}$, $\hat{d_B}$ and $\hat{d}$ to be as close as possible to $50\mu s$, $50\mu s$ and $100\mu s$ respectively, with no more than 1% of error.

B. Results

Synchronization performance of the SB running in controlled conditions is acceptable since it can stay synced and timestamp sensor measurements with time precision higher than that of the DAVIS and the LIDAR sensor event resolution.

The scope screen with infinite persistence in Figure 5(c) shows the phase jitter of the SB's output signal against the input synchronization signal during 5 min of operation. None

[2]https://literature.cdn.keysight.com/litweb/pdf/5990-6619EN.pdf?id=2002858



(a) (b) (c)

Fig. 5: Time synchronization histograms: (a) histogram of phase lag between synchronization input (SE) and synced output signal (SO); (b) timestamp value histogram. Oscilloscope screen (c) - synchronization phase jitter during 5 min of operation for synchronization input and synced output signal.

of the SO rising edges was outside of 100 ns interval, this time corresponds to 8 processor cycles. This jitter maintains a relatively constant phase lag of approximately 230 ns, which is introduced by CPU's interrupt latency. As the time lag is constant, it does not affect the synchronization process. Measurements were made using quality synchronization signal in laboratory conditions from 50 ppm source. After start-up, synchronization is achieved in approximately 20 s, however this time can be improved by tuning PID coefficients or using a better regulator.

The results of the 10 Hz edge event timestamping test are shown on the histogram in Figure 5(b). In this test, the SB timestamped events at nominal 100 ms periods. Note that the real measured value using a frequency counter was 99.99 ms due to the inaccuracy of the function generator. The bins represent the difference from the nominal 100 ms value in microseconds. The negative values mean that a longer time period was measured. The histogram shows two spikes because the timestamping of some events was interrupted by the synchronization algorithm. However, in all cases, timestamp error was within 4 us interval.

Counted times pairwise. The DAVIS assigns timestamps for trigger events with precision of 1 μs. We tested this by recording trigger events with the DAVIS sensor, while it was synchronized to the 10 kHz external clock. Even when the synchronization clock was purposefully different from 10 kHz (within the 1% of error), DAVIS still relied on it very accurately — over tens of minutes long time periods between triggers a difference in recorded timestamps was accurate to 1 μs with respect to the synchronization clock (meaning — *as skewed in time as the synchronization clock itself*). Therefore, the rate of counting for τ_{SB} and τ_{DAVIS} has less than ± 1 μs of absolute error. The average value of $\Delta s_{LIDAR,SB} = -5.48 \cdot 10^{-8}$, meaning average time drift of the LIDAR against the SB of -197.3 μs per hour. Local time drift of LIDAR clock against the SB clock per every second evaluated as standard deviation of $\Delta s_{LIDAR,SB}$ was 27.39 μs. Standard deviation of error between τ_{CM} and τ_{LIDAR} synchronized via PTP was 143 μs.

Trigger registration. GPIO trigger generated by the CM has timestamp error of ± 25 μs and is registered by the SB with error less than 4 μs, and by the DAVIS sensors with less than 1 μs error.

DAVIS clock. The SB internal clock τ_{SB} compared to the calibrated 4 MHz clock from the oscilloscope showed average period $\hat{d} = 100.018$ μs ($\hat{d_A} = 50.01$ μs and $\hat{d_B} = 50.008$ μs). Standard deviation of d_A, d_B and d was 0.054 μs, 0.047 μs and 0.068 μs respectively. The overall error did not exceed our goal of 1%.

V. Conclusions

The proposed approach, with minimal hardware additions, allows to achieve synchronization in the range of microseconds during data acquisition. This approach can increase data set capture quality in case of drifting clocks and eliminate necessity of synchronization in data post-processing, in which IMU data is often used, and this implies accuracies in the range of tens of miliseconds. The proposed time synchronization algorithm can be used to integrate microcontroller-based units into time-critical applications like event-based vision, IMU, range sensor, and other sensor systems. The proposed time synchronization solution will be used in future work in order to create a new Event-Camera dataset using a mobile robot platform in a variety of agricultural environments.

Issues. Time of the stereo camera was not synchronized within the overall solution due to the unavailablity of the input signal for synchronization. This imperfection may be solved by measuring time offsets between the central computing module and the stereo camera, where frames of data from the stereo camera are timestamped. Although our CM's NIC supports hardware PTP mode, unfortunately, by this time we only managed to run software-based PTP. We believe it is the reason why PTP synchronization error was over 50 μs stated in the technical specification of the used LIDAR.

With software-based PTP, period of LIDAR's output is very unstable and the synchronization with the SB is too frequently lost. Therefore, numbers provided in the section IV-B are without the PTP sync.

REFERENCES

[1] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128 x128 120 dB 15 us Latency Asynchronous Temporal Contrast Vision Sensor," *Solid State Circuit*, vol. 43, no. 2, pp. 566–576, 2008.

[2] R. Berner, C. Brandli, M. Yang, and S. C. Liu, "A 240-by-180 10mW 12us latency sparse-output vision sensor for mobile applications," *VLSI Circuits (VLSIC), 2013 Symp.*, pp. 186–187, 2013. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6578656%5Cnpapers2://publication/uuid/3AF9D27E-5313-4407-9CA4-354081F5C082

[3] P. Lichtsteiner and T. Delbruck, "A 64 x 64 AER logarithmic temporal derivative silicon retina," *2005 PhD Res. Microelectron. Electron. - Proceedingsof Conf.*, vol. II, no. 1, pp. 406–409, 2005.

[4] C. Posch, D. Matolin, and R. Wohlgenannt, "A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, 2011.

[5] C. Brandli, R. Berner, M. Yang, S. C. Liu, and T. Delbruck, "A 240 x 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[6] K. A. Boahen, "Point-to-Point Connectivity Between Neuromorphic Chips Using Address Events," vol. 47, no. 5, pp. 416–434, 2000.

[7] F. Barranco, C. Fermuller, Y. Aloimonos, and T. Delbruck, "A dataset for visual navigation with neuromorphic methods," *Frontiers in Neuroscience*, vol. 10, no. FEB, pp. 1–9, 2016.

[8] T. Delbrück, B. L. Barranco, E. Culurciello, and C. Posch, "Activity-Driven , Event-Based Vision Sensors," *Circuits and Systems (ISCAS) Proceedings of 2010 IEEE International Symposium on*, pp. 2426–2429, 2010.

[9] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM," *Int. J. Rob. Res.*, vol. 36, no. 2, pp. 142–149, feb 2017. [Online]. Available: http://arxiv.org/abs/1610.08336http://dx.doi.org/10.1177/0278364917691115http://journals.sagepub.com/doi/10.1177/0278364917691115

[10] A. Z. Zhu, D. Thakur, T. Ozaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The Multi Vehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3D Perception," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2032–2039, 2018. [Online]. Available: http://arxiv.org/abs/1801.10202%0Ahttp://dx.doi.org/10.1109/LRA.2018.2800793

[11] J. Binas, D. Neil, S.-C. Liu, and T. Delbruck, "DDD17: End-To-End DAVIS Driving Dataset," pp. 1–9, 2017. [Online]. Available: http://arxiv.org/abs/1711.01458

[12] B. Rueckauer and T. Delbruck, "Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor," *Frontiers in Neuroscience*, vol. 10, no. APR, 2016.

[13] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt, "Event-based 3D SLAM with a depth-augmented dynamic vision sensor," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 359–364, 2014.

[14] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 142–149, 2 2017. [Online]. Available: http://arxiv.org/abs/1610.08336http://dx.doi.org/10.1177/0278364917691115http://journals.sagepub.com/doi/10.1177/0278364917691115

[15] H. Rebecq, D. Gehrig, and D. Scaramuzza, "Esim: an open event camera simulator," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 969–982. [Online]. Available: http://proceedings.mlr.press/v87/rebecq18a.html

[16] A. Censi and D. Scaramuzza, "Low-latency event-based visual odometry," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 703–710, 2014.

[17] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? the UZH-FPV drone racing dataset," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 6713–6719, 2019.

[18] A. Noda, S. Tabata, M. Ishikawa, and Y. Yamakawa, "Synchronized High-Speed Vision Sensor Network for Expansion of Field of View," *Sensors*, vol. 18, no. 4, p. 1276, apr 2018. [Online]. Available: http://www.mdpi.com/1424-8220/18/4/1276

[19] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965. [Online]. Available: http://ieeexplore.ieee.org/document/5388473/