Time Synchronization between SOKUIKI Sensor and Host Computer using Timestamps

Alexander Carballo[†], Yoshitaka Hara[†⋆], Hirohiko Kawata[†],
Tomoaki Yoshida[‡], Akihisa Ohya[†] and Shin'ichi Yuta[†]

***Abstract*— Time is crucial in applications such as sensor data fusion, autonomous mobility and SLAM. However clocks at end systems are rarely synchronized and often running at different speeds. Lack of synchronization therefore reduces the accuracy of sensor readings. The SOKUIKI scanning laser range finder allows acquiring time values by timestamping range readings. Our work consists in a method for time synchronization, with clock skew estimation, between a laser range finder sensor and a host computer.**

I. Introduction

Keeping an accurate notion of time is a very important requirement in several areas like machine control, navigation, distributed computing, communications, etc. In the case of robot control, task planning depends in knowing the specific time required to complete a given step. In the case of robot navigation, the robot uses sensor's readings to detect landmarks in the environment to better position itself, possibly integrating data from different sensors in a fusion step. Here the readings from sensors must be synchronized with the robot's on-board time, if not readings from the future or past may mislead the robot into a different direction or into missing the destination point.

Time synchronization is possible if both sending and receiving systems have access to some time source, as it is the case of host computers, sensors, however, are rarely provided with a time reference. That is the case for most laser range finders (LRF) and ultrasonic sensors, they send distance information with no time source. Then the host computer controlling the robot must assume that acquired data is consistent with the current time slot and proceed with motion commands, but if the acquired data corresponds to a previous time slot then the robot may collide with an obstacle not detected in time.

This form of synchronization must consider data acquisition and processing time in the sensor side, data transmission time and data extraction and processing time in the computer side, in order to determine the possible control response to sensor events. However these timing values differ from sensor to sensor and to synchronize several sensors simultaneously could be difficult.

If the sensor is equipped with some local time reference, then its time value can be sent back to the host computer as *timestamp* within sensor data. This *timestamp* can be compared with the local clock on the host computer to validate *freshness* of readings. If timestamp value corresponds to some instant Ts_i while computer's time value is Th_i, if the sensor's time satisfies $Th_i - \Delta t < Ts_i < Th_i + \Delta t$, for a given Δt, then the range scan is considered fresh, otherwise it can be discarded.

The above is valid if both time sources, Ts_i and Th_i are mutually synchronized, in other words each system's clock value reflects the same instant in true time ($|Ts_i - Th_i| \to 0$). Due to finite speed of data signals and processing speed, perfect *true-time* synchronization is impossible [1].

Also the frequency of clocks in two different systems may also differ due to oscillator or resonator properties, temperature, etc., meaning that one clock could run a little faster when compared to another clock, introducing *clock skew* and even *clock drifting* problems. However current clock synchronization algorithms, in modern computer networks and distributed systems, achieve reasonable results with precisions in the order of microseconds, enough for most robot control applications.

The Hokuyo *URG-04LX SOKUIKI* scanning laser range finder sensor [2] (figure 1, SOKUIKI sensor in the rest of the paper) is provided with an internal time source and, since its *SCIPver2.0* specification [3], the current 24-bits time value is stamped on every range scan sent back to the host computer, so synchronization with host computer can be achieved while the sensor is providing range readings.

Fig. 1: Hokuyo *URG-04LX* "SOKUIKI" sensor.

The present work proposes a simple yet effective synchronization method between a host computer and a LRF sensor by using logical clocks in the host side and the timestamp value provided by the sensor. The synchronization method uses a simple clock skew elimination technique, to remove errors accountable to relative differences in frequency of the clocks, including an estimation of the drifting rates.

[†]Intelligent Robot Laboratory, University of Tsukuba, Japan. {acs,har,kwt,ohya,yuta}@roboken.esys.tsukuba.ac.jp

[‡]Future Robotics Technology Center, Chiba Institute of Technology, Japan. yoshida@furo.org

[⋆]Mr. Hara is now affiliated with Hitachi, Ltd.

The rest of the paper is organized as follows. Section II briefly reviews related work, then section III provides a theoretical background of the related subjects in this research. Section IV provides a description of the methodology used in solving this problem. Section V presents the achieved results. And finally, conclusions and future work are left for the last section.

II. Previous Work

We consider the problem of host computer and laser range finder synchronization under the same light and with the same considerations as computer networks or distributed process synchronization.

A pioneer study on time and causal relations among events on a distributed system is attributed to Lamport [1]. Lamport's work considered both a strong clock condition using physical true-time clocks for the ordering of events and a more relaxed clock condition using logical clocks or timestamps.

Perhaps the most known and used method for computer time synchronization is due to David Mills' Network Time Protocol (*NTP*)[4]. Four timestamp values are exchanged, then the client computes the *time offset* θ and the client also selects the best server among a group. Our work uses partially the NTP time offset calculation procedure.

Moon *et al.* [5] proposed three properties skew estimation algorithms must have, including complexity, robustness and non-negative delays, an a linear programming algorithm was proposed to find the closest line under the data points of the time vs delay relationship. Zhang *et al.* [6] found under estimations of clock skew in [5] and they proposed using a convex hull method, addressing also the problem of clock resets. Khlifi [7] combined the convex hull approach with a sliding window algorithm for on-line clock skew estimation and removal. Bi *et al.* [8] considered not only the clock skew problem but also the clock drifting in their piecewise reliable skew estimation approach. And in different direction, Cristian and Fetzer [9] proposed a probabilistic method, achieving also a good response to skew and drifting of clocks.

These works achieved very good results in estimation and cancellation of skew and drifting in network traces. Our approach instead is focused towards online host-sensor synchronization.

Kawata *et al.* [10] describe the capabilities of the SOKUIKI sensor and an application of timestamps for map construction is mentioned. The problem of synchronization between a host computer and a rotating SOKUIKI sensor was studied by Ueda *et al.* in [11]. Their approach used the rectangular synchronization signal sent by the sensor at the end of every scan, using a dedicated hardware. Our method is based on logical clocks, so only the sensor's internal time value sent as timestamp with every scan.

III. Background

In this section we introduce the problem of clock skew and basic terminology around the problem, following [5] borrowing Lamport's notation[1].

A. Properties of clocks

A *clock* $\mathcal{C}_t$ is a continuous function providing an approximated value of the *true time* t, $\mathcal{C}_t(t) = \mathtt{t}$. Hardware clocks and logical clocks are examples of such approximation function.

Let a clock $\mathcal{C}_t$ such that $\mathcal{C}_t^{'}(t) = \partial\mathcal{C}_t(t)/\partial t$ and $\mathcal{C}_t^{''}(t) = \partial^2\mathcal{C}_t(t)/\partial t^2$ exists, and $\mathcal{C}_s$ and $\mathcal{C}_r$ be clocks on a sender and receiving systems.

- **Delay** (δ) is the amount of time that an event takes, $\mathcal{C}_t(t_{end}) - \mathcal{C}_t(t_{start}) = \delta$ for some clock $\mathcal{C}_t$.
- **Offset** (θ) is the difference in time of the two clocks $\mathcal{C}_s(t) - \mathcal{C}_r(t)$. If both $\mathcal{C}_s$ and $\mathcal{C}_r$ are perfect clocks then the local system can know the actual time on the remote system by simply $\mathcal{C}_r + \theta = \mathcal{C}_s$.
- **Skew** (α) is the difference in the frequency domain of the clock and a perfect clock, for $\mathcal{C}_s$ and $\mathcal{C}_r$ this is $\mathcal{C}_s^{'}(t) - \mathcal{C}_r^{'}(t)$.
- **Drift** (ρ) refers to the fact that some clocks do not run at the right speed when compared with others. Drifting of clocks differs depending on their quality, temperature, the power drained from battery, etc. The drift of $\mathcal{C}_s$ relative to $\mathcal{C}_r$ is $\mathcal{C}_s^{''}(t) - \mathcal{C}_r^{''}(t)$ and it is a parameter provided by the manufacturer.

B. The Clock Skew problem

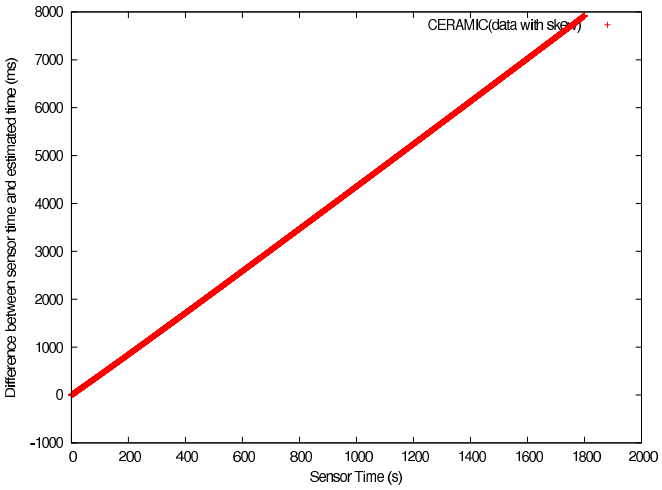
As mentioned above, two ideal clocks $\mathcal{C}_s(t)$ and $\mathcal{C}_r(t)$ may differ up to some offset θ, so knowing this time difference will suffice to estimate the time of the remote clock by simple addition. Let $\widetilde{\mathcal{C}_s(t)} = \mathcal{C}_r + \theta$ be the estimated clock of the remote system from the local system perspective. A plot of the true $\mathcal{C}_s(t)$ vs the difference of the true value and the estimated value, $(\mathcal{C}_s(t) - \widetilde{\mathcal{C}_s(t)})$, should be equal to the horizontal line $y = \theta$.

However real systems are far different. The plot of figure 2 was obtained from a SOKUIKI laser range finder with $10Hz$ scanning rate, provided with a ceramic resonator clock. Here the horizontal axis corresponds to the "true" clock value $\mathcal{C}_s(t)$ and the vertical is the $(\mathcal{C}_s(t) - \widetilde{\mathcal{C}_s(t)})$ difference for this sensor, during 30 minutes continuous scan (figure 2a for the $30min$ period and 2b a enhancement of the first 20 seconds).

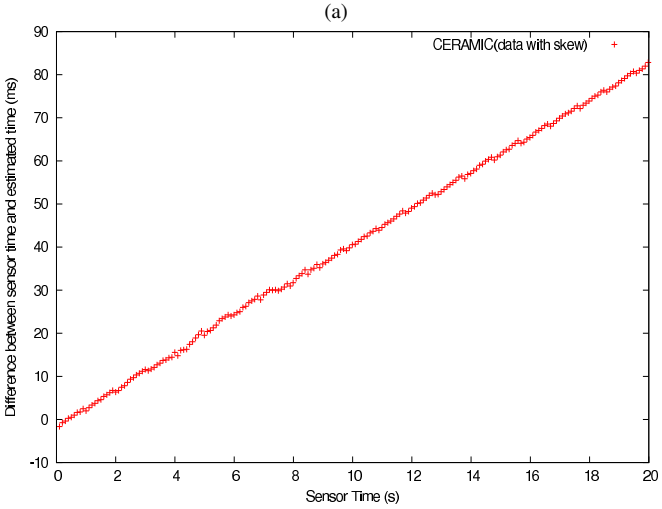
A constantly increasing behaviour is observed in the figures, very different from the expected horizontal line $y = \theta$. It is also worth to mention that the ascending line is not smooth (fig. 2b), several disturbances may be seen in the plot. These fluctuations have several sources, most of them due to drifting of clocks and due to processing in the receiving side. In the later case, while the receiving process is reading time data from the sender, the local operating system may switch to some other tasks. Reading from the remote system and storing data in local files involves using the input and output system which adds processing costs.

IV. Time Synchronization

The goal of this research is to achieve time synchronization between a host computer and an SOKUIKI sensor using timestamps from the later. There is no physical clock update



(a)



(b)

Fig. 2: Sensor Time vs Difference, (a) 30 minutes test and (b) zoom of the first 20 seconds, horizontal axis is the sensor time (seconds), vertical is the mentioned difference (in milliseconds).

function in the SOKUIKI sensor, it provides the current clock counter value since power up. So the neither the sensor nor the computer physical clocks are updated, instead a logical clock, a function of the local system's hardware clock value, is kept. In this section the method used for synchronization with skew cancellation is explained.

A. Estimation of clock offset

As explained above, synchronization involves computing the time offset θ, the relative skew information α and the drifting rate ρ. A physical clock provides an approximation of a true time function bounded by a drifting rate, usually provided by the manufacturer. In this sense, the sensor time value is limited by a drift value of ρ_s so we set the sensor time as $C_s(t)(1-\rho_s)$, the host clock is bounded by ρ_h, so it's set accordingly.

Reading the remote system clock value might require several time variables. In the case of *NTP*[4] four clock values (timestamps) are used, as depicted in figure 3: two from the client when sending to the server the time request command and receiving the answer ($C_h(t_1)$ and $C_h(t_4)$ respectively) and two from the server when receiving the request and when sending back the answer ($C_s(t_2)$ and $C_s(t_3)$ respectively). In our case only one timestamp value is available from the server (the sensor), $C_s(t_3) = C_s(t_2) = \textit{read timestamp}$. We compute the offset value as $\theta = C_s(t_2) - \frac{(C_h(t_4) - C_h(t_1))}{2}$, since only one time value is provided by the sensor.

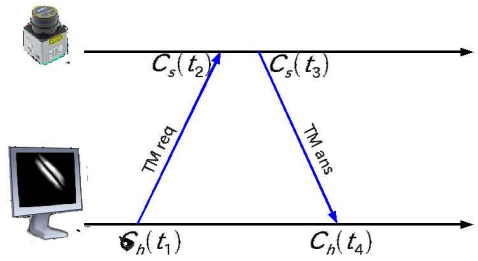


Fig. 3: NTP remote clock reading method, a host computer requesting the sensor's time.

B. Synchronization with clock skew cancellation

From the stated in III-B and specially considering figure in 2a, it is tempting to visualize the skew function as a lineal equation (as also reported in [5], [6], [7], [8]) of the form $y = \alpha x + \beta$. Knowing the time offset θ and the line parameters α and β, the local system can easily estimate future time values of the remote system by just using the line equation. The line parameters can be estimated using least squares method.

Let $y = C_s(t_i) - (C_h(t_i) + \theta)$ and $x = C_s(t_i)$ in the line equation $y = \alpha x + \beta$. Then new estimation of $\widetilde{C_s(t_i)}$ is computed as :

$$\widetilde{C_s(t_i)} = \frac{C_h(t_i) + \theta + \beta}{1 - \alpha} \quad (1)$$

This new estimation of $\widetilde{C_s(t_i)}$ in 1 involves the skew line parameters α in order to cancel its effect.

V. Experimental Results

In this section the achieved results for time synchronization with the skew cancellation method are presented. Results presented here were computed on a x86 host computer with an Intel core duo at $2.13GHz$ and $2Gb$ RAM, and Linux kernel 2.6.17.11 with timer interrupt frequency variable $HZ = 250$, the Intel's SpeedStep function was deactivated to avoid processor frequency changes.

The sensor used for synchronization test was a *URG-04LX* SOKUIKI scanning laser range finder from *Hokuyo Automatic Co. Ltd.*[2], communication between the host computer and the sensor used the USB Abstract Control Model (usbacm) in USB2.0 mode. The sensor is small ($50x50x70$mm) but capable of $10mm$ resolution in distances

up to 5.6m, angular resolution of 0.36^o, wide angular range of 240^o, and a scan rate of 10Hz.

To obtain the local computer time, the `clock_gettime` function of the `RTL` library was used. For the case of the time value of the SOKUIKI sensor, the range scanning command (`GDGS`) of its *SCIP2.0 Protocol Specification*[3] was used, after every scan the timestamp value of the sensor is included, so we simply extracted it. The sensor is also provided with a command (`TM`) to request only the current timestamp value (clock reading), without performing any range scan, it returns far much faster (a few milliseconds) than a normal range scan. We decided not to use it and instead synchronize clocks while acquiring range scan data.

No clock resets are considered in our study, the sensor's clock starts from zero since power-up instant and reaches the round up condition in 4.6 hours, sufficient if compared with robot battery lifetime. All the experiments are online, as soon as data is available from the sensor, the clock skew is computed and then cancelled. The resolution of all the experiments is 100ms due to the sensor's scanning rate.

A. Ceramic vs Quartz

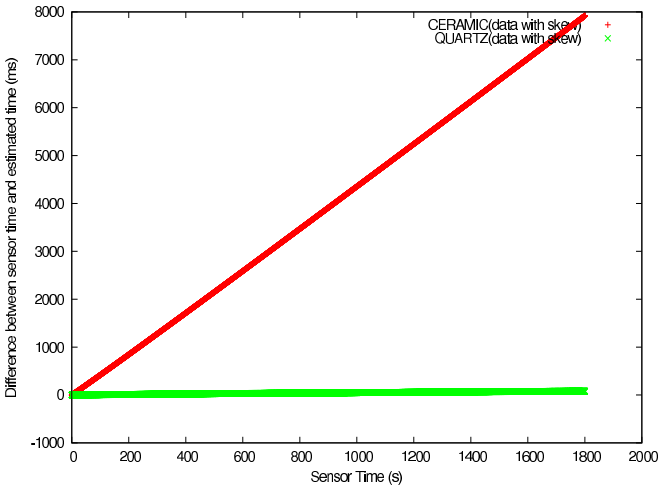
The first experiment compares the clock skew behaviour of two different versions of the SOKUIKI sensor, one equipped with a ceramic resonator and another with a quartz oscillator, for a period of 30 minutes. The commercially available version of the Hokuyo URG-04LX scanning laser range finder has a ceramic resonator, the quartz version used in this experiment is a prototype kindly provided to our research laboratory by *Hokuyo, Ltd.*

Ceramic resonators have low cost but bad frequency tolerance when compared to quartz. As expected, results in figure 4a show a very rapidly ascending skew value for the ceramic version of the sensor (red) when compared to quartz which looks almost flat (green). Figure 4b presents only the behaviour of the quartz version of the sensor, it is certainly not flat but shows clock skew. From figure 4a the ceramic resonator skew is about 8$secs$ after 30min while figure 4b presents a skew for the quartz clock of 80ms after the same period of time.

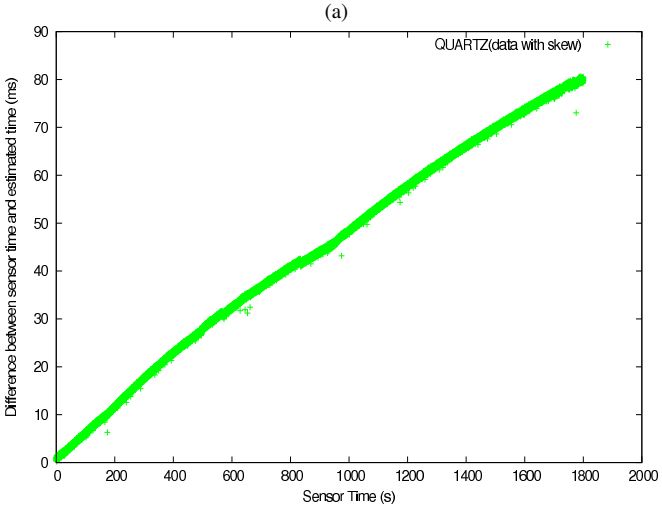
B. Skew cancellation with periodic adjustments

The effects of skew and especially drifting are rather difficult to estimate and therefore to cancel by just one attempt. Instead, periodic adjustments are necessary to keep a proper reduction and maintain synchronization.

The next experiment is designed to keep synchronization by periodically estimating and cancelling the skew effects. Results are shown in figure 5, the initial skew value is estimated (red) and the skew cancellation was applied (green) in a quartz version of the sensor. After some amount of time, every 10 minutes in this case, the new skew values were re-estimated from samples of 30 seconds (fig. 5a, represented in blue color) and the same cancellation method was applied (green). The experiment was run for a period of 30 minutes. Please note that this experiment is different from the one explained in previous section where no cancellation is done.



(a)



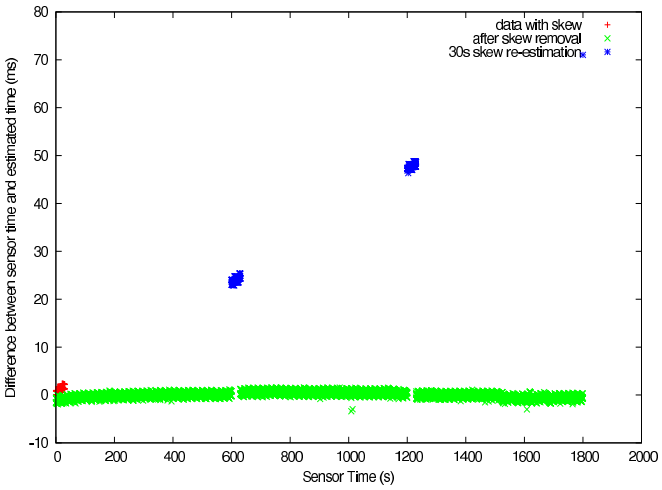
(b)

Fig. 4: Sensor time vs Difference in ceramic and quartz clocks, (a) plot of ceramic clock (red) vs quartz clock (green), (b) only the quartz clock data, the sensor time axis is in seconds while the difference axis is in milliseconds.

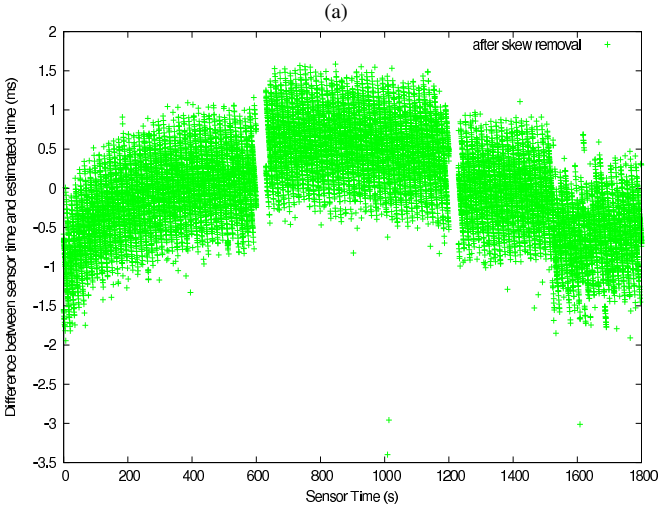
To better understand the results, figure 5b shows only the data with skew effects removed, the small gaps are due to the skew re-estimation. Skew is drastically reduced, all data tends to be grouped in the band of $\pm 2ms$. Figure 5c presents the first 120 seconds of the skew cancellation data, the stepping appearance is due to factors like differences of clock frequency, clock resolution, input/output interruptions, etc.

Figure 6 shows similar results but for a test over one hour. In this case there is no initial skew estimation ($\alpha = 0$) so the first part (red) shows an steep ascending behaviour, however after the first 10 minutes there is a skew re-estimation and it is reduced, and again data tends to be within the $\pm 2ms$ band (green).

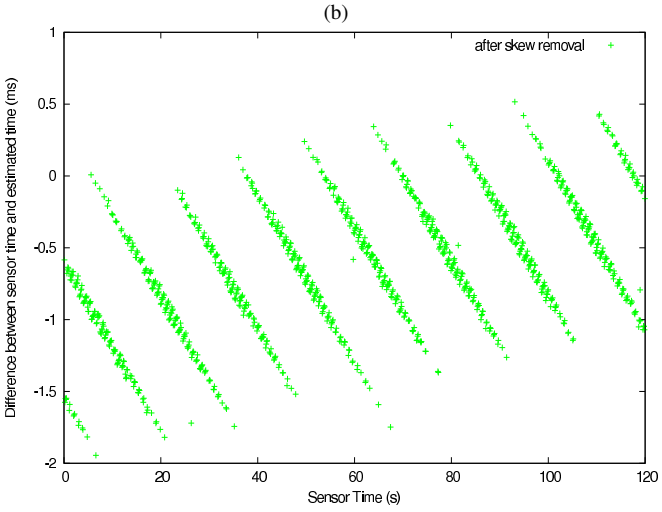
As a comparison of the skew cancellation method, a similar test was perform on the ceramic version of the sensor for over 30 minutes, and results are presented in figure 7. The initial skew value is estimated (red) then skew cancellation



(a)



(b)



(c)

Fig. 5: Sensor time vs Difference over 30 minutes, with 30 seconds skew re-estimation, every 10 minutes (a), (b) only data of skew cancellation and (c) enhancement of the first 2 minutes. The sensor time axis is in seconds while the difference axis is in milliseconds.

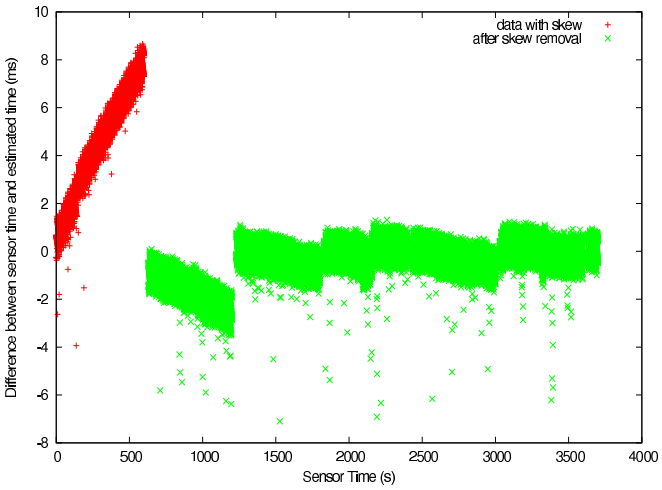


Fig. 6: Sensor time vs difference over one hour, with 30 seconds skew re-estimation, every 10 minutes. The sensor time axis is in seconds while the difference axis is in milliseconds.

was applied (green).

The difference with previous tests is that the 30 seconds skew re-estimation is performed every 2 minutes (more frequent gaps in the skew cancellation curve, re-estimation data is not show for visibility). Since skew effects are bigger in the ceramic clock, longer spacing of the re-estimation, although convenient for there are fewer interruptions in the scan process, yielded very poor results in skew reduction. Data for ceramic clock tend to be in the the $\pm 20ms$ band.

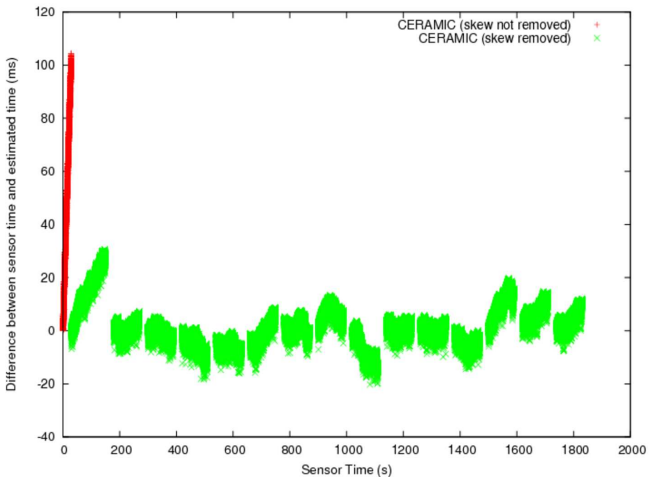


Fig. 7: Sensor time vs difference for the ceramic version, test over 30 minutes with 30 seconds skew re-estimation, every 2 minutes. The sensor time axis is in seconds while the difference axis is in milliseconds.

C. Discussion

From these results we can safely conclude that periodic updates in the synchronization parameters (offset θ, skew α and β) are necessary, in both the ceramic and the quartz

version of the sensor. The results depicted in figure 5 and in a longer term as in figure 6 show a good skew cancellation effect.

The results of skew cancellation in the ceramic version of the SOKUIKI sensor show an error of approximately $\pm 20ms$, while the error for the quartz version is about $\pm 2ms$. To better understand this difference, lets consider two different scenarios for a mobile robot:

1) A robot moving in a straight line with a linear velocity ν of 1 m/s
2) A robot spinning in the same position with an angular velocity ω of 180 deg/s

In scenario 1 an error of $\pm 2ms$ with speed ν means an error in position of scanned objects of $\pm 2mm$, small when compared with odometry resolution and within sensor range resolution. But the $\pm 20ms$ case of the ceramic version of the sensor, moving at ν, means an error in scanned objects position of $\pm 20mm$, a rather big value if the robot depends on the sensor for navigation.

In scenario 2, the quartz version of the sensor having an error of $\pm 2ms$ with speed ω means an error in orientation of $\pm 0.36^o$, which is within the angular resolution of the SOKUIKI sensor, as described above. That means that the scan data is shifted at most by one scan point. However for the ceramic resonator case, $\pm 20ms$ at ω means an orientation error of $\pm 3.6^o$, which means that scan data is shifted several points due to error

In our results however, a time varying behaviour is still present in the cancellation line (it is not flat), this means that:

- It is possible that more optimal line or even polynomial approximation algorithms deliver better estimation of the skew value
- Temperature fluctuations on the testing environment and the normal heating of the sensor due to operation may have increased the drifting rate

These considerations shall be studied in a future work.

VI. Conclusions

Time synchronization is a very important task in applications such as robot control, navigation and fusion of sensor data, however perfect synchronization is impossible. An ideal method consists in obtaining the time offset θ between the two systems and use logical clocks to keep the time account of the remote system. However clocks often fail to provide a good value of the true time, the relative clock skew and drift values must be estimated and then removed.

Clock skew is present in every time value reported by both the sender and the receiving systems. Its behaviour tends to a lineal function when the real time of the sender is compared with the estimated value on the local system. A skew cancellation was achieved, however the effect of clock drifting of the sensor is still significant.

Differences in the physical properties of clocks types, materials, variations in temperature, clock drifting, etc., affect importantly not only the behaviour of the skew function, but also the robot navigation since the error in obstacle position and the scan points orientation varies with the sensor's clock type.

Periodic estimation of skew is necessary due to heating and other conditions on the sensor, accumulating drifting effects. Furthermore, the unpredictable behaviour of host computer task switching and communications delays makes necessary the adjustment. In this work we made no attempt on measuring the later cases.

References

[1] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. Communications of the ACM. v. 21 (7), pp. 558-565, 1978.

[2] Hokuyo Automatic Co., Ltd. http://www.hokuyo-aut.co.jp/

[3] Hirohiko Kawata. URG Series Communication Protocol Specification SCIP-Version2.0. Drawing C-42-03320B, Hokuyo Automatic Co. Ltd. Nov., 2006.

[4] David L. Mills. Internet Time Synchronization: The Network Time Protocol. Global States and Time in Distributed Systems,IEEE Computer Society Press, 1994.

[5] Sue Moon, Paul Skelly, Don Towsley. Estimation and removal of clock skew from network delay measurements. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings of the IEEE INFOCOM'99. v. 1, 1999.

[6] Li Zhang, Zhen Liu, Cathy Honghui Xia. Clock Synchronization Algorithms for Network Measurements. In Proceedings of IEEE INFOCOM 2002.

[7] Hechmi Khlifi, Jean-Charles Gregoire. Estimation and Removal of Clock Skew From Delay Measures. Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04), 2004.

[8] Jingping Bi, Qi Wu, Zhongcheng Li. On estimating clock skew for one-way measurements. Computer Communications. v. 29 (8), pp.1213-1225, 2006.

[9] Flaviu Cristian, Christof Fetzer. Probabilistic Internal Clock Synchronization. Proceedings of the 13th IEEE Symposium on Reliable Distributed Systems, pp. 22-31, 1994.

[10] Hirohiko Kawata, Satofumi Kamimura, Akihisa Ohya, Jun'ichi Iijima, Shin'ichi Yuta. Advanced Functions of the Scanning Laser Range Sensor for Environment Recognition in Mobile Robots. Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Heidelberg, Germany, pp. 414-419, 2006.

[11] Tatsuro Ueda, Hirohiko Kawata, Tetsuo Tomizawa, Akihisa Ohya and Shin'ichi Yuta. Mobile SOKUIKI Sensor System −Accurate Range Data Mapping System with Sensor Motion−. International Conference on Autonomous Robots and Agents, Dec. 2006.