Time Synchronization between SOKUIKI Sensor and Host Computer using Timestamps

Alexander CARBALLO[†], Yoshitaka HARA[†], Hirohiko KAWATA[†],
Tomoaki YOSHIDA[‡], Akihisa OHYA[†], Shin'ichi YUTA[†]
[†]University of Tsukuba [‡]Chiba Institute of Technology

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, therefore this lack of synchronization reduces the accuracy of sensor readings. The new SCIP2.0 protocol of SOKUIKI sensor allows acquiring time values by timestamping range readings. Our work consists in a method for time synchronization with clock skew estimation, between a Sokuiki sensor and a host computer.

Keywords: Sokuiki Sensor, Time Synchronization, Clock Skew

1 Introduction

Keeping an accurate notion of time is a very important requirement in several areas like machine control, navigation (GPS), distributed computing, communications, etc. In the particular case of robot control, task planning depends in knowing the specific time required to complete every step. Also the robot uses sensors to detect landmarks in the environment to better position itself, here the readings from sensors must be synchronized in time with the robot's onboard time, if not then 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 are provided with 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 of most laser range finders (LRF) or ultrasonic sensors, they send distance information with no time source. Then it is duty of the host computer controlling the robot to assume that acquired data is consistent to the current time slot and proceed with the 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.

If the sensor is equipped with some local time reference, then the local time can be provided to the host computer in every data sent. 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 $Th_i - \Delta < Ts_i < Th_i + \Delta$ 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 ($|Ts_i - Th_i| \to 0$). It is known that due to finite speed of data signals, processor speed and even light, perfect true-time synchronization is impossible [7], however current clock synchronization algorithms achieve reasonable results with precisions in the order of microseconds, enough for most robot control applications.

The Hokuyo SOKUIKI sensor (a scanning laser range finder) (figure 1, `URG sensor` in the rest of the paper), is provided with an internal time source (a ceramic resonator) and, since its *SCIPver2.0* specification [5], the current 24-bits time value is stamped on every range scan sent back to the host computer. A good description of the sensor's capabilities is presented in [4] and a demonstration in mapping with timing registration is presented in [10].

Figure 1: SOKUIKI sensor [3].

This work consists in the implementation of a synchronization method between a host computer and a `URG` sensor by using the timestamp value provided by the sensor, logical clocks are used in the host side. The synchronization method uses a clock skew elimination technique to remove errors accountable to relative differences in frequency of the clocks, including an estimation of the drifting rates.

The rest of the paper is organized as follows. Section 2 briefly reviews related work, then section 3 provides a theoretical frame of the related subjects in this research. Section 4 provides a description of the methodology used in solving this problem. Section 5 presents the achieved results. And finally, conclusions and future work are left for the last section.

2 Previous Work

We consider the problem of host computer and scanning laser sensor synchronization under the same light, with the same considerations, as computer network or distributed process synchronization.

A pioneer study on time and causal relations among events on a distributed system is attributed to Lamport [7]. 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. Then, perhaps the most known method for computer time synchronization is due to David Mills' Network Time Protocol (NTP) [8].

Moon *et al.* [9] 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.* [11] found under estimations of clock skew in the later approach, so they proposed using a convex hull method, addressing also the problem of clock resets. Khlifi [6] combined the convex hull approach with a sliding window algorithm for online clock skew estimation and removal. The work of Bi *et al.* considered not only the clock skew problem but also the clock drifting in their piecewise reliable skew estimation approach. In rather different direction, Cristian and Fetzer [2] proposed a probabilistic method, achieving also a good response to skew and drifting of clocks.

The work of Kawata *et al.* [4] describes 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 moving SOKUIKI sensor was studied by Ueda *et al.* in [10]. 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.

3 Background

In this section we introduce the problem of clock skew and basic terminology around the problem, following [9] in Lamport's notation.

3.1 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 $\mathcal{C}_t^{''}(t) = \partial^2\mathcal{C}_t(t)/\partial t^2$ exist, 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 don't 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.

3.2 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 is an example. Here the horizontal axis corresponds to the true clock value $\mathcal{C}_s(t)$ and the vertical is the above difference. A constantly increasing behaviour is observed. It is also worth to mention that the ascending line is never smooth, several disturbances may be seen in the plot. These fluctuations in time have several sources, most of them due to drifting of clocks and to the receiving side operation. 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.

The image in figure 2 also shows a stepping behaviour. The clocks on both sides have different frequencies, especially the sender's clock which in this case has a resolution of 1 ms while the receiving side is able to provide time with microseconds resolution. This huge difference is accountable for the stepping flow of the line.

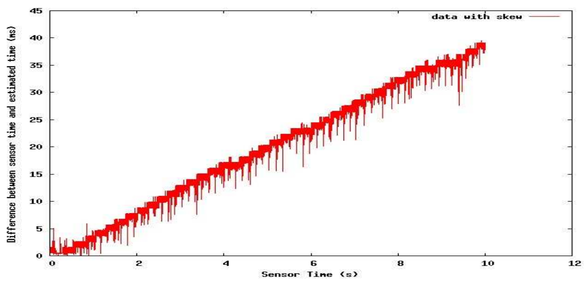

Figure 2: Sensor Time vs Difference, horizontal axis is the sensor time (seconds), the vertical is the mentioned difference (in milliseconds).

4 Time Synchronization

The goal of this research is to achieve time synchronization between a host computer and an `URG` sensor using timestamps from the later. The physical clocks are not 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.

4.1 Estimation of clock offset

As explained above, synchronization involves computing the time offset θ, the relative skew information and the drifting rate. 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 $\mathcal{C}_s(t)(1 - \rho_s)$, 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` four clock (timestamp) values are used, as depicted in figure 3: two from the client when sending the server time request and receiving the answer ($\mathcal{C}_h(t_1)$ and $\mathcal{C}_h(t_4)$ 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 available from the server (the sensor), $C_s(t_3) = C_s(t_2) =$ 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.

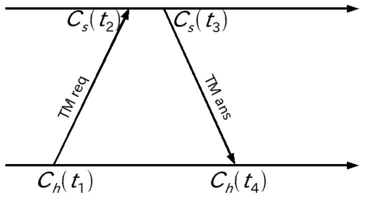

Figure 3: NTP remote clock reading method.

4.2 Synchronization with clock skew cancellation

From the stated in 3.2, it is tempting to visualize the skew function as a lineal equation (as also reported in [1, 6, 9, 11]) 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. To estimate this line we used a simple 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 in order to cancel its effect.

5 Experimental Results

In this section the achieved results for time synchronization with the skew cancellation method are presented. Results presented here were computed using a x86 host computer with a 1500 MHz Intel Pentium IV processor and 384Mb RAM, and a Linux kernel 2.6.20.1 with HZ=300. The sensor used for synchronization was a `Hokuyo URG-04LX` scanning laser range finder; communication between the host computer and the sensor used the USB Abstract Control Model (usbacm) in USB2.0 mode. The Intel's SpeedStep function was deactivated to avoid processor frequency changes.

To obtain the local computer time, the `clock_gettime` function, of the `RTL` library was used, in this way the current time on the host computer was obtained with nanoseconds resolution. For the case of the time value of the `URG` sensor, the `TM` command of the *`SCIP2.0` Protocol Specification* [5] was used.

The drifting parameters used are $\rho_h = 1x10^{-6}$, a common value for quartz crystal oscillators, and $\rho_s = 4x10^{-4}$ (the precise type of ceramic resonator used in the sensor was not know at the time of writing this paper, we used then an upper bound).

No clock resets are considered in our study, the sensor's clock starts at zero from the power up instant and reaches the round up condition in 4.6 hours, sufficient if compared with operational robot battery lifetime.

5.1 Effect of Skew Cancellation

The first positive results of the skew cancellation method are presented in figure 4. Here two samples, one of 300 seconds and another of 1800, were used to probe the skew cancellation. The offset value θ was estimated once and the skew α was obtained over a sample of 20 seconds (up) and 600 seconds (down) (red color), and after that another sample of 300 and 1800 seconds, respectively, was read but this time the skew was notoriously reduced (green). The case for 1800 still shows a small slope, with periodic adjustments this slope can also be reduced.

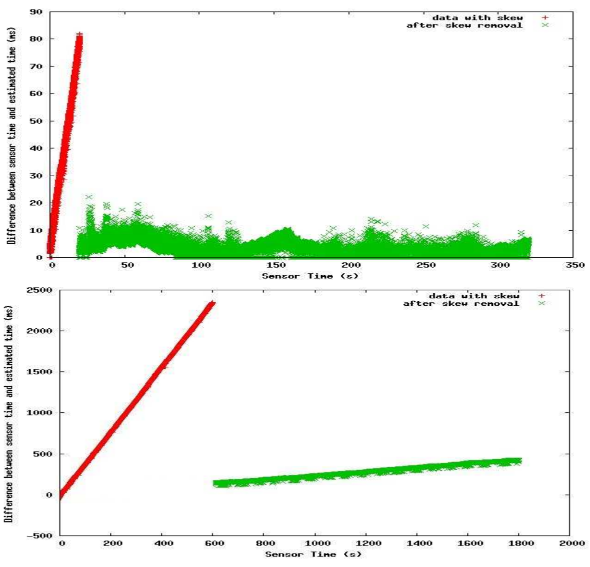

Figure 4: Sensor time vs difference with skew (red) and with skew cancellation (green), for 300 seconds (up) and for 1800 seconds (down). The sensor time axis is in seconds while the difference axis is in milliseconds.

5.2 Skew cancellation with periodic adjustments

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

In the next experiment, the initial skew value is estimated (red) and the cancellation was applied (green). Then new skew values were re-estimated from samples of 2, 5 and 30 seconds (blue), every 120 seconds, and the same cancellation method was applied (green). The experiment lasted for a total of 600 seconds. Note that this experiment is different from the one explained in section 5.1 where synchronization occurred only once. Results are depicted in figure 5.

In the case of 2 and 5 second skew adjustments, the first skew value (red) is under-estimated, but successive estimations correct this problem. A longer test was also studied, this time 30 minutes, as presented in figure 5, test parameters are similar as above: 30 seconds skew re-estimation, every 2 minutes.

From these results we can safely conclude that periodic updates in the synchronization parameters (offset θ, skew

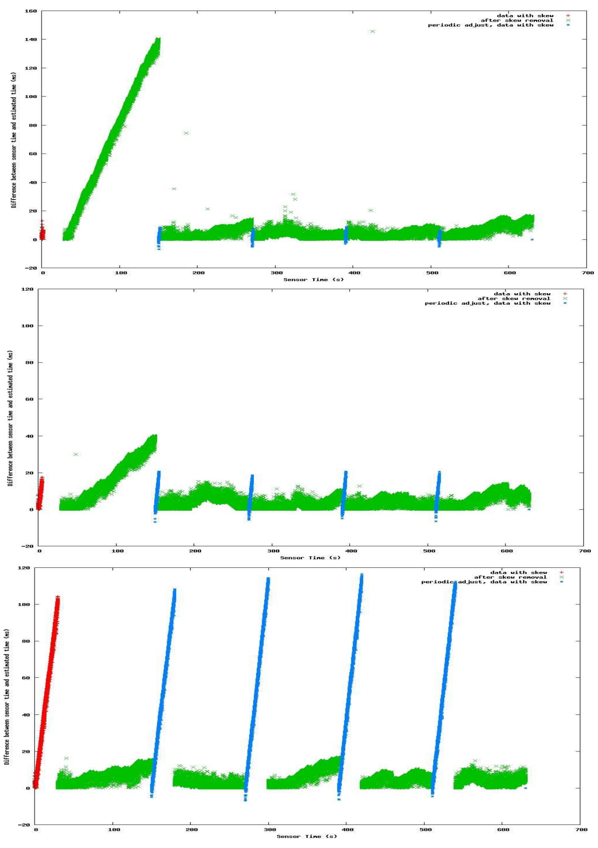

Figure 5: Periodical clock skew removal during 600 seconds, adjustments every 2 mins., 2, 5 and 30 seconds skew estimations (up, middle, down).

α and β) are necessary. The results in figure 4 show some degree of skew removal but those depicted in figure 5 for different adjustment periods and in longer term as in figure 6 exhibit better cancellation effect. However a time varying behaviour is still present in the cancellation line, this means that:

- It is possible that more optimal line estimation algorithms delivering 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

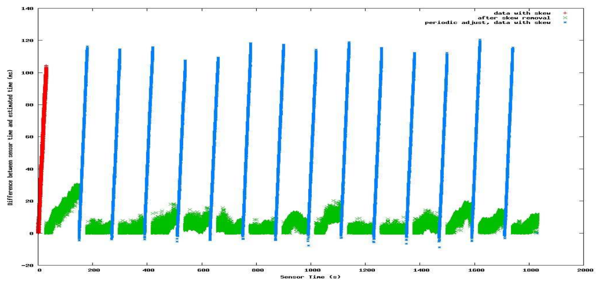

Figure 6: A 30 minutes test, periodic adjustments every 2 minutes, 30 seconds skew estimation.

6 Conclusions

Time synchronization is a very important tasks in applications such as robot control, however perfect synchronization is impossible. An ideal method consists in obtaining the offset θ between the two systems, 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, a constant change in a clock frequency, 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 very significant.

Periodic synchronization is necessary due to heating and other conditions on the sensor. Furthermore, the unpredictable behaviour of host computer task switching and interrupt rates makes necessary the adjustment. In this work we made no attempt on estimating the later rates.

REFERENCES

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

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

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

[4] 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.

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

[6] 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.

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

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

[9] 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.

[10] 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.

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