pre-print: *The 15th Int'l Conference on Advanced Robotics (ICAR)*, 20-23 June 2011, Tallinn, Estonia, pp. 501 - 506.

Active, lifelong sensor synchronization: a Kalman filtering approach

Eric Demeester, Johan Philips, Alexander Hüntemann, Emmanuel Vander Poorten and Hendrik Van Brussel

***Abstract*— In robotics and in many other fields, time synchronization between a host computer and sensors or other computers is essential for many reasons. Whereas past approaches have focused on phase and frequency estimation, little discussion seems to be available in literature on adopting uncertainty on these estimates to determine whether synchronization is out of bounds and to actively decide when to synchronize again to reduce uncertainty. This paper presents an algorithm to perform such active synchronization based on a Kalman filtering approach. The performance of the algorithm is illustrated by synchronizing a Hokuyo URG laserscanner with a computer.**

I. INTRODUCTION

Time synchronization between different clocks is essential in many domains for many reasons. In robotics for example, such time synchronization is important when fusing data of different sensors, which may provide data at different frequencies. This is especially important when robots are moving at high speed as compared to their sensors' refresh rate, as was experimentally proven in the Darpa Grand and Urban Challenges [1]. Furthermore, it is useful to predict when new sensor data will become available, so that they can be processed as soon as they are available and that no time is lost by actively polling the sensor to retrieve data that is not yet there.

The time synchronization algorithm described in this paper was adopted on our wheelchair platforms [2] (one is shown in Fig. 1(a)) for synchronization between a PC and several Hokuyo and other (in-house built) laserscanners. It proved to be useful for various reasons. First of all, time synchronization proved to be important for expressing data from different sensors in a central coordinate frame. Simply expressing the most recent scans from both laserscanners in the robot's coordinate frame at the time of reception of a new scan yields inconsistencies between the scans when the robot is moving, as shown in Fig. 1(c). Therefore it is important to know the time at which data are generated. Secondly, we have used this time synchronization algorithm to correct scans for motion blur, which occurs when the robot is moving while a scan is taken, see e.g. Fig. 2. Thirdly, using scan matching and accurate timestamps, we can estimate the sensor's speed, and thus the robot's speed. Fourthly, we used the timestamps to predict when new sensor data will be available, in order to avoid polling the data buffers in vain and in order to access new sensor data from the moment they are available to increase reaction time to new data.

All authors are with the Department of Mechanical Engineering, Catholic University of Leuven, 3001 Heverlee, Belgium, eric.demeester@mech.kuleuven.be.

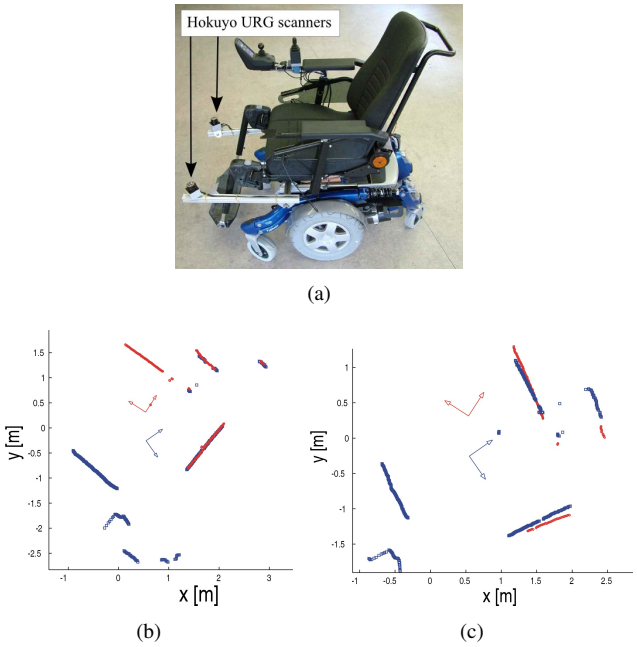


Fig. 1. One of our robotic wheelchairs equipped with two Hokuyo laserscanners (a). Naively time stamping the scans when receiving them, and plotting the two most recent ones is fine as long as the robot is not moving (b), but yields inconsistent results once the robot moves, even at moderate wheelchair speeds and for close-by objects (c). The coordinate frames indicate the pose of the two Hokuyo laserscanners.

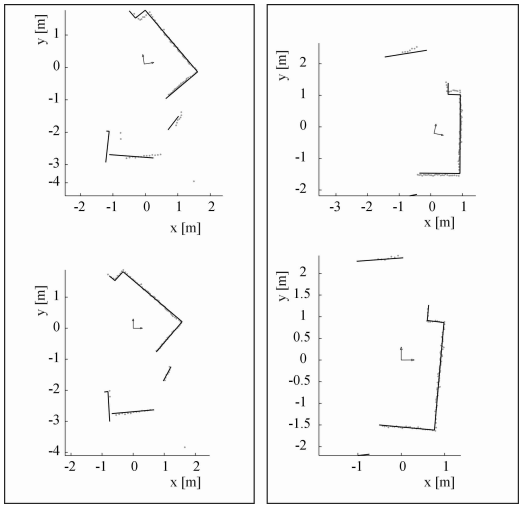


Fig. 2. The left and right figure respectively show motion deblurring for a laserscan taken when rotating in clockwise and in counterclockwise direction. The coordinate frame denotes the lidar's frame. The top figures show the scan before motion deblurring, positioned on an a priori map using a scan matching algorithm. The bottom figures show the scans after motion deblurring. Such motion deblurring can be performed thanks to accurate time stamps.

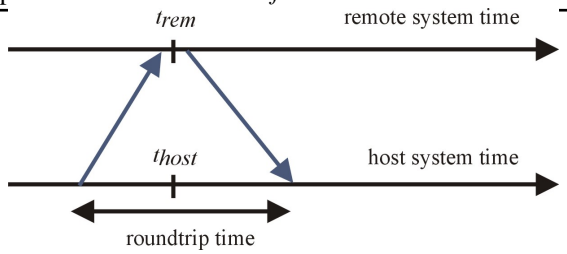


Fig. 3. Performing a "synchronization action" or a "time request" refers to the host system asking the remote system for its current time t_{rem}. Using this time and the measured roundtrip delay, synchronization between the two systems can be performed.

The remainder of this paper is organized as follows. Sections I-A and I-B respectively introduce some concepts and requirements regarding time synchronization, which will be used throughout the paper. Section II gives a concise discussion on the large amount of literature regarding time synchronization. Section III describes a Kalman Filtering approach to time synchronization, which allows to decide whether or not the clocks are synchronized, when to send new synchronization messages, and when to poll the data buffers to retrieve the latest sensor information. Section IV describes the performance of this Kalman filtering approach when applied to synchronization between a computer and a Hokuyo URG laserscanner. Section V concludes the paper and presents future work.

A. Concepts and notation

In the remainder of the paper, we will refer to a *host system* trying to synchronize with a *remote system*, where the host system wants to express time stamps coming from the remote system in its own time reference. In order to do so, the host will perform "synchronization actions" or "send time requests", i.e. the host will ask the remote system for the remote time, as shown in Fig. 3. The host typically measures the roundtrip time and uses this for synchronization.

B. Requirements and functionality

With the synchronization algorithm in Section III, we pursue the following functionality. The main purpose of the algorithm is to synchronize a host clock with a remote clock *continuously*, so that the robot does not have to be halted for any period in order to synchronize. Synchronization means both correct phase and correct frequency synchronization. A synchronization approach should provide four functions:

1) Estimate the phase and frequency of the remote clock (**F1**).
2) Convert remote times to host times using a clock model (**F2**).
3) Decide when to take a new synchronization measurement (**F3**).
4) Decide whether or not the host and remote clocks are synchronized (**F4**), and be aware of synchronization loss.

Furthermore, we would like a synchronization algorithm to adhere to the following requirements as much as possible:

- **Accuracy (R1).** Ideally, a synchronization accuracy of ms or sub-ms is required for our robotic wheelchair application. Suppose we want to estimate the robot's speed from subsequent distance measurements and their time stamps. A timestamp error ε, a distance sensor with a refresh period of Δt and a robot with a maximum linear velocity v_{max} result in a maximal error e_v on the linear velocity estimate of:

$$e_v = \frac{s_{\text{max}}}{\Delta t} - \frac{s_{\text{max}}}{\Delta t + 2\varepsilon} = \frac{s_{\text{max}} 2\varepsilon}{\Delta t\,(\Delta t + 2\varepsilon)},$$

  where ε can be both positive and negative and s_{max} represents the distance travelled during Δt when driving with a linear velocity of v_{max} (similar for rotational velocities). For a Hokuyo URG sensor with a period of around 10 Hz, an electric wheelchair with a maximum linear speed of around 15 $km/h = 4.17\ m/s$, and an allowed error on the velocity estimate of $e_{\text{v,rel}} = e_v/v_{\text{max}} = 5\%$ of the maximum speed, the allowed synchronization error equals:

$$\varepsilon = \frac{e_{\text{v,rel}} \Delta t}{2\,(1 - e_{\text{v,rel}})} = 2.6\ ms.$$

- **Fast synchronization (R2).** The host should synchronize as fast as possible with a remote clock at startup or after loss of synchronization. At startup, at most a couple of seconds are allowed. Re-synchronization after loss of synchronization should be even faster, as the robot may be up and running and may have to be halted to await synchronization. Waiting may be an option for autonomous robots, but it is not acceptable when it affects a human driver as well.
- **Robustness (R3).** Ideally, the synchronization algorithm is robust against loss of synchronization, clock resets, and time-varying clock characteristics.
- **Independence of other sensors (R4).** Synchronization may be obtained by using additional internal or external sensors, for which robot motion (or some other coupling) is required to detect the same events with various sensors. Since this would limit the applicability of the approach, we prefer to have an algorithm that does not require these other sensors.
- **Limited use of resources (R5).** The computational complexity should be as low as possible, with both a limited use of CPU time and memory.
- **Generic (R6).** The algorithm should work on as many types of sensors as possible.
- **As little loss of data as possible (R7).** Asking a sensor for its current time may cause loss of data. Hence, these synchronization requests should be limited in frequency.
- **Assess the uncertainty on the phase and frequency estimation (R8).** This uncertainty is e.g. required when estimating the uncertainty on the robot's velocity from an uncertain estimate of travelled distance and an uncertain estimate of passed time.

II. RELATED WORK

Time synchronization between clocks of various systems has been studied since the end of the seventies [3], and a large amount of research exists in this field.

Many existing algorithms are meant to synchronize computers in networks and distributed systems [4]. Perhaps the most known and used method for computer time synchronization is the Network Time Protocol (NTP) [5]. In this protocol, four timestamp values are exchanged, after which a host system computes the time offset and also selects the best server among a group.

Also, several algorithms have been designed to synchronise computers and sensors. In [6], synchronization between a Hokuyo laserscanner and a computer is performed using NTP timestamps. A large set of time stamps is gathered from the laserscanner after which the time offset and clock drift are estimated using a least squares method. A similar approach is followed in [7], who propose to perform synchronization actions during 30 s, each 6 minutes. Ideally, such synchronization should be performed continuously with as little hinder for robot motion as possible (**R2**).

Another way to synchronize a host system and one or more sensors is to use events that are detectable by several of them. For example, robot motion is detectable by both vision sensors and odometry sensors. In [8], this idea is used to synchronize both vision and odometry to each other. In the absence of robot motion however, this approach cannot be used (**R4**).

In [9], a jitter estimation method is proposed. This method does not actively ask for the sensor time, but uses only the time stamps given by the sensor to esimate possible jitter on the arrival time of sensor data.

To the best of our knowledge, none of the approaches that synchronize with sensors give a principled way to determine (1) when synchronization is lost (or when new synchronization actions are required in order to prevent loss of synchronization), and (2) when to sample data buffers. For this reason, we explored the possibility of using a Kalman filter that assesses the uncertainty on its estimates and uses this uncertainty for functionality **F1** to **F4**. Since this approach contains logic to decide when synchronization actions should be taken to reduce uncertainty on synchronization, we denoted it an *active* approach.

III. KALMAN FILTER BASED TIME SYNCHRONIZATION

A. Representation of the remote clock model

Since the host system has to take all synchronization-related decisions, the functions below are described in terms of a host clock that estimates the time of a remote clock. The relationship between the host and the remote clock can be a non-linear function:

$$t_{\text{rem}} = f(t_{\text{host}}).$$

In many cases, this relationship can be approximated rather well (certainly for small time intervals) as a linear model with a time offset t_{offset} and a scale factor α (the clock drift rate):

$$t_{\text{rem}} = \alpha \cdot (t_{\text{host}} - t_{\text{ref}}) + t_{\text{offset}},$$

where t_{ref} is the host clock time corresponding to the remote clock time t_{offset}.

B. General procedure for Kalman filter based synchronization

We adopt a Kalman filter [10] to estimate the model's parameters α and t_{offset}. The general procedure to update the model of the remote clock is depicted in Algorithm 1. This algorithm returns $true$ in case the clock model is updated successfully. The model starts in a non-initialized state, i.e. $filterInitialised \leftarrow false$. From time to time, this model will be updated with a measurement of the remote clock. For this, the host sends a time request to the remote system, and measures the round trip time when receiving the remote clock's time t_{rem}. The covariance σ^2_{rem} on the remote time is assumed to be known, e.g. by calibration. In our implementation, the host time t_{host} corresponding to the remote time is taken to lie in the middle of the measured round trip interval, and its standard deviation σ_{host} is taken to equal half the round trip interval. The measured values $\{t_{\text{host}}, \sigma^2_{\text{host}}\}$ and $\{t_{\text{rem}}, \sigma^2_{\text{rem}}\}$ are used as input to Algorithm 1. Sections III-C, III-D and III-E describe the functions of Algorithm 1 in more detail. Section III-H describes the logic to decide when to request a new remote time, and thus to execute Algorithm 1.

Algorithm 1 Remote clock model update.

1: Input: $\{t_{\text{host}}, \sigma^2_{\text{host}}\}$ and $\{t_{\text{rem}}, \sigma^2_{\text{rem}}\}$.
2: **if** $filterInitialised \equiv false$ **then**
3: **initialise_state** (t_{rem}, 1, $\mathbf{P}_{\text{init}}$, t_{host}, σ^2_{host}).
4: $filterInitialised \leftarrow true$.
5: return $true$.
6: **else**
7: **predict_state** ($t_{\text{host}}, \sigma^2_{\text{host}}$).
8: **if correct_state** ($t_{\text{host}}, \sigma^2_{\text{host}}, t_{\text{rem}}, \sigma^2_{\text{rem}}$) $\equiv true$ **then**
9: return $true$.
10: **else**
11: **initialise_state** (t_{offset}, α, $\mathbf{P}_{\text{init}}$, t_{ref}, σ^2_{ref}).
12: **predict_state** ($t_{\text{host}}, \sigma^2_{\text{host}}$).
13: return **correct_state** ($t_{\text{host}}, \sigma^2_{\text{host}}, t_{\text{rem}}, \sigma^2_{\text{rem}}$).
14: **end if**
15: **end if**

C. Kalman filter state and state initialization

The Kalman filter state $\mathbf{x}$ is chosen to consist of two elements:

$$\mathbf{x} = \begin{bmatrix} t_{\text{offset}} \\ \alpha \end{bmatrix}.$$

The covariance $\mathbf{P}$ on the state $\mathbf{x}$ is continuously updated as well:

$$\mathbf{P} = \begin{bmatrix} \sigma^2_{oo} & \sigma^2_{o\alpha} \\ \sigma^2_{\alpha o} & \sigma^2_{\alpha\alpha} \end{bmatrix}.$$

Furthermore, two additional variables are kept in memory, the reference time t_{ref} and its variance σ^2_{ref}.

The Kalman filter state is initialized the first time a synchronization measurement is obtained (line 3 in Algorithm 1), and whenever state correction did not succeed (line 11 in Algorithm 1). Function **initialise_state()** does the following operations:

$$\begin{aligned} \mathbf{x} &= \begin{bmatrix} t_{\text{rem}} \\ 1 \end{bmatrix} \text{ for line 3 or } \begin{bmatrix} t_{\text{offset}} \\ \alpha \end{bmatrix} \text{ for line 11,} \\ \mathbf{P} &= \mathbf{P}_{\text{init}}, \\ \begin{bmatrix} t_{\text{ref}} \\ \sigma^2_{\text{ref}} \end{bmatrix} &= \begin{bmatrix} t_{\text{host}} \\ \sigma^2_{\text{host}} \end{bmatrix} \text{ for line 3 or } \begin{bmatrix} t_{\text{ref}} \\ \sigma^2_{\text{ref}} \end{bmatrix} \text{ for line 11.} \end{aligned}$$

D. State prediction

In order to predict the state $\mathbf{x_p}$ and covariance $\mathbf{P_p}$ for a given future time t_{host} with covariance σ^2_{host} (lines 7 and 12 in Algorithm 1), a system model $\mathbf{x_p} = f(\mathbf{x}, \mathbf{u})$ is adopted, where the input $\mathbf{u}$ equals $\Delta = t_{\text{host}} - t_{\text{ref}}$ and the uncertainty on the input equals $\sigma^2_\Delta = \sigma^2_{\text{host}} + \sigma^2_{\text{ref}}$:

$$f : \begin{cases} t_{\text{offset,p}} = \alpha \cdot \Delta + t_{\text{offset}} \\ \alpha_{\text{p}} = \alpha \end{cases}$$

Furthermore, the covariance is predicted:

$$\mathbf{P_p} = \nabla f_x \mathbf{P} \nabla f_x^T + \nabla f_u \sigma^2_\Delta \nabla f_u^T + \mathbf{Q},$$

where $\mathbf{Q}$ represents the process noise on the system model f, and where

$$\begin{aligned} \nabla f_x &= \begin{bmatrix} 1 & \Delta \\ 0 & 1 \end{bmatrix}, \\ \nabla f_u &= \begin{bmatrix} \alpha \\ 0 \end{bmatrix}. \end{aligned}$$

E. State correction

The state is corrected using the remote system time t_{rem} as a measurement, with variance σ^2_{rem}. For this, first the Normalised Innovation Squared (NIS) value is determined:

$$\begin{aligned} \nu &= t_{\text{rem}} - t_{\text{offset,p}}\ , \\ s &= \sigma^2_{\text{rem}} + \sigma^2_{oo,p}\ , \\ NIS &= \frac{\nu^2}{s}\ . \end{aligned}$$

The NIS value is χ^2 distributed. A NIS value above a chosen threshold `MAX_NIS` or below `MIN_NIS` indicates that the prediction of the remote time is outside the chosen confidence interval, meaning that either the model or the measurement is incorrect. In that case, the state correction is considered to be unsuccessful and **correct_state()** returns $false$. If on the contrary the NIS value lies within the boundaries of acceptance, the state $\mathbf{x}$ is updated as follows:

$$\begin{aligned} \mathbf{H} &= \begin{bmatrix} 1 & 0 \end{bmatrix}\ , \\ K &= \frac{\mathbf{P_p} \cdot \mathbf{H}^T}{s} = \begin{bmatrix} \sigma^2_{oo,p}/s \\ \sigma^2_{os,p}/s \end{bmatrix}\ , \\ t_{\text{offset}} &= t_{\text{offset,p}} + K_{11} \cdot \nu\ , \\ \alpha &= \alpha_{\text{p}} + K_{21} \cdot \nu\ . \end{aligned}$$

In the above equations, $\mathbf{H}$ represents the measurement function which expresses the measurement t_{rem} as a function of the state $\mathbf{x}$: $t_{\text{rem}} = \mathbf{Hx} = t_{\text{offset}}$. Furthermore, the reference time t_{ref} and its variance σ^2_{ref} are set to t_{rem} and σ^2_{rem} respectively. The covariance $\mathbf{P}$ is updated as follows:

$$\mathbf{P} = (\mathbf{I} - \mathbf{KH}) \cdot \mathbf{P_p}\ .$$

F. Decision: synchronized or not at time t?

In order to determine whether the model for the remote clock still yields precise enough predictions at a certain time t_{host} with covariance σ^2_{host}, the state $\mathbf{x}$ is first predicted at time t_{host} using **predict_state()**. We propose to decide that the host and remote system are still synchronized if the uncertainty on the offset t_{offset} and on the clock drift factor α are below a chosen threshold:

$$\begin{aligned} \sigma^2_{oo} &\leq \texttt{THRESHOLD\_P\_OO\_SYNCH}, \\ \sigma^2_{\alpha\alpha} &\leq \texttt{THRESHOLD\_P\_AA}. \end{aligned}$$

G. Conversion of remote time to host time

If data are accompanied by time stamps, these time stamps are expressed in the remote system's time frame. These need to be converted into the host's time frame. If we get as input t_{rem} and variance σ^2_{rem}, we can compute the corresponding host time t_{host} and variance σ^2_{host} (once the Kalman filter is initialised) as follows:

$$\begin{aligned} t_{\text{host}} &= \frac{t_{\text{rem}} - t_{\text{offset}}}{\alpha} + t_{\text{ref}} \\ &= g\left(t_{\text{rem}}, t_{\text{offset}}, \alpha, t_{\text{ref}}\right) \end{aligned}$$

The uncertain variables on the right hand side of this expression are t_{rem}, t_{offset}, α and t_{ref}. Their joint covariance matrix is modelled to be:

$$\mathbf{P}_{\text{tot}} = \begin{bmatrix} \sigma^2_{\text{rem}} & 0 & 0 & 0 \\ 0 & \sigma^2_{\text{ref}} & 0 & 0 \\ 0 & 0 & \sigma^2_{oo} & \sigma^2_{o\alpha} \\ 0 & 0 & \sigma^2_{\alpha\,\text{o}} & \sigma^2_{\alpha\alpha} \end{bmatrix}.$$

Hence, the uncertainty on t_{host} equals:

$$\begin{aligned} \sigma^2_{\text{host}} &= \nabla g \cdot \mathbf{P}_{\text{tot}} \cdot \nabla g^T \\ &= \frac{\sigma^2_{\text{rem}}}{\alpha^2} + \sigma^2_{\text{ref}} + \nabla b \cdot \mathbf{P} \nabla b^T\ , \end{aligned}$$

where

$$b = \begin{bmatrix} -\frac{1}{\alpha} & -\frac{t_{\text{rem}} - t_{\text{offset}}}{\alpha^2} \end{bmatrix}\ .$$

H. Determination of new synchronization actions

The determination of when to ask for a new remote time is based on the prediction of when the uncertainty on the state $\mathbf{x}$, σ^2_{oo} and $\sigma^2_{\alpha\alpha}$, will exceed a chosen threshold:

1) If the model is not yet initialised, or if the predicted variance $\sigma^2_{\alpha\alpha} + q_{\alpha\alpha}$ on the scale factor α is larger than threshold `MAX_P_AA`, it is decided to synchronize as soon as possible.
2) Else, it is computed for which future time instant t_{host} the additional condition of limited variance on the offset is violated. Let $\Delta = t_{\text{host}} - t_{\text{ref}}$. The future time

instant can be computed by considering the predicted variance on the offset:

$$\sigma_{oo}^2 + 2\Delta\sigma_{os}^2 + \Delta^2\sigma_{\alpha\alpha}^2 + \alpha^2\left(\sigma_{\text{ref}}^2 + \sigma_{\text{host}}^2\right) + q_{oo} \leq \texttt{MAX\_P\_OO\_PRED}$$

This is a quadratic inequality in Δ. If no solution is found, it is decided to synchronize as soon as possible. If more than one solution is found, the smallest Δ is chosen. It is decided to send a new synchronization message in time $t_{\text{ref}} + \Delta$.

IV. EVALUATION OF KALMAN FILTER BASED SYNCHRONIZATION

This section evaluates the proposed synchronization with respect to the requirements of Section I-B and discusses its performance when applied to the synchronization of a computer with a Hokuyo URG scanner.

A. Evaluation with respect to requirements

The synchronization algorithm described in Section III addresses all four functional requirements: estimate phase and frequency (**F1**), conversion of remote times to host times (**F2**), decision of when to send a new synchronization message (**F3**) and being aware of synchronization loss (**F4**).

The requirements regarding accuracy (**R1**), speed of synchronization (**R2**), robustness (**R3**) and loss of data (**R7**) will be discussed for the application to the Hokuyo URG in Section IV-B.

The approach does not require other sensors or robot motion to perform synchronization, and therefore adheres to **R4**. The algorithm needs only very limited CPU and memory resources (**R5**), and proceeds in a constant time fashion $O(1)$ with a fixed and known number of multiplications and additions per model update.

The proposed Kalman Filter synchronization algorithm is generic (**R6**) in the sense that it does not assume any specific information regarding sensors. The only assumption it makes is that the sensor's current time can be asked for.

The approach assesses the uncertainty on its estimates, and therefore adheres to requirement **R8**.

Furthermore, the algorithm is straight forward to implement.

B. Verification of requirements for Hokuyo URG

We applied the algorithm of Section III to synchronize actively and continuously a host computer Atom Mini-ITX 1.6 GHz running Windows XP with a Hokuyo URG laserscanner. For this, the parameters in Table I were adopted. Fig. 4(a) shows a plot of PC time versus Hokuyo time. The red circles denote actual measurements of the Hokuyo time at certain PC times. These measurements correspond to the synchronization measurements. The blue points in the figure denote predictions by the Kalman filter synchronization algorithm. As can be seen from the figure, the linear model for the remote clock is acceptable for the Hokuyo URG. As a consequence, the Kalman filter performs quite well. Fig. 4(b) shows the first seconds of a typical start-up. As

TABLE I

PARAMETERS OF THE KALMAN FILTER BASED TIME SYNCHRONIZATION ALGORITHM FOR THE HOKUYO URG SCANNER.

Parameter	Value
$\mathbf{P}_{\text{init}}$	$\begin{bmatrix} 1\cdot10^{6} & 0 \\ 0 & 1\cdot10^{6} \end{bmatrix}$
$\mathbf{Q}$	$\begin{bmatrix} 6\cdot10^{-10} & 0 \\ 0 & 8\cdot10^{-9} \end{bmatrix}$
σ_{rem}^2	$1\cdot10^{-9}$
MIN_NIS	$1\cdot10^{-3}$
MAX_NIS	5
THRESHOLD_P_OO_SYNCH	$1\cdot10^{-4}$
THRESHOLD_P_AA	1
MAX_P_OO_PRED	$25\cdot10^{-6}$

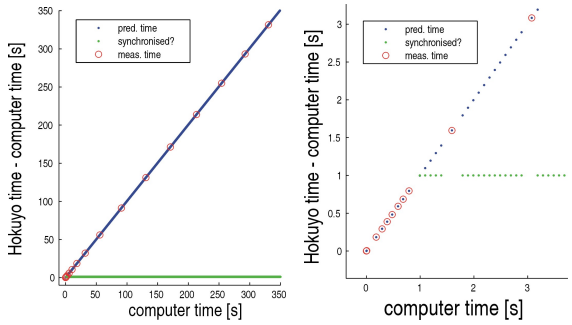
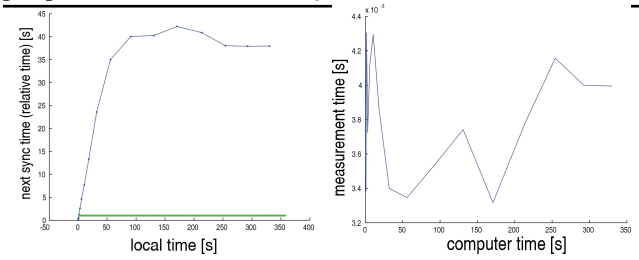


Fig. 4. PC time versus Hokuyo time (a) and zoom into the first seconds (b). Both times start from zero, since their first time stamp was subtracted from all subsequent time measurements, to make the plots more easily interpretable. The green dots indicate at which time stamps the PC and Hokuyo are synchronized. When sending time requests (red circles), no green dots are plotted.

can be seen, more synchronization messages are performed at the beginning in order to reduce the uncertainty on the state estimate of the remote clock. As shown in the figure, synchronization typically occurs within one second, and is seldomly lost afterwards. This adheres to requirement **R2**.

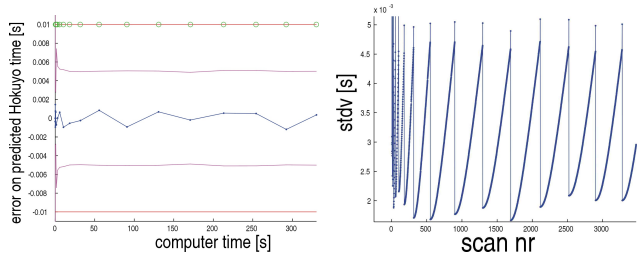
Fig. 5(a) shows the interval between synchronization messages. At startup, synchronziation messages are typically sent at a higher frequency. This situation evolves to a constant interval of around 40 s for the (conservative) parameter settings of Table I. Typical roundtrip times for our PC and Hokuyo lie around 4 ms, as shown in Fig. 5(b).

The accuracy on the time conversion and predictions is shown in Fig. 6(a). This figure shows the error between the Hokuyo time as predicted by the Kalman filter, and the real Hokuyo time, and this for each moment at which a synchronization action is performed. As can be seen from the figure, the error on the time prediction is typically below 2 ms, for time periods of around 40 s. In between these synchronization actions, the time error is even lower. Hence, this adheres to requirement **R1**. The estimated uncertainty on remote times is shown in Fig. 6(b). As expected from Fig. 4(a), the scan period of the Hokuyo is not changing



(a) (b)

Fig. 5. Time between successive synchronization requests (a) and typical roundtrip times (b).

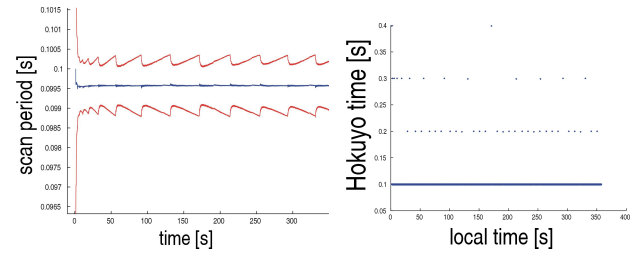


(a) (b)

Fig. 6. Error between predicted time by the Kalman filter and the actual Hokuyo time t_{rem} (a), and uncertainty on the estimate of t_{rem} (b).

very fast, see Fig. 7(a). The Kalman filter estimates the scan period well, with sub-ms resolution. The uncertainty on the scan period even seems slightly conservative and could be even improved.

When processing time synchronization requests by the PC, the Hokuyo scanner goes into a special timing mode during which no scans are taken. Hence, a number of scans may be lost when asking for the Hokuyo time. Fig. 7(b) shows the time difference between successive scans. From this, it can be seen that 1 to 2, and from time to time 3, successive scans are lost when performing a synchronization operation. For our wheelchair application this was acceptable, though we would like to improve on this (**R7**). Since both scanners observe the front of this robotic wheelchair, a safe strategy would be to synchronize both scanners at different times, so that the front of the wheelchair is always being observed. Though dedicated tests on robustness (**R3**) still have to be performed, during our experiments the wheelchair PC did not loose synchronization with the two Hokuyos, even when running time-consuming localisation, planning and obstacle avoidance algorithms.



(a) (b)

Fig. 7. Estimated Hokuyo scan period (a), which is known to lie around 100 ms, and the uncertainty thereupon. Time between successive scans (b), which should be around 0.1 if no scans are lost.

V. CONCLUSIONS AND FUTURE WORKS

This paper presented a synchronization algorithm based on Kalman filtering, with a focus on synchronization with sensors that run at a fixed frequency. Several requirements for such synchronization algorithms were set up, and existing approaches were compared to these requirements. Our algorithm was evaluated by synchronizating a PC with the popular Hokuyo URG sensor. The algorithm adheres to most of the requirements, though robustness was not specifically tested yet. Furthermore, the algorithm assumes the remote system can reply with its own time. In the near future we would like to test the algorithm on other types of Hokuyos and hardware.

VI. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Communitys Seventh Framework Programme FP7/2007-2013 - Challenge 2 - Cognitive Systems, Interaction, Robotics - under grant agreement No. 248873-RADHAR.

REFERENCES

[1] Martin Buehler, Karl Iagnemma, and Sanjiv Singh, *The DARPA Urban Challenge, Autonomous Vehicles in City Traffic*, vol. 56 of *Springer Tracts in Advanced Robotics*, Springer Berlin / Heidelberg, 2009.

[2] Eric Demeester, Alexander Hüntemann, Dirk Vanhooydonck, Gerolf Vanacker, Hendrik Van Brussel, and Marnix Nuttin, "User-adapted plan recognition and user-adapted shared control: A bayesian approach to semi-autonomous wheelchair driving", *Autonomous Robots*, vol. 24, no. 2, pp. 193 – 211, February 2008.

[3] Leslie Lamport, "Time, clocks, and the ordering of events in a distributed system", *Communications of the ACM*, vol. 21, no. 7, pp. 558 – 565, 1978.

[4] B. Koninckx, H. Van Brussel, B. Demeulenaere, J. Swevers, N. Meijerman, F. Peeters, and J. Van Eijk, "Closed-loop, fieldbus-based clock synchronisation for decentralised motion control systems", in *Proceedings of the CIRP 1st international conference on agile, reconfigurable manufacturing*, Ann Arbor, MI, 2001.

[5] David L. Mills, "Internet time synchronization: The network time protocol", *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482 – 1493, October 1991.

[6] Alexander Carballo, Yoshitaka Hara, Hirohiko Kawata, Tomoaki Yoshida, Akihisa Ohya, and Shin'ichi Yuta, "Time synchronization between sokuiki sensor and host computer using timestamps", in *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Seoul, Korea, August 20 - 22 2008, pp. 261 – 266.

[7] Fredy Tungadi and Lindsay Kleeman, "Time synchronisation and calibration of odometry and range sensors for high-speed mobile robot mapping", in *Proceedings of the Australasian Conference on Robotics and Automation*, Jonghyuk Kim and Robert Mahony, Eds. December 3 - 5 2008, Canberra, Australia, ISBN 978-0-646-50643-2.

[8] Munir Zaman and John Illingworth, "Interval-based time synchronisation of sensor data in a mobile robot", in *Proceedings of the Intelligent Sensors, Sensor Networks and Information Processing Conference*, December 14 - 17 2004, pp. 463 – 468.

[9] E. Olson, "A passive solution to the sensor synchronization problem", in *Proceedings the 2010 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, Taipei, Taiwan, October 18 – 22 2010, pp. 1059 – 1064.

[10] Yaakov Bar-Shalom, X.-Rong Li, and Thiagalingam Kirubarajan, *Estimation with Applications to Tracking and Navigation*, John Wiley & Sons, Inc., 2001.