2011 IEEE Intelligent Vehicles Symposium (IV)
Baden-Baden, Germany, June 5-9, 2011

Precise Timestamping and Temporal Synchronization in Multi-Sensor Fusion

Tobias Huck*, Antje Westenberger*, Martin Fritzsche*, Tilo Schwarz* and Klaus Dietmayer†
* Department Environment Perception, Daimler AG, Ulm, Germany,
{tobias.huck, antje.westenberger, martin.fritzsche, tilo.schwarz}@daimler.com
† Institute of Measurement, Control, and Microtechnology, University of Ulm, Germany, klaus.dietmayer@uni-ulm.de

***Abstract*—This paper presents a new approach to exact timestamping of asynchronous measurements in a multi-sensor setup. In order to improve the performance of a single sensor, driver assistance systems use several different sensors which have different latencies that usually cannot be measured directly. These unknown latencies pose a problem in data association and temporal synchronization. Consequently, a method to estimate or incorporate the latencies is needed in all sensor fusion algorithms in order to derive the real time of a measurement. In this paper a method is described to compensate the sensor latencies even if they cannot be observed directly.**

I. Introduction

Modern driver assistance systems are based on an increasing amount of different sensors that may be complementary or cooperative. Much research is therefore done on reliable multi-sensor fusion algorithms. The crucial first part of every sensor data fusion is the data association, including the temporal synchronization of measurements from different sensors. The exact time at which the measurement was taken is needed. Unfortunately, not all sensors provide such information.

Todays automotive sensors can be classified in two categories: The first are sensors that can be externally triggered or provide a separate counter value with each measurement, like some cameras do, or provide a strobe pulse like some free-running sensors. The second are sensors which cannot be externally triggered or provide additional information like a counter. Most sensors in this category are free-running like radars or laserscanners.

Hence, the earliest timestamp available for a radar measurement is the time of arrival at a data acquisition system, where this measurement data is processed. The processing time between the sensor measurement and the arrival at the acquisition system buffer is unknown. In very time critical applications, e.g., in pre-crash situations or automatic emergency breaking, these unknown latencies cannot be neglected, as they may add up to several hundred milliseconds.

In previous work, sensor data was often fused without considering the possibility that these data may be unsynchronized (e.g. [1]). In [2] a special parallel hardware device is used to timestamp sensor data with GPS accuracy. But as mentioned earlier, this approach cannot be applied to all sensors. In [3] a method for software timestamping is proposed using UTC time including a procedure to improve the timestamps, yet the data are only timestamped after they have arrived in the buffer. In [4] a method for synchronizing the clocks of several computers to a common reference clock is described. Again, this does not solve the problem of estimating the sensor latencies.

The idea of the paper at hand is as follows: First, jittering timestamps are compensated using filtering techniques. A new procedure regarding the minimization of measurement delays is proposed to get timestamps as close as possible to the unknown measurement timestamps. Then a special approach is performed with a test vehicle where sensors measure a common spatial target with varying yaw angle of the vehicle. The measured yaw angle differs between the measurements from different sensors due to the different sensor specific latencies. These latencies can be exactly determined from the evaluation of this test drive using a cross correlation.

The outline of the paper is as follows: In chapter II an overview of the main problems arising in temporal synchronization is given. Chapter III proposes detailed solutions to the different subproblems. Chapter IV presents the practical evaluation of the described methods.

II. Main problems arising in timestamping

A. The measurement latency

Consider a general sensor setup with multiple sensors. Let A_i be a measurement taken by one of these sensors at time t_{A_i}, the *true sampling timestamp*, where $i \in \mathbb{N}$ denotes the number of the measurement. The delay before the measurement arrives at the acquisition system is the *measurement latency* $\Delta\tau_i$ (see Figure 1). Several factors contribute to this latency,

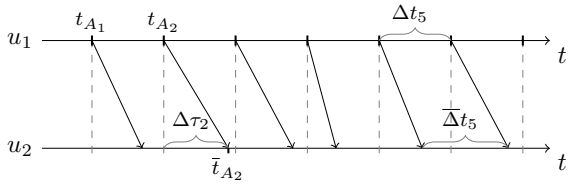

Fig. 1. Sensor clock u_1 with acquisition system clock u_2

e.g., the measurement acquisition time, the pre-processing time, the communication transfer, the buffering, the computer scheduling, etc. These factors are not deterministic and are usually subject to fluctuations and jitter. The mean values of these m delaying factors are denoted by δ_k $(k = 1, ..., m)$ and

the arrival time at the acquisition system by

$$\bar{t}_{A_i} := t_{A_i} + \sum_{k=1}^{m} \delta_k + v_i, \tag{1}$$

where v_i is assumed to be zero-mean white noise. These delayed timestamps are referred to as *software timestamps*. The measurement latency can be written as

$$\Delta\tau_i = \sum_{k=1}^{m} \delta_k + v_i. \tag{2}$$

Note that this latency is sensor specific, which means that there are actually several latencies $\Delta\tau_i^{s_j}$, one for each sensor s_j. Yet for convenience, a general latency is denoted by $\Delta\tau_i$, the other variables are used analogously in this chapter.

B. The measurement cycle duration

Usually there is no system-wide reference clock, but an internal sensor clock u_1 independent of the computer clock u_2. In general, the internal clock u_1 cannot be observed or accessed directly. It is assumed that there is a certain period of time between two measurements, the sensor specific *measurement cycle duration* Δt that is assumed constant with respect to the internal clock u_1. One faces the problem that

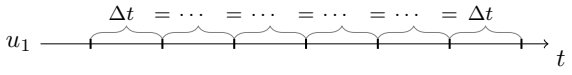

Fig. 2. Linearly drifting sensor clock u_1 from u_1's perspective

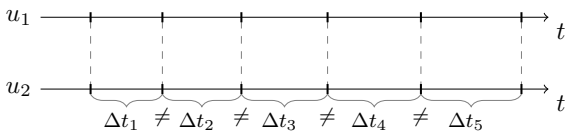

Fig. 3. Linearly drifting sensor clock u_1 from u_2's perspective

the internal clock u_1 may drift, which means that the cycle duration Δt may only be constant with respect to the internal clock and may drift over time with respect to UTC time. This fact is graphed in Figures 2 and 3. Consequently the cycle duration is variable,

$$\Delta t_i = t_{A_{i+1}} - t_{A_i}. \tag{3}$$

In the case of an additive drift d_i at timestamp t_{A_i}, one gets for the i-th cycle duration

$$\Delta t_i = \Delta t + \sum_{k=2}^{i} d_k, \tag{4}$$

where $\Delta t := \Delta t_1$ is used as a base cycle duration.

Due to the delayed arrival time $\bar{t}_{A_i}$ at the acquisition system, instead of the cycle duration Δt_i, only the disturbed duration

$$\overline{\Delta} t_i = \bar{t}_{A_{i+1}} - \bar{t}_{A_i} \tag{5}$$

between the arrival times of two consecutive measurements A_i and A_{i+1} can be observed (see Figure 1). The disturbed duration is not constant because of the jitter and the drift. According to (1) it can be written as

$$\overline{\Delta} t_i = \Delta t_i + v_{i+1} - v_i. \tag{6}$$

Hence the first subproblem consists of filtering the exact cycle duration Δt_i from the observable disturbed duration $\overline{\Delta} t_i$. The second subproblem consists of estimating the unknown measurement latency $\Delta\tau_i$. If these two quantities were known, one would be able to determine exactly the measurement time t_{A_i} and subsequent measurement time $t_{A_{i+1}}$, and the problem of exact timestamping would be solved.

III. A SYNCHRONIZATION METHOD

A. Estimation of the cycle duration

The key to overcoming the previously described problems is the following: First, the exact cycle duration Δt_i is estimated using common filtering methods, e.g., a Kalman filter, mean value filter or exponential filter. A standard linear Kalman filter is used in this paper in order to predict future timestamps as proposed in [5]. It is important to not only estimate the cycle duration Δt_i but separately the drift d_i according to equation (4) as well. For the i-th state vector one gets

$$x_i = \begin{bmatrix} \Delta t_i \\ d_i \end{bmatrix}. \tag{7}$$

Because of equation (4) the dynamics of the state vector can be modeled by

$$x_{i+1} = F x_i + w_i, \quad F := \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \tag{8}$$

where w_i is a zero-mean white noise. The Kalman filtering algorithm with application in timestamping is stated in algorithm I. As usual, P denotes the error covariance, R the measurement covariance, Q the process noise covariance and $H := \begin{bmatrix} 1 & 0 \end{bmatrix}$ the transition matrix from the state space to the measurement space. A detailed description of the Kalman filtering process can be found, e.g., in [6].

Algorithm I
CYCLE DURATION ESTIMATION ALGORITHM

%initial values
$\widehat{x}_1^+ = \begin{bmatrix} \overline{\Delta} t_1 \\ 0 \end{bmatrix}$
$P_1^+ = I$
for $i = 1$ to n **do**
  %predict
  $\widehat{x}_{i+1}^- = F\widehat{x}_i^+$
  $P_{i+1}^- = F P_i^+ F^T + Q$
  %update
  $K_{i+1} = P_{i+1}^- H^T (H P_{i+1}^- H^T + R)^{-1}$
  $\widehat{x}_{i+1}^+ = \widehat{x}_{i+1}^- + K_{i+1}(\overline{\Delta} t_{i+1} - H\widehat{x}_{i+1}^-)$
  $P_{i+1}^+ = (I - K_{i+1} H) P_{i+1}^-$
end for

The result of this linear filtering process is an estimated cycle duration $\widehat{\Delta} t_i$ $(i = 1, 2, ...)$ that is free of jitter and takes drift into account.

B. Timestamping

Suppose now there is a measurement A_i with arrival time $\bar{t}_{A_i}$ at the acquisition system. The main goal is to improve these inaccurate software timestamps as well as predict future timestamps. As the estimation $\widehat{\Delta}t_i$ is known, the natural first

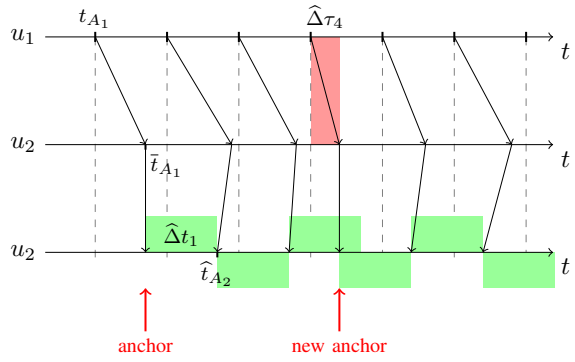

Fig. 4. Timestamp correction on the acquisition system

prediction of the next arrival time would be

$$\widehat{t}_{A_{i+1}} = \bar{t}_{A_i} + \widehat{\Delta}t_i. \tag{9}$$

This method can be continued. For example, $\bar{t}_{A_1}$ is used as a base timestamp (or anchor, respectively) and the estimated cycle duration is added several times in order to predict the arrival time of an arbitrary measurement A_{i+1}:

$$\widehat{t}_{A_1} = \bar{t}_{A_1}, \tag{10}$$

$$\widehat{t}_{A_{i+1}} = \widehat{t}_{A_i} + \widehat{\Delta}t_i = \widehat{t}_{A_1} + \sum_{k=1}^{i} \widehat{\Delta}t_k \tag{11}$$

$(i = 1, 2, ...)$.

In order to get more accurate timestamps, what should really be estimated is the true measurement time t_{A_i} instead of the arrival time $\bar{t}_{A_i}$. The goal is an approximation satisfying

$$t_{A_i} \approx \widehat{t}_{A_i} < \bar{t}_{A_i}. \tag{12}$$

This can be achieved as follows: The method described above is used to add the estimated cycle duration to a base timestamp as long as

$$\widehat{t}_{A_i} \leq \bar{t}_{A_i}, \tag{13}$$

i.e. as long as the predicted arrival time is earlier than the real arrival time. This means the prediction is a more adequate approximation of the exact timestamp than the measured arrival time: Assuming that the Kalman estimation $\widehat{\Delta}t_i$ is a sufficiently accurate approximation to the exact cycle duration Δt_i, clearly

$$t_{A_i} < \widehat{t}_{A_i}, \tag{14}$$

hence from equation (13) and (14)

$$\left|t_{A_i} - \widehat{t}_{A_i}\right| \leq \left|t_{A_i} - \bar{t}_{A_i}\right|. \tag{15}$$

Consequently one would use $\widehat{t}_{A_i}$ instead of $\bar{t}_{A_i}$ to approximate the true timestamp. On the other hand, once

$$\widehat{t}_{A_i} > \bar{t}_{A_i}, \tag{16}$$

as the arrival time $\bar{t}_{A_i}$ cannot be earlier than the true timestamp t_{A_i}, one gets analogously

$$\left|t_{A_i} - \widehat{t}_{A_i}\right| > \left|t_{A_i} - \bar{t}_{A_i}\right|, \tag{17}$$

which means that the measured arrival time is a better approximation of the exact timestamp than the estimation. Hence a new estimation

$$\widehat{t}_{A_i} := \bar{t}_{A_i} \tag{18}$$

is defined and used as a new base timestamp (or new anchor respectively). Again the estimated cycle duration is added to this new base timestamp as long as (13) is fulfilled, otherwise a new base timestamp is set according to (18) and so on. One therefore obtains a new estimated latency based on these improved timestamps:

$$\widehat{\Delta}\tau_i := \widehat{t}_{A_i} - t_{A_i}. \tag{19}$$

By applying this method recursively, it is clear that the latency $\widehat{\Delta}\tau_i$ is minimized,

$$\widehat{\Delta}\tau_i \longrightarrow \min \quad (i \longrightarrow \infty), \tag{20}$$

assuming sufficiently accurate Kalman estimations. Hence after some iterations, $\widehat{t}_{A_i}$ is the best possible approximation to the exact timestamp t_{A_i}. After a certain amount of iterations, the base timestamp will not change significantly anymore due to (20), so the method has stabilized. Yet the problem remains that $\Delta\tau_i$ is still unknown. The estimated minimal latency $\widehat{\Delta}\tau_i$ cannot be explicitly given either due to the unknown timestamp t_{A_i} in (19), although it is known to be minimized. A solution to this problem will be given in chapter III-C.

The algorithm for estimating the minimal sensor specific delay is summarized in algorithm II. A graphical presentation can be found in Figure 4.

Algorithm II
MINIMAL DELAY ESTIMATION ALGORITHM

%initial value
$\widehat{t}_{A_1} = \bar{t}_{A_1}$
for $i = 1$ to n **do**
  calculate $\widehat{\Delta}t_i$ via algorithm I
  %first timestamp prediction
  $\widehat{t}_{A_{i+1}} = \widehat{t}_{A_i} + \widehat{\Delta}t_i$
  if $\widehat{t}_{A_{i+1}} > \bar{t}_{A_{i+1}}$ **then**
    %timestamp corrected
    $\widehat{t}_{A_{i+1}} = \bar{t}_{A_{i+1}}$
  end if
end for

C. Synchronization using a UTC reference clock

Now consider the problem of synchronizing data from different sensors that have been timestamped using the method described in section III-B. First, it is assumed that there is one sensor providing exact UTC hardware timestamps that can be used as a reference. Such an exact timestamp can be set, e.g., by a parallel hardware device called SyncBox.

To get an exact timestamp from the camera, the SyncBox sends out a trigger pulse with an associated timestamp through

a separate data channel to the data acquisition system (e.g. the non-realtime PC). Both timebases of the SyncBox and the PC are synchronized with GPS. Thus all timestamps from sensors without a trigger input and the timestamp from the reference sensor can be matched. Due to the special hardware used for the SyncBox, trigger and generated timestamp differ only by about 5μs. See Figure 5 for details.

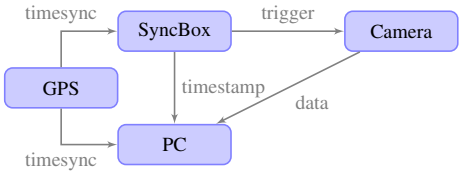

Fig. 5. System overview: camera triggered by a SyncBox

The test vehicle is equipped with the reference sensor (e.g. a camera with a SyncBox) and several other sensors that have to be synchronized. In addition, a calibration object is used which consists of several components to which the used sensors respond best. For example in the case of a radar and a camera, one would use a corner reflector as well as a checkerboard pattern. These components have to be arranged so their centers lie on the same vertical axis.

The calibration object is now placed in front of the test vehicle at a distance of, e.g., 100 meters and a curved approach towards the calibration object is performed, for example a sine wave-form trajectory. Each sensor provides the measured position of the object, which is timestamped as soon as possible according to III-B. It is assumed that the spatial estimates contain negligible error. Hence the measurements from distinct sensors only differ in the associated timestamps due to the sensor specific latencies $\Delta\tau_i^{s_j}$.

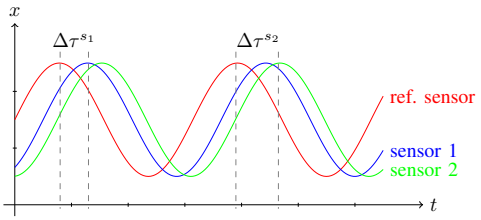

Fig. 6. Sensor specific latencies between a reference sensor and two additional sensors

The position of the object with respect to the vehicle changes according to the driven trajectory. Plotting the relative object position for each sensor with respect to the sensor specific timestamps results in sine curves with a difference of phase (see Figure 6). Now the sine curve of the reference sensor is compared with one of the other sensors. As the reference sensor provides the exact timestamps, the corresponding sine curve is the correct trajectory. The phase shift between the reference trajectory and one of the other trajectories represents exactly the sensor specific minimal latency

$$\Delta\tau^{s_j} := \min_i \Delta\tau_i^{s_j}. \tag{21}$$

Given that the Kalman estimations are sufficiently accurate, the approximation $\widehat{\Delta}\tau_i^{s_j}$ converges towards this minimal latency due to (20) and hence the approximated latency is nearly constant after some time. Thus one can now precisely tell the minimal latency for each sensor according to the phase shift. Consequently, the software timestamps from algorithm II can be corrected by simply subtracting the latency:

$$t'_{A_i} := \widehat{t}_{A_i} - \widehat{\Delta}\tau_i^{s_j}. \tag{22}$$

Using these corrected timestamps instead of the software timestamps $\widehat{t}_{A_i}$, all the sine curves in Figure 6 coincide. Consequently, the temporal synchronization is completed and one can tell the exact timestamps, even for sensors that do not provide such information.

A similar procedure using a curved drive has been proposed in [7]. On the one hand, no UTC reference timestamps are used, but the GPS position of the ego vehicle is used as a reference and converted into a temporal error. The disadvantage of this approach is that the GPS position is often not accurate enough, resulting in very rough estimated latencies. On the other hand, the approach cannot be generalized to the case without GPS reference data, in contrast to the approach presented in this paper, see the next section.

D. Synchronization without exact reference clock

The setup described in chapter III-C is applicable only in research situations where a specially equipped research vehicle can be used. In practical applications, usually no reference sensor providing exact UTC timestamps is available. In this case, one must dispense with exact absolute timestamps. Yet one can temporally synchronize measurements from different sensors by correcting the phase shift of the sine curves as described above. The absolute timestamps are not obtained but timestamps relative to the other sensors. This is accurate enough for most applications since the focus is often put on the fusion of different sensors, not necessarily on the absolute UTC time of a measurement. For sensor fusion in general, the relative timestamp is mostly sufficient.

The main advantage of the proposed algorithm remains that the sensor specific delays are taken into consideration. Usually the synchronization is done up to a constant error using software timestamps only, as in [5]. What is often neglected is the fact that there is not one constant error but one for each sensor, and as these errors may differ significantly, they cannot be ignored. This fact is adequately compensated using the method described earlier, even if there are no absolute UTC timestamps.

IV. Experimental Results

A. Simulation

In this section, simulated results are presented to test the described approach. First, the estimation of the cycle duration was simulated as described in section III-A. The true sampling timestamps were generated according to (4) using $\Delta t := 40$ms as a base cycle duration and a constant additive drift of $d_i := 0.001$ms $(i = 1, 2, ...)$. The assumption of a drift is

realistic due to, e.g., temperature variation of the quartz in the sensor. Note that this drift will not be constant in reality, but corresponding to the temperature, for example. Yet it can be assumed constant during a certain period of time.

A jittering measurement latency $\Delta\tau_i$ was produced according to (2) with mean $\mu := 30\text{ms}$ and variance $\sigma^2 := 0.1\text{ms}^2$, leading to a jittering disturbed cycle duration $\overline{\Delta}t_i$ as described in (5). The cycle duration was estimated via a linear Kalman filter according to algorithm I. The results are presented in Figure 7.

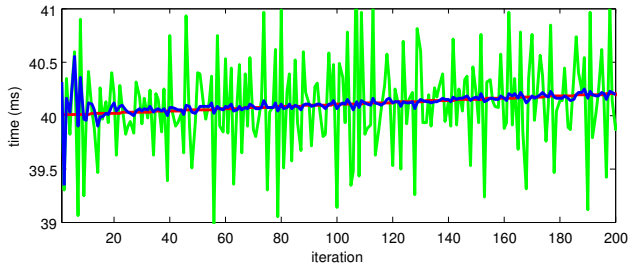

Fig. 7. Simulation results for the cycle duration: true cycle duration Δt_i with drift (red), jittering and drifting cycle duration $\overline{\Delta}t_i$ (green) and Kalman estimation $\hat{\Delta}t_i$ (blue)

As one can see, the jitter of the disturbed green curve can amount to several milliseconds. The performance of the Kalman filter depends among other things on the selection of the process and measurement noise covariance; in this case a measurement noise covariance of $R := 0.1(\text{ms})^2$ and a process noise covariance of

$$Q := \begin{bmatrix} 10^{-6} & 0 \\ 0 & 10^{-6} \end{bmatrix} (\text{ms})^2$$

was used. The Kalman filtering process results in a smoothed curve depicted in blue that approximates the true cycle duration curve in red. It is plain to see that filtering improves the jittering timestamps significantly.

Now the estimation of the minimal latency via the method proposed in section III-B is demonstrated. First, the exact drifting cycle duration Δt_i is assumed to be known. Then algorithm II is applied using this exact cycle duration instead of the Kalman estimation $\hat{\Delta}t_i$, leading to a minimal latency according to (21). The result is shown in Figure 8. The red curve demonstrates the predicted effect of minimizing the latency until a relative stabilization can be observed. Further iteration of the algorithm will not lead to a significant change in this minimal latency.

Now as the exact cycle duration Δt_i is not known in practical applications, the Kalman estimation $\hat{\Delta}t_i$ that was shown in Figure 7 is used. Applying algorithm II results in the estimated minimal latency $\hat{\Delta}\tau_i$ that is shown via the blue curve in Figure 8. As one can see, the estimation still jitters, but this effect is negligible. Hence with filtered instead of exact data the algorithm is still applicable. Note that the minimal latency can only be calculated in theoretical simulations due to the unknown true sampling time t_{A_i} needed in (19). In real-world applications one would need the procedure described in section III-C.

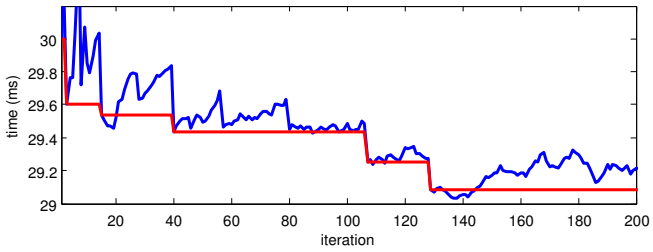

Fig. 8. Simulation results for the minimal latency: exact minimal latency $\Delta\tau_i$ (red) and estimated minimal latency $\hat{\Delta}\tau_i$ (blue)

One may wonder why the blue curve in Figure 8 is not monotonically decreasing, as one would assume intuitively according to the minimization procedure in algorithm II. This effect is caused by the small errors in the filtering process that lead to errors in the approximated timestamps because of (11). A small additive positive error leads to an overestimation of the minimal latency. Yet this effect is of no concern for the performance of the algorithm.

There is another much more critical effect that must be avoided. It must be assured that the Kalman filter is applied in a way that there is no systematic error, as one knows from the Kalman filter theory. E.g., if only the cycle duration is estimated without the drift, using a one-dimensional filtering process, the cycle duration is systematically underestimated because of the positive additive drift in the example on hand. With a systematic small error $0 < \varepsilon_i \ll 1$, one gets an estimated cycle duration

$$\widehat{\Delta}t_i = \Delta t_i - \varepsilon_i. \tag{23}$$

Of course the assumption of mean value free noise in Kalman filtering would be violated, but this may not be critical in other applications. Using such an estimation in the proposed method on the other hand can cause dramatic effects, however small these systematic errors may be. The reason for this purpose lies in the fact that all estimated drifts are added in this method, as one can see in equation (11). As long as (13) is fulfilled, one would get approximated new timestamps

$$\widehat{t}_{A_{i+1}} = \widehat{t}_{A_1} + \sum_{k=1}^{i} \widehat{\Delta}t_k = \widehat{t}_{A_1} + \sum_{k=1}^{i} \Delta t_k - \sum_{k=1}^{i} \varepsilon_k. \tag{24}$$

Clearly e.g. in the case of a constant error $\varepsilon_i = \varepsilon > 0$,

$$\sum_{k=1}^{i} \varepsilon_i \longrightarrow \infty \quad (i \longrightarrow \infty) \tag{25}$$

As a cycle duration usually only accounts for several milliseconds, these errors may add up very quickly. One has to pay particular attention to the correct application of the filtering process.

In order to assure the correct filtering without effects like in (25), an additional test can be included in algorithm II: Due to the asymptotic behavior (20), one would expect

$$\left|\widehat{t}_i - \overline{t}_i\right| < \text{const.} \tag{26}$$

Consequently in each iteration, one could test if

$$\left|\widehat{t}_i - \overline{t}_i\right| < \overline{\Delta} t_i, \tag{27}$$

is fulfilled, otherwise the Kalman estimation is not accurate enough and has to be adjusted.

Undesirable effects similar to (25) can be induced by rounding errors. E.g., if the Kalman estimates are done up to nanosecond accuracy, but only microseconds are used in algorithm II and the rounding is not accurately done, these small rounding errors may add up as well, as described above. In this case one should avoid rounding and use more precise timestamps instead.

B. Practical evaluation with real-world data

In this section, practical tests of the algorithms using real-world measurements are presented. The curved approach proposed in section III-C was done using a test vehicle equipped with two different cameras. The first camera, denoted by camera 1, is an industrial camera which can be externally triggered. Hence camera 1 can serve as a reference sensor, and the delay can be determined immediately using the SyncBox,

$$\Delta\tau^{\text{camera 1}} = 29 \text{ ms}. \tag{28}$$

The second camera, in the following denoted by camera 2, is a production camera that cannot be triggered, thus the measurement latency is unknown.

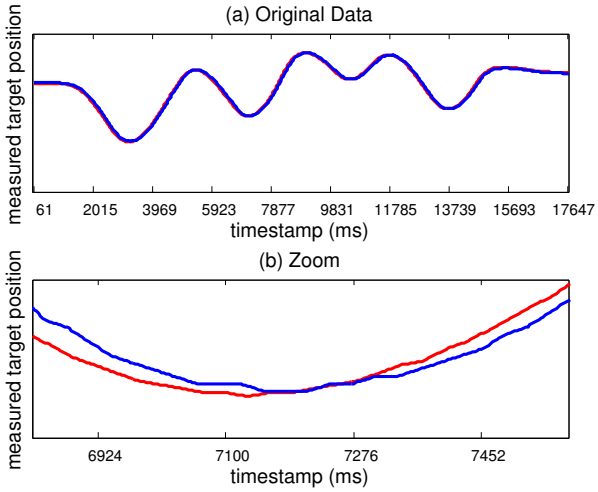

Fig. 9. Real-world evaluation of the curved approach: target position measured via camera 1 with SyncBox timestamps (red) as ground truth and camera 2 (blue) with delayed timestamps

The results of the curved approach are presented in Figure 9 (note that the UTC timestamps are shortened for convenience). The target position measured via camera 1 using GPS timestamps from the SyncBox served as ground truth. The delayed measurements from camera 2 are depicted in blue. Clearly one can see the time delay of the blue curve with respect to the red curve from the reference sensor.

This delay can be calculated using a cross correlation. In the current sensor setup, a delay of

$$\Delta\tau^{\text{camera 2}} = 42 \text{ ms} \tag{29}$$

according to (21) was determined. Thus the data from camera 2 can be temporally corrected as described in equation (22).

The same evaluation was done using a radar and a camera. The radar data is jittering more due to the relatively rough azimuth resolution of the radar. Nevertheless the delay of the radar data can be determined as described above, resulting in an approximate radar latency of

$$\Delta\tau^{\text{radar}} = 128 \text{ ms}. \tag{30}$$

Hence all used sensors can be exactly timestamped according to (22).

V. Conclusion

A filter-based method for achieving accurate timestamping and temporal synchronization in a multi-sensor setup was proposed in this paper. A procedure for estimating the sensor specific cycle durations and measurement latencies between the unknown sampling timestamps and the delayed software timestamps was described. These latencies were minimized resulting in a correction of the jitter as well as in a best possible software timestamp. In a multi-sensor setup, a method for temporal synchronization of the different sensors and hence compensation of the different latencies was proposed. The described algorithms were tested using simulated examples as well as real-world data.

Acknowledgment

This work was partially supported by the project Ko-PER, which belongs to the project initiative Ko-FAS funded by the German *Bundesministerium für Wirtschaft und Technologie* (Federal Department of Commerce and Technology) under grant number 19S9022B and 19S9022H. This support is gratefully acknowledged.

References

[1] C. Kwok, D. Fox and M. Meila. *Real-time particle filters*. IEEE, Sequential State Estimation, 92(2), 2004.

[2] M. Kais, D. Millescamps, D. Bétaille, B. Lusetti and A. Chapelon. *A Multi-Sensor Acquisition Architecture and Real-Time Reference for Sensor and Fusion Methods Benchmarking*. Intelligent Vehicles Symposium 2006, June 13-15, 2006, Tokyo, Japan.

[3] V. Di Lecce, A. Amato and M. Calabrese. *GPS-aided lightweight architecture to support multi-sensor data synchronization*. IEEE International Instrumentation and Measurement Technology Conference, Victoria, Vancouver Island, Canada, May 12-15, 2008.

[4] O. Bezet and V. Cherfaoui. *On-Line Timestamping Synchronization in Distributed Sensor Architectures*. Proceedings of the 11th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS'05), 2005.

[5] J.-O. Nilsson and P. Händel. *Time Synchronization and Temporal Ordering of Asynchronous Measurements of a Multi-sensor Navigation System*. Position Location and Navigation Symposium (PLANS), 2010.

[6] R. Kalman. *A New Approach to Linear Filtering and Prediction Problems*. Journal of Basic Engineering 82 (1960), Nr. 1, S. 35-45.

[7] R. Lindl. *Tracking von Verkehrsteilnehmern im Kontext von Multisensorsystemen*. PhD thesis, TU München, 2009.