Robust recursive network clock synchronization

John-Olof Nilsson and Peter Händel

Department of Signal Processing, ACCESS Linnaeus Centre
KTH Royal Institute of Technology
Osquldas väg 10, SE-100 44 Stockholm, Sweden

***Abstract*—A simple and statistically robust method for passive clock synchronization in sensor networks is presented. The method is not limited to passive (one-way communication) synchronization, but this scenario justifies the method. The recursive nature of the method and the targeted passive setup mean that it adds a minimum of requirements on the system in which it is used. Statistical characteristics of the method are quantified and real measurements are used to illustrate the robustness and performance gain relative to a naive Kalman filter based clock synchronization. Finally, C++ code that implements the suggested clock synchronization method, is provided in this article.**

I. Introduction

Time information is needed for the control and state estimation of any dynamic system. For distributed systems, this adds the requirement to perform clock synchronization between different parts of the system; see [1,2,3,4] for reviews on the topic. The clock synchronization, i.e. timing estimation, must be based on distributed time measurements and communication in/between different parts of the system [5,6]. For distributed systems, due to the communication solutions (packet networks), there are often no guarantees of communication delays. The same lack of guarantee also arises from prevailing non-real-time operating systems or application programming interfaces. Consequently, to perform time-synchronization in such systems or with such interfaces, statistical robustness to large delays is desirable [7]. Unfortunately, this is not ensured in the timing estimation of many clock synchronization algorithms. See [8] and references therein for a general treatment of the statistical robustness concept.

With two-way communication (message exchange), large delay outliers can easily be detected. Unfortunately, with only one-way communication, large outliers cannot easily be detected without relying on previous clock relation estimates and thereby introducing feedback, and potential stability/dead lock problems in the system. Two-way message exchange typically requires special purpose communication (unless it can piggy-back on some other communication). Consequently, it may not be desirable, or even possible, to implement this special clock synchronization communication. However, any communication carries time information. This may be exploited to perform a passive clock synchronization with the only requirement that the transmitted data is time stamped at its source or sent with regular intervals (which is commonly the case for sensory data) [6]. This setup is illustrated in Fig. 1. Letting the clock synchronization piggy-back on one-way communication adds a minimum of requirements on the system in which it is to be implemented.

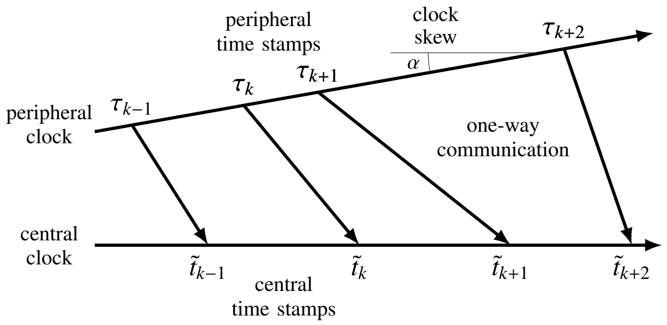

Fig. 1: A passive network clock synchronization setup. A central node is (passively) receiving time stamped information from a peripheral node with varying delays. Both nodes have their own clock and the problem is to estimate the timing of peripheral time stamping events with respect to the central clock.

Statistical robustness could be achieved with brute force by applying a particle filter [1] or batch processing with consistency checks/outlier rejections. However, the complexity, the computational cost, and memory requirements of such solutions make them undesirable. For simplicity, while still retaining reasonable performance, recursive estimation methods are desirable. We have shown earlier that a Kalman filter works fine for benign communication delays [9]. Unfortunately, as will be shown, for packet networks and malign communication delays it will introduce systematic errors due to lack of statistical robustness. Consequently, in this article we suggest a simple recursive statistically robust network clock synchronization method. The method can operate with only one-way communication but is not limited to this. The synchronization method exploits robust estimation techniques to gracefully discard time measurements resulting from large communication delays. It features an easily adjustable likelihood function for the communication delay. A generally applicable likelihood function is suggested and the resulting statistical robustness is demonstrated in terms of the timing estimation influence function. Finally, the operation of the estimation is demonstrated with real data from an Android implementation for a cooperative pedestrian localization system [10].

Reproducible research: C++ code that implements the suggested algorithm is provided in this article.

II. Clock and communication models

There are two physical processes whose timing characteristics have to be considered/modeled to perform the timing estimation/clock synchronization, the clocks and the communication delays. The estimation problem is to estimate the time t_k, where k is a time index, with respect to the central clock of the time stamped peripheral event, given peripheral time stamps τ_k, central time stamps $\tilde{t}_k$, and models for the clock relation and the communication delays.

The communication delays may have non-stationary heavy tailed characteristics, while the clocks will have relatively benign, stable, and stationary characteristics. For most state estimation problems, the clocks will have a short time stability that is much greater than the dynamics bandwidth of the system, i.e. we are estimating the physical state of the system. Since the accuracy of our clock pace is so much better than the communication jitter, a simple model for the clock will suffice. An integrated and discretized Wiener process is a suitable choice. For one-way communication, the information that time stamps τ_k of the source carry is the time with respect to the peripheral clock which has passed since the last time stamp $dt_k = \tau_k - \tau_{k-1}$, i.e. the difference between the current time stamp and the preceding time stamp.[1] This gives the process model

$$\begin{bmatrix} t_k \\ \alpha_k \end{bmatrix} = \begin{bmatrix} 1 & dt_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_{k-1} \\ \alpha_{k-1} \end{bmatrix} + \begin{bmatrix} dt_k \\ 0 \end{bmatrix} + \begin{bmatrix} \bar{w}_k \\ w_k \end{bmatrix} \tag{1}$$

where α_k is the clock skew, of the peripheral clock relative to the central clock, modeled as a sampled Wiener process with the integrated process noise w_k and $\bar{w}_k$ is the integrated Wiener process. See [6] for further details.

Each incoming communication is assumed time stamped with a time $\tilde{t}_k$ with respect to the central clock. This will give the measurements related to the process model. The time stamps will simply equal the time of the measurement event plus a communication delay,

$$\tilde{t}_k = t_k + v_k. \tag{2}$$

The communication delay v_k will contain both deterministic components (a minimum delay) and random components. In case the communication is polled, i.e. triggered by some command from the central node, the central node can timestamp both the outgoing and incoming communications. This will mean that there are two time stamps. However, the same model can still be exploited by simply taking the mean of the time stamps as a single time stamp of the event. In this case v_k becomes the delay difference which can be both positive or negative.

In the case of a one-way communication (or unsymmetrical delays for a two-way communication) the minimum delay will naturally be unobservable. However, there will still be a mean delay relative to which we can estimate the timing. Since timing is of interest to start with, there will most likely be some dynamics in the system. These dynamics can be used to estimate the mean delay, see e.g. [11,12].

[1] In case there is no time stamps but there is a fixed communication (sampling) rate, the sampling period (potentially unknown) with respect to the peripheral clock carries the same information [6].

III. Robust clock synchronization

The random components of the communication delay v_k is most likely correlated in time, non-stationary, and could potentially show heavy tailed characteristics. Consequently, an accurate model for the delay will often not be available and we will simple have to make the timing estimation robust to the malign delay characteristics. Fortunately, as previously indicated, the clock pace α_k is most likely good enough to bridge over any correlation in v_k. This means that for the propagation in time, a simple covariance based solution will suffice. However, for the measurement updates based on the measurement model (2), statistical robustness needs to be incorporated.

A. Time update

The well behaved clock means that over time we can live in an assumed Gaussian world. Propagation of state estimates is trivially made with

$$\begin{bmatrix} \hat{t}_k^- \\ \hat{\alpha}_k^- \end{bmatrix} = \begin{bmatrix} 1 & dt_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{t}_{k-1}^+ \\ \hat{\alpha}_{k-1}^+ \end{bmatrix} + \begin{bmatrix} dt_k \\ 0 \end{bmatrix} \tag{3}$$

where $\hat{\cdot}_k^-$ and $\hat{\cdot}_k^+$ are the prior and posterior means, respectively, and $\hat{t}_k^{-/+}$ and $\hat{\alpha}_k^{-/+}$ are the (estimated) time of the peripheral time stamped event and the peripheral clock skew (relative to the central clock), respectively. Further, the system and covariance matrices are

$$F_k = \begin{bmatrix} 1 & dt_k \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad Q_k = \sigma^2 \begin{bmatrix} \frac{dt_k^3}{3} & \frac{dt_k^2}{2} \\ \frac{dt_k^2}{2} & dt_k \end{bmatrix}$$

with which the covariance is propagated by

$$P_k^- = F P_{k-1}^+ F^\top + Q_k \tag{4}$$

where σ^2 is the variance of the Wiener process and $P_k^- = \mathrm{cov}([\hat{t}_k^-, \hat{\alpha}_k^-])$ and $P_{k-1}^+ = \mathrm{cov}([\hat{t}_{k-1}^+, \hat{\alpha}_{k-1}^+])$ are the prior and posterior covariance, respectively.

B. Measurement update

The measurements (central time stamp $\tilde{t}_k$) are only dependent on a subset of the system states. Consequently, a marginalization and sampling based update as suggested in [10,13] could potentially be used. Unfortunately, the large numerical values of the time estimates $\hat{t}_k$ and time stamps $\tilde{t}_k$ potentially in the order of 10^9 (POSIX/Unix time), on one side, and the small numerical values of the clock skew estimates $\hat{\alpha}_k$ and the measurement residuals potentially in the order of 10^{-6} make the measurement update methods not directly applicable due to potential numerical problems.

The marginalization and sampling based updates are based on the idea that the posterior (conditional statistics) is first calculated from the subset of the state vector of which the measurement is dependent, i.e. $\hat{t}_k$. Following this the total posterior is calculated based on the assumption that the marginal posterior is Gaussian. Assuming that the subset of the state vector is low dimensional, the marginal posterior can be calculated with great freedom by sampling the (assumed)

Gaussian prior and reweighting the samples by a likelihood function. Switching between the Gaussian and non-Gaussian assumptions is obviously somewhat statistically inconsistent, but it is a common practice to get practical estimation algorithms and it is used in the popular sigma-point (unscented) Kalman filters, for example. Further, as previously argued, the delays will often be non-stationary, so optimality claims are hard to make in practice anyway.

The sampling of the prior can efficiently be implemented by precalculated samples and weights of the unit normal distribution. Denote the covariance components by

$$P_k^- = \begin{bmatrix} P_t^{-/+} & P_{t,\alpha}^{-/+} \\ P_{\alpha,t}^{-/+} & P_\alpha^{-/+} \end{bmatrix}$$

where for notational convenience the time indices of the subcomponents have been dropped. Samples of the assumed Gaussian prior of t_k are chosen as

$$\hat{t}_k^- + s_k^{(i)} \quad \text{where} \quad s_k^{(i)} = u^{(i)} \cdot (P_t^-)^{1/2}$$

where $u^{(i)}$ are samples of the unit normal distribution. This way the weights of the samples become time independent (ignoring normalization) since

$$w_{(i)}^- = \frac{1}{\sqrt{2\pi P_t^-}} e^{-\frac{(s_k^{(i)})^2}{2P_t^-}} \sim \frac{1}{\sqrt{2\pi}} e^{-\frac{(u^{(i)})^2}{2}}$$

where $\sim$ denotes proportionality, and can therefore be precalculated. We have typically come to use 13 uniformly distributed samples over the range ± 3 (standard deviations).

To get weighted samples of the posterior, the prior weights are to be reweighed with the likelihood function. The likelihood function is the key to the statistical robustness and should be heavy-tailed to achieve this. In principle, the likelihood function should be one-tailed since the delay must be positive (for one-way communication). However, since the prior is sampled over a finite range, a two-tailed distribution is needed to avoid a potential lock in the system, otherwise this range could fall outside the support of the likelihood function. A suitable and easily implemented function is the Cauchy distribution giving the likelihood weights (ignoring normalization)

$$\ell_{(i)} = \frac{1}{\gamma^2 + (s_k^{(i)} + (t_k^- - \tilde{t}_k))^2} \tag{5}$$

where γ is the scale of the distribution. However, any two-tailed likelihood function could be used. With the likelihood weights, the posterior weights are given by

$$w_{(i)}^+ = w_{(i)}^- \ell_{(i)}.$$

This in turn gives the posterior mean

$$\hat{t}_k^+ = \hat{t}_k^- + \frac{1}{w} \sum_i s_k^{(i)} w_{(i)}^+$$

and covariance

$$P_t^+ = \frac{1}{w} \sum_i (s_k^{(i)} + (\hat{t}_k^- - \hat{t}_k^+))^2 w_{(i)}^+$$

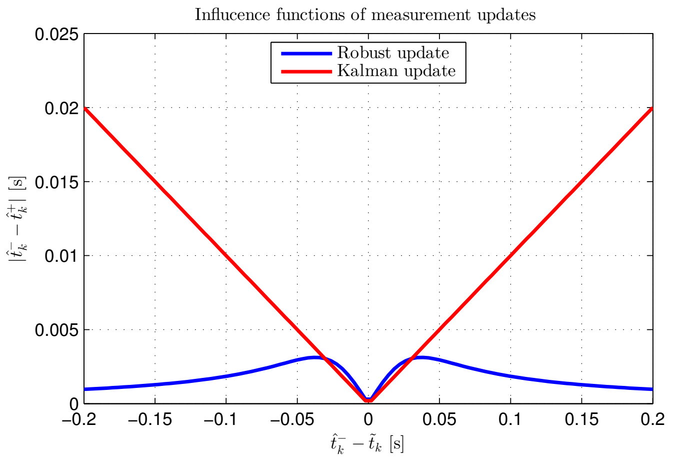

Fig. 2: The influence functions of the suggested robust measurement update and the corresponding Kalman measurement update for $P_t^- = 10^{-4}$ (standard deviation 0.01 [s]), $\gamma = 0.03$ [s], and a measurement noise standard deviation for the Kalman update of 0.03 [s]. The robust update gracefully neglects measurements beyond three standard deviations of the Gaussian prior while the Kalman update indiscriminately corrects the state estimate for the measurements.

where $w = \sum_i w_{(i)}^+$ is a normalizing factor. The samples and weights could be kept for the next measurement update, to implement a marginal particle filter. However, for simplicity, we prefer to use the static samples $u^{(i)}$ and weights $w_{(i)}^-$ at every update and to keep only the posterior mean and covariance.

With the posterior mean $\hat{t}_k^+$ and covariance P_t^+ and assuming that the posterior is Gaussian, the conditional mean and covariance of the full state can be calculated with updates rules presented in [10]. However, the rules will be greatly simplified in the scalar case and have to be rewritten to avoid numerical problems. Straightforward manipulations give the conditional mean, covariance, and cross covariance of α_k

$$\begin{aligned} \hat{\alpha}_k^+ &= \hat{\alpha}_k^- + d\alpha_k \\ P_\alpha^+ &= P_\alpha^- + u\,(u\,P_t^+ - P_{\alpha,t}^-) \\ P_{\alpha,t}^+ &= u\,P_t^+ \end{aligned}$$

where $u = P_{\alpha,t}^- / P_t^-$ and $d\alpha_k = u\,(t_k^- - t_k^+)$.

C. Robustness

The suggested measurement update has a complexity comparable to a standard Kalman measurement update, but the result is fundamentally different. No matter how wrong a measurement is, the Kalman measurement update will just treat it and the prior as very unlikely events and produce a result in between. Consequently, large outliers will give erroneous results. This can be mitigated by rejecting measurements with too large measurement residuals (difference between predicted and measured time). However, there is always the risk that the predication is actually wrong. This could cause a lock in the system after which all of the measurements are rejected. Of course this could be solved with additional logics. However, in the end this would make the Kalman update more complex than the suggested robust update.

Instead of using logics to reject outliers, the suggested method exploits the heavy tailed likelihood function to gracefully neglect measurements with large measurement residuals. A large residual will make the likelihood function (5) be evaluated far out on the tail. There the difference between the weights will be small, and therefore the magnitude of the correction $\hat{t}_k^- - \hat{t}_k^+$ will be small. This property can be quantified by the influence function of the update. The influence function is the value $|\hat{t}_k^- - \hat{t}_k^+|$ as a function of the measurement residual $\hat{t}_k^- - \tilde{t}_k$. The influence function of both the Kalman measurement update and the suggested update are shown in Fig. 2 for the prior covariance $P_t^- = 10^{-4}$ (standard deviation 0.01 [s]) and $\gamma = 0.03$ [s]. For the Kalman update the standard deviation of the measurement noise is also 0.03 [s]. For small measurement residuals, the robust update and the Kalman update are seen to behave approximately the same. However, the robust update is seen to gracefully neglect measurements beyond 3 standard deviations while the Kalman update indiscriminately includes the measurements and produces corrections that are proportional to the measurement residual.

IV. Code

A basic C++ class that implements the robust synchronization method is provide in Table I. The class method `double sync(double,double)` of the class `class Rsync` returns the timing estimate $\hat{t}_k^+$ given the arguments: central time stamp `double t_c` and peripheral time stamp `double t_p`. The private class variables `double x[2]` and `double P[3]` hold the state and covariance estimates, respectively. Note that since the covariance matrix is symmetric only three values need to be stored. The constant `static const int N_pts` contains the number of samples used. The array `static const double sp_unit[N_pts]` contains the samples of the unit normal distribution $\mathcal{N}(0,1)$ while the array `static const double prior[N_pts]` contains the corresponding values/prior weights, normalized with the value of the maximum weight. An instance of `class Rsync` is created for each peripheral device and as soon as there is an incoming information package, a time stamp is taken and the class method `double sync(double,double)` is called to perform the timing estimation/clock synchronization. Increasing peripheral time stamps are assumed.

The parameters of the method is hardcoded in the method. The constants `static const double sigma2, g2;` are the process noise covariance σ^2 and the Cauchy distribution scale squared γ^2, respectively. The clock skew and the state estimate covariance variables are given hardcoded initial values in the constructor.

V. Experimental results

The robust clock synchronization has been implemented on an Android platform to support a cooperative pedestrian localization system [10]. Measurements are time stamped at its peripheral source and transmitted to a central node. Upon arrival the information packages are time stamped again with respect to the central clock. The timing is estimated followed by a sensor fusion based on this timing.

TABLE I: C++ class that implements the suggested robust clock synchronization method. An `Rsync` object is allocated for each peripheral data source. The class method `sync(double,double)` returns the time estimate given the arguments: central time stamp `double t_c` and peripheral time stamp `double t_p`. Increasing peripheral time stamps are assumed.

```cpp
/* ----------------- Rsync.hpp ------------------ */
class Rsync {
public:
  Rsync();
  double sync(double t_c,double t_p);
private:
  double x[2], P[3], t_p_last;
  bool init;
  static const double sigma2 = 1e-10, g2 = 1e-2;
  static const unsigned int N_pts = 13;
  static const double sp_unit[], prior[];
};
/* -------------------------------------------- */


/* ----------------- Rsync.cpp ------------------ */
#include "Rsync.hpp"
#include <cmath>

Rsync::Rsync() {
  x[1]=0; P[0]=1.0; P[1]=0; P[2]=1e-6; init=false;
}

// Samples and weights of unit normal distribution
const double Rsync::sp_unit[N_pts] = {
  -3,-2.5,-2,-1.5,-1,-0.5,0,0.5,1,1.5,2,2.5,3 };
const double Rsync::prior[N_pts] = { 0.01111,
  0.04394,0.13534,0.32465,0.60653,0.88250,1.00000,
  0.88250,0.60653,0.32465,0.13534,0.04394,0.01111 };

// Estimate timing of time stamped event
double Rsync::sync(double t_c,double t_p){
  if(init){
    // Time update
    double dt = t_p-t_p_last; double dt2 = dt*dt;
    x[0] += (x[1] + 1.0)*dt;
    P[0] += 2.0*dt*P[1]+dt2*P[2]+0.33*dt2*dt*sigma2;
    P[1] += dt*P[2] + 0.5*dt2*sigma2;
    P[2] += dt*sigma2;

    // Measurement update
    double sp[N_pts], posterior[N_pts];
    double sqrtP0 = sqrt(P[0]), u = P[1]/P[0];
    double t_ud = 0, sum_post = 0;
    P[0] = 0;
    for(unsigned int i=0;i<N_pts;i++) {
      sp[i] = sp_unit[i]*sqrtP0;
      posterior[i] = prior[i]/(1.0+(sp[i]
              +(x[0]-t_c))*(sp[i]+(x[0]-t_c))/g2);
      sum_post += posterior[i];
      t_ud += sp[i]*posterior[i];
    }
    t_ud /= sum_post; t_ud += x[0];
    for(unsigned int i=0;i<N_pts;i++) {
      double tmp = sp[i]+(x[0]-t_ud);
      P[0] += tmp*tmp*posterior[i];
    }
    P[0] /= sum_post;
    x[1] += u*(t_ud-x[0]);
    x[0] = t_ud;
    P[2] += u*(u*P[0]-P[1]);
    P[1] = u*P[0];
  } else {
    x[0]=t_c; init=true;
  }
  t_p_last=t_p;
  return x[0];
}
/* -------------------------------------------- */
```

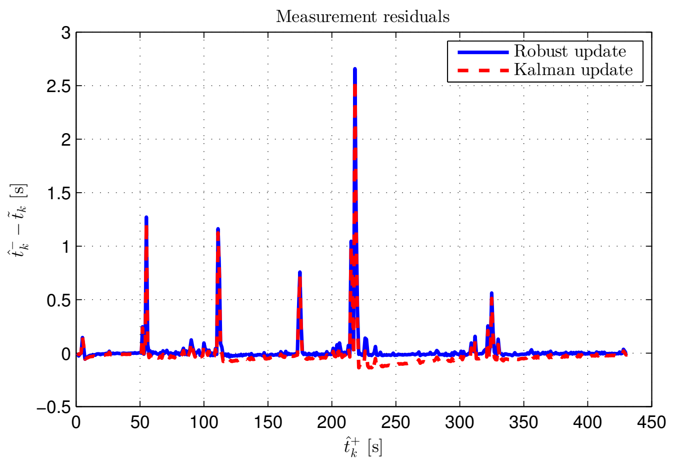

Fig. 3: Measurement residuals for the suggested robust clock synchronization and a naive Kalman based clock synchronization. After each large outlier, the Kalman filter is seen to have a systematic error which is not presented in the robust synchronization.

The communication in the localization system shows varying connectivity, and consequently the communication delays can be large. Fig. 3 shows the innovations of the timing estimation with apparent large outliers taken from a 7 [min] long run with the localization system. The robust synchronization will ignore the outliers, consequently they appear isolated, and there is no effect on subsequent residuals. In contrast, for the Kalman based synchronization, the outliers are seen to introduce systematic errors. After each outlier there is a decaying over-predication of the central time stamp/measurement.

VI. Conclusions

We have argued that network clock synchronization ought to be made statistically robust to be applicable to many distributed systems. The relative stability of the clocks makes this achievable by only making the measurement updates statistically robust. Thereby the complexity can be kept low compared with a full particle filter solution. A sampling based measurement update has been suggested to achieve the statistical robustness. The robustness has been demonstrated in terms of the influence function. Finally, real data has been used to demonstrate the effectivenss of the method and to compare it with a naive Kalman filter based clock synchronization.

References

[1] Y.-C. Wu, Q. Chaudhari, and E. Serpedin, "Clock synchronization of wireless sensor networks," *Signal Processing Magazine, IEEE*, vol. 28, no. 1, pp. 124–138, 2011.

[2] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: A survey," *Ad Hoc Networks (Elsevier*, vol. 3, pp. 281–323, 2005.

[3] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: A survey," *Network, IEEE*, vol. 18, no. 4, pp. 45–50, 2004.

[4] S. Lasassmeh and J. Conrad, "Time synchronization in wireless sensor networks: A survey," in *IEEE SoutheastCon 2010 (SoutheastCon), Proceedings of the*, pp. 242–245, 2010.

[5] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," *Commun. ACM*, vol. 21, pp. 558–565, July 1978.

[6] J.-O. Nilsson and P. Händel, "Timing estimation in distributed sensor and control systems with central processing," *CoRR*, vol. abs/1309.1864, 2013.

[7] Q. Chaudhari, "A simple and robust clock synchronization scheme," *Communications, IEEE Transactions on*, vol. 60, no. 2, pp. 328–332, 2012.

[8] A. Zoubir, V. Koivunen, Y. Chakhchoukh, and M. Muma, "Robust estimation in signal processing: A tutorial-style treatment of fundamental concepts," *Signal Processing Magazine, IEEE*, vol. 29, no. 4, pp. 61–80, 2012.

[9] J.-O. Nilsson and P. Handel, "Time synchronization and temporal ordering of asynchronous sensor measurements of a multi-sensor navigation system," in *Position Location and Navigation Symposium (PLANS), 2010 IEEE/ION*, (Palm Springs, CA, US), pp. 897–902, 3-6 May 2010.

[10] J.-O. Nilsson, D. Zachariah, I. Skog, and P. Händel, "Cooperative localization by dual foot-mounted inertial sensors and inter-agent ranging," *EURASIP Journal on Advances in Signal Processing*, vol. 2013:164, 2013.

[11] J.-O. Nilsson, I. Skog, and P. Handel, "Joint state and measurement time-delay estimation of nonlinear state space systems," in *Information Sciences Signal Processing and their Applications (ISSPA), 2010 10th International Conference on*, (Kuala Lumpur, Malaysia), pp. 324–328, 10-13 May 2010.

[12] S. Julier and J. Uhlmann, "Fusion of time delayed measurements with uncertain time delays," in *American Control Conference, 2005. Proceedings of the 2005*, (Portland, OR, USA), pp. 4028–4033 vol. 6, 8-10 June 2005.

[13] D. Zachariah, I. Skog, M. Jansson, and P. Handel, "Bayesian estimation with distance bounds," *Signal Processing Letters, IEEE*, vol. 19, no. 12, pp. 880–883, 2012.