The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Integrating Asynchronous Observations for Mobile Robot Position Tracking in Cooperative Environments

Andreu Corominas Murtra[1], Josep M. Mirats Tur[1], Alberto Sanfeliu[1,2]

Abstract— **This paper presents an asynchronous particle filter algorithm for mobile robot position tracking, taking into account time considerations when integrating observations being delayed or advanced from the prior estiamate time point. The interest of that filter lies in cooperative environments and in fast vehicles. The paper studies the first case, where a sensor network shares perception data with running robots that receive accurate obeservations with large delays due to acquisition, processing and wireless communications. Promising simulated results comparing a basic particle filter and the proposed one are shown. The paper also investigates a situation where a robot is tracking its position, fusing only odometry and observations from a camera network partially covering the robot path.**

I. INTRODUCTION

Position tracking of mobile robots is an active and large research field, given the key interest of the task [1], [2]. For autonomous navigation purposes, a mobile robot need to be robustly localized, either in a local frame or in a global one. This position tracking need to be done in real-time since the tracking output usually closes a control loop in charge of following a path to reach a given goal position. This role of the position tracking in the autonomous navigation framework imposes real-time constraints when computing it.

In most of the mobile robots, position tracking is computed by means of a filter process that integrates observations coming from different perception subsystems onboard the robot, and even other observations coming from remote observers not onboard the robot, when the robot is running in a *cooperatie environment* as that described in [3], where a sensor network shares perception information with a set of operating robots. Whether these observations are onboard or not, they have all an observation *time stamp*, that rarely coincides with the time that the observation data is available at the filtering processing unit. Moreover, the filtering process has to set a time stamp for its output estimation and this time neither coincides with the observation time stamps.

The above outlined time considerations become more serious in two cases of interest: cooperative robotics and fast vehicles. In the first case the observations arriving at filtering process may be done by remote observers (computers) connected with the tracking computer (usually onboard) by means of a wireless network. These observations suffer from a considerable delay provoked by acquisition, processing and communications. These delays can reach the order of hundreds of milliseconds, enough time for a mobile robot to perform non negligible displacements. In the second case, little delays, even provoked by latencies of the onboard sensors, are traduced to large displacements due to the high speed of the vehicle.

Previous works on considering delayed observations for position tracking purposes are based on Kalman filter [4], [5], but adapting particle filters to integrate asynchronous observations remains a less exploarated topic. In the first work, the authors proposes a close optimal solution, but assumes that the observations arrive during the last sampling interval. In the second case, the authors propose a fixed-lag Kalman filter that delivers the best estimate with a latency of p iterations, thus the best estimate is not available at the current iteration. Moreover these works only focuses on integration of *delayed* observations but in real applications, some observations arrives *after* prior estimates, thus the concept of *advanced* observation arises. This occurs in filters that integrates multiple observations, spending a while for each observation integration and, thus new observations can arrive after prior estimate time.

This paper presents a new framework for particle filter position tracking that integrates observations taking into account their time stamps. The filter does not propagate the particle set once per iteration as classical approaches do. The proposed algorithm propagates the particle set only when a new observation arrives with a time stamp greater than the last propagation. At this moment, the filter propagates with the kinematic model and the odometric observation and, then, integrates the observation as a delayed one. In order to integrate delayed observations, the proposed approach keeps an historic of the last estimations and backpropagate particles to compute observation models at positions where particles were expected to be at the observation time. The paper shows results of simulated experiments comparing the presented approach with a classic particle filter, showing clearly the improvements and potentialities of the new approach.

The paper is organized as follows: section II describes the implemented basic particle filter, used in this work to compare the performance of the proposed approach. Section III describes the aynchronous particle filter that integrates observations taking into account time considerations. Simulated results comparing the accuracy of both filters are presented in section IV. In order to show potentialities of the proposed algorithm, section V includes promising simulated

Authors are with: [1]Institut de Robòtica i Informàtica Industrial, IRI (UPC-CSIC). Barcelona, Spain. www-iri.upc.es. [2] Universitat Politècnica de Catalunya, UPC. Barcelona, Spain. www.upc.edu. [acorominas, jmirats, asanfeliu] @iri.upc.edu

Research conducted at the Institut de Robòtica i Informàtica Industrial of the Universitat Politècnica de Catalunya and Consejo Superior de Investigaciones Científicas. Partially supported by Consolider Ingenio 2010, project CSD2007-00018, CICYT project DPI2007-61452, and IST-045062 of the European Community Union.

results on tracking the position of a mobile robot using only odometry and observations coming from a camera network partially covering the robot path. Finally, the paper outlines the conclusions of the presented work.

II. BASIC PARTICLE FILTER

A particle filter is a recursive algorithm that estimates a probability density function of the state of a system given a set of observations and, optionally, a kinematic or dynamic model of that system. The representation of the density function is made by a set of samples as vectors in the state space, each one having a weight related with the likelihood that the system state has in that point given the observations. The pair formed by a sample vector and a weight is called a *particle*. Further details on particle filters and its applications on mobile robotics can be found in [1], [6].

Our implementation of the particle filter for mobile robot position tracking uses a geometric map, described with the standard Geographical Information Systems (GIS) format. The estimated state of the robot is a vector in the continuous space of positions in the map frame. Let be $X_r^\tau = (x_r^\tau, y_r^\tau, \theta_r^\tau)$ the robot *true* state at time τ. This true state remains always unknown and is the target variable to estimate. The output of the t^{th} iteration of the position tracking process is the position estimate $\hat{X}_r^t = (\hat{x}_r^t, \hat{y}_r^t, \hat{\theta}_r^\tau)$, the estimated covariance matrix $\hat{C}_r^t$ and the time stamp of these estimates τ^t. Please note that τ refers to continuous time and t indicates an iteration index.

Being $\mathcal{X}$ the state space, 3-dimensional and continuous,

$$X_r^\tau, \hat{X}_r^t \in \mathcal{X} = \{(x_{min}, x_{max}), (y_{min}, y_{max}), (-\pi, \pi]\} \tag{1}$$

That is, the robot is assumed to be always in the working area. If we integrate data coming from $N_B + 1$ observers the approximation made by the sample representation of the density function at iteration t can be written as:

$$p(X_r^\tau | \{o_k^\tau\}) \ \sim \ P^t = \{(X_i^t, w_i^t)\} \tag{2}$$

The above expression indicates that the probability density function is approximated with the set P^t formed by N_P particles ($i = 1..N_P$). The output estimate takes into account all the observations from the start of the filter execution, denoted as $\{o_k^\tau\}$, being o_k^τ a single observation with time stamp τ made by the k^{th} observer. An observation o_k^τ is assumed to be inside the k^{th} observation space,

$$o_k^\tau \in \mathcal{O}_k \tag{3}$$

When an observation is integrated at t^{th} iteration, we denote it as o_k^t. In order to compute a likelihood of the i^{th} particle given the observation o_k^t, we compute:

$$p(X_i^t | o_k^t) = L_k(o_k^t, o_k^s(X_i^t)) \tag{4}$$

where the L_k function is a likelihood function between two observations: the one made by the k^{th} observer, o_k^t, and the *expected* one, $o_k^s(X_i^t)$, computed using the k^{th} observation model. The likelihood function $L_k()$ is:

$$L_k : \mathcal{O}_k^2 \rightarrow \mathbb{R} \in [0, 1] \tag{5}$$

thus, we define a likelihood function for each observer ($k = 1..N_B$) and the outputs of such functions are always bounded to $[0, 1]$ interval. The $k = 0$ index is reserved for the odometric observation which is integrated by means of a kinematic model instead of a likelihood function.

Algorithm 1 summarizes an iteration of the implemented basic particle filter. Firstly, the **propagate()** function propagates the particle set with the last odometric increments and the kinematic model of th platform $f()$.

$$X_i^t = f(X_i^{t-1}, o_0^t); \ \forall i = 1..N_P \tag{6}$$

Algorithm 1 Basic particle filter iteration

INPUT: $P^{t-1}, o_k^t \ \forall k$
OUTPUT: $P^t, (\hat{X}^t, \hat{C}^t, \tau^t)$
  P^t=propagate(P^{t-1}, o_0^t)
  for $k = 1..N_B$ **do**
    for $i = 1..N_P$ **do**
      $p(X_i^t | o_k^t) = L_k(o_k^t, o_k^s(X_i^t))$
      $w_i^{t'} = w_i^{t'} \cdot p(X_i^t | o_k^t)$
    end for
  end for
  $(\hat{X}^t, \hat{C}^t, \tau^t)$ =setEstimate(P^t)
  $\hat{X}^{t^+}$=propagate($\hat{X}^t, o_0^{t^+}$)
  publish($\hat{X}^{t^+}, \hat{C}^t, \tau^t$)
  resampling(P^t)

After propagation, a **correction** loop integrates the available observations without taking into account time considerations. This correction step can be formalized as:

$$w_i^{t'} = \prod_{k=1}^{N_B} p(X_i^t | o_k^t); \ \forall i = 1..N_P, \ \forall k = 1..N_B \tag{7}$$

The **setEstimate()** function parametrizes the particle set as a Gaussian density function. This Gaussian estimation is computed in order to publish a close result ready to be used by other real-time processes but the particle set remains the genuine output of the particle filter. In this function the time stamp of the tracking process, τ^t is also set. To compute the Gaussian density parameters, a normalization is performed to assure that the sum of all weights is 1:

$$w_i^t = \frac{w_i^{t'}}{\sum_{j=1}^{N_P} w_j^{t'}}; \ \forall i = 1..N_P \tag{8}$$

The parameters of the Gaussian density function are:

$$\hat{x}_r^t = \sum_{i=1}^{N_P} x_i^t \cdot w_i^t; \quad (\hat{\sigma}_x^t)^2 = \sum_{i=1}^{N_P} (x_i^t - \hat{x}_r^t)^2 \cdot w_i^t \tag{9}$$

$$\hat{y}_r^t = \sum_{i=1}^{N_P} y_i^t \cdot w_i^t; \quad (\hat{\sigma}_y^t)^2 = \sum_{i=1}^{N_P} (y_i^t - \hat{y}_r^t)^2 \cdot w_i^t \tag{10}$$

$$\hat{\theta}_r^t = atan(\frac{\sum_{i=1}^{N_P} sin\theta_i^t \cdot w_i^t}{\sum_{i=1}^{N_P} cos\theta_i^t \cdot w_i^t}) \tag{11}$$

$$(\hat{\sigma}_\theta^t)^2 = \sum_{k=1}^{N_P} (acos(cos(\theta_i^t - \hat{\theta}_r^t)))^2 \cdot w_i^t \quad (12)$$

$$\hat{\sigma}_{xy}^t = \sum_{k=1}^{N_P} (x_i^t - \hat{x}_r^t) \cdot (y_i^t - \hat{y}_r^t) \cdot w_i^t; \;\; \hat{\sigma}_{x\theta}^t = \hat{\sigma}_{y\theta}^t = 0; \quad (13)$$

To avoid large latency delays due to the processing of the correction loop, the filter propagates the estimated state just before publishing it, thus it computes a prior estimate,$\hat{X}^{t^+}$, using the acumulated odometry from the last propagation() call, $o_0^{t^+}$. Please note that particles are not affected at this step. Publishing a prior allow us to compare the basic filter with the proposed one in a more proper conditions.

The **publish()** function sends through a TCP port the prior estimate computed by the filter. Processes requiring real-time position data, such as navigation or monitoring, should connect to this port to receive it. Finally, the **resampling()** step generates a new particle set resampling the current one by means of the regularized resampling method [6]. When an old particle is choosen to be resampled, the new one sets its weight to $1/N_P$ and draws a new state vector following a random-normal centered on the old particle state with standard deviations derived from the platform size.

III. ASYNCHRONOUS PARTICLE FILTER

This section describes the proposed approach to take into account the moment of the observations when they are integrated in the particle filter. In order to outline the proposed algorithm, we introduce some definitions:

- $\Omega_k^t = (o_k^t, C_k^t, \tau_k^t, s_k^t)$ is an observation o_k^t, with covariance matrix C_k^t, arriving to the computing unit at iteration t, coming from the k^{th} observer, made at continuous time τ_k^t and with status s_k^t.
- Ω^t is the set composed by the last observation from each of the N_B observer. This set changes dynamically while filtering advances, since data reception is done concurrently and asynchronously with filtering at the tracking processing unit.
- $H^t = \{(\hat{X}^{t-\Delta}, \hat{C}^{t-\Delta}, \tau^{t-\Delta}), \ldots, (\hat{X}^{t-1}, \hat{C}^{t-1}, \tau^{t-1}), (\tilde{X}^t, \tilde{C}^t, \tau^t)\}$ is a set keeping the filter history of the Δ last posterior estimates and the last prior etimate made by the filter.

The t^{th} iteration of the proposed asynchronous particle filter, integrating observations coming from N_B observers, is outlined in the algorithm 2. Figure 1 depicts the case when $j < t$, thus the observation o_k^t is delayed with respect to the last prior time stamp τ^t. This figure shows how the particle X_i^t is backpropagated in order to compute observation models at positions where that particle was expected to be at the observation time. The other case, evaluated in the algorithm with the statement $IF\ j == t$, appears when the observation time stamp is advanced with respect the last prior time stamp. At this case, the filter propagates the particle set with the current odometric increments, and the advanced observation becomes a delayd one since the prior time stamp is updated at setEstimate() calling.

Algorithm 2 Asynchronous particle filter iteration

INPUT: $P^{t-1}, H^{t-1}, \Omega^t$
OUTPUT: $P^t, (\hat{X}^t, \hat{C}^t, \tau^t), H_\Delta^t$

```
P^t=propagate(P^{t-1}, o_0^t)
(X̃^t, C̃^t, τ^t) =setEstimate(P^t)
H^t.pushBackCurrentEstimate((X̃^t, C̃^t, τ^t))
for k = 1..N_B do
    j =max ι ∈ {t−Δ, ..., t}|τ^ι ≤ τ_k^t
    if j == t then
        P^t=propagate(P^t, o_0^{t+})
        (X̃^t, C̃^t, τ^t) =setEstimate(P^t)
        H^t.replaceLastEstimate((X̃^t, C̃^t, τ^t))
        j = t − 1
    end if
    α = (τ^{j+1} − τ_k^t) / (τ^{j+1} − τ^j)
    X̂^H = αX̂^j + (1 − α)X̂^{j+1}
    ΔX = X̃^t − X̂^H
    for i = 1..N_P do
        X_i^{H,t} = X_i^t − ΔX
        p(X_i^{H,t}|o_k^t) = L_k(o_k^t, o_k^s(X_i^{H,t}))
        w_i^{t'} = w_i^t · p(X_i^{H,t}|o_k^t)
    end for
end for
(X̂^t, Ĉ^t, τ^t) =setEstimate(P^t)
X̂^{t+}=propagate(X̂^t, o_0^{t+})
publish(X̂^{t+}, Ĉ^t, τ^t)
H^t.replaceLastEstimate((X̂^t, Ĉ^t, τ^t))
resampling(P^t)
```

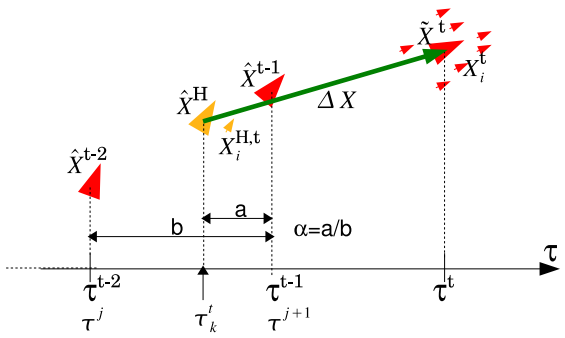

Fig. 1. Backward propagation of the particle X_i^t when integrating observation o_k^t.

IV. SIMULATION RESULTS: COMPARISON OF THE TWO FILTERS

In order to evaluate the performance of the proposed algorithm, we have executed an expriment consisting of a simulation of a mobile robot running on an environment of $10.000m^2$ at speed of about $2m/s$, completing a path of about $300m$. The simulated robot is equipped with two laser scanners, a compass and a GPS (coverage of about 60% of the path). Moreover, a camera network is deployed on the environment, covering about the 55% of the path and providing observations of the robot (x, y) location. Table I

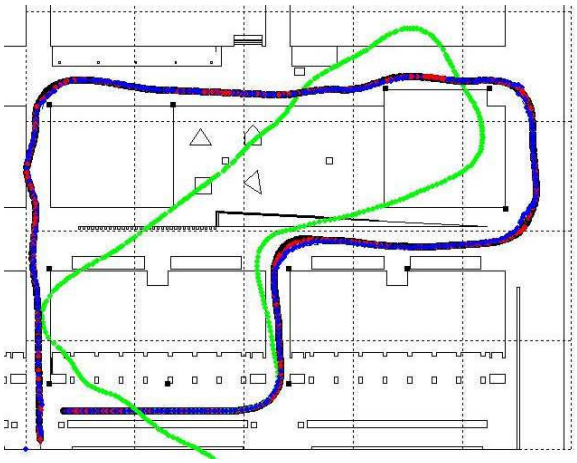

Fig. 2. Ground truth (black), basic (blue) and asynchronous (red) particle filter estimates, and odometry (green) on the map. Cameras are drawn as little black squares. Dotted lines build a 20mx20m grid.

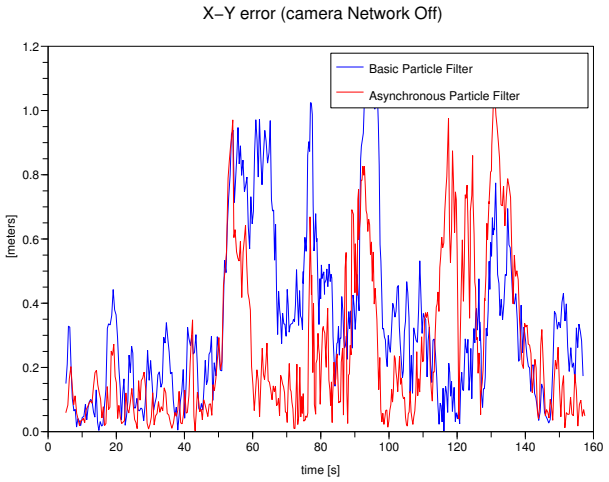

Fig. 3. XY error of both filters when the camera netwok is switched off.

summarizes the rates, the latencies and the mean of the simulated gaussian noise of each observer. These values were set taking into account real devices and systems.

TABLE I

RATES AND LATENCIES OF THE OBSERVERS

	Observer	Rate(Hz)	Latency(ms)	Mean noise
o_0^t	odometry	20	~0	$5\%(\delta XY), 10\%(\delta\theta)$
o_1^t	front laser	4	50	$5cm\ (range)$
o_2^t	back laser	4	50	$5cm\ (range)$
o_3^t	compass	5	20	$1.5^o(\theta)$
o_4^t	GPS	1	20	$2m\ (x, y)$
o_5^t	CameraNet	1	500	$0.4m\ (x, y)$

The experimental testbench was composed by two computers. The computer 1 was executing the simulator, the basic and the asynchronous particle filters. The computer 2 executed the GUI and was saving the frames in order to produce the attached video. This scenario allows to compare the two filters in real-time with the same conditions since they are running on the same simulation execution. Further details on the software can be found in [7]. For both filters, the number of particles was set to $N_P = 100$. Figure 2 shows the map, the ground truth positions in black, the basic filter estimates in blue, the asynchronous filter estimates in red and the odometric positions in green. In this figure the cameras are also drawn as small black squares.

Using this testbench we present two experiments. The experiment A was switching off the camera network, thus both filters were integrating only the observations provided by the onboard sensors. In the experiment B we have switched on the camera network, thus both filters integrate also remote observations provided by the camera network. To evaluate the performance of the filters we evaluate the following error figures:

$$e_x = \sqrt{(\hat{x}_r^t - x_r^\tau)^2};\ e_y = \sqrt{(\hat{y}_r^t - y_r^\tau)^2}$$
$$e_\theta = \sqrt{(\hat{\theta}_r^t - \theta_r^\tau)^2};\ e_{xy} = \sqrt{(\hat{x}_r^t - x_r^\tau)^2 + (\hat{y}_r^t - y_r^\tau)^2} \quad (14)$$

To compute this error we lineary approximate the simulated ground truth $(x_r^\tau, y_r^\tau, \theta_r^\tau)$ data at exact times where estimations are computed. This is done by considering the ground truth sample just before the estimate and just after the estimate. The ground truth process was running at 20Hz.

A. Experiment A: Camera Network Off

Figure 3 shows the error e_{xy} when the camera network was switched off. In this case the observers provide data with low latencies, therefore the asynchronous filter does not take clearly advantage of its properties. However, the proposed approach performs slightly better, since the observations are integrated properly considering their time stamps.

B. Experiment B: Camera Network On

When a camera network is switched on, we put in the scenario a very accurate observer that, however, provides observations at low rate and with large latencies. In this scenario, the proposed asynchronous filter performs much better than the basic one as figure 4 shows. The asynchronous filter outperforms the basic one with the exception of a short passage, where two filters have demonstrated a good recovery behaviour. This execution is recorded and presented in the attached video, where the particle sets of each filter can be seen with the simulated ground truth psition of the robot.

For this second experiment the error for each estimated variable is presented, each one accompanied with the estimated covariance. The following figures 5, 6, 7 show how the filter error remains in the most time inside the covariance bound of 1σ.

C. Discussion

Table II summarizes the mean errors for both filters and both experiments A & B. As expected, the proposed approach works much better when an accurate but delayed observer plays in the scene, as the case when the camera network is on. We can also see how the θ estimate does not improve its performance since it depends basically of the

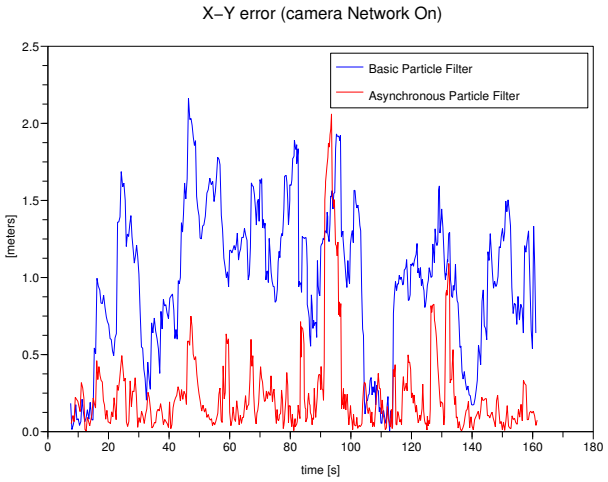

Fig. 4. e_{xy} error of both filters when the camera netwok is switched on.

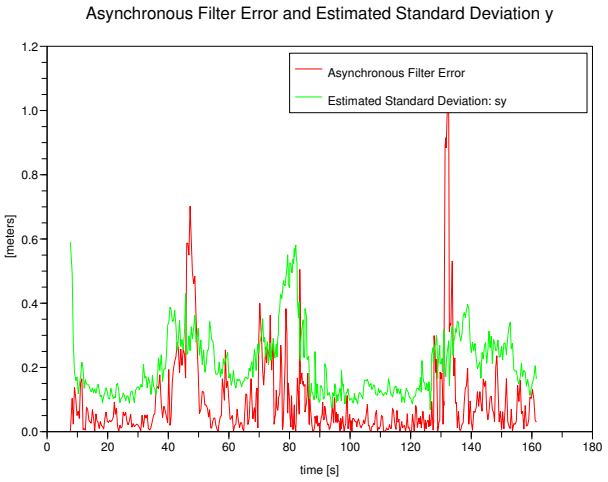

Fig. 6. e_y and the estimated covariance $\hat{\sigma}_y$ for the asynchronous filter.

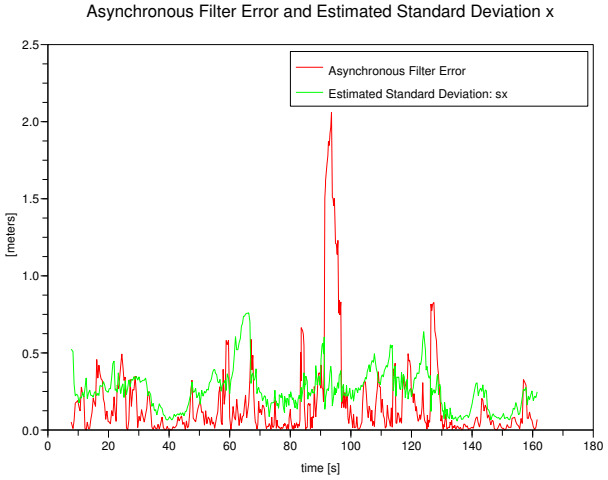

Fig. 5. e_x and the estimated covariance $\hat{\sigma}_x$ for the asynchronous filter.

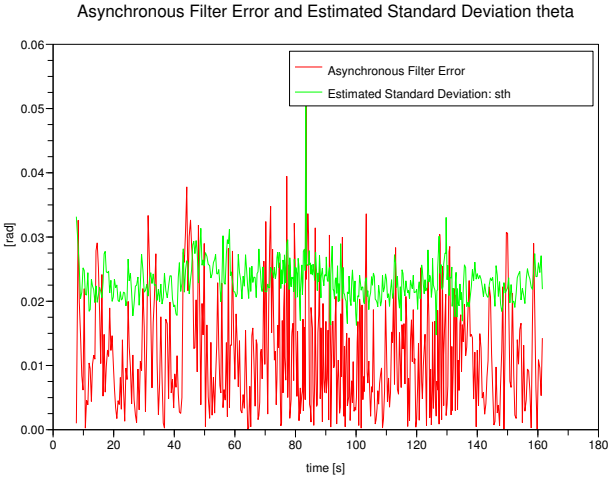

Fig. 7. e_θ and the estimated covariance $\hat{\sigma}_\theta$ for the asynchronous filter.

odometry and the compass, and these two observers have high rates and low latencies.

TABLE II
MEAN ERRORS OF A & B EXPERIMENTS

	Basic PF		Asynchronous PF	
CamNet	$\mu(e_{xy})[m]$	$\mu(e_\theta)[rad]$	$\mu(e_{xy})[m]$	$\mu(e_\theta)[rad]$
OFF	0.36	0.013	0.28	0.012
ON	1.05	0.013	0.26	0.012

On table II the reader can also compare the asynchronous filter with and without the camera network and discover that only a little improvement appears in terms of position estimate accuracy, but gains in terms of robustness since another observer is integrated on the filter. From this point, we want to evaluate the feasability of tracking the position of a robot with only the odometry and the camera network, in order to consider the proposed algorithm as a practical solution to be onboard of cheap robots running on environments where a camera network has been deployed. The following section presents results on this issue.

V. SIMULATION RESULTS: POSITION TRACKING WITH ODOMETRY AND CAMERA NETWORK

Once the asynchronous filter has shown good properties integrating observations with high latencies, we want to investigate a fusion scheme with only odometry and the camera network. This two observations are very complementary since odometry has a high rate, a small latency and a good accuracy in short displacements, while a camera network provides absolute and accurate (x, y) observtaions with a large latency, but does not suffer from accumulated drifts as odometry does. In this experiment we use the same testbench as the previous ones but we execute only the asynchronous filter since the basic filter was unable to track the robot position in a robust way.

Figures 8, 9, 10 depicts the error of this experiment for the three estimated variables. These figures shows how the

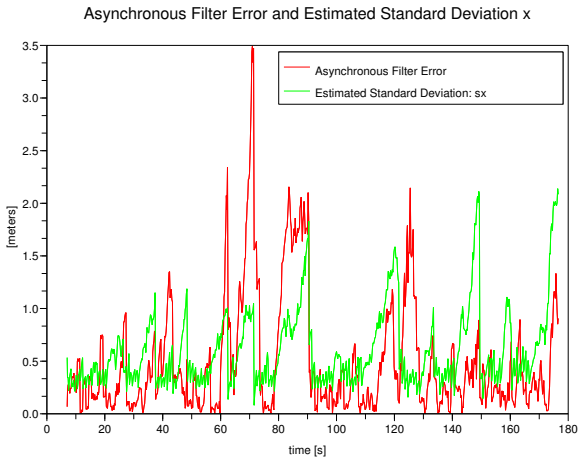

Fig. 8. e_x and the estimated covariance $\hat{\sigma}_x$ for the asynchronous filter only integrating odometry and camera network observations.

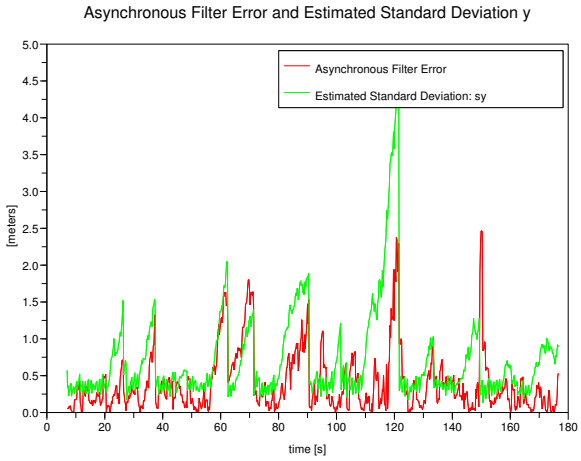

Fig. 9. e_y and the estimated covariance $\hat{\sigma}_y$ for the asynchronous filter only integrating odometry and camera network observations.

error increments when the robot is out of coverage of the camera network and redcuces when the robot integrates remote observations coming from the camera network. Even if the coverage of the camera network is partial (55% of the path) the proposed approach is able to track the position of the robot with an acceptable error. Mean errors for this experiment were $\mu(e_{xy}) = 0.7\ m$ and $\mu(e_\theta) = 0.07\ rad$. They can be compared with those errors of table II where the filters integrated all observations from all sensory subsystems. Obviously, the filter integrating all observations performs better, but the interest of this second result lies in the fact that cheap robots with only wheel encoders could track its position taking benefit of observations coming from a deployed camera network with partial coverage of the environment.

VI. CONCLUSIONS

This paper presents an asynchronous particle filter algorithm that takes into account time considerations when

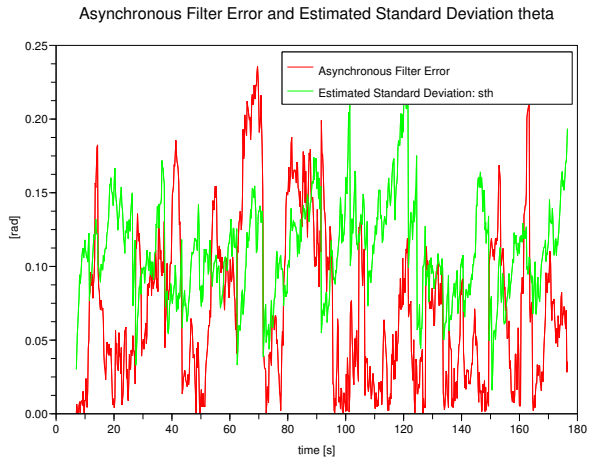

Fig. 10. e_θ and the estimated covariance $\hat{\sigma}_\theta$ for the asynchronous filter only integrating odometry and camera network observations.

integrates observations coming from a set of asynchronous observers (delayed or advanced from the prior estimate). The paper considers the position tracking task as a real-time process playing a key role in autonomous navigation systems and, therefore, the design of the proposed approach wants to publish the best position estimate without latencies or delay assumptions as previous approaches do. The proposed solution is tested in a simulated testbench comparing a basic particle filter and the asynchronous one, and promising results are presented and discussed. A practical situation fusing only odometry an observations coming from a camera network is also evaluated showing the potentialities of the proposed approach in cooperative environments where a camera network shares information with running robots.

REFERENCES

[1] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, pp. 99–141, 2001.

[2] J. Levinson, M. Montemerlo, and S. Thrun, "Map-Based Precision Vehicle Localization in Urban Environments," in *Proceedings of the Robotics: Science and Systems Conference*, (Atlanta, USA. June, 2007.).

[3] A. Sanfeliu and J. Andrade-Cetto, "Ubiquitous networking robotics in urban settings," in *Proceedings of the IEEE/RSJ IROS Workshop on Network Robot Systems*, (Beijing, China. October, 2006.).

[4] Y. Bar-Shalom, "Update with Out-of-Sequence Measurements in Tracking: Exact Solution," *IEEE Transactions on Aerospace and Electronic Systems*, no. 3, pp. 769–778, 2002.

[5] A. Ranganathan, M. Kaess, and F. Dellaert, "Fast 3D Pose Estimation With Out-of-Sequence Measurements," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (San Diego, California. October, 2007.).

[6] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, " A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," *Transactions on Signal Processing*, vol. 50, pp. 174–188, February 2002.

[7] A. Corominas Murtra, J. Mirats Tur, O. Sandoval, and A. Sanfeliu, "Real-time Software for Mobile Robot Simulation and Experimentation in Cooperative Environments," in *Proceedings of the Simulation, Modelling and Programming for Autonomous Robots (SIMPAR). Lecture Notes on Artificial Intelligence 5325. Springer ed. ISBN 978-3-540-89075-1.*, (Venice, Italy. November, 2008), pp. 135–146.