A SIMULATION PLATFORM FOR LOCALIZATION AND MAPPING

Thomas Hanefeld Sejerøe[*], **Niels Kjølstad Poulsen** [**]
Ole Ravn [*,1]

[*] *Ørsted●DTU, Automation,*
The Technical University of Denmark,
Building 326, DK-2800 Kgs. Lyngby, Denmark
E-mail: `or@oersted.dtu.dk`
[**] *Informatics and Mathematical Modelling,*
The Technical University of Denmark,
Building 321, DK-2800 Kgs. Lyngby, Denmark
E-mail:`nkp@imm.dtu.dk`

Abstract: In this paper we present a simulation platform for evaluate methods for simultaneous location and mapping. The platform is based on The Kalmtool 3 toolbox which is a set of MATLAB tools for state estimation for nonlinear systems. The toolbox contains functions for extended Kalman filtering as well as for two new filters called the DD1 filter and the DD2 filter. It also contains function for Uncented Kalman filters as well as three versions of particle filters. The toolbox requires MATLAB ver. 6, but no additional toolboxes are required. *Copyright© IFAC 2006.*

Keywords: Simulation, toolbox, State estimation, Kalman filtering, nonlinear systems, autonomous robots

1. INTRODUCTION

The simulation platform described in this paper, is build on **Kalmtool** which is a collection of Matlab implementations for simulation and estimation in connection to nonlinear dynamic systems. The development of the toolbox has been driven by the application, which is navigation of mobile robots. In this context location and mapping is corner stones.

During the work it was found that the extended Kalman filter was somewhat inconvenient to use in some of our applications. A small modification of the application sometimes had serious implications on the EKF implementation. Moreover, it was often difficult to implement. Our problem was that the EKF requires a linearization of the system model. Sometimes this is easy to find but sometimes it can be pretty hard. In any case, it makes things inflexible. If a small change is made in the model, one has to work out a new set of derivatives. This is particularly inconvenient in model calibration where certain model parameters are temporarily included in the state vector and estimated simultaneously with the actual states.

Since it was suggested, the extended Kalman filter (EKF) has undoubtly been the dominating technique for nonlinear state estimation. Nevertheless, the EKF is known to have several drawbacks. These are mainly due to the Taylor linearization of the nonlinear transformations around the current state estimate. The linearization requires that Jacobians of state transition and observation equations are derived, which is often a quite complex task. Moreover, sometimes there are points in which the Jacobians are not defined. In addition to the difficulties with implementation, conver-

[1] Corresponding author

gence problems are often encountered due to the fact that the linearized models describe the system poorly.

There have been significant focus on this area recently and previous work include several toolboxes and other platforms. ReBEL (Recursive Bayesian Estimation Library) van der Merwe (2004) is a Matlab toolkit of functions and scripts, designed to facilitate sequential Bayesian inference (estimation) in general state space models. The CAS Robot Navigation Toolbox Arras (2004) is a tool for doing off-line off-board localization and SLAM on mobile robots. The design of the CAS toolbox decouples robot model, sensor models, features and algoritms used giving the user ability to adapt the toolbox by just modifying or adding the pieces in question. The toolbox does not in its present form support the generation of realtime code for use on the robot.

The present platform Kalmtool 3 has its root in Kalmtool (v. 2) but focus here is on comparision and transparency giving the developer more control over the process of adapting changes and keeping housekeeping code minimal. The implemented methods are described in more detail in Sejerøe et al. (2005).

The paper is organized as follows: first the overall design philosophy behind the platform is described. Secondly, a description of the the toolbox and the implementation of the estimation algorithms are given. This includes the extended Kalman filter, the Uncented Kalman filter and different types of particle-filters. Next the driving application i.e. location and mapping in connection to mobile robots are discussed. Section 3 gives an extensive example study as well as a demonastration of the platform for comparing algortims for navigation of a mobile robot. Finally conclusions and references are given.

2. THE PLATFORM AND TOOLBOX

The overall design philosophy has been to put focus on making a simple, transparent, yet powerfull platform that and makes life easy to use both for application and algorithm developer.

Transperancy overcomes the barrier effect that is often expirenced when using tools that at first sight seem very user friendly but when used on real problems becomes difficult to handle due to the inherent complexity.

The approach taken uses MATLAB as a numerical and graphical basis for developing the platform. The platform is driven from Simulink as this provides a shorter path to implementation using for instance Realtime Workshop and makes is simple to use real data for comparison.

The philosophy of the update to the Kalmtool toolbox is to provide a more open structure (see figure 1, which is easier to use and which enables the user to investigate the inner workings of the estimation algorithms. With this in mind, the structure of the Kalmtool 3 functions have been opened up an a larger selection of functions made available. The functions are made to work in an online setting, one timestep - one update, though this does not prohibit its use in offline environments. The main changes are the breakup of the estimation loop and the introduction of an a evaluator function.

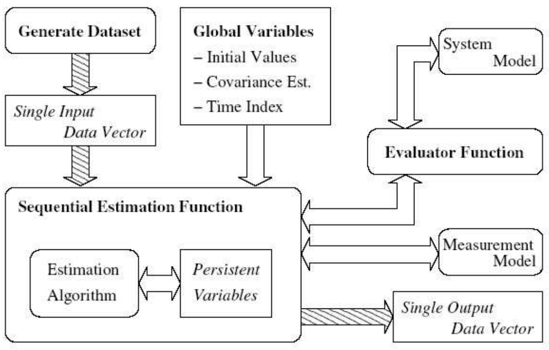


Fig. 1. *The structure of Kalmtool 3.*

Changing the loop means that the functions can now be used directly in an online setting providing that there are sufficient resources available.

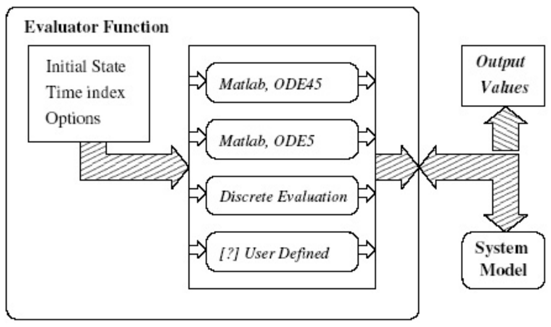


Fig. 2. *The evaluator function.*

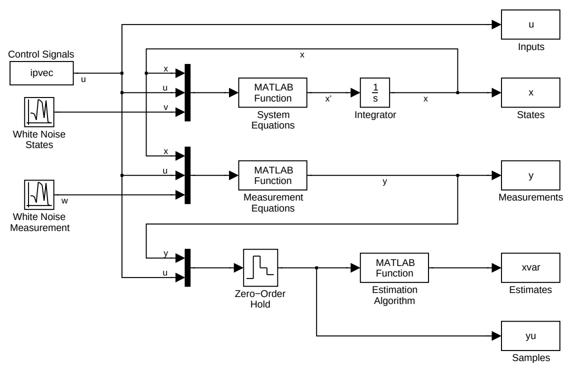


Fig. 3. *The Simulink layout of a continous system.*

As seen in the above figures the user can easily add new algorithm into the platform by modifying the MATLAB function in the Estimation block and change the system by modifying the system and measurement MATLAB blocks.

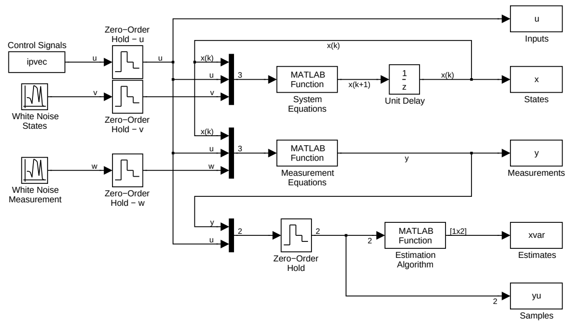


Fig. 4. *The Simulink layout of a discrete time systems*

Procedure	functionality
divdiff1	the Divided difference first order
divdiff2	the Divided difference second order
ekfcntns	the Extented Kalman filter with cont. eval.
ekfdscrt	the Extented Kalman filter with disc. eval.
pfexpuns	a raw version of the Particle Filter
pfgenerc	a generic Particle Filter using SIR
pfgnrcmh	a generic Particle Filter using SIR and MHR
ukfsimple	the Unscented Kalman filter
ukfscaled	the scaled Unscented Kalman filter

Table 1. Table of estimation functions available in Kalmtool 3.

3. EXAMPLE STUDY

The versatility of the simulation framework is most evident when implementing a number of examples. For the purpose of this demonstration, a discrete time difference equation system and a continuous time differential equation system are selected. The example studies concludes with a simultanous location and mapping problem.

3.1 Nonlinear state estimation

The first example i a discrete time system and is an academic example of a nonlinear system (though in a simplified form), which has been used previously as a benchmark for testing filter algorithms (Netto et al. (1978)).

In the example the process is a nonlinear equation with a linear and noisy measurements. First, the process equation, x_{k+1} is listed, next the measurement equation, y_k.

$$x_{k+1} = \frac{1}{2}x_k + \frac{25x_k}{1+x_k^2} + 8\cos(1.2k) + v_n \tag{1}$$

$$y_k = x_k + w_k;$$

Note that, both the noise sources, v_k and w_k, are zero mean Gaussian white noise with variances of 10.0 and 1.0 respectively. As was the case with the small robot model, a Monte Carlo series of simulations was made with a variety of estimation algorithms. Two examples

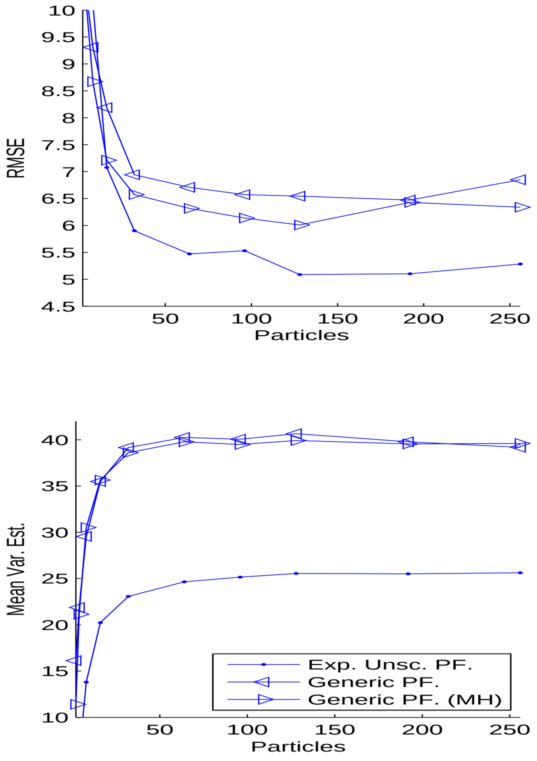


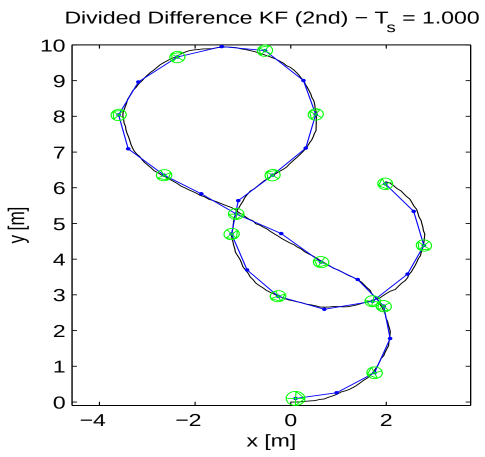
Fig. 5. *Two graphs depicting the effect of varying the particle count per update on the benchmark system. The mean RMSE and mean variance estimates are seen to converge rather quickly to relatively stationary values at around 100 particles per time update.*

of the appearance of a simulation can be found in figure 7.

The result of the Monte Carlo simulation can be seen in table 2. The Kalman filter type algorithms were simulated 1000 times and the means of the root mean square errors (RMSE) were found as well as the means of the variance estimates. The Particle Filter types were simulated 100 times with 200 particles per time update in all filters.

Algorithm	Mean RMSE	Mean Var. Est.	Time
C.D. EKF	0.9573	0.9206	1.000
Std. UKF	0.9472	0.9238	0.126
Scl. UKF	0.9503	0.9247	0.151
DD1	0.9417	0.9221	0.133
DD2	0.9260	0.9238	0.137
Exp. PF	0.9513	0.9165	2.067
Gen. PF	4.2326	31.595	5.917
PF (MH)	4.0238	28.165	8.543

Table 2. *Table of results for a Monte Carlo series of simulations on the discrete nonlinear and noisy system.*



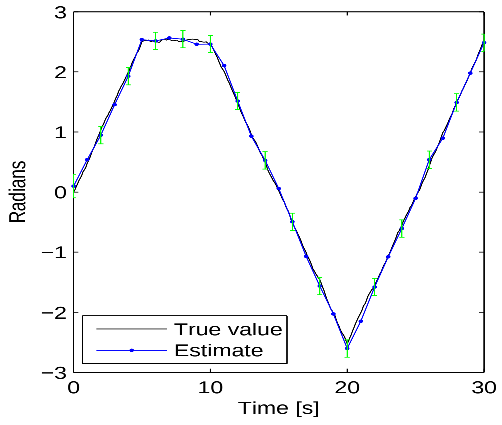


Fig. 6. *A path traced by the small unicycle robot. The estimation routine employed is the 2nd order Divided Difference filter with a sampling frequency of 1 Hz. At every other estimated state, the 95% confidence intervals are drawn as ellipses or bars respectively. The estimate is at no point outside the confidence intervals.*

Finally, in order to compare the precision of the three particle filters as a function of the number of particles per time update, a Monte Carlo series of simulations was made using the benchmark system. The results can be seen in figure 5. The series consisted of 100 runs per particle count, from 2 to 256 particles in increasing steps. The algorithms converge rather quickly as the particle count increases.

3.2 Location and Mapping

When maneuvering an autonomous guide vehicle (AGV) it is important to know the position and orientation of the vehicle. This is often done by using the odonometry of the vehicle. This is basically just to use the measured traveling distance and measured change of orientation. This is also denoted as dead-reckoning. It is however well known that this method has an inherit nature of accumulating errors. The determination of the position and orientation is therefore supplied with measurement of the robot position and orientation in relation to some guide marks with known (or relative well known) positions. This, dead-reckoning and eventually the use of some guide marks is denoted as location or navigation.

The model of the mobile robot (unicycle type) is given by the set of equations which are slightly nonlinear. The equations yield a position as well as a heading. The input signals (i.e. control signals) are the velocity, γ, and turnrate, ω.

$$\frac{d}{dt}\begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} = \begin{pmatrix} \gamma_t\, cos(\theta_t) \\ \gamma_t\, sin(\theta_t) \\ \omega_t \end{pmatrix} + v_t \qquad (2)$$

The process noise is in the (later) example studies simulated as $N(0, 0.01\, I_3)$. The estimation procedure in the location part and the mapping part is based on a sampled version of the above process equation. The sampling can be done analytically (for this simple example) or by means of a numerical ODE solver.

Location in relation autonomous guided vehicle is based on a fusion of results from several sensors. Normally one of the sensors set is the odometry, i.e. noisy measurements of the speed of the wheels:

$$\omega_r = \frac{2\gamma_t + b\omega_t}{2r_r}$$
$$\omega_l = \frac{2\gamma_t - b\omega_t}{2r_l}$$

Another set of measurements is the relative position between a guide marks and the robot. Assume a guide mark has a position which is known with some precision embedded in

$$\begin{bmatrix} x_g \\ y_g \end{bmatrix} \in \mathbf{N}(0, P_g)$$

The position of the robot is also known with some precision reflected by

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} \in \mathbf{N}\left(\begin{bmatrix} \hat{x}_t \\ \hat{y}_t \\ \hat{\theta}_t \end{bmatrix}, P_t\right)$$

The actual measurement is the distance and the direction to the guide mark which can be transformed into a set of Cartesian measurement:

$$y_t = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ x_g \\ y_g \end{bmatrix} + e_t \qquad (3)$$

The mesurement noise is assumend to be $\mathbf{N}(0, R_2)$ where R_2 reflects the transformation of the uncertainty in the mesurements of the distanse and the direction from the robot to the guide mark.

Both location and mapping is based on the same principle. In connection to mapping a newly observed guide mark is assumed to have a position given by the a'priori distribution

$$\begin{bmatrix} x_g \\ y_g \end{bmatrix} \in \mathbf{N}(p, P_0) \qquad (4)$$

reflecting the lack of knowledge. As a limit it can be assumed to be totally flat.

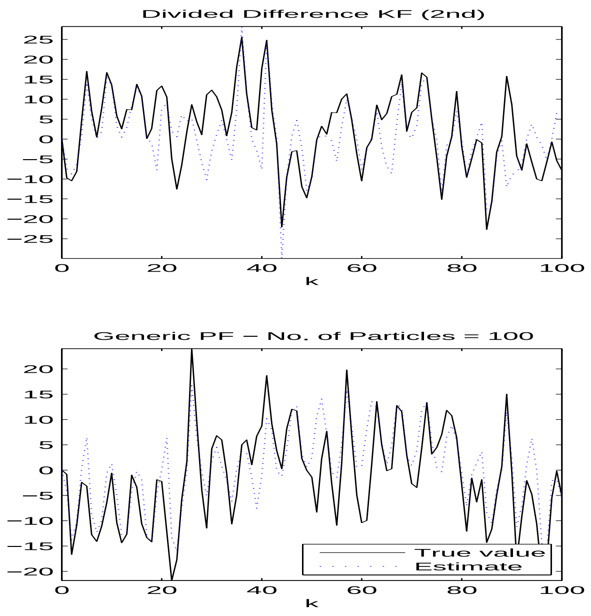


Fig. 7. *Two examples of the highly nonlinear and noisy system given in equation 1. The topmost is the 2nd order Divided Difference filter, while the bottommost is a generic Particle Filter.*

3.3 Dead-reckoning

The next example is a continuous time system and is a very simple model of a dead-reckoning guidance for a small mobile robot (see equation (2)). In Figure 6 the results of a simulation using the Divided Difference (2nd order) as estimator can be seen. The integral of the control signal, ω, is seen in the lower panel below the path traced by the robot (upper panel).

In order to compare a range of techniques implemented in the framework, accuracy results are given in table 3. Attempting to find a fair estimate of the accuracy, 100 runs were made with each algorithm and the average values were found. The particle filters all used 200 particles per time update. The table contains the "worst case" values for the three states.

Also listed in the table is the computational burden of each algorithm. The latter is given as a relative number compared to the runtime of a continuous-discrete extended Kalman filter (C.D. EKF). The times are relative, as other processor speeds and types will yield different absolute results. Furthermore, the algorithms and their runtimes may well benefit from numerical

Algorithm	Max. RMSE	Max. Var. Est.	Time
C.D. EKF	0.06799	0.005717	1.000
Std. UKF	0.07399	0.006779	2.678
Scl. UKF	0.07331	0.006693	3.673
DD1	0.07251	0.006779	2.640
DD2	0.07177	0.006781	2.658
Exp. PF	0.07591	0.006479	16.86
Gen. PF	0.09698	0.039829	17.82
PF (MH)	0.08960	0.058690	18.53

Table 3. *Small mobile robot, worst value of mean estimate (x,y,θ) and maximum mean variance estimate of 100 Monte Carlo simulations. The table is split into Kalman filter variants (top) and particle filters (bottom). The particle filters all used 200 particles.*

optimizations in application specific implementations. The algorithms used a fixed step integration (Matlab, Dormand-Prince, order 5) to solve equation 2. The standard Unscented Kalman filter (Std. UKF) performs very well, while it's scaled version gives a lower mean RMSE and a slightly lower mean variance estimates. The DD1 and DD2 both give low mean RMSE and consistent variance estimates - in this case, the second order parts of the DD2 does not yield much.

3.4 Simultanuous location and mapping I

The next two examples are related to location while a map of the guide marks is build. The dynamics involved is the AGV given in (2) with a sensor fusing between the odometry (dead-reckoning) and the relatiove postioning of the guide marks. Both the location and mapping is based on the oberservation equation, (3), where the guide mark is the actual guide mark under observation. The robot is assumed to have an active view sector in front which is 90 degree wide and has a range of 4 m. The active guide marks are the guide marks visible within the robot view sector.

In this context the map consists of a database containing the estimated locations of the guide marks and their respective uncertainty. Besides the database the location and mapping consists of a routine for handling the information related to the active guide marks.

In the first example related to simultaneous location and mapping the task is to navigate the robot along a wall and drive through the door opening and return. The door opening is defined in terms of two set of guide marks. The navigation is performed by means of way points located in in front and behind the door opening. The positions of the way points are assumed to be known. The control implementation is described in Bak (2000), but is beyond the scope of this paper.

The results are illustrated in Figure 8 where the applied estimation technique is based on the DD2 method described in Sejerøe et al. (2005).

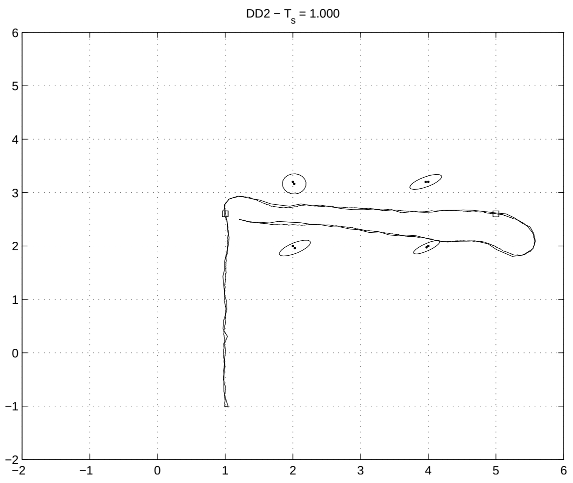


Fig. 8. *Navigation of a mobile robot through a door opening. The map is build simultanuouly while controlling the robot. The control is based on the location i.e. the estimation of the position and orientation of the robot. The positions and their incertainty are anotated by 99% confidence areas (ellipsoids).*

3.5 Simultanuous location and mapping II

This example is quite similar to the previous example, except that in this case it is a bit more complex and the robot has to follow a coridor equipped with guide marks. The results can be seen in Figure 9. The true robot path (which is known due to the simulation) and the estimated are indicated by solid lines. The location of the 4 way points are also indicated.

As the map is build the position of the guide marks are introduced. The estimated positions and their uncertainties are indicated with a dot and a 99% confidence area (ellipsoids). Notice, that in some case the correct position of a guide mark is outside the confidence area.

In this work we have applied an earth fixed coordiante system in which both position (and orientation) of the robot and the guide marks are related. The result is positions of robot and guide marks in an absolute scale. However, the dynamic is related to the robot only. Another approach is to apply a robot fixed coordinate system. Then the position of the robot and guide marks are relative. In a robot fixed coordinate system the process equation for the guide marks are no longer the identity but a result of the movement of the robot (and the coordinate system).

4. CONCLUSION

In this paper we have presented a simulation platform for simultaneous location and mapping. The platform is based on the toolbox Kalmtool which a set of MATLAB tools for state estimation for nonlinear systems. It contains functions for extended Kalman filtering as well as for the two new filters, the DD1 filter and the DD2 filter. It also contains functions for Unscented

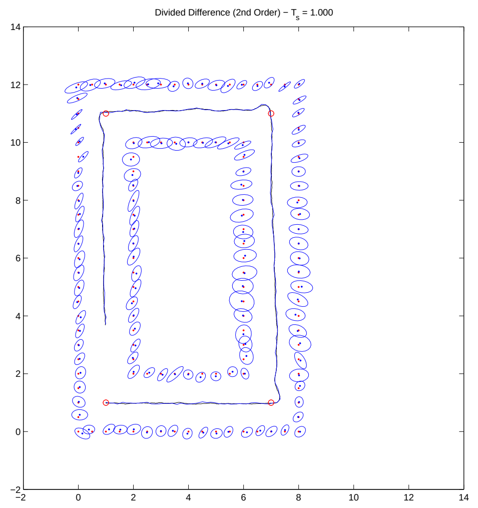


Fig. 9. *Navigation of a mobile robot along a corridor with guide marks located on the walls. The map is build simultanuouly while controlling the robot.*

(standard and scaled) Kalman filter as well as three versions of particle filters.

The paper contain some examples to illustrate the methods and the results maninly based on divided difference approach (DD2) to estimation in nonlinear dynamic systems. The dynamics is mainly related to (small) mobile robots and simultanuous location and mapping.

The toolbox is available at

`server.oersted.dtu.dk/personal/or/kalmtool3`

ACKNOWLEDGMENT

The support from the Danish Center for Scientific Computing (DCSC) (under grant CPU-1101-30) is gratefully acknowledged.

REFERENCES

Kai O. Arras. The cas robot navigation toolbox, quick guide. Technical report, CAS, KTH, January 2004.

M. Bak. *Control of Systems with Constraints*. PhD thesis, IAU, DTU, 2000.

A.M.L. Netto, L. Gimeno, and M.J. Mendes. A new spline algorithm for non-linear filtering of discrete time systems. *Proceedings of the 4th IFAC Symposium on Identification and System Parameter Estimation, Tbilisi, U.S.S.R.*, pages 2123–2130, 1978.

Thomas Hanefeld Sejerøe, Niels Kjølstad Poulsen, and Ole Ravn. A new evaluation platform for navigation systems. In *16'th IFAC World Congress, Prague, Czech Republic*, pages Tu–A18–TO/2, ID: 03891, 2005.

Rudolph van der Merwe. Quick-start guide for rebel toolkit. Technical report, Oregon Health and Science University, February 2004.