Nonlinear Estimation Framework in Target Tracking

Ondřej Straka, Miroslav Flídr,
Jindřich Duník, Miroslav Šimandl
Department of Cybernetics
University of West Bohemia
Plzeň, Czech Republic.
{straka30,flidr,dunikj,simandl}@kky.zcu.cz

Erik Blasch
Defence R&D Canada-Valcartier
2459 Pie-XI Blvd. North
Québec City, QB G3J 1X5
erik.blasch@drdc-rddc.gc.ca

Abstract – *The goal of the article is to describe a software framework designed for nonlinear state estimation of discrete-time dynamic systems. The framework was designed with the aim to facilitate implementation, testing and use of various nonlinear state estimation methods. The main strength of the framework is its versatility due to the possibility of either structural or probabilistic model description. Besides the well-known basic nonlinear estimation methods such as the extended Kalman filter, the divided difference filters and the unscented Kalman filter, the framework implements the particle filter with advanced features. As the framework is designed on the object oriented basis, further extension by user-specified nonlinear estimation algorithms is extremely easy. The paper describes the individual components of the framework, their key features and use. The paper demonstrates easy and natural application of the framework in target tracking which is illustrated in two examples - tracking a ship with unknown control and tracking three targets based on raw data.*

Keywords: Nonlinear state and parameter estimation; Bayesian approach; Tracking; EKF; UKF; Particle filter.

1 Introduction

Nonlinear state estimation of discrete-time stochastic dynamic models is a rapidly developing field of study which plays a crucial role in many areas such as target tracking [1, 2], satellite navigation, signal processing, fault detection and adaptive and optimal control problems. It constitutes an essential part of any decision-making process.

The general solution of state estimation problem can be found by means of Bayesian functional relations (BFR) [3]. However, a closed-form solution to the BFR can be found only for a few special cases [4, 5], where a linear Gaussian system is such a typical case and the solution corresponds to the well-known Kalman filter. Nevertheless, usually it is necessary to apply a suitable approximation.

A useful idea is to approximate the model in such a way that further proceeding within a Kalman filtering framework is possible. A way to achieve this is to substitute the nonlinear functions describing the model by the first few terms of their Taylor series expansion [6] or the Stirling's interpolation formula [7, 8]. It is also possible to solve the problem numerically. In this case the integrals occurring in the BFR are numerically solved using an appropriate quadrature rule e.g. Gauss-Hermite quadrature rule or unscented transformation [9, 8].

Although these methods are usually computationally moderate, validity of the estimate is usually limited. To extend the estimate validity, the Gaussian sum method [10, 11] has been proposed. This method expects all considered probability density functions (pdf's) in the form of a mixture of Gaussian pdf's. The sought conditional pdf's of the state are then also represented by such a mixture.

Since the nineties, the particle filters (PF) based on the Monte Carlo method [12, 13] have become very popular. They approximate the conditional pdf's by empirical pdf's consisting of random samples and associated weights and provide globally valid estimates. Their popularity stems from their easy implementation in very general settings and cheap and formidable computational power. Although the global methods provide extended validity of estimates, it is paid by a growth of theoretical and computational demands.

The vast number of different state estimation methods with their theoretical demands, the number of design parameters and impossibility to design recommendations, which could aid the user who looks for a suitable method for his estimation task, called for the development of several software toolboxes implementing various estimation methods for comparison.

There are several software packages dealing with nonlinear state estimation. The most notable representatives are KalmTool [14], ReBEL (Recursive Bayesian Estimation Library) [15], NEF (Nonlinear Estimation Framework) [16] and EKF/UKF Toolbox [17]. All the packages are very powerful tools, which provide extensive support for all the currently well-known local estimation methods and an elementary PF.

The NEF is a collection of mutually linked MATLAB functions for modelling system behavior, state estimation and evaluation of the results with a simple user interface. Its

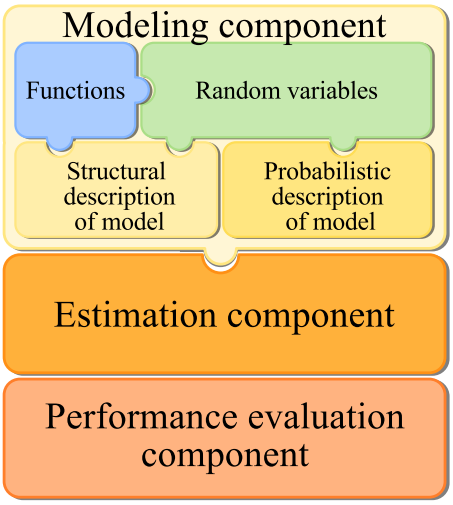

Figure 1: NEF components

development has been driven by the need for a tool that can evaluate the quality of a state estimation method in an arbitrary case, compare performance of several state estimators and provide means for rapid prototyping of new state estimators with minimal effort. The NEF represents the most recent generation of software toolboxes developed by authors and follows the previous generations represented by NFTools [18] and NFToolsCD [19]. Its basic components are illustrated in Fig. 1. The NEF employs the object oriented approach making it possible to spare the casual user from unnecessary details. But it also offers the experienced user full control over the experiments and provides means for complex extensions. The key features of the NEF are

- structural and probabilistic modelling,
- support for time-varying models,
- support for filtering, multi-step prediction and fixed-lag smoothing,
- implementation of both standard and numerically stable estimation algorithms,
- full estimator parametrization by means of the standard MATLAB property-value mechanism,
- complete evaluation of estimate quality.

The goal of the paper is to briefly introduce the NEF and its key components (modelling, estimation and performance evaluation) and thoroughly illustrate its application in two target tracking examples.

The paper is organized as follows. The following sections 2, 3 and 4 introduce the key NEF components. An application with tracking a ship with unknown control is a subject matter of Subsection 5.1 and tracking with raw data using the NEF is presented in Subsection 5.2. Finally, Section 6 will be devoted to concluding remarks.

2 Modelling component

The modelling component is one of two underlying components of the NEF. It aims primarily at specification of the model for estimation purposes but it also allows simulation of the model to obtain state and measurement data in the case that the model is also used as a data generator.

From the theoretical point of view, models considered within the NEF represent a discrete-time stochastic system in state-space representation with a continuous state.

There are two ways to specify a model. Either it might be given by a structural description

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k), \qquad k = 0, 1, \ldots, \tag{1}$$

$$\boldsymbol{z}_k = \boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k), \qquad k = 0, 1, \ldots, \tag{2}$$

where $\boldsymbol{x}_k \in \mathcal{R}^{n_x}$, $\boldsymbol{z}_k \in \mathcal{R}^{n_z}$ and $\boldsymbol{u}_k \in \mathcal{R}^{n_u}$ are state, measurement and control of the model, respectively, $\boldsymbol{w}_k \in \mathcal{R}^{n_x}$ and $\boldsymbol{v}_k \in \mathcal{R}^{n_z}$ are state and measurement white noises described by $p(\boldsymbol{w}_k)$ and $p(\boldsymbol{v}_k)$, respectively. Both noises are mutually independent and they are also independent of the known initial state $\boldsymbol{x}_0$ pdf $p(\boldsymbol{x}_0)$. The functions $\boldsymbol{f}_k$ and $\boldsymbol{h}_k$ are supposed to be known.

Or, the second way of model specification is probabilistic description given by the transient pdf (3) and the measurement pdf (4):

$$p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k, \boldsymbol{u}_k),\ k = 0, 1, \ldots, \tag{3}$$

$$p(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{u}_k),\ k = 0, 1, \ldots. \tag{4}$$

The initial condition of the state is given by $p(\boldsymbol{x}_0)$.

It should be emphasized that both structural and probabilistic description may be time-varying. The way how the model is specified is often enforced by the chosen estimator. For estimators exclusively within the Kalman filtering framework, the structural description is required. The PF and the Gaussian sum method allow specification of the model by any of the two ways.

The following two subsections are dedicated to structural and probabilistic description of the models and the third subsection to description of functions and random variables.

2.1 Structural and probabilistic description

For a structural description of a model, it is necessary to specify two possibly nonlinear functions $\boldsymbol{f}_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k)$ and $\boldsymbol{h}_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k)$ and three pdf's $p(\boldsymbol{w}_k)$, $p(\boldsymbol{v}_k)$ and $p(\boldsymbol{x}_0)$ as is illustrated in Fig. 2 (block `nefEqSystem`).

For a probabilistic description of a model, it is necessary to specify three pdf's $p(\boldsymbol{x}_{k+1}|\boldsymbol{x}_k, \boldsymbol{u}_k)$, $p(\boldsymbol{z}_k|\boldsymbol{x}_k, \boldsymbol{u}_k)$ and $p(\boldsymbol{x}_0)$ as is illustrated in Fig. 2 (block `nefPDFSystem`).

Distributions of the random variables can be chosen from a predefined set in the NEF. If a user wants to define his own distribution, the root class `nefRV` provides a template that can be used as a starting point. The fact that all parameters of the distribution can be specified as a function of the state, input or time makes it possible to define conditional distributions, in an efficient manner.

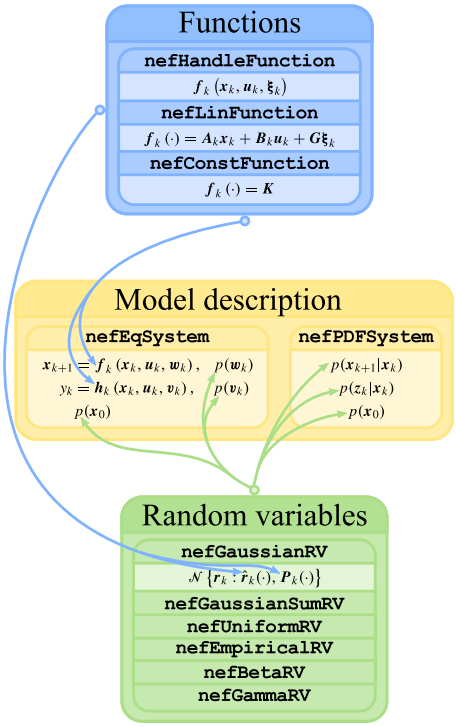

Figure 2: Scheme of NEF modelling component.

2.2 Functions and random variables

Within the NEF, the functions are specified as objects of the abstract class `nefFunction` or its subclasses. The NEF offers basic functions as a constant function (`nefConstFunction`) and a linear function (`nefLinFunction`). The class `nefHandleFunction` is the most useful one because it utilizes the MATLAB handle function mechanism which makes it possible to describe a vast number of functions. In the case that the offered classes are not sufficient, no matter how unlikely this is, a user-defined class can be designed based on the template provided by the root class `nefFunction`.

The functions can have in general at most four arguments, which are the state, input, noise and time. Any function object provides means to:

- evaluate the function at an arbitrary point,
- evaluate its first derivative with respect to state at an arbitrary point,
- evaluate its first derivative with respect to noise at an arbitrary point.

The first derivatives are required only if a Kalman filter based estimator for a nonlinear model is to be used. Usually, the derivatives must be supplied by the user upon the function object creation only. In the case of constant or linear function they are computed automatically.

Similarly to the functions, the random variables are also specified using objects. The root class is called `nefRV` and its currently implemented subclasses correspond to distributions

- multivariate uniform (`nefUniformRV`),
- multivariate Gaussian (`nefGaussianRV`),
- multivariate Gaussian sum (`nefGaussianSumRV`),
- multivariate empirical pdf (`nefEmpiricalRV`),
- univariate Beta (`nefBetaRV`),
- univariate Gamma (`nefGammaRV`).

Note that the parameters of the distributions are specified as functions, which allows description of a non-stationary process.

Each random variable object provides means to:

- draw a sample of the random variable,
- evaluate its pdf at a given point,
- provide mean and variance,
- provide mode or median in some cases,
- display a plot of the pdf.

3 Estimation component

The estimation component of the NEF provides implementation of a number of estimators used to solve the estimation problem where the main task is to find a conditional probability density function of the state conditioned by available measurements. The estimate of the state $\boldsymbol{x}_k$ is given by the posterior pdf $p(\boldsymbol{x}_k|z^\ell, \boldsymbol{u}^\ell)$, where z^ℓ is the sequence of measurements up to time instant ℓ, i.e., $z^\ell \triangleq [z_0^T, z_1^T, \ldots, z_\ell^T]^T$.

The estimation problem itself can be divided according to the relation of k and ℓ into the following three special cases.

- If $\ell = k$, the problem is called *filtering*.
- If $\ell < k$, the problem is called *prediction*.
- If $\ell > k$, the problem is called *smoothing*.

Solution to all of the three problems is provided by the BFR [3] that are analytically tractable only for a few special models. Nevertheless, most of the state estimators can be seen as an approximative solution to the BFR and therefore the core of the NEF estimation component, the `nefEstimator` class follows the BFR idea. It provides the `estimate` method which is a universal interface for any estimation task.

The method calls the following methods

`timeUpdate`: $p(\boldsymbol{x}_{k+i}|z^k, \boldsymbol{u}^{k+i-1}) \rightarrow p(\boldsymbol{x}_{k+i+1}|z^k, \boldsymbol{u}^{k+i})$,

`measurementUpdate`: $p(\boldsymbol{x}_k|z^{k-1}, \boldsymbol{u}^{k-1}) \rightarrow p(\boldsymbol{x}_k|z^k, \boldsymbol{u}^k)$,

`smoothUpdate`: $p(\boldsymbol{x}_{k-i}|z^k, \boldsymbol{u}^k) \rightarrow p(\boldsymbol{x}_{k-i-1}|z^k, \boldsymbol{u}^k)$

Table 1: Estimators implemented in the NEF estimation component

name in NEF	estimators
`nefKalman`, `nefSKalman` `nefUDKalman`	(extended) Kalman filter (standard, square-root and UD versions)
`nefDD1`, `nefSDD1`, `nefDD2`	central difference Kalman filter, divided difference filter (1^{st} and 2^{nd} order) (standard and square-root version)
`nefUKF`, `nefSUKF`	unscented Kalman filter (standard and square-root version), cubature Kalman filter
`nefItKalman`	iterated Kalman filter based on any of the above local filter
`nefGSM`	Gaussian sum filter based on any of the above local filter
`nefPF`	bootstrap filter, generic particle filter, auxiliary particle filter, unscented particle filter.

Table 2: Estimation tasks supported by individual estimators

estimator	filtering	prediction	smoothing
`nefKalman`	✓	✓	✓
`nefSKalman`	✓	✓	✓
`nefUDKalman`	✓	✓	
`nefItKalman`	✓	✓	✓
`nefDD1`	✓	✓	✓
`nefSDD1`	✓	✓	✓
`nefDD2`	✓	✓	✓
`nefUKF`	✓	✓	✓
`nefSUKF`	✓	✓	✓
`nefGSM`	✓	✓	
`nefPF`	✓	✓	

that are implemented by individual estimators and in this way any of the three estimation problems can be solved by individual estimation methods.

Currently, the NEF provides implementation of the estimators given in Table 1. Details of estimators with estimation tasks they can solve are given in Table 2.

4 Performance evaluation component

As was mentioned above, the aim of estimation problem is to find a conditional pdf of the state. All the NEF estimators provide the estimates in the form of a distribution of the state. A common task is to measure estimation error and compare performance of several estimators against the true value of the state. Obtaining such a performance index requires a procedure consisting of i) collecting data from Monte Carlo (MC) simulations, ii) extracting appropriate indicators from the conditional distribution of the state provided by individual estimators and iii) evaluating the performance index. Within the NEF, this process is provided by the performance evaluator component. It contains methods for initialization, data processing and evaluating the performance index which is selected during initialization as well as the number of estimators to be compared and the expected number of MC simulations. The performance index can be evaluated at any time, not necessarily after completing all the MC simulations.

Table 3: Performance indices implemented in the NEF performance evaluation component

ABSOLUTE ERROR MEASURES	
MSEM	mean squared error matrix
RMSE	root mean squared error
AEE	average Euclidean error
HAE	harmonic average error
GAE	geometric average error
MEDE	median error
MODE	mode error
RELATIVE ERROR MEASURES	
RMSRE	root mean squared relative error
ARE	average Euclidean relative error
BEEQ	Bayesian estimation error quotient
EMER	estimation error relative to measurement error
PERFORMANCE MEASURES	
NCI	non-credibility index
ANEES	average normalized estimation error squared

Currently, the performance evaluation component provides performance indices [20, 21] given in Table 3.

5 NEF application in target tracking

In the following two subsections, application of the NEF in two target tracking examples will be illustrated.

5.1 Tracking a ship with unknown control

Suppose a ship with unknown control is tracked [22]. Its state is given by position and velocity in x and y directions, $\boldsymbol{x}_k = [x_k, y_k, \dot{x}_k, \dot{y}_k]$.

State transition is given by

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{u}_k + \boldsymbol{w}_k, \quad (5)$$

where $T = 0.02$ and $\boldsymbol{w}_k$ is a white noise normally distributed with zero mean and covariance matrix $Q = 10^{-6} \cdot \boldsymbol{I}$, where $\boldsymbol{I}$ is identity matrix.

As the control $\boldsymbol{u}_k$ is unknown, for state estimation purposes it will be appended to the state variable.

The following code creates the state function describing the above defined state transition with augmented unknown control:

```
F = [1 0 T 0;
     0 1 0 T;
     0 0 1 0;
     0 0 0 1];
G = [0 0;
     0 0;
     1 0;
     0 1];
Fm = [F G;zeros(2,4) eye(2)];
f = nefLinFunction(Fm,[],eye(6));
```

and the state noise is specified as

```
Q = 1e-6*eye(4);
Qm = [Q zeros(4,2);...
  zeros(2,4) 1e-3*eye(2)];
w = nefGaussianRV(zeros(6,1),Qm);
```

The position of the ship is measured in polar coordinates, i.e.

$$z_k = \begin{bmatrix} \arctan \frac{y_k}{x_k} \\ \sqrt{y_k^2 + x_k^2} \end{bmatrix} + v_k, \tag{6}$$

where v_k is a white noise normally distributed with zero mean and covariance matrix $R = \begin{bmatrix} 4 \cdot 10^{-4}(\pi/180) & 0 \\ 0 & 1 \cdot 10^{-4} \end{bmatrix}$. The measurement function h_k in (6) is nonlinear and is specified using the `nefHandleFunction` class as

```
h = nefHandleFunction(@(x,u,v,t) ...
 [atan(x(2)/x(1));sqrt(x(1)^2+x(2)^2)]...
 + v,[6 0 2 0]);
```

with the measurement noise given by

```
R = [4e-4*(pi/180)^2 0; 0 1e-4];
v = nefGaussianRV(zeros(2,1),R);
```

The initial state condition has normal distribution with mean $x_0 = [20,\ 50,\ 0,\ -12]^{\mathrm{T}}$ and covariance matrix $P_0' = I$. The initial state, state noise and measurement noise are considered to be independent, thus

```
m0 = [20 50 0 -12 0 0]';
P0 = diag([1e1 1e1 1e1 1e1 1e-1 1e-1]);
x0 = nefGaussianRV(m0,P0);
```

As the PF is to be used for estimation with a structurally specified model, also the loglikelihood of the measurement pdf must be specified:

```
LLH = @(z,x,u,t) -0.5*diag(...
  (z-[atan(x(2,:)./x(1,:));...
  sqrt(x(1,:).^2+x(2,:).^2)])'*inv(R)*...
  (z-[atan(x(2,:)./x(1,:));...
  sqrt(x(1,:).^2+x(2,:).^2)]))';
```

Now, the model is specified by

```
model = nefEqSystem(f,h,w,v,x0,...
  'logLikelihood',LLH);
```

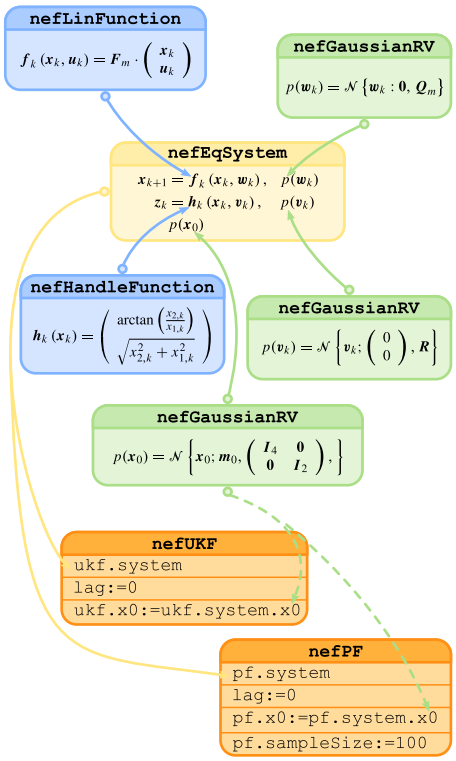

Figure 3: Scheme of the experiment setup in example 1.

Suppose, the unscented Kalman filter (UKF) and the PF will be used for estimation and specified by

```
PF = nefPF(model,'sampleSize',1000);
UKF = nefUKF(model);
```

Note that for the PF a probabilistic description of the model could also be used, but a structural description was already required by the UKF so it was also used for the PF. For illustration, the experiment setup is shown in Fig 3. Performance of the estimators will be compared in terms of the root mean square error [20] and the non-credibility index (NCI) [21]. The performance evaluators are set up for two estimators and 10 MC simulations as

```
filters = 2;
mcrun = 10;
RMSE_PE = nefPerformanceEvaluator(...
  model,K,mcrun,filters,'method','RMSE',...
  'idxState',[1:4]);
NCI_PE = nefPerformanceEvaluator(...
  model,K,mcrun,filters,'method','NCI',...
  'idxState',[1:4]);
```

Note, that only the first four states (i.e. position and velocity) will be taken into account during performance evaluation and the last two states (unknown control) will be ignored.

Now, 10 MC simulations will be performed with processing the results obtained from individual estimators in performance evaluators

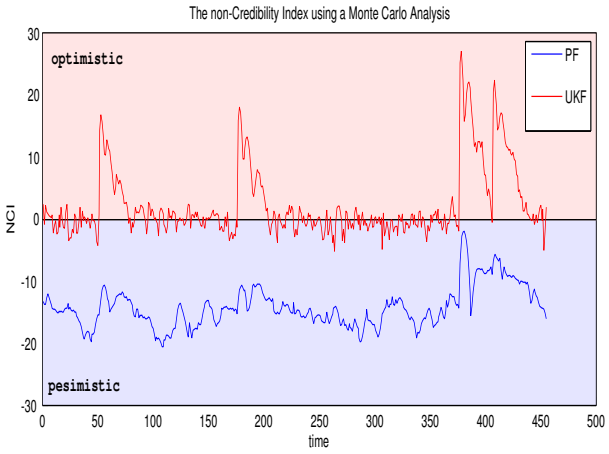

Figure 4: Performance indices computed in example 1.

```
for i = 1:mcrun
  [val_PF] = estimate(PF,z,u);
  [val_UKF] = estimate(UKF,z,u);
  for k = 1:K
    Data.state{1,k} = [x(:,k); u(:,k)];
    estData{1,1,k} = val_PF{k};
    estData{2,1,k} = val_UKF{k};
  end
  processData(RMSE_PE,Data,estData);
  processData(NCI_PE,Data,estData);
end
```

Finally, the performance indices are obtained using

```
rmse = performanceValue(RMSE_PE);
nci = performanceValue(NCI_PE);
```

with results depicted in Fig. 4.

5.2 Tracking with image data

In the second example suppose a scenario with three targets moving within a 2D space [23] is given and raw unthresholded data are used to track the targets [24].

The state of the target at time instant k is defined by the position (x_k, y_k) and velocity $(\dot{x}_k, \dot{y}_k)$ of the target in the x and y directions. The return intensity ℓ of the target, is considered to be known. Thus the target state is given by

$$\boldsymbol{x}_k = [x_k, \dot{x}_k, y_k, \dot{y}_k]^{\mathrm{T}}.$$

The l-th target state evolves according to the discrete-time linear Gaussian model [25] as

$$\boldsymbol{x}_{l,k+1} = \boldsymbol{F}\boldsymbol{x}_{l,k} + \boldsymbol{\Gamma}\boldsymbol{w}_k, \tag{7}$$

where $\boldsymbol{w}_k$ is a zero mean white Gaussian noise with covariance matrix $\boldsymbol{Q}$. The matrix $\boldsymbol{\Gamma}$ was added to take into account possible complex behavior of the target. This adjustment is necessary for a PF with prior sampling density. As the

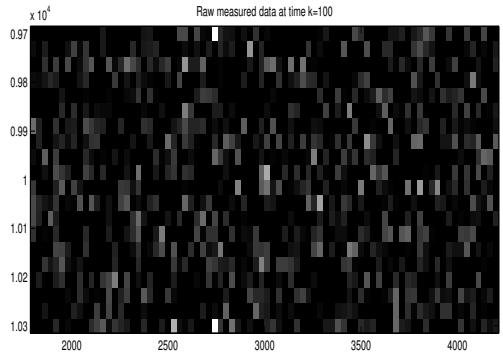

Figure 5: An example of measured raw data.

constant velocity process model is considered, the transition matrix $\boldsymbol{F}$ is given by

$$\boldsymbol{F} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

and the covariance matrix of the state noise by

$$\boldsymbol{Q} = \begin{bmatrix} q_s T^3/3 & q_s T^2/2 & 0 & 0 \\ q_s T^2/2 & q_s T & 0 & 0 \\ 0 & 0 & q_s T^3/3 & q_s T^2/2 \\ 0 & 0 & q_s T^2/2 & q_s T \end{bmatrix}, \tag{9}$$

where T is the sampling period and q_s is the power spectral density of the acceleration noise in the spatial dimensions. The prior pdf of the target state upon its appearance at time k is denoted as $p_b(\boldsymbol{x}_k)$ and is known for each of the targets.

Within the NEF, the model dynamics is specified by

```
T = 1;
qs = 0.001;
F = [1 T 0 0;
     0 1 0 0;
     0 0 1 T;
     0 0 0 1];
Q = [qs/3*T^3 qs/2*T^2 0         0;
     qs/2*T^2 qs*T     0         0;
     0        0        qs/3*T^3 qs/2*T^2;
     0        0        qs/2*T^2 qs*T];
Gamma = sqrt(diag([1e4 1e3 1e4 1e3]));
f = nefLinFunction(F,[],eye(4));
w = nefGaussianRV(zeros(4,1),...
  Gamma*Q*Gamma');
```

The measurements obtained as a result of sensor signal processing are in the form of a sequence of images [25]. The measurement $\bar{\boldsymbol{z}}_k$ at time instant k is assumed to be a two-dimensional image consisting of n_x cells in the x direction and n_y cells in the y direction. An example of the measurement at a time instant is in Fig. 5 Hence, each measurement $\bar{\boldsymbol{z}}_k$ is a set $\{z_k^{(i,j)}\}_{i=1,j=1}^{n_x,n_y}$ of values of measured intensity at

each cell. Each cell (i, j) contains a contribution of the l-th target denoted as $h_l^{(i,j)}(\boldsymbol{x}_k)$ and noise denoted as $v_k^{(i,j)}$, i.e.

$$z_k^{(i,j)} = \sum_{l=1}^{3} h_l^{(i,j)}(\boldsymbol{x}_k) + v_k^{(i,j)} \tag{10}$$

The measurement noise $v_k^{(i,j)}$ is considered to be a white zero-mean Gaussian noise with variance R for all cells. Note that the noise is independent between cells.

The nonlinear function $h_l^{(i,j)}(\boldsymbol{x}_k)$ representing contribution of the l-th target to each cell is given by

$$h_l^{(i,j)}(\boldsymbol{x}_k) = \frac{\Delta_x \Delta_y \ell}{2\pi\Sigma^2} \exp\left(-\frac{(\Delta_x i - x_{l,k})^2 + (\Delta_y j - y_{l,k})^2}{2\Sigma^2}\right), \tag{11}$$

where Δ_x and Δ_y represent the size of a cell in the x and y directions respectively, ℓ is the known intensity.

The spread of the measurement contributed by the target is modelled by a two-dimensional Gaussian distribution. As the state estimation methods assume the measurement to be a vector, let us consider the measurement $\bar{z}_k$ and the set of measurement functions $\{h^{(i,j)}(\boldsymbol{x}_k)\}_{i=1,j=1}^{n_x,n_y}$ in (11) to be stacked up as columns of $n_x \cdot n_y$ elements as $\boldsymbol{z}_k$ and $\boldsymbol{h}(\boldsymbol{x}_k)$, respectively.

So, in the NEF the measurement part of the model is specified as

```
nz = nx*ny;
h = nefHandleFunction(@(x,u,v,k) ...
  hfun(opt,x,u,v,k),[4 0 nz 0]);
v = nefGaussianRV(zeros(nz,1),...
  eye(nz)*sigma^2);
```

where `hfun` is a name of an m-file implementing the function (11) and `opt` is a structure containing all parameters of (11).

Now, the individual models will be specified together with their initial conditions

```
x1= [2000 10 10000 0]';
x2= [2000 13 10250 0]';
x3= [2000 13 9750 0]';
Px0 = 1e-6*eye(4);
x0_1 = nefGaussianRV(x1,Px0);
x0_2 = nefGaussianRV(x2,Px0);
x0_3 = nefGaussianRV(x3,Px0);
model1 = nefEqSystem(f,h,w,v,x0_1,...
  'logLikelihood',logLikelihood_handle);
model2 = nefEqSystem(f,h,w,v,x0_2,...
  'logLikelihood',logLikelihood_handle);
model3 = nefEqSystem(f,h,w,v,x0_3,...
  'logLikelihood',logLikelihood_handle);
```

The specification of `logLikelihood_handle` property is mandatory if the PF with prior sampling density (default setting) is used with a structurally defined model. The value

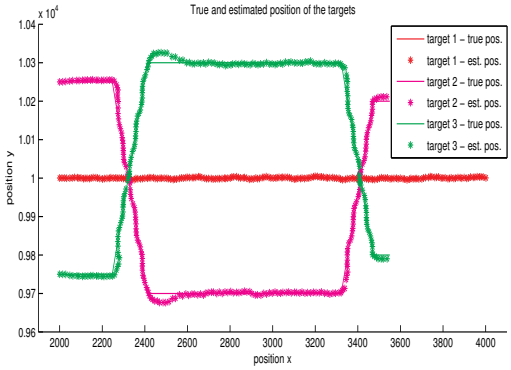

Figure 6: True positions of the targets and their estimates

is a handle to a MATLAB function providing a natural logarithm of the measurement pdf $p(\boldsymbol{z}_k|\boldsymbol{x}_k)$. Now a PF is set up for each model

```
PF1 = nefPF(model1,'sampleSize',1000);
PF2 = nefPF(model2,'sampleSize',1000);
PF3 = nefPF(model3,'sampleSize',1000);
```

and the estimation process is executed using

```
est_PF1 = estimate(PF1,z,u);
est_PF2 = estimate(PF2,z,u);
est_PF3 = estimate(PF3,z,u);
```

The method `estimate` provides the conditional pdf of the state $p(\boldsymbol{x}_k|\boldsymbol{z}^k)$. To obtain the mean value from the pdf, the method `evalMean` can be used in the following manner

```
for k = 1:K
  xm_PF1(:,k) = evalMean(est_PF1{k});
  xm_PF2(:,k) = evalMean(est_PF2{k});
  xm_PF3(:,k) = evalMean(est_PF3{k});
end
```

The result of estimation can be seen in Fig. 6 for three targets crossing [23].

6 Conclusion

The paper presented the Nonlinear Estimation Framework[1] that facilitates implementation, testing, use and evaluation of nonlinear state estimation methods. It is directed at both the casual and the experienced user. It provides easy to use, however, powerful tools for straightforward estimation experiment design. The framework consists of three components. The underlying component aims at describing a nonlinear stochastic discrete-time dynamic state-space model and simulating its behavior. The most extensive component contains implementation of many state estimation algorithms both local and global. The third, recently introduced component focuses on evaluation of performance of

[1]The framework is freely available for non-commercial use. Information about availability can be found at the web page `http://nft.kky.zcu.cz/nef`.

the estimators and offers a number of various performance measures.

The usage of the NEF was illustrated in two target tracking examples, more specifically tracking a ship with unknown control and tracking three object using raw data.

Acknowledgement

The work was supported by the Ministry of Education, Youth and Sports of the Czech Republic, project 1M0572, and by the Czech Science Foundation, project GACR 102/08/0442.

References

[1] E. Blasch. *Derivation of a belief filter for high range resolution radar simultaneous target tracking and identification*. PhD thesis, Wright State University, 1999.

[2] W. Koch. Fixed-interval retrodiction approach to Bayesian IMM-MHT for maneuvering multiple targets. *IEEE Transactions on Aerospace and Electronic Systems*, 36(1):2–14, 2000.

[3] J. V. Candy. *Bayesian signal processing*. Wiley, 2008.

[4] M. Šimandl. State Estimation For Non-Gaussian Models. In *Preprints of the 13th IFAC World Congress*, volume 2, pages 463–468, San Francisco, USA, 1996. IFAC, Elsevier Science.

[5] M. Šimandl. Multi-process models and outliers elimination. In *Proceedings of IASTED Int. Conf. SIP'97*, pages 320–325, New Orleans, USA, 1997. IASTED.

[6] A.H. Jazwinski. *Stochastic Processes and filtering Theory*. Academic Press, 1970.

[7] M. Nörgaard, N.K. Poulsen, and O. Ravn. New developments in state estimation for nonlinear systems. *Automatica*, 36(11):1627–1638, 2000.

[8] K. Ito and K. Xiong. Gaussian Filters for Nonlinear Filtering Problems. *IEEE Trans. on Automatic Control*, 45(5):910–927, 2000.

[9] S.J. Julier, J.K. Uhlmann, and H.F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482, 2000.

[10] H.W. Sorenson and D.L. Alspach. Recursive Bayesian Estimation Using Gaussian Sums. *Automatica*, 7:465–479, 1971.

[11] M. Šimandl and M. Flídr. Nonlinear nonnormal dynamic models: State estimation and software. In *Computer-Intensive Methods in Control and Signal Processing*. Birkhäuser, Boston, 1997.

[12] N. Gordon, D. Salmond, and A.F.M. Smith. Novel approach to nonlinear/ non-Gaussian Bayesian state estimation. *IEE proceedings-F*, 140:107–113, 1993.

[13] M. Šimandl and O. Straka. Functional sampling density design for particle filters. *Signal Processing*, 88(11):2784–2789, 2008.

[14] M. Nörgaard, N.K. Poulsen, and O. Ravn. Kalmtool for use with MATLAB. In *Proceedings of the 13th IFAC Symposium on System Identification, SYSID03*, pages 1490–1495. IFAC, 2003. http://server.oersted.dtu.dk/personal/or/kalmtool3/.

[15] R. van der Merwe and E. A. Wan. ReBEL - Recursive Bayesian Estimation Library, 2006. http://choosh.csee.ogi.edu/rebel/.

[16] O. Straka, M. Flídr, J. Duník, and M. Šimandl. A Software Framework and Tool for Nonlinear State Estimation. In *Proceedings of the 15th IFAC Symposium on System Identification*, pages 510–515, Saint-Malo, France, 2009.

[17] S. Särkkä and J. Hartikainen. EKF/UKF Toolbox, 2007. http://www.lce.hut.fi/research/mm/ekfukf/.

[18] M. Flídr, J. Duník, O. Straka, M. Šimandl, and J. Švácha. Framework for implementing and testing nonlinear filters. In *Proceedings of the 7th IFAC Symposium on Advances in Control Education*, Madrid, Spain, 2006.

[19] J. Švácha, M. Šimandl, O. Straka, and M. Flídr. Nonlinear filtering toolbox for continuous stochastic systems with discrete measurements. In *Proceedings of the 7th IFAC Symposium on Advances in Control Education*, Madrid, Spain, 2006.

[20] X.R. Li and Z. Zhao. Evaluation of estimation algorithms part I: incomprehensive measures of performance. *IEEE Transactions on Aerospace and Electronic Systems*, 42(4):1340–1358, 2006.

[21] E.P. Blasch, A. Rice, and C. Yang. Nonlinear track evaluation using absolute and relative metrics. In *Proceedings of SPIE, the International Society for Optical Engineering*, pages 62360–62360. SPIE, 2006.

[22] J. V. Candy. *Model-Based Signal Processing*. John Wiley & Sons, Hoboken, New Jersey, 2006.

[23] E. Blasch and D. Kahler. Multiresolution EO/IR Target Track and ID. In *SIF Proc. Fusion05*, 2005.

[24] HM Shertukde and Y. Bar-Shalom. Tracking of crossing targets with imaging sensors. *IEEE Transactions on Aerospace and Electronic Systems*, 27(4):582–592, 1991.

[25] B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House, 2004.