Bayesian Filters in Practice

Low-cost mobile robot position tracking using bearing only beacons

Jiří Krejsa
Centre of Mechatronics
Institute of Thermomechanics AS CR, v.v.i.
Brno, Czech Republic
krejsa@fme.vutbr.cz

Stanislav Věchet
Faculty of Mechanical Engineering
Brno University of Technology
Brno, Czech Republic
vechet.s@fme.vutbr.cz

***Abstract*—Bayesian filters represent the most commonly used tool for state estimation not only in mobile robotics. The filters are widely used in sensor data fusion and robot localization problems. The paper describes in detail our experiences with the filters in robot localization using bearing only beacons. Bearing only beacons are easy to implement, therefore can be realized by the students and relatively complex task of Bayesian filtering can be explained using real data. Both simulation and practical results with Extended Kalman filter and Unscented Kalman filter are given, taking into consideration not only the precision of obtained estimate, but also its robustness against the noise and memory and computational requirements that must be considered when computational resources are limited.**

Mobile robot localization, bearing only beacons, Bayesian filters

I. INTRODUCTION

Bayesian filters are commonly used tools whenever certain quantity can not be expressed as a single value / vector, but estimate is used instead, taking into consideration the probabilistic nature of the quantity. Such situation often appears in data fusion, when data measured by the sensors are affected with certain level of noise. As an example the fusion of odometry readings and compass measurements can be given. Typical task that requires the probabilistic approach in quantities description is the position tracking problem (local localization). Bayesian filters are widely used to address this problem even to the extent of global localization and simultaneous localization and mapping problem, when the ability to model the desired quantity (robot position) with multimodal probability distribution is essential [1], [2].

The task of position tracking requires the fusion of robot motion model and sensor data about the environment. This task is essential in mobile robot navigation, and students in the field are often introduced to the state estimation problem when examining the problem.

Robot motion can use either the velocity information from the controllers, or the odometry information read from IRC sensors on robot wheels. Environment data are usually in the form of landmarks, e.g. visual landmarks extracted from the images acquired by the camera [3], [4]. Such approach has the advantage of relatively low cost, as the prices of image acquiring systems are low and still dropping. However the computational cost of image processing is rather high. Our goal was to develop a position tracking method usable in cases when computational resources are rather limited (that is often the case in simple low cost educational robots). The method, described further in more detail uses infrared beacons placed on known locations and receiver located on the robot, capable of detecting the relative angle between the beacons and the robot, together with the information about the beacon identification, thus solving the mapping problem. Robot position is estimated by Bayesian filter that combines the motion model with the receiver measurement.

Low cost of bearing only beacons predestinates its use in education of robotics. Robot that uses beacons based localization can be built by the students themselves in reasonable time, as the further described method does not require odometry readings. Understanding the Bayes filters is essential for almost any task in robotics whenever coping with uncertainty. However although the literature on this field is vast, e.g. [5], it is not easy to find a text directly usable by the students to write down a real robot application. Hopefully this paper can serve as a guide not only to understand the use of the filters in beacon based localization, but also to address the practical problems often neglected in theoretical literature.

The paper is organized in following way. Chapter II gives detailed overview of the task in question and describes Bayesian filters used. Chapter III shows the results of simulations while chapter IV shows the results obtained by the experimental robot and chapter V brings the conclusions.

II. MATERIALS AND METHODS

A. The task in detail

The problem to be solved is to determine the position of the robot on the plane (2D problem) given the motion model of the robot, beacons relative angles measurement, beacons positions and initial position. In other words, we know the control actions of the robot and absolute fixed position of the beacons and we measure the relative angles of beacons with respect to the robot. From these data we determine the position of the robot.

The position is given by x and y coordinate of the robot $\left[x^R, y^R\right]$ and its heading angle φ^R with respect to fixed

global coordinate system. As the position changes in time, such position in time step k is further denoted as the state vector $\mathbf{x}_k$:

$$\mathbf{x}_k = \begin{bmatrix} x_k^R & y_k^R & \varphi_k^R \end{bmatrix}^T \tag{1}$$

The state changes as the robot moves. State change is evoked by actions: the translational and rotational velocities of the robot u^t and u^r. Motion model gives the relation between the actions and state vector change.

There are N beacons available in known fixed locations given by the x and y coordinates in global coordinate system. See the overview in Fig. 1.

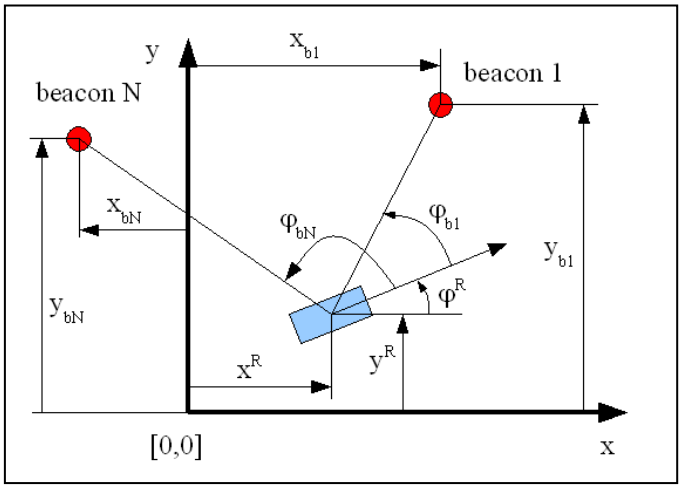

Figure 1. The state of the robot, beacons positions and beacons measurement

Due to the imperfections of the motion model and noise in the measurement, the state can not be represented only as the vector of values and probabilistic approach must be considered. As the problem in hand is of the position tracking, the multimodal probability density can be avoided and simple Gaussian approximation is used. The description of the underlying model can therefore be defined as

$$\begin{aligned} \mathbf{x}_{k+1} &= f\left(\mathbf{x}_k, \mathbf{u}_k, k\right) + \mathbf{v}_k \\ \mathbf{y}_k &= h\left(\mathbf{x}_k, k\right) + \mathbf{w}_k \end{aligned} \tag{2}$$

where $\mathbf{x}_k$ is n dimensional vector of states (robot position), $\mathbf{u}_k$ is action vector (velocities given by the controller), $\mathbf{v}_k$ is white Gaussian process noise (representing the imperfections of motion model) with zero mean and covariance matrix $\mathbf{V}_k$, $\mathbf{y}_k$ is system output, $\mathbf{w}_k$ is white Gaussian measurement noise with zero mean and covariance matrix $\mathbf{W}_k$ and f and h are continuously differentiable nonlinear functions. While in general those functions can depend on time step index k, if further examples such dependency is not used.

The state transition function f defines how the state (robot position) changes when action is applied:

$$\mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_k\right) + \mathbf{v}_k = \begin{bmatrix} \cos\varphi_k^R u_k^t + x_k^R \\ \sin\varphi_k^R u_k^t + y_k^R \\ u_k^r + \varphi_k^R \end{bmatrix} + \mathbf{v}_k \tag{3}$$

Regarding the measurements, as there are N beacons generally available (however, there is no guarantee that all the beacons are detected), their positions are (see Fig. 1): applied:

$$\mathbf{x}_{bi} = \begin{bmatrix} x_{bi}, y_{bi} \end{bmatrix}, i = 1, 2, \ldots, N \tag{4}$$

For a single beacon the output equation is

$$\mathbf{y}_{1k} = \begin{bmatrix} h_1\left(\mathbf{x}_k, \mathbf{x}_{b1}\right) \end{bmatrix} + \begin{bmatrix} \mathbf{w}_{1k} \end{bmatrix} \tag{5}$$

where

$$h_1\left(\mathbf{x}_k, \mathbf{x}_{b1}\right) = \begin{bmatrix} \operatorname{atan2}\left(y_k^R - y_{b1}, x_k^R - x_{b1}\right) - \varphi_k^R \end{bmatrix} \tag{6}$$

For N beacons the equations are simply added one by one depending on what beacon was measured. In general:

$$\mathbf{y}_k = \begin{bmatrix} h_1\left(\mathbf{x}_k, \mathbf{x}_{m1}\right) \\ h_2\left(\mathbf{x}_k, \mathbf{x}_{m2}\right) \\ \vdots \\ h_{n_m}\left(\mathbf{x}_k, \mathbf{x}_{mn_m}\right) \end{bmatrix} + \begin{bmatrix} \mathbf{w}_{1k} \\ \mathbf{w}_{2k} \\ \vdots \\ \mathbf{w}_{n_m k} \end{bmatrix} \tag{7}$$

Now with the task properly defined, our goal is to produce the state estimate with mean $\hat{\mathbf{x}}_k$ and covariance matrix $\mathbf{P}_k$. This can be done by Bayesian filters.

B. Extended Kalman filter

There is a number of Bayesian filters suitable for the problem in hand. They differ by the representation of the estimate. The Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) both represent the estimate by random vector with Gaussian distribution. Both filters work in usual predictor/corrector manner. The difference is in the way the filters use for linearization of nonlinear functions f and h from (2). EKF uses the Taylor Expansion and therefore requires partial derivatives of the functions, while UKF uses linearization via the Unscented Transform. Let's first take a look at EKF.

The estimator works in two stages. First the motion model is applied (estimate changes when action is performed) and new position estimate is predicted. In second stage the measurement is taken into account (beacons relative angles)

and position estimate is corrected. Therefore the input is the current estimate (at the beginning of the whole process this estimate is equal to initial estimate of robot position), the action and the measurement. Necessary relations are given by following equations:

Prediction (state estimate change when action is applied:

$$\begin{aligned}\hat{\mathbf{x}}_{k+1|k} &= f\left(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k, k\right)\\ \mathbf{P}_{k+1|k} &= \mathbf{F}_k \mathbf{P}_{k|k} \mathbf{F}_k^T + \mathbf{V}_k\end{aligned} \tag{7}$$

Update (correction of the estimate using measured data):

$$\begin{aligned}\hat{\mathbf{x}}_{k+1|k+1} &= \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}\tilde{\mathbf{y}}_{k+1}\\ \mathbf{P}_{k+1|k+1} &= \mathbf{P}_{k+1|k} - \mathbf{K}_{k+1}\mathbf{H}_{k+1}\mathbf{P}_{k+1|k}\end{aligned} \tag{8}$$

where:

$$\mathbf{F}_k = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k|k}}$$

$$\tilde{\mathbf{y}}_{k+1} = \mathbf{y}_{k+1} - h\left(\hat{\mathbf{x}}_{k+1|k}, k+1\right)$$

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^T\mathbf{S}_{k+1}^{-1}$$

$$\mathbf{S}_{k+1} = \mathbf{H}_{k+1}\mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^T + \mathbf{W}_{k+1}$$

$$\mathbf{H}_{k+1} = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k+1|k}}$$

EKF requires the partial derivatives of f and h functions. For the motion model, the partial derivatives forming Jacobian $\mathbf{F}_k$ are given by:

$$\mathbf{F}_k = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{k|k}} = \begin{bmatrix} 1 & 0 & -\sin\hat{\varphi}_{k|k}^R u_k^t \\ 0 & 1 & \cos\hat{\varphi}_{k|k}^R u_k^t \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

And Jacobian for a single beacon is (the matrix is transposed to fit within the text):

$$\mathbf{H}_{k+1,1} = \begin{bmatrix} \dfrac{1}{1+\left(\dfrac{y_{b1}-\hat{y}_{k+1|k}^R}{x_{b1}-\hat{x}_{k+1|k}^R}\right)^2}\dfrac{y_{b1}-\hat{y}_{k+1|k}^R}{\left(x_{b1}-\hat{x}_{k+1|k}^R\right)^2} \\ -\dfrac{1}{1+\left(\dfrac{y_{b1}-\hat{y}_{k+1|k}^R}{x_{b1}-\hat{x}_{k+1|k}^R}\right)^2}\dfrac{1}{x_{b1}-\hat{x}_{k+1|k}^R} \\ -1 \end{bmatrix}^T \tag{10}$$

Jacobains for more beacons are added the same way as the output equations (7).

C. *Unscented Kalman filter*

The Taylor series expansion applied by the EKF is not the only way to linearize the transformation of Gaussians. The Unscented Kalman Filter uses so-called unscented transform. The principle is following: UKF deterministically extracts so-called sigma points from the Gaussian and passes them through nonlinear function. Resulting mean and covariances are then extracted from transformed sigma points. This method might remind of the Monte Carlo method or Particle filter that uses samples from the distribution as distribution representation. The difference is that the sigma points are chosen deterministically, not randomly.

The generation of sigma points $\boldsymbol{\mathcal{X}}$ for n-dimensional random vector $\mathbf{x}$ in UKF is given by the following rule:

$$\begin{aligned}\boldsymbol{\mathcal{X}}_0 &= E(\mathbf{x})\\ \boldsymbol{\mathcal{X}}_i &= E(\mathbf{x}) + \left(\sqrt{(n+\kappa)\mathbf{P}_\mathbf{x}}\right)\\ \boldsymbol{\mathcal{X}}_{n+i} &= E(\mathbf{x}) - \left(\sqrt{(n+\kappa)\mathbf{P}_\mathbf{x}}\right)\end{aligned} \tag{11}$$

Where $E(\mathbf{x})$ is the mean of random vector $\mathbf{x}$ and $\mathbf{P}_\mathbf{x}$ is the covariance matrix. Parameter κ determines how far the sigma points are spread around the mean. n-dimensional random vector produces $2n+1$ sigma points.

Sigma points are transformed by the nonlinear function (in our case there two such functions, f and h).

$$\boldsymbol{\mathcal{Y}}_i = f\left(\boldsymbol{\mathcal{X}}_i\right) \tag{12}$$

Each sigma point is accompanied by the weight used to determine the mean and covariance after the transformation.

$$\begin{aligned}W_0 &= \kappa/(n+\kappa)\\ W_i &= 1/2(n+\kappa),\ i = 1\ldots 2n\end{aligned} \tag{13}$$

The mean is given as the weighted mean of transformed sigma points

$$E(\boldsymbol{\mathcal{Y}}) = \sum_{i=0}^{2n} W_i \boldsymbol{\mathcal{Y}}_i \tag{14}$$

And corresponding covariance matrix as

$$\mathbf{P}_y = \sum_{i=0}^{2n} W_i \left(\boldsymbol{\mathcal{Y}}_i - E(\boldsymbol{\mathcal{Y}})\right)\left(\boldsymbol{\mathcal{Y}}_i - E(\boldsymbol{\mathcal{Y}})\right)^T \tag{15}$$

In order to use UKF in position tracking task, the predictor / corrector structure is kept, however the way the mean and covariance of the estimate is calculated is different from EKF. As there are two nonlinear functions (process and measurement), the unscented transform must be used twice. During the prediction first the sigma points are generated according to (11) from the current estimate mean and its covariance matrix, together with corresponding weights (13). Sigma points are then transferred through process function f (12) and prediction mean and covariance are calculated according to (14, 15), with process noise added. The prediction step formulas are therefore as follows:

- $\boldsymbol{\mathcal{X}}_{k|k} = \begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ \hat{\mathbf{x}}_{k|k} + \left(\sqrt{(n+\kappa)\mathbf{P}_{k|k}}\right) \\ \hat{\mathbf{x}}_{k|k} - \left(\sqrt{(n+\kappa)\mathbf{P}_{k|k}}\right) \end{bmatrix}^T$
- $W_0 = \kappa / (n+\kappa)$
- $W_i = 1/2(n+\kappa),\ for\ \ i = 1...2n$
- $\breve{\boldsymbol{\mathcal{X}}}_{k+1|k} = f\left(\boldsymbol{\mathcal{X}}_{k+1|k}, \mathbf{u}_k, k\right)$
- $\hat{\mathbf{x}}_{k+1|k} = \sum_{i=0}^{2n} W_i \breve{\boldsymbol{\mathcal{X}}}_{i,k+1|k}$
- $\mathbf{P}_{k+1|k} = \sum_{i=0}^{2n} W_i \left(\breve{\boldsymbol{\mathcal{X}}}_{i,k+1|k} - \hat{\mathbf{x}}_{k+1|k}\right)\left(\breve{\boldsymbol{\mathcal{X}}}_{i,k+1|k} - \hat{\mathbf{x}}_{k+1|k}\right)^T + \mathbf{V}_k$

In the correction (update) step, the new set of sigma points is generated (based on the mean and covariance from prediction step), sigma points are again passed through h (nonlinear function of measurement) and the mean is calculated together with the covariance. Finally the mean and covariance of the new estimated is calculated. The correction step formulas are as follows:

- $\boldsymbol{\mathcal{Z}}_{k+1|k} = \begin{bmatrix} \hat{\mathbf{x}}_{k+1|k} \\ \hat{\mathbf{x}}_{k+1|k} + \left(\sqrt{(n+\kappa)\mathbf{P}_{k+1|k}}\right) \\ \hat{\mathbf{x}}_{k+1|k} - \left(\sqrt{(n+\kappa)\mathbf{P}_{k+1|k}}\right) \end{bmatrix}^T$
- $\breve{\boldsymbol{\mathcal{Z}}}_{k+1|k} = h\left(\boldsymbol{\mathcal{Z}}_{k+1|k}, k+1\right)$
- $\hat{\mathbf{z}}_{k+1|k} = \sum_{i=0}^{2n} W_i \breve{\boldsymbol{\mathcal{Z}}}_{i,k+1|k}$
- $\mathbf{S}_{k+1} = \sum_{i=0}^{2n} W_i \left(\breve{\boldsymbol{\mathcal{Z}}}_i - \hat{\mathbf{z}}_{k+1|k}\right)\left(\breve{\boldsymbol{\mathcal{Z}}}_i - \hat{\mathbf{z}}_{k+1|k}\right)^T + \mathbf{W}_{k+1}$
- $\mathbf{P}^{x,z}_{k+1|k} = \sum_{i=0}^{2n} W_i \left(\breve{\boldsymbol{\mathcal{X}}}_i - \hat{\mathbf{x}}_{k+1|k}\right)\left(\breve{\boldsymbol{\mathcal{Z}}}_i - \hat{\mathbf{z}}_{k+1|k}\right)^T$
- $\mathbf{K}_{k+1} = \mathbf{P}^{x,z}_{k+1|k} \mathbf{S}^{-1}_{k+1}$
- $\tilde{\mathbf{y}}_{k+1} = \mathbf{y}_{k+1} - \hat{\mathbf{z}}_{k+1|k}$
- $\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}\tilde{\mathbf{y}}_{k+1}$
- $\mathbf{P}_{k+1|k+1} = \mathbf{P}_{k+1|k} - \mathbf{K}_{k+1}\mathbf{S}_{k+1}\mathbf{K}^T_{k+1}$

The UKF does not require the calculation of Jacobian matrices and it better copes when the resulting distribution is far from Gaussian (typically when there are large differences in variances). However, these nice features are devalved by increased computational expense.

III. Simulation Results

Both methods were first implemented for simulation purposes using Matlab. The implementation is straight forward from the equations above, as Matlab supports vector / matrix operations. The number of simulation tests were performed with various levels of process and measurement noise, different probability of the beacons to actually work at all, etc. The example of comparing the estimate generated by both EKF and UKF is shown on Fig. 2. and Fig. 3. Fig. 2 shows the mean course together with the confidential ellipses drawn for x and y coordinate only (angle is not considered). Fig. 3. shows the same example with the robot chassis sketch, so the actual angle of the robot can be seen.

During the tests the probability of the beacon to be seen was set to 50% in each step. Simulated variance of beacons relative angle measurement was set to $\pi/8$.

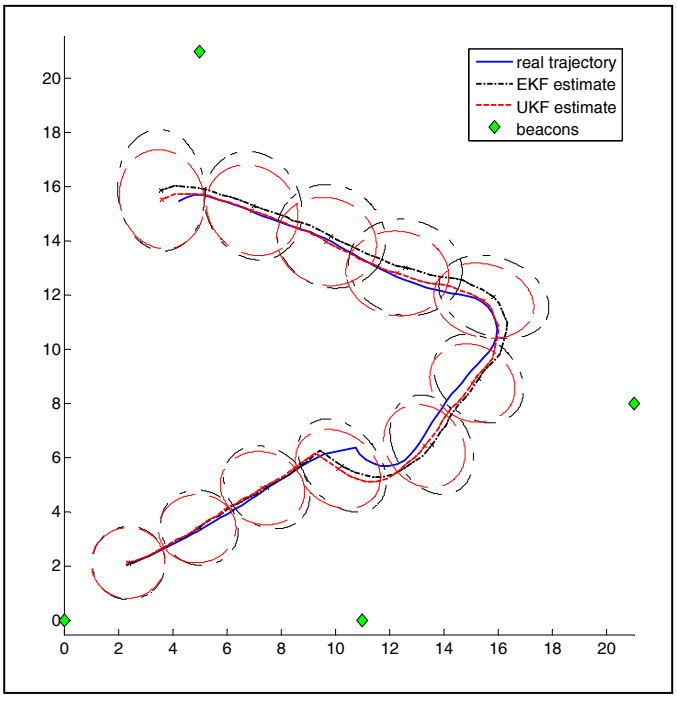

Figure 2. Comparison of real (simulated) trajectory and estimates produced by EKF and UKF filters.

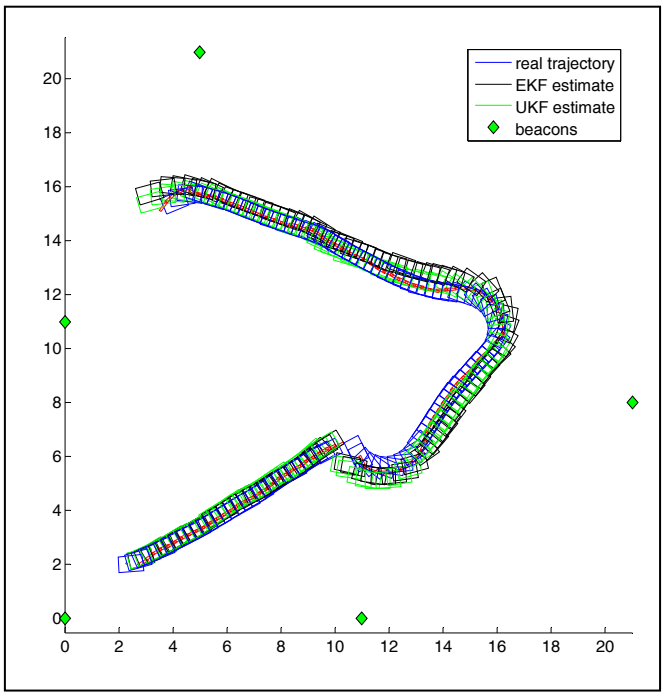

Figure 3. Comparison of real trajectory and the estimates with robot actual size drawn

To illustrate the robustness of the method, during the test a sudden change in robot's position is applied (robot is moved for 1 meter in x direction, 0.5m in y direction and turned clockwise for $\pi/2$). Actions are generated by the simple planner (motion towards randomly generated goal).

We can see that both filters can localize the robot successfully; there is no significant difference in the mean nor the covariances. Filters can handle the sudden change in robot position and converge towards correct position. Further tests indicate that localization still converges for probability of beacons to be detected dropped to 15 % (for 4 beacons).

While the implementation of UKF is somewhat simpler (there is no need to calculate the Jacobians), the single step of the filter is about 4 times slower compared to EKF. This is not a huge difference, however, when computational resources are limited, such difference can be essential. This was proved during the experiments with real robot, described in following chapter.

IV. Experimental Results

Both methods were verified with the experimental robot Leela, equipped with the infrared beacons scanner, see Fig. 4. for the ring of receivers around the neck. The scanner covers full 360 degrees range. The robot has differential drive chassis and while equipped with IRC sensors on wheels, the odometry readings were not used to determine the actions, both translational and rotational velocities were taken from the controller. Controller performance imperfections, wheel slippage and other noise sources were all modeled by the process noise matrix $\mathbf{V}_k$.

Figure 4. Experimental robot Leela used for filter verification

The beacon scanner and the emitters communicate with each other using two wireless technologies. The low power comsuption radio modules with free 433MHz modulation are used for one way data transmision, in the direction from the scanner to the beacons. This way the proper beacon is selected for transmission and thus beacon identification problem is solved. The beacon starts with data transmition immediatelly after the proper identification number is received. The transmition from selected beacon to the scanner is also one way and it is based on infrared principle. The beacon uses infrared LED (880nm light-wave) with carrier frequency 38kHz. This signal is detected by infrared receivers on the scanner. As the emitted infrared light is easy to missdetect, software filter is implemented to calculate proper beacon relative angle.

In order to determine the true position of the robot, the image processing of images acquired during the test by the static camera mounted above the experimental grounds was used.

The implementation of both filters for the robot was done in C language, with our own implementation of necessary matrix operations. All the code runs on 8-bit AVR family processor. ATmega128 is a low cost 8-bit processor with programmable 128kB flash memory, two serial communication interfaces and eight analog to digital converters running on 16MHz frequency. The processor acts as the main control unit and it runs both the localization algorithm and path planner.

The path planner is based on the position estimate, so we can not directly compare the filters during single run. Moreover, both filters can not run simultaneously as the processing time would prolong the sampling rate to unusable level. Therefore the examples, given on Fig. 5 and Fig. 6. show independently the comparison between real trajectory of the robot and position estimate produced by the filters during the travel. The tests were performed in the lobby of A4 building at the Faculty of Mechanical Engineering, the obstacles are drawn on the figures just to give an idea about the environment.

One can see that with the real robot the estimation corresponds to the simulation results for both filters. The UKF

position estimation error is slightly higher, however, this is not caused by the behavior of the filter itself, but by the lower sampling rate due to higher computational demands of the filter.

.

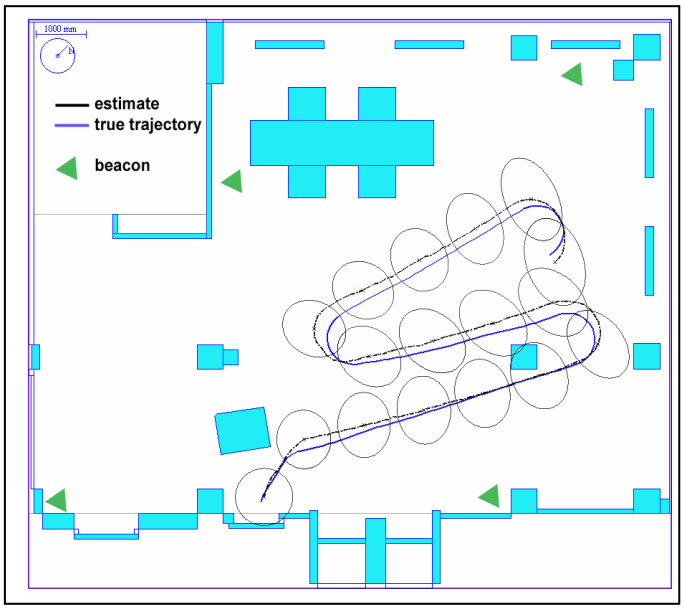

Figure 5. The real trajectory of the robot and corresponding estimate obtained using Extended Kalman filter

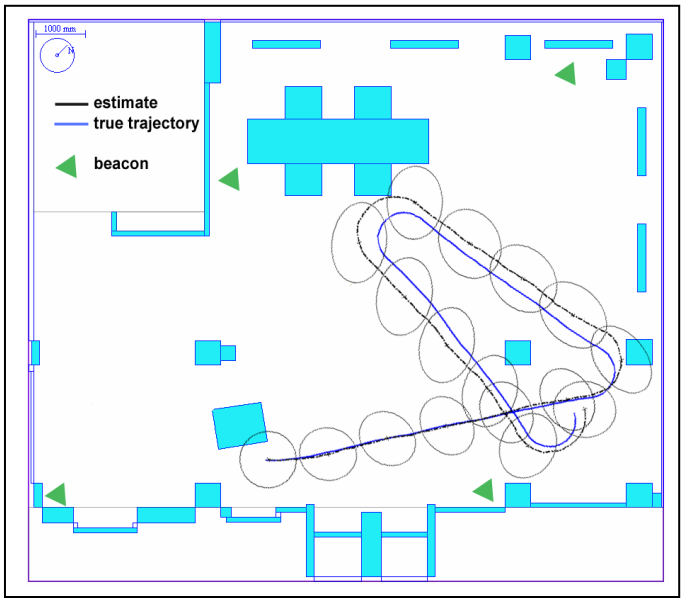

Figure 6. The real trajectory of the robot and corresponding estimate obtained using Unscented Kalman filter

V. CONCLUCIONS AND FUTURE WORK

Bayes filters are the basic tools for state estimation in robotics. The Extended Kalman filter and Unscented Kalman filter are among others (e.g. Particle filter) the most popular estimators nowadays. The mobile robot localization problem can be considered solved using landmarks extraction and further Bayesian filtering, however, there are applications where the beacons can justify its main drawback – the necessity to position the beacons in given locations. Such application can profit from very low computational requirements set for such a system. Thus the beacons based localization is (apart from other uses) ideal for the students in robotic teams, as they can solve nontrivial task of localization with cheap hardware.

Further advantage of beacons based localization is in the fact that number of robots that use the beacons is virtually unlimited. Therefore beacons can provide the base for the students to compete with different localization approaches while all their robots (of possibly different nature) have the same environment information. According to our experiences this aspect is strongly underestimated in competitions (frustration of the students with uneven conditions).

The aim of the paper was to describe in understandable way the method ideal for low cost robot localization that can be implemented by the students while helping them to understand the basics of Bayesian filtering. What type of filter to use in the task is not the main goal, however, when computational power restrictions are strong, the EKF simply wins the race. UKF benefits from better approximation of true non-Gaussian distribution are erased by higher computational cost leading to longer sampling rates and less stable estimates.

The future work will be focused on using the beacon / receiver combinations mounted on the robots, so the multiple robots can help to localize themselves. This should lead to reduction of number of beacons needed to install statically, however such need can not be reduced to zero, as only the relative information can be extracted from dynamic beacons measurement.

ACKNOWLEDGMENT

Published results were acquired with the support of the Academy of Sciences of the Czech Republic under the research plan AV0Z20760514 "Complex dynamical systems in thermodynamics, fluid and solid mechanics" and with the support of the Operational Programme Education for Competitiveness, project number CZ.1.07/2.3.00/09.0162 „Knowledge and Skills in Mechatronics - Innovations Transfer to Practice“. Authors would also like to thank Petr Schreiber for his help with C coding and Michal Růžička for his work on beacons hardware.

REFERENCES

[1] H. Choset, et al., “Principles of Robot Motion", MIT Press, 2005.

[2] S. Thrun, W. Burgard, D. Fox, “Probabilistic Robotics”, MIT Press, 2006.

[3] K. Arras, N.Tomatis, “Improving Robustness and Precision in Mobile Robot Localization by Using Laser Range Finding and Monocular Vision”, *Third European Workshop on Advanced Mobile Robots (EUROBOT ‘99), Zurich, Switzerland*, Sept.6-8, 1999.

[4] A. J. Davison, “Real-time simultaneous localisation and mapping with a single camera”, *9th ICCV*, vol 2, pp. 1403–1410, October 2003.

[5] K. Bekris, M. Glick, L. Kavraki, “Evaluation of Algorithms for Bearing-Only SLAM", *IEEE Intl. Conf. on Robotics and Automation, May, Orlando, FL*, p. 1937–1944, 2006.