Self-Calibration in Differential Drive Dynamics/Kinematics Model

Martins Ekmanis
Department of Systems Theory and Design
Riga Technical University
Riga, Latvia
Martins.Ekmanis@rtu.lv

***Abstract*—Localization and path planning is an essential elements of mobile robots but it relies on model availability. It is difficult to obtain robot dynamics/kinematics model due to lot of constants and variables. This paper proposes a highly adaptable and universal method for obtaining differential drive robot model by self-calibration. The obtained model is based on linear time invariant assumption and accepts sensor-less prediction as well as hybrid mode, where encoder's data are filtered against the model. Theoretical background and experimental proof of model quality is included in this paper. The algorithms are implemented in software code and experimentally verified on physical robot. The experimental results confirm models accuracy by means of model based track following tests.**

Keywords—dynamics model; self-calibration; localization; robotics

I. INTRODUCTION

Self-localization is the most important capability for a mobile robot to effectively perform its tasks. Without it, the task assigned to a robot cannot be completed, and provides the basis for development. This paper focuses on a model based approach to track the position within an indoor space.

The pose estimation is a challenging field of research as there is no magic sensor to cover all needs. Even if GPS could be used in outdoor environment, its precision, refresh intervals and variance does not fit most of the practical application requirements. Existing research focuses on artificial landmark usage like short-range radio signals [1], infrared [2] or magnetic transducers [3]. The results look promising, but infrastructure requirements not always are rational to use outside laboratory environment.

Recently significant progress is reached in computer vision [4] [5] and depth cameras with point cloud processing algorithms [6]. Unfortunately light conditions outside laboratories are variable and the process of computing is resource consuming. Since the frame rate on mobile solutions are limited by power consumption, CPU and memory resources, visual landmark identification can be used as a global localization, but it cannot be used as the only localization source. Obtained position from cameras is successfully used to reduce positioning errors while recognizable landmark is visible. Therefore most solutions focus on multiple sensor data fusion for better position quality estimation.

Data fusion in robotics usually is performed with an Extended Kalman filter [7] or a Particle filter [8] [9] [16]. In each case a good robot kinematic and dynamics model is needed to make reasonable prediction. Noisy sensor data helps to correct the prediction keeping accumulated error within limits. Match with hardcoded constants could be used as a model to calculate the next robot position after given control signals, but in real-life good constants are not available since the surrounding environment is highly dynamic and uncertain.

Significant part of mobile robots uses a differential drive system. The encoders capture wheel rotation and give feedback to motor controllers. Moreover, robot movement can be calculated using encoders. Unfortunately surface parameters, wheel wear, cargo, center of mass etc. affects distance travelled. Using average values leads to significant errors in position estimation specially angle estimate. It is better than nothing and can be utilized in modern data fusion mechanisms. However having better model would require not so frequent landmark updates and can save on other sensors, landmarks and CPU requirements.

Existing research is focusing on error correction by data fusion leaving an open question about model suitability [1] [2] [5]. This paper discusses the use of a dynamic self-calibrating model. Proposed model is based on a linear time-invariant assumption that may mislead in the beginning as motors are not linear and parameters are time dependent. However in a small time interval, the approximation reaches acceptable precision. If parameters are obtained dynamically, model follows environment changes keeping error rate as low as possible. Obtained dynamics/kinematics model can be used as well for path planning, simulation or fault detection but it is not in the scope of this paper.

Experimental tests are performed with a Roomba 560 vacuum cleaning robot, which is programmed to follow closed loop trajectory estimating its position only using the model. After loop closure, the robot stops near starting position, where mismatch distance is measured and used for model evaluation. Results show the model ability to adapt to environment and are compared with the traditional odometry model [10].

The paper is organized as follows. Section II explains linear models, their calibration, and application for different drive robot. Experimental setup, models testing method and main results are discussed in section III. Finally, Section IV gives conclusions and the directions of future work are highlighted.

II. Materials And Methods

A. Linear time-invariant model

A general technique for the design of non-linear system models does not exist and usually linear models are used instead [11]. Often non-linear models are linearized around a given operating point if possible otherwise methods, like artificial neural networks, genetic algorithms etc. can be applied. These methods can give a good solutions but result cannot be guaranteed.

A linear system produces its output as a linear combination of its current and previous inputs and previous outputs. It is called as time-invariant if the system parameters do not change over a given time interval. This approach is successfully applied to different problems to automatically seek for the best models of unknown systems [12].

The linear time-invariant (LTI) system can be represented by (1), where output signal y and input signal x. The equation includes scalars b_j for $j = 0 \dots q$ and a_k for $k = 1 \dots p$ where the maximum of p and q is the order of the system. [13]

$$y(n) = \sum_{j=0}^{q} b_j x(n-j) - \sum_{k=1}^{p} a_k y(n-k) \quad (1)$$

By re-arranging (1) and transforming into the s-domain, systems transfer function H(s) is obtained (2), where Y(s) is the system response to control signal X(s).

$$H(s) = Y(s)/X(s) \quad (2)$$

The general linear system transfer function can have three different types depending on the form of transfer function H(s). [13]

When the numerator of the transfer function is constant the model is autoregressive (AR) type. AR model attempts to predict an output of a system based on the previous outputs [13]. Because the model excludes control signals it is not applicable in applications addressed by this paper.

The moving–average model assumes that the denominator of transfer function is a constant. This is nonrecursive system also called finite-impulse response (FIR). This type of systems has no stability problems as the impulse response is finite length. This model is appropriate for inertial systems (smoothing) [14].

Third and most general case is the mixed pole/zero model called autoregressive moving-average (ARMA) model. It is used to model systems with infinite-impulse response (IIR) [14]. ARMA is used to model systems with feedback like PID controllers.

In the context of this research second and third model are applied. In the next subsections each of them is described in details.

B. Moving-average model

As previously mentioned the transfer function (2) is system output and input ratio. If we could pass Dirac delta function $\delta(t)$ (3) as an input to system, its response will be equal to the transfer function (4). This follows from Dirac delta function Laplace transformation (5) with is equal to 1.

$$\delta(t) = \begin{cases} +\infty, & t = 0 \\ 0, & t \neq 0 \end{cases} \quad (3)$$

$$H(s) = Y(s)/1 \quad (4)$$

$$\int_{-\infty}^{+\infty} \delta(t)dt = 1 \rightarrow L[\delta(t)] = 1 \quad (5)$$

Unfortunately this is not useful case, because Dirac delta function having infinitely high and infinitely short impulse cannot be executed in physical system. Instead step function $u(t)$ (6) can be easily reproduced on physical system like robot. The step functions derivate (7) is Dirac delta function.

$$u(t) = \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases} \quad (6)$$

$$\frac{d}{dt}u(t) = \delta(t) \quad (7)$$

The transition process should be captured and its derivate version of sensor readings will represent transfer function.

In practice robot is not just a single motor. Even differential drive system (Fig.1) have separate control signal for each motor and at least two outputs namely linear speed $\vartheta(t)$ and angular speed $\omega(t)$.

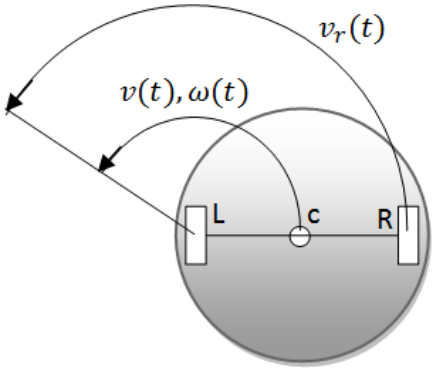

Fig. 1. Differential drive system.

Relation (2) can be converted to matrix equation (8), where predicted output vector is transfer matrix multiplied by control signal $[\mathrm{L(s)}, \mathrm{R(s)}]^{\mathrm{T}}$. The transfer matrix includes each output response for each input attribute, where $\mathrm{H_{Lv}(s)}$ is left motor response to linear speed, $\mathrm{H_{L\omega}(s)}$ is left motor response to angular speed, $\mathrm{H_{Rv}(s)}$ is right motor response to linear speed and $\mathrm{H_{R\omega}(s)}$ is right motor response to angular speed.

$$\begin{bmatrix} \mathrm{v(s)} \\ \mathrm{\omega(s)} \end{bmatrix} = \begin{bmatrix} \mathrm{H_{Lv}(s)} & \mathrm{H_{Rv}(s)} \\ \mathrm{H_{L\omega}(s)} & \mathrm{H_{R\omega}(s)} \end{bmatrix} \cdot \begin{bmatrix} \mathrm{L(s)} \\ \mathrm{R(s)} \end{bmatrix} \quad (8)$$

From here time domain model (9) can be obtained providing linear $v(t)$ and angular $\omega(t)$ speeds prediction. The equation (9) contains convolution operators (10) as a result of reverse Laplace transformation of function multiplication.

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} \frac{d}{dt} u_{Lv}(t) * L(t) + \frac{d}{dt} u_{Rv}(t) * R(t) \\ \frac{d}{dt} u_{L\omega}(t) * L(t) + \frac{d}{dt} u_{R\omega}(t) * R(t) \end{bmatrix} \quad (9)$$

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau) h(t-\tau) d\tau \quad (10)$$

Cartesian coordinate and angle represent the robot position $X = [x, y, \theta]^T$. Therefore repeating recursively differential drive update equation (11) with predicted speeds the following robot positions can be calculated.

$$X_{t+1} = \begin{cases} \begin{bmatrix} x_t + v \cdot \Delta t \cos(\theta_t) \\ y_t + v \cdot \Delta t \sin(\theta_t) \\ \theta_t \end{bmatrix} & , \omega = 0 \\ \begin{bmatrix} x_t - \frac{v}{\omega} \sin(\theta_t) + \frac{v}{\omega} \sin(\theta_t + \omega \Delta t) \\ y_t + \frac{v}{\omega} \cos(\theta_t) - \frac{v}{\omega} \cos(\theta_t + \omega \Delta t) \\ \theta_t + \omega \Delta t \end{bmatrix} & , \omega \neq 0 \end{cases} \quad (11)$$

In other words the model includes robot kinematics and dynamics obtained through calibration maneuver. If to provide model with empirical step response functions $u_{Lv}(t)$, $u_{Rv}(t)$, $u_{L\omega}(t)$, $u_{R\omega}(t)$ and planed control signals $R(t)$, $L(t)$ then the output is predicted robot position in time.

See model implementation details and experimental results in results section.

C. Autoregressive moving-average model

ARAMA [12] has been used in second model. This model is more general approach and can be used in recursive mode. Therefore not just control inputs but also previous encoders reading have influence on next prediction.

Determination of model order is important step for successful prediction. If the model has higher order then necessary than the impulse response estimates calculated from the resulting ARMA parameters will be overly sensitive to corruption by noise. If the model has lower order then necessary than the model will lack the degrees of freedom necessary to represent the dynamics of the corresponding physical system.

$$A = \begin{bmatrix} x_n^1 & x_{n-1}^1 & \dots & x_{n-q}^1 & y_{n-1}^1 & y_{n-2}^1 & \dots & y_{n-p}^1 \\ x_n^2 & x_{n-1}^2 & \dots & x_{n-q}^2 & y_{n-1}^2 & y_{n-2}^2 & \dots & y_{n-p}^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_n^i & x_{n-1}^i & \dots & x_{n-q}^i & y_{n-1}^i & y_{n-2}^i & \dots & y_{n-p}^i \end{bmatrix}$$

$$x = [b_0 \quad b_1 \quad \dots \quad b_q \quad a_1 \quad a_2 \quad \dots \quad a_p]^T$$

$$\mathrm{y} = [y_n^1 \quad y_n^2 \quad \dots \quad y_n^i]^T$$

$$Ax = y \quad (12)$$

Once the ARMA model order is known the scalars b_j and a_k can be calculated from linear system of equations (12) where x_n^i is the system input in i run, n timeslot and y_n^i is measured system response in i run and n timeslot. The system ratios can be calculated from linearly independent equation system having the same number of equations as unknowns.

D. Models evaluation method

The models were tested by physical robots in laboratory environment. During the test robot have predefined route (Fig.2) to follow where numbers show checkpoint sequence. Route is a closed loop therefore robot should return to the starting position after completing its job. Distance between start and final positions was measured and processed by statistical methods to determine variance and mean error. As the loop is closed initial angular displacement caused by setting robot in starting position does not affect results.

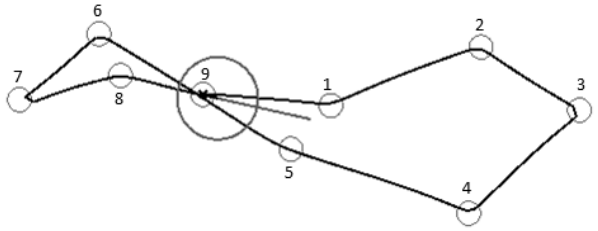

Fig. 2. Predefined route for model evaluation, where numbers mean sequence of checkpoints.

For each model the robot performs 30 independent attempts. To compare models with odometry third test is performed with classic odometry model [10] where position is purely calculated from encoders (11).

III. RESULTS AND DISCUSSION

A vacuum cleaning robot Roomba 560 was modified by removing the cleaning unit and replacing with a mini ITX board (Fig.3). Algorithms are coded using C++.

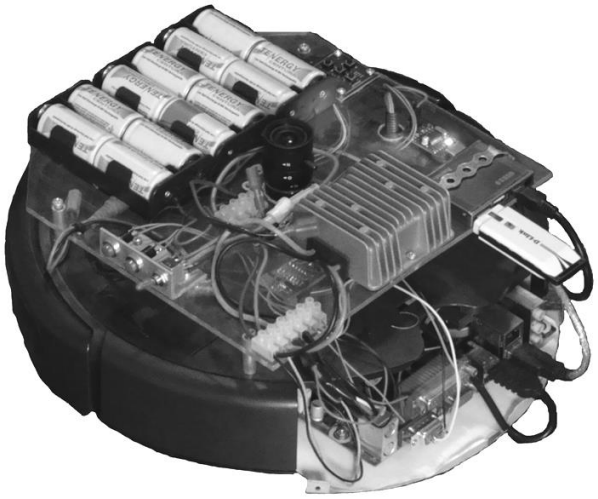

Fig. 3. Modified Roomba 560 robot for the evaluation of different models.

All parameters are obtained during calibration operations except distance between wheels. All other variables are derived from this constant during the calibration, therefore adapting to it as a distance unit.

A. *Encoder calibration*

Built-in light bumper sensors are used to calibrate the encoders. The robot is given instruction to rotate while light bumper signal is captured and processed. When robot makes more than one full rotation the signal becomes periodic (Fig.4). If robot is near single obstacle like wall or table, simple peak detector can be used to detect period. In more general case when surrounding environment is unstructured the captured signal should be processed with auto correlation function to detect repeating patterns. The x axis on graphic (Fig.4) is in encoder ticks, therefore relation between signal level and rotation period is time and speed independent.

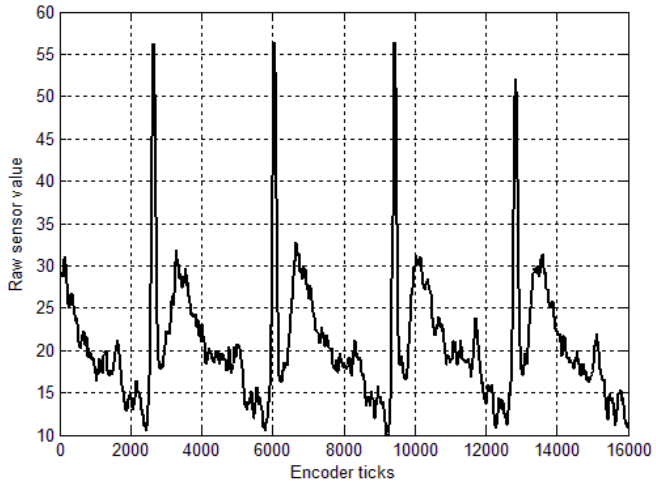

Fig. 4. Bumper sensor signal during rotation.

Each wheel travel $d = 2\pi L$ meters during one full rotation, where L=0.33m is given distance between wheels. Therefore scale factor *sc* can be calculated from encoder delta Δ_{enc} found during single rotation period (13).

$$sc = 2\pi L/\Delta_{enc} \tag{13}$$

The sc value depends on surface where the robot is operating. For example, on carpet robot move slower with same wheel rotation speed then on solid surface because of wheel slipping.

B. *Transfer calibration*

As it is mentioned before the transfer function of robot can be obtained by turning on drives (6). The transfer matrix includes each output response for each input attribute. In differential drive case calibration maneuver should include sequence of four actions.

1. Left motor should be turned on while capturing linear $u_{Lv}(t)$ and angular $u_{L\omega}(t)$ speeds.
2. Left motor turns off and waits while robot fully stops.
3. Right motor turns on and right motor response $u_{Rv}(t)$ and $u_{R\omega}(t)$ is captured.
4. Right motor turns off and waits while robot fully stops.

If the built in PID control is switched off the falling of speed is significantly slower than acceleration due to the inertia. Since the transfer function capture only acceleration it is assumed then both parts are symmetric. The presence of closed loop PID control makes the system response symmetrical therefore supporting the use of linear approximation.

Figure 5 depicts wheel speed response to step function. One can notice that the falling edge is different than regular damping. Robot controller performs PID control; therefore growing and falling edges are almost symmetric letting us to approximate it as a linear system.

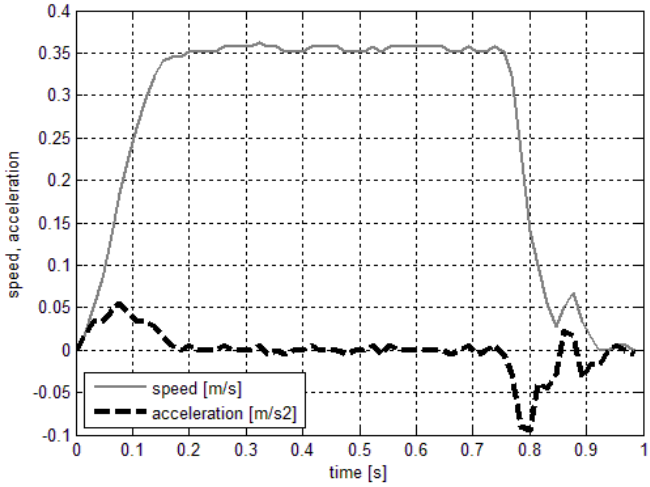

Fig. 5. Wheel step response with enabled PID control.

The transfer functions (Fig.6) are obtained during calibration where all non-zero values are before 15th steps (step corresponds to sensor reading period that is 18ms for Roomba). It means that the response is finite and fit as a finite transfer function for moving-average model.

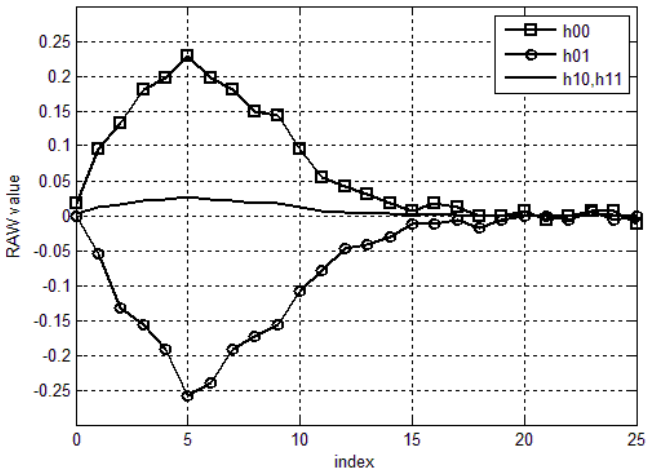

Fig. 6. Empirical transfer functions obtained during the calibration.

C. ARMA calibration

ARMA model order is determined experimentally as a balance between model accuracy and CPU consumption. The specific experiment with Roomba robot gave an order 9 as the best value. Its calculation took almost 2 seconds and the accuracy did not increase with order 10 and further. Therefore equation (12) includes ten $b_{0\ldots9}$ and nine $a_{1\ldots9}$ scalars. In general, model order depends on several factors including sensor capture interval, mechanical inertia, feedback controllers etc. Numbers given here reflect specific experimental robot configuration and should be considered as an example.

As it mentioned before the linear system ratios can be calculated from linearly independent equation system having the same number of equations as unknowns. Equations should be independent to solve it but acquired data from sensors does not insure that. For initial calibration sequence normally distributed set of control signals are used. To accumulate sensor readings the system is over defined, therefore having more than needed equations in system. Such system can be solved by Singular Value Decomposition method (SVD).

SVD is a method for transforming correlated variables into a set of uncorrelated ones that better expose the various relationships among the original data items. SVD method itself is not a topic of this research, therefore will not discussed here in details. OpenCV function 'cv::solve' is used to calculate over defined equation systems. [15]

Fragment of such randomly distributed calibration sequence is shown in (Fig.7). Response is delayed because of software, hardware and communication that can be observed in response graphic as well as in calculated ratios.

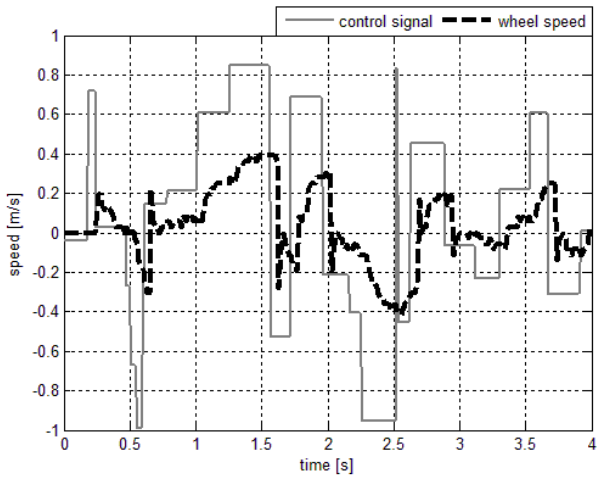

Fig. 7. Fragment of random LTI calibration sequence.

After filling the matrix (12) and resolving the equation a_k and b_j ratios (Table I) are ready to be used for prediction. Obtained b_0 value shows that prediction mostly relays on the last value because of system inertia. The control influence a is delayed by 3-6 steps (50-100ms) that indicates to the presence of delay and jitter in control and communication circuits. Therefore obtained scalars confirm delays shown in response graphic (Fig.7).

TABLE I. CALCULATED LTI RATIOS

Index i	Scalars a_i	Scalars b_i
0		0.5395
1	-0.0009	-0.1391
2	0.0013	0.1447
3	-0.0004	0.0971
4	0.0178	0.0403
5	0.0769	0.0225
6	0.0725	-0.0195
7	-0.0323	-0.0148
8	0.014	-0.0361
9	0.0012	0.5395

Function (1) can be used in full recursion mode therefore providing sensor less prediction in the same manner as transfer function approach. The main difference is in calibration process and amount of CPU required. The transfer function requires start and stop sequence of drives to calibrate where ARAM model can use any stochastic sequence of control signals. Calibration of transfer function does not consume any additional CPU resources but ARAM requires solving huge equation system having hundreds of equations.

ARAM model can be used with sensor data as well. Therefore y_{n-1} to y_{n-q} are historical sensor readings and y_n is model output. This approach works as a filter fitting sensor data to the model; therefore calculated robot position is more noise resistant and provides better position tracking accuracy compared with pure odometry (Table 1).

TABLE II. POSITION TRACKING ACCURACY PER MODEL

Model	Average offset [m]	Standard deviation [m]
Odometry	0.068	0.027
Moving-average	0.332	0.095
ARMA	0.258	0.021

Robot generates motion plan and executes it without any feedback when sensor less model is used, thereby running in open loop mode. Standard deviation in this case (Table II) depends only on environment because signal provided to drives every run is the same. Variance in this case is bigger compared to pure odometry, but still maintains a reasonable range.

IV. CONCLUSIONS

Robot dynamics model usually is difficult to build as there is significant uncertainty about environment. This paper proposes use of simple linear time invariant model as a differential drive robot dynamic model. The main focus in this approach is on auto calibration. If the calibration can be performed seamless during robot operation, even the simple model provides good accuracy.

Experimental setup was made to test auto calibration and measure model accuracy in physical environment. Experimental results (Table II) show low variance even in sensor-less mode, where precision is equivalent to the traditional sensor-based approach. Purposed model is

applicable as a back-up positioning method in domains, where high reliability is required or cost effectiveness is crucial. Model includes kinematics and dynamics constrains, therefore can be used for path planning, simulation and fault detection.

Positioning precision can be improved by applying the ARMA model. The model includes prior knowledge of robot dynamics by ratios calculated during calibration. Such model can be used as a filter to discard faulty sensor readings. The faulty sensor data do not fit model therefore can be easily discovered and discarded. Moreover, comparing model output with sensor signals will notify when environment has significantly changed and recalibration is required.

Our future research will focus on static deviation from the planned trajectory. As it is shown in Table II, the average offset after loop closure for linear models are higher than sensor based approach can provide. This side effect is caused by nonlinear effects currently absent in our model.

ACKNOWLEDGMENT

The research described in this paper is supported by funding of the ERDF Project “Development of technology for multi-agent robotic intelligent system”, project implementation contract No. 2010/0258/2DP/ 2.1.1.1.0/10/ APIA/VIAA/005.

REFERENCES

[1] Y. Jin, W.S. Soh and W.C. Wong, "Indoor Localization with Channel Impulse Response Based Fingerprint and Nonparametric Regression," IEEE Transactions on Wireless Communications, pp. 1120-1127, Mar 2010.

[2] J. Krejsa and S. Vechet, "Infrared Beacons based Localization of Mobile Robot," Electronics and Electrical Engineering, vol. 1, no. 117, pp. 17-22, 2012.

[3] Won Suk You, Byung June Choi, Bumsoo Kim, Hyungpil Moon, Ja Choon Koo, Wankyun Chung and Hyouk-Ryeol Choi, "Global localization for a small mobile robot using magnetic patterns," in 2010 IEEE International Conference on Robotics and Automation (ICRA), Anchorage, AK, 2010.

[4] Zhiwei Zhu, T. Oskiper, S. Samarasekera, R. Kumar and H. Sawhney, "Real-Time Global Localization with A Pre-Built Visual Landmark Database," in 26th IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, 2008.

[5] T. Goedemé, M. Nuttin, T. Tuytelaars and L. V. Gool, "Omnidirectional Vision Based Topological Navigation," International Journal of Computer Vision, vol. 74, no. 3, pp. 219-236, 2007.

[6] J. Cunha, E. Pedrosa, C. Cruz, A. J. Neves and N. Lau, "Using a Depth Camera for Indoor Robot Localization and Navigation," in Robotics Science and Systems 2011 Workshop on Advanced Reasoning with Depth Cameras, Los Angeles, California, USA, 2011.

[7] S. Russel and P. Norvig, “Kalman Filters,” in Artificial Intelligence A Modern Approach Third Edition, New Jersey, Pearson Education, Inc., 2009, pp. 584-590.

[8] S. Thrun, "Particle Filters in Robotics," in Proceedings of the Eighteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-02), San Francisco, CA, 2002.

[9] A. Doucet, N. de Freitas and N. Gordon, "Particle Filters far Mobile Robot Localization," in Sequential Monte Carlo Methods in Practice, UK, Springer, 2001, pp. 401-428.

[10] C. Wang, "Location estimation and uncertainty analysis for mobile robots," in 1988 IEEE International Conference on Robotics and Automation, Philadelphia, PA, 1988.

[11] K. Najim, Control of Continuous Linear Systems, London, UK: Wiley-ISTE, 2006.

[12] M. H. Perrott and R. J. Cohen, "An Efficient Approach to ARMA Modeling of Biological Systems with Multiple Inputs and Delays," IEEE Transactions On Biomedical Engineering, vol. 43, no. 1, pp. 1-14, 1996.

[13] C. L. Phillips and J. M. Parr, Signals, Systems, and Transforms. Fourth Edition., NJ, USA: Peason Education, Inc., 2008.

[14] Y. M. Kadah, "Digital Signal Processing: Discrete-Time Signals and Systems," [Online]. Available: http://ymk.k-space.org/ DSP_chapter8.pdf. [Accessed 04 Jan 2013].

[15] G. Bradski and A. Kaebler, "Matrix and Image Operators," in Learning OpenCV, Sebastopol, CA, O'Reilly Media, Inc., 2008, pp. 47-77.

[16] B. Sugandi, H. Kim, J.K. Tan, S. Ishikawa, “Multiple objects tracking method based on particle filter”, Recent Researches in Circuits, Systems, Mechanics and Transportation Systems, WSEAS Press, pp. 64-69. 2011.