4

Model based Kalman Filter Mobile Robot Self-Localization

Edouard Ivanjko, Andreja Kitanov and Ivan Petrović
University of Zagreb, Faculty of Electrical Engineering and Computing,
Department of Control and Computer Engineering
Croatia

1. Introduction

Mobile robot self-localization is a mandatory task for accomplishing its full autonomy during navigation. Various solutions in robotics community have been developed to solve the self-localization problem. Developed solutions can be categorized into two groups: relative localization (dead-reckoning) and absolute localization. Although very simple and fast, dead-reckoning algorithms tend to accumulate errors in the system as they utilize only information from proprioceptive sensors as odometer readings (e.g. incremental encoders on the robot wheels). Absolute localization methods are based on exteroceptive sensors information. These methods yield a stable locating error but are more complex and costly in terms of computation time. A very popular solution for achieving online localization consists of combining both relative and absolute methods. Relative localization is used with a high sampling rate in order to maintain the robot pose up-to-date, whereas absolute localization is applied periodically with a lower sampling rate to correct relative positioning misalignments Borenstein et al. (1996a).
As regards absolute localization within indoor environments, map-based approaches are common choices. In large majority of cases, it is assumed that a map (model) of the workspace has been established. The environment model can either be pre-stored as architects drawing CAD model or built online simultaneously with localization using sensor data fusion or structure from motion technics. The classical approach to model-based localization consists of matching the local representation of the environment built from sensor information with the global model map and will be used in this chapter also.
So, this chapter presents two approaches to mobile robot self-localization regarding used perceptive sensor combined with an environment model. First approach uses sonar range sensor and second approach uses monocular camera. Environment model in the first approach is an occupancy grid map, and second approach uses a 3D rendered model.
Sonar range sensor is often used in mobile robotics for localization or mapping tasks Lee (1996); Wijk (2001) especially after occupancy grid maps were introduced Moravec & Elfes (1985). Mostly feature maps are used because of their more accurate environment presentation. In such a case when sonar range measurement prediction is done additional steps have to be made. First, appropriate environment feature has to be detected and then corresponding uncertainty that detected feature is correct has to be computed. This uncertainty adds also additional computation step into mostly used Kalman filter framework for non-linear

systems making pose estimation more computationally complex. In this chapter an occupancy grid map is used as the environment model in combination with Extended Kalman filter (EKF) and Unscented Kalman filter (UKF). In this way computation of detected feature uncertainty is avoided and doesn't have to be included in the Kalman filter localization framework with sonar sensor.

General Kalman filter consist of a prediction (time update) and correction (measurement update) step Welch & Bishop (2000). Prediction step makes an update of the estimated state. In this case estimated state is mobile robot pose (combined mobile robot position and orientation). So a kinematic mobile robot model is used for state prediction. In field of mobile robotics such a kinematic model is also known as odometry Borenstein et al. (1996b). It uses measured drive wheel speeds to compute mobile robot pose displacement from a known start pose. It is accurate for short distances because of its aproximate quadratical error growth rate. Such a growth rate arises from the fact that pose error in current pose estimation time step is added to all previous made errors. Fortunately some odometry errors can be taken into account slowing down the error growth rate Borenstein et al. (1996b). This is done by odometry calibration and is also used in work described in this chapter.

Although navigation using vision has been addressed by many researchers, vision is not commonly used on its own but usually in conjunction with other exteroceptive sensors, where multi-sensor fusion techniques are applied, see e.g. Arras et al. (2001) and Li et al. (2002). However, cameras have many advantages as range sensors comparing to sonars and laser rangefinders as they are passive sensors, provide much more data about environment, can distinguish between obstacles based on color etc. A great deal of work has been done on stereo vision, see e.g. Guilherme & Avinash (2002), but for reasons of size and cost monocular vision based navigation has been addressed by a number of researchers, e.g. Aider et al. (2005); Jeon & Kim (1999); Kosaka & Kak (1992); Neira et al. (1997).

When using monocular vision, the localization process is composed of the five following stages Guilherme & Avinash (2002); Kosaka & Kak (1992): 1) image acquisition from current robot pose; 2) image segmentation and feature extraction; 3) model rendering; 4) matching 2D-image and 3D-model features and 5) camera pose computation. It is observed that each stage may be time-consuming due to large amount of data involved. The strategy ultimately adopted for each phase must then be very well-assessed for real-time use. For example, an efficient real-time solution to the feature matching problem is presented in Aider et al. (2005), where interpretation tree search techniques were applied. For mobile robot working environment modelling and rendering professional freeware computer graphics tool Blender *www.blender3d.org* (1995) was used, which is an open source software for 3D modelling, animation, rendering, post-production, interactive creation and playback. It is available for all major operating systems and under the GNU General Public License. The main advantage of that choice is getting powerful 3D modelling tool while being able to optimize the code for user application and use some external well proven computer graphics solutions that Blender incorporates: OpenGL and Python. It also gives external renderer Yafray. Real-time image segmentation for complex and noisy images is achieved by applying Random Window Randomized Hough Transform (RWRHT) introduced in Kälviäinen et al. (1994) which is here used for the first time for robot self-localization to the best of our knowledge. We also implemented and improved robot (camera) pose estimation algorithm developed in Kosaka & Kak (1992).

This chapter is organized as follows. Second section describes used sensors including their mathematical models. Following section describes applied sensor calibration procedures.

Fourth section describes sonar based localization, followed with the fifth section describing monocular vision based localization. Sixth section gives experimental results obtained with a differential drive mobile robot including experiment condition description and comment of localization results. Chapter ends with conclusion.

2. Mobile robot and sensors models

As mentioned in the introduction, used sensors are encoders, sonars and a mono camera. Whence encoders measure mobile robot velocity which is used for odometric pose estimation this section will describe mobile robot kinematic model used for odometric pose estimation. Besides odometry, sonar and camera models will be described.

2.1 Mobile robot kinematics model

As mentioned above a differential drive mobile robot for indoor environments is used in experiments. Its kinematics configuration with denoted relevant variables is presented in Fig. 1. Such a mobile robot configuration has three wheels. Two front wheels are drive wheels with encoder mounted on them and the third wheel is a castor wheel needed for mobile robot stability. Drive wheels can be controlled independently. Kinematic model of differential drive mobile robot is given by the following relations:

$$x(k+1) = x(k) + D(k) \cdot \cos\left(\Theta(k) + \frac{\Delta\Theta(k)}{2}\right), \tag{1}$$

$$y(k+1) = y(k) + D(k) \cdot \sin\left(\Theta(k) + \frac{\Delta\Theta(k)}{2}\right), \tag{2}$$

$$\Theta(k+1) = \Theta(k) + \Delta\Theta(k), \tag{3}$$

$$D(k) = v_t(k) \cdot T, \tag{4}$$

$$\Delta\Theta(k) = \omega(k) \cdot T, \tag{5}$$

$$v_t(k) = \frac{\omega_L(k)R + \omega_R(k)R}{2}, \tag{6}$$

$$\omega(k) = \frac{\omega_R(k)R - \omega_L(k)R}{b}, \tag{7}$$

where are: $x(k)$ and $y(k)$ coordinates of the center of axle [mm]; $D(k)$ travelled distance between time step k and $k+1$ [mm]; $v_t(k)$ mobile robot translation speed [mm/s]; $\omega(k)$ mobile robot rotational speed [°/s]; T sampling time [s]; $\Theta(k)$ angle between the vehicle and x-axis [°]; $\Delta\Theta(k)$ rotation angle between time step k and $k+1$ [°]; $\omega_L(k)$ and $\omega_R(k)$ angular velocities of the left and right wheel, respectively [rad/s]; R radius of the two drive wheels [mm], and b mobile robot axle length [mm]. This general model assumes that both drive wheels have equal radius. Sampling time T was 0.1 $[s]$.

In case of real world mobile robot operations, drive wheel speed measurements are under measurement error influence and some mobile robot parameters values aren't exactly known. Measurement error is mostly a random error with zero mean and can't be compensated. Unknowing exact mobile robot parameters present systematic error and can be compensated

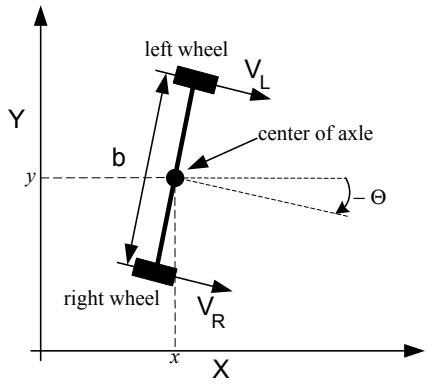


Fig. 1. Differential drive mobile robot kinematics

by means of calibration. Crucial unknown parameters are drive wheel radii and axle length. Wheel radii effect the measurement of the drive wheel circumference speed and axle length measurement affects the mobile robot rotational speed. Equations (6) and (7) can so be rewritten to present mentioned error influence:

$$v_t(k) = \frac{(\omega_L(k)R + \varepsilon_L) + (\omega_R(k)R + \varepsilon_R)}{2}, \tag{8}$$

$$\omega(k) = \frac{(\omega_R(k)R + \varepsilon_R) - (\omega_L(k)R + \varepsilon_L)}{b + \varepsilon_b}, \tag{9}$$

where ε_L, ε_R and ε_b are the error influences on the drive wheel circumference speed measurements and axle length, respectively. It can be noticed here that axle length is also under influence of systematic and random errors. Systematic error obviously comes from unknowing the exact axle length. In this case random error is caused by the effective axle length, which depends on the wheel and surface contact points disposition. Contact points disposition may wary during mobile robot motion due to uneven surfaces and influence of non-symmetric mass disposition on the mobile robot during rotation.

2.2 Sonar model

An interpretation of measured sonar range is given in Fig. 2. It can be seen that in 2D a sonar range measurement can be presented as a part of a circle arc. Size of circle part is defined by the angle of the main sonar lobe and is typical for of the shelf sonar's between 20 and 30 degrees. Therefore, the detected obstacle is somewhere on the arc defined by measured range and main sonar's lobe angle. Background of Fig. 2 shows a grid which is used for creation of occupancy grid map. When a sonar range measurement is interpreted in an occupancy grid framework usually a one to two cells wide area around the measured range is defined as the occupied space. Space between the sonar sensor and measured range is empty space. The sonar is a time of flight sensor, which means it sends a wave (acoustic in this case) and measures the time needed for returning the wave reflected from an obstacle back to the sonar. Generated acoustic wave has its most intensity along its axis, as denoted

in Fig. 2, therefore resulting a more accurate distance measurement of obstacles that are inline and perpendicular to the sonar axis. Whence wave intensity decreases with traversed distance, absolute range measurement accuracy also decreases with wave-traversed distance. This is related with requirement of a big range measurement which is a longer open time window to accept the reflected wave and therefore enable more specular reflections and outliers. Specular reflections and outliers present in this case false readings, which decrease the quality of the obtained map. To take this sonar range measurement features into account a stronger emphasis is given to the range measurements closer to the sonar sensor and environment parts closer to the main sonar axis. Mathematically this can be expressed with following equations [3]:

$$\alpha(\Theta) = \begin{cases} 1 - \left(\frac{\Theta}{\Theta_0}\right)^2 & 0 \leq \Theta \leq \Theta_0 \\ 0 & |\Theta| > \Theta_0 \end{cases}, \tag{10}$$

$$\Delta(\rho) = 1 - \frac{1 + \tanh(2(\rho - \rho_v))}{2}, \tag{11}$$

where $\alpha(\Theta)$ presents angular modulation function i.e., main lobe pattern of the used sonar sensor; Θ angle between sonar axis and currently updated cell [°]; Θ_0 is one half of the sonar main lobe angle [°]; ρ distance from the sonar sensor and currently updated cell $[m]$; $\Delta(\rho)$ presents radial modulation function and ρ_v presents visibility radius where less emphasis is given to the sonar range measurement further away from visibility radius $[m]$. Parameter Θ_0 value depends from the used sonar sensor and for our Polaroid 6500 sonar sensor it is $12.5\,[^\circ]$. Parameter ρ_v decreases influences of outlier readings and recommended value for an indoor environment is $1.2\,[m]$. Sonar range measurement uncertainty is modeled with angular and radial modulation functions. Most accurate range measurements are so within main sonar axis which is used later in sonar range measurement prediction.

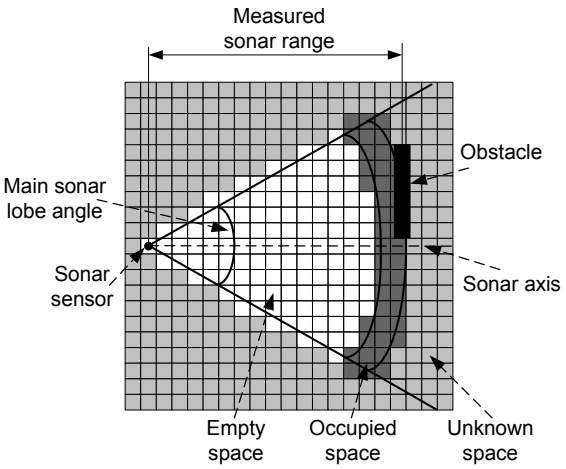


Fig. 2. Interpretation of a sonar range measurement

2.3 Camera model

Generally, a camera has 6 degrees of freedom in three-dimensional space: translations in directions of axes x, y and z, which can be described with translation matrix $T(x, y, z)$, and rotations around them with angles α, β and γ, which can be described with rotation matrices $R_x(\alpha), R_y(\beta)$ and $R_z(\gamma)$. Camera motion in the world coordinate system can be described as the composition of translation and rotation matrices:

$$C = T(x, y, z)\ R_z(\gamma)\ R_y(\beta)\ R_x(\alpha), \tag{12}$$

where

$$\mathbf{R_x}(\mathbf{a}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha & 0 \\ 0 & sin\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{R_y}(\mathbf{b}) = \begin{bmatrix} cos\beta & 0 & sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\beta & 0 & cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{R_z}(\mathbf{g}) = \begin{bmatrix} cos\gamma & -sin\gamma & 0 & 0 \\ sin\gamma & cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{T}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Inverse transformation C^{-1} is equal to extrinsic parameters matrix that is

$$C^{-1}(\alpha, \beta, \gamma, x, y, z) = R_x(-\alpha)\ R_y(-\beta)\ R_z(-\gamma)T(-x, -y, -z). \tag{13}$$

Perspective projection matrix then equals to $P = S\ C^{-1}$ where S is intrinsic parameters matrix determined by off-line camera calibration procedure described in Tsai (1987). The camera is approximated with full perspective pinhole model neglecting image distortion:

$$\left[(x, y)^\top = \left(\frac{\alpha_x X_c}{Z_c} + x_0, \frac{\alpha_y Y_c}{Z_c} + y_0 \right)^\top \right], \tag{14}$$

where $\alpha_x = f/s_x$ and $\alpha_y = f/s_y$, s_x and s_y are pixel height and width, respectively, f is camera focal length, (X_c, Y_c, Z_c) is a point in space expressed in the camera coordinate system and $(x_0, y_0)^\top$ are the coordinates of the principal (optical) point in the retinal coordinate system. The matrix notation of (14) is given with:

$$\begin{bmatrix} WX \\ WY \\ W \end{bmatrix} = \underbrace{\begin{bmatrix} \alpha_x & 0 & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{S} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}. \tag{15}$$

In our implementation, the mobile robot moves in a plane and camera is fixed to it at the height h, which leaves the camera only 3 degrees of freedom. Therefore, the camera pose is equal to the robot pose $\mathbf{p}$. Having in mind particular camera definition in Blender, the following transformation of the camera coordinate system is necessary $C^{-1}(-\pi/2, 0, \pi + \varphi, p_x, p_y, h)$ in order to achieve the alignment of its optical axes with z, and its x and y axes with the retinal coordinate system. Inverse transformation C^{-1} defines a new homogenous transformation of 3D points from the world coordinate system to the camera coordinate system:

$$C^{-1} = \begin{bmatrix} -cos\varphi & -sin\varphi & 0 & cos\varphi\ p_x + sin\varphi\ p_y \\ 0 & 0 & -1 & h \\ sin\varphi & -cos\varphi & 0 & -sin\varphi\ p_x + cos\varphi\ p_y \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{16}$$

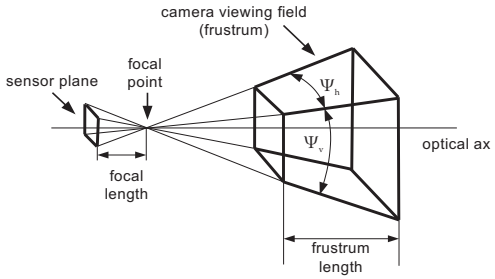


Fig. 3. Visible frustrum geometry for pinhole camera model

Apart from the pinhole model, the full model of the camera should also include information on the camera field of view (frustrum), which is shown in Fig. 3. The frustrum depends on the camera lens and plane size. Nearer and further frustrum planes correspond to camera lens depth field, which is a function of camera space resolution. Frustrum width is defined with angles Ψ_h and Ψ_v, which are the functions of camera plane size.

3. Sensors calibration

Sensor models given in the previous section describe mathematically working principles of sensors used in this article. Models include also influence of real world errors on the sensors measurements. Such influences include system and nonsystem errors. System errors are constant during mobile robot usage so they can be compensated by calibration. Calibration can significantly reduce system error in case of odometry pose estimation. Sonar sensor isn't so influenced by error when an occupancy grid map is used so its calibration is not necessary. This section describes used methods and experiments for odometry and mono-camera calibration. Obtained calibration parameters values are also given.

3.1 Odometry calibration

Using above described error influences, given mobile robot kinematic model can now be augmented so that it can include systematic error influence and correct it. Mostly used augmented mobile robot kinematics model is a three parameters expanded model Borenstein

et al. (1996b) where each variable in the kinematic model prone to error influence gets an appropriate calibration parameter. In this case each drive wheel angular speed gets a calibration parameter and third one is attached to the axle length. Using this augmentation kinematics model given with equations (8) and (9) can now be rewritten as:

$$v_t(k) = \frac{(k_1\omega_L(k)R + \varepsilon_{Lr}) + (k_2\omega_R(k)R + \varepsilon_{Rr})}{2}, \tag{17}$$

$$\omega(k) = \frac{(k_2\omega_R(k)R + \varepsilon_{Rr}) - (k_1\omega_L(k)R + \varepsilon_{Lr})}{k_3 b + \varepsilon_{br}}, \tag{18}$$

where ε_{Lr}, ε_{Rr}, and ε_{br} are the respective random errors, k_1 and k_2 calibration parameters that compensate the unacquaintance of the exact drive wheel radius, and k_3 unacquaintance of the exact axle length.
As mentioned above, process of odometry calibration is related to identification of a parameter set that can estimate mobile robot pose in real time with a minimal pose error growth rate. One approach that can be done is an optimization procedure with a criterion that minimizes pose error Ivanjko et al. (2007). In such a procedure firstly mobile robot motion data have to be collected in experiments that distinct the influences of the two mentioned systematic errors. Then an optimization procedure with a criterion that minimizes end pose error can be done resulting with calibration parameters values. Motion data that have to be collected during calibration experiments are mobile robot drive wheel speeds and their sampling times. Crucial for all mentioned methods is measurement of the exact mobile robot start and end pose which is in our case done by a global vision system described in details in Brezak et al. (2008).

3.1.1 Calibration experiments

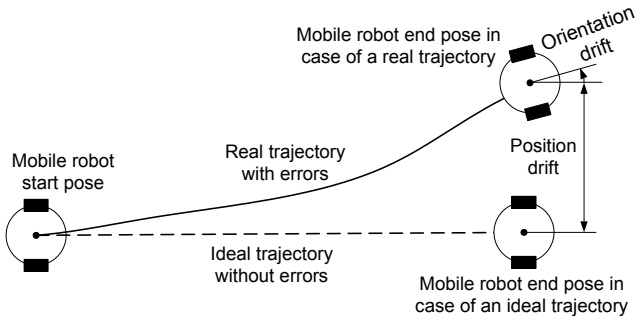


Fig. 4. Straight line experiment

Experiments for optimization of data sets collection must have a trajectory that can gather needed information about both, translational (type B) and rotational (type A) systematic errors. During the experiments drive wheel speeds and sampling time have to be collected, start and end exact mobile robot pose has to be measured. For example, a popular calibration and benchmark trajectory, called UMBmark test Borenstein & Feng (1996), uses a 5 $[m]$ square trajectory performed in both, clockwise and counterclockwise directions. It's good for data collection because it consist of straight parts and turn in place parts but requires a big room.

In Ivanjko et al. (2003) we proposed a set of two trajectories which require significantly less space. First trajectory is a straight line trajectory (Fig. 4), and the second one is a turn in place trajectory (Fig. 5), that has to be done in both directions. Length of the straight line trajectory is $5\,[m]$ like the one square side length in the UMBmark method, and the turn in place experiment is done for $180\,[^\circ]$. This trajectories can be successfully applied to described three parameters expanded kinematic model Ivanjko et al. (2007) with an appropriate optimization criterion.

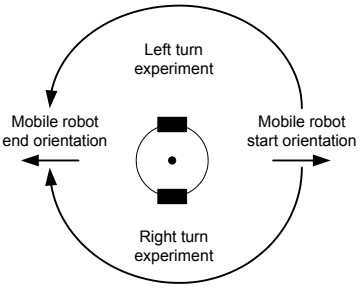


Fig. 5. Turn in place experiments

During experiments collected data were gathered in two groups, each group consisting of five experiments. First (calibration) group of experiments was used for odometry calibration and second (validation) group was used for validation of the obtained calibration parameters. Final calibration parameters values are averages of parameter values obtained from the five collected calibration data sets.

3.1.2 Parameters optimization

Before the optimization process can be started, an optimization criterion I, parameters that will be optimized, and their initial values have to be defined. In our case the optimization criterion is pose error minimum between the mobile robot final pose estimated using the three calibration parameters expanded kinematics model and exact measured mobile robot final pose. Parameters, which values will be changed during the optimization process, are the odometry calibration parameters.

Optimization criterion and appropriate equations that compute the mobile robot final pose is implemented as a m-function in software packet MATLAB. In our case such function consists of three parts: (i) experiment data retrieval, (ii) mobile robot final pose computation using new calibration parameters values, and (iii) optimization criterion value computation. Experiment data retrieval is done by loading needed measurements data from textual files. Such textual files are created during calibration experiments in a proper manner. That means file format has to imitate a ecumenical matrix structure. Numbers that present measurement data that have to be saved in a row are separated using a space sign and a new matrix row is denoted by a new line sign. So data saved in the same row belong to the same time step k. Function inputs are new values of the odometry calibration parameters, and output is new value of the optimization criterion. Function input is computed from the higher lever optimization function using an adequate optimization algorithm. Pseudo code of the here needed optimization m-functions is given in Algorithm 1 where $X(k)$ denotes estimated mobile robot pose.

Algorithm 1 Odometric calibration optimization criterion computation function pseudo code

Require: New calibration parameters values {Function input parameters}
Require: Measurement data: drive wheel velocities, time data, exact start and final mobile robot pose {Measurement data are loaded from an appropriately created textual file}
Require: Additional calibration parameters values {Parameters k_1 and k_2 for k_3 computation and vice versa}
1: $\omega_L, \omega_R \Leftarrow$ drive wheel velocities data file
2: $T \Leftarrow$ time data file
3: $X_{start}, X_{final} \Leftarrow$ exact start and final mobile robot pose
4: **repeat**
5: $X(k+1) = X(k) + \Delta X(k)$
6: **until** experiment measurement data exist
7: compute new optimization criterion value
8: **return** Optimization criterion value

In case of the expanded kinematic model with three parameters both experiments (straight line trajectory and turn in place) data and respectively two optimization m-functions are needed. Optimization is so done iteratively. Facts that calibration parameters k_1 and k_2 have the most influence on the straight line experiment and calibration parameter k_3 has the most influence on the turn in place experiment are exploited. Therefore, first optimal values of calibration parameters k_1 and k_2 are computed using collected data from the straight line experiment. Then optimal value of calibration parameter k_3 is computed using so far known values of calibration parameters k_1 and k_2, and collected data from the turn in place experiment. Whence the turn in place experiment is done in both directions, optimization procedure is done for both directions and average value of k_3 is used for the next iteration. We found out that two iterations were enough. Best optimization criterion for the expanded kinematic model with three parameters was minimization of the mobile robot final orientations differences. This can be explained by the fact that the orientation step depends on all three calibration parameters as given with (7) and (18). Mathematically used optimization criterion can be expressed as:

$$I = \Theta_{est} - \Theta_{exact}, \tag{19}$$

where Θ_{est} denotes estimated mobile robot final orientation $[^\circ]$, and Θ_{exact} exact measured mobile robot final orientation $[^\circ]$. Starting calibration parameters values were set to 1.0. Such calibration parameters value denotes usage of mobile robot nominal kinematics model.
Above described optimization procedure is done using the MATLAB Optimization Toolbox *** (2000). Appropriate functions that can be used depend on the version of MATLAB Optimization Toolbox and all give identical results. We successfully used the following functions: `fsolve`, `fmins`, `fminsearch` and `fzero`. These functions use the Gauss-Newton non-linear optimization method or the unconstrained nonlinear minimization Nelder-Mead method. It has to be noticed here that `fmins` and `fminsearch` functions search for a minimum m-function value and therefore absolute minimal value of the orientation difference has to be used. Except mentioned MATLAB Optimization Toolbox functions other optimization algorithms can be used as long they can accept or solve a minimization problem. When mentioned optimization functions are invoked, they call the above described optimization m-function with new calibration parameters values. Before optimization procedure is started

appropriate optimization m-function has to be prepared, which means exact experiments data have to be loaded and correct optimization criterion has to be used.

3.1.3 Experimental setup for odometry calibration

In this section experimental setup for odometry calibration is described. Main components, presented in Fig. 6 are: differential drive mobile robot with an on-board computer, camera connected to an off-board computer, and appropriate room for performing needed calibration experiments i.e. trajectories. Differential drive mobile robot used here was a Pioneer 2DX from *MOBILE*ROBOTS. It was equipped with an on-board computer from VersaLogic including a WLAN communication connection. In order to accurately and robustly measure the exact pose of calibrated mobile robot by the global vision system, a special patch (Fig. 7) is designed, which should be placed on the top of the robot before the calibration experiment.

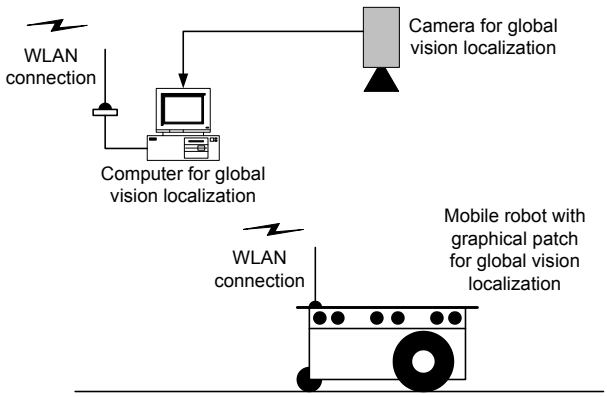


Fig. 6. Experimental setup for odometry calibration based on global vision

Software application for control of the calibration experiments, measurement of mobile robot start and end pose, and computation of calibration parameters values is composed from two parts: one is placed (run) on the mobile robot on-board computer and the other one on the off-board computer connected to the camera. Communication between these two application parts is solved using a networking library ArNetworking which is a component of the mobile robot control library ARIA *** (2007). On-board part of application gathers needed drive wheel speeds measurements, sampling time values, and control of the mobile robot experiment trajectories. Gathered data are then send, at the end of each performed experiment, to the off-board part of application. The later part of application decides which particular experiment has to be performed, starts a particular calibration experiment, and measures start and end mobile robot poses using the global vision camera attached to this computer. After all needed calibration experiments for the used calibration method are done, calibration parameters values are computed.

Using described odometry calibration method following calibration parameters values have been obtained: $k_1 = 0.9977$, $k_2 = 1.0023$, and $k_3 = 1.0095$. From the calibration parameters values it can be concluded that used mobile robot has a system error that causes it to slightly turn left when a straight-forward trajectory is performed. Mobile robot odometric system also overestimates its orientation resulting in k_3 value greater then 1.0.

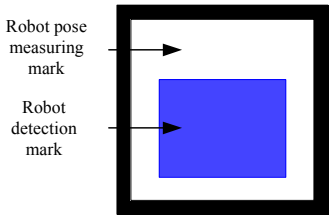


Fig. 7. Mobile robot patch used for pose measurements

3.2 Camera calibration

Camera calibration in the context of threedimensional (3D) machine vision is the process of determining the internal camera geometric and optical characteristics (intrinsic parameters) or the 3D position and orientation of the camera frame relative to a certain world coordinate system (extrinsic parameters) based on a number of points whose object coordinates in the world coordinate system $(X_i, i = 1, 2, \cdots, N)$ are known and whose image coordinates $(x_i, i = 1, 2, \cdots, N)$ are measured. It is a nonlinear optimization problem (20) whose solution is beyond the scope of this chapter. In our work perspective camera's parameters were determined by off-line camera calibration procedure described in Tsai (1987).

$$min \sum_{i=1}^{N} \left(SC^{-1} X_i - x_i \right)^2 \tag{20}$$

By this method with non-coplanar calibration target and full optimization, obtained were the following intrinsic parameters for SONY EVI-D31 pan-tilt-zoom analog camera and framegrabber with image resolution 320x240:

$$\alpha_x = \alpha_y = 379 \text{ [pixel]},$$
$$x_0 = 165.9 \text{ [pixel]}, \; y_0 = 140 \text{ [pixel]}.$$

4. Sonar based localization

A challenge of mobile robot localization using sensor fusion is to weigh its pose (i.e. mobile robot's state) and sonar range reading (i.e. mobile robot's output) uncertainties to get the optimal estimate of the pose, i.e. to minimize its covariance. The Kalman filter Kalman (1960) assumes the Gaussian probability distributions of the state random variable such that it is completely described with the mean and covariance. The optimal state estimate is computed in two major stages: time-update and measurement-update. In the time-update, state prediction is computed on the base of its preceding value and the control input value using the motion model. Measurement-update uses the results from time-update to compute the output predictions with the measurement model. Then the predicted state mean and covariance are corrected in the sense of minimizing the state covariance with the weighted difference between predicted and measured outputs. In succession, motion and measurement models needed for the mobile robot sensor fusion are discussed, and then EKF and UKF algorithms for mobile robot pose tracking are presented. Block diagram of implemented Kalman filter based localization is given in Fig. 8.

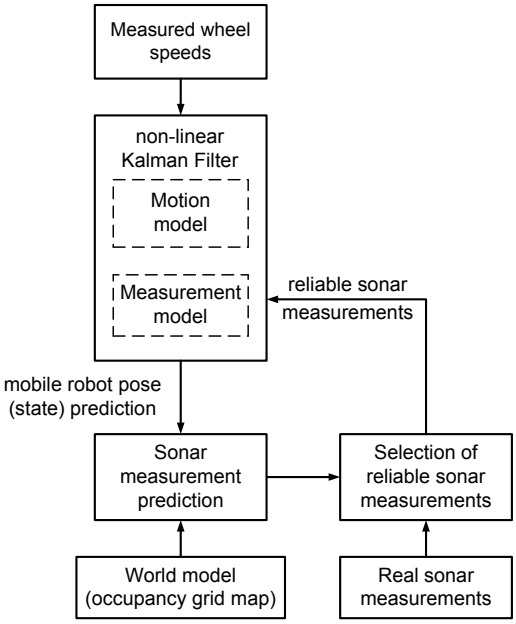


Fig. 8. Block diagram of non-linear Kalman filter localization approaches.

4.1 Occupancy grid world model

In mobile robotics, an occupancy grid is a two dimensional tessellation of the environment map into a grid of equal or unequal cells. Each cell represents a modelled environment part and holds information about the occupancy status of represented environment part. Occupancy information can be of probabilistic or evidential nature and is often in the numeric range from 0 to 1. Occupancy values closer to 0 mean that this environment part is free, and occupancy values closer to 1 mean that an obstacle occupies this environment part. Values close to 0.5 mean that this particular environment part is not yet modelled and so its occupancy value is unknown. When an exploration algorithm is used, this value is also an indication that the mobile robot has not yet visited such environment parts. Some mapping methods use this value as initial value. Figure 9 presents an example of ideal occupancy grid map of a small environment. Left part of Fig. 9 presents outer walls of the environment and cells belonging to an empty occupancy grid map (occupancy value of all cells set to 0 and filled with white color). Cells that overlap with environment walls should be filled with information that this environment part is occupied (occupancy value set to 1 and filled with black color as it can be seen in the right part of Fig. 9). It can be noticed that cells make a discretization of the environment, so smaller cells are better for a more accurate map. Drawback of smaller cells usage is increased memory consumption and decreased mapping speed because occupancy information in more cells has to be updated during the mapping process. A reasonable tradeoff between memory consumption, mapping speed, and map accuracy can be made with cell size of 10 [cm] x 10 [cm]. Such a cell size is very common when occupancy grid maps are used and is used in our research too.

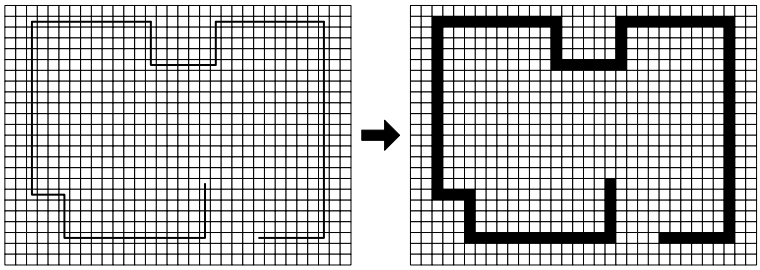


Fig. 9. Example of occupancy grid map environment

Obtained occupancy grid map given in the right part of Fig. 9 does not contain any unknown space. A map generated using real sonar range measurement will contain some unknown space, meaning that the whole environment has not been explored or that during exploration no sonar range measurement defined the occupancy status of some environment part.
In order to use Kalman filter framework given in Fig. 8 for mobile robot pose estimation, prediction of sonar sensor measurements has to be done. The sonar feature that most precise measurement information is concentrated in the main axis of the sonar main lobe is used for this step. So range measurement prediction is done using one propagated beam combined with known local sensor coordinates and estimated mobile robot global pose. Measurement prediction principle is depicted in Fig. 10.

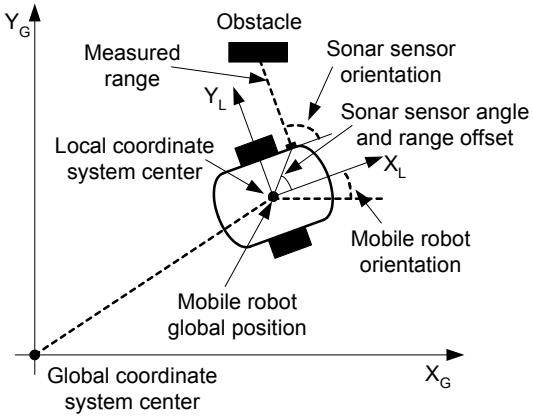


Fig. 10. Sonar measurement prediction principle.

It has to be noticed that there are two sets of coordinates when measurement prediction is done. Local coordinates defined to local coordinate system (its axis are denoted with X_L and Y_L in Fig. 10) that is positioned in the axle center of the robot drive wheels. It moves with the robot and its x-axis is always directed into the current robot motion direction. Sensors coordinates are defined in this coordinate system and have to be transformed in the global coordinate system center (its axis are denoted with X_G and Y_G in Fig. 10) to compute relative

distance between the sonar sensor and obstacles. This transformation for a particular sonar sensor is given by the following equations:

$$S_{XG} = x + S_{offD} \cdot cos\left(S_{off\Theta} + \Theta\right), \tag{21}$$

$$S_{YG} = y + S_{offD} \cdot sin\left(S_{off\Theta} + \Theta\right), \tag{22}$$

$$S_{\Theta G} = \Theta + S_{sens\Theta}, \tag{23}$$

where coordinates x and y present mobile robot global position $[mm]$, Θ mobile robot global orientation $[^\circ]$, coordinates S_{XG} and S_{YG} sonar sensor position in global coordinates $[mm]$, $S_{\Theta G}$ sonar sensor orientation in the global coordinate system frame $[^\circ]$, S_{offD} sonar sensor distance from the center of the local coordinate system $[mm]$, $S_{off\Theta}$ sonar sensor angular offset towards local coordinate system $[^\circ]$, and $S_{\Theta G}$ sonar sensor orientation towards the global coordinate system $[^\circ]$.
After above described coordinate transformation is done, start point and direction of the sonar acoustic beam are known. Center of the sound beam is propagated from the start point until it hits an obstacle. Obtained beam length is then equal to predicted sonar range measurement. Whence only sonar range measurements smaller or equal then 3.0 m are used, measurements with a predicted value greater then 3.0 m are are being discarded. Greater distances have a bigger possibility to originate from outliers and are so not good for pose correction.

4.2 EKF localization

The motion model represents the way in which the current state follows from the previous one. State vector is expressed as the mobile robot pose, $\mathbf{x}_k = [x_k\ y_k\ \Theta_k]^T$, with respect to a global coordinate frame, where k denotes the sampling instant. Its distribution is assumed to be Gaussian, such that the state random variable is completely determined with a 3×3 covariance matrix $\mathbf{P}_k$ and the state expectation (mean, estimate are used as synonyms). Control input, $\mathbf{u}_k$, represents the commands to the robot to move from time step k to $k+1$. In the motion model $\mathbf{u}_k = [D_k\ \Delta\Theta_k]^T$ represents translation for distance D_k followed by a rotation for angle $\Delta\Theta_k$. The state transition function $\mathbf{f}(\cdot)$ uses the state vector at the current time instant and the current control input to compute the state vector at the next time instant:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k), \tag{24}$$

where $\mathbf{v}_k = \left[v_{1,k}\ v_{2,k}\right]^T$ represents unpredictable process noise, that is assumed to be Gaussian with zero mean, ($\mathbb{E}\{\mathbf{v}_k\} = [0\ 0]^T$), and covariance $\mathbf{Q}_k$. With $\mathbb{E}\{\cdot\}$ expectation function is denoted. Using (1) to (3) the state transition function becomes:

$$\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{v}_k) = \begin{bmatrix} x_k + (D_k + v_{1,k}) \cdot \cos(\Theta_k + \Delta\Theta_k + v_{2,k}) \\ y_k + (D_k + v_{1,k}) \cdot \sin(\Theta_k + \Delta\Theta_k + v_{2,k}) \\ \Theta_k + \Delta\Theta_k + v_{2,k} \end{bmatrix}. \tag{25}$$

The process noise covariance $\mathbf{Q}_k$ was modelled on the assumption of two independent sources of error, translational and angular, i.e. D_k and $\Delta\Theta_k$ are added with corresponding uncertainties. The expression for $\mathbf{Q}_k$ is:

$$\mathbf{Q}_k = \begin{bmatrix} \sigma_D^2 & 0 \\ 0 & \Delta\Theta_k^2 \sigma_{\Delta\Theta}^2 \end{bmatrix}, \tag{26}$$

where σ_D^2 and $\sigma_{\Delta\Theta}^2$ are variances of D_k and $\Delta\Theta_k$, respectively.
The measurement model computes the range between an obstacle and the axle center of the robot according to a measurement function Lee (1996):

$$h_i(\mathbf{x}, p_i) = \sqrt{(x_i - x)^2 + (y_i - y)^2}, \tag{27}$$

where $p_i = (x_i, y_i)$ denotes the point (occupied cell) in the world model detected by the ith sonar. The sonar model uses (27) to relate a range reading to the obstacle that caused it:

$$z_{i,k} = h_i(\mathbf{x}_k, p_i) + w_{i,k}, \tag{28}$$

where $w_{i,k}$ represents the measurement noise (Gaussian with zero mean and variance $r_{i,k}$) for the ith range reading. All range readings are used in parallel, such that range measurements $z_{i,k}$ are simply stacked into a single measurement vector $\mathbf{z}_k$. Measurement covariance matrix $\mathbf{R}_k$ is a diagonal matrix with the elements $r_{i,k}$. It is to be noted that the measurement noise is additive, which will be beneficial for UKF implementation.
EKF is the first sensor fusion based mobile robot pose tracking technique presented in this paper. Detailed explanation of used EKF localization can be found in Ivanjko et al. (2004) and in the sequel only basic equations are presented. Values of the control input vector $\mathbf{u}_{k-1}$ computed from wheels' encoder data are passed to the algorithm at time k such that first time-update is performed obtaining the prediction estimates, and then if new sonar readings are available those predictions are corrected. Predicted (prior) state mean $\hat{\mathbf{x}}_k^-$ is computed in single-shot by propagating the state estimated at instant $k-1$, $\hat{\mathbf{x}}_{k-1}$ through the true nonlinear odometry mapping:

$$\hat{\mathbf{x}}_k^- = \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbb{E}\{\mathbf{v}_{k-1}\}). \tag{29}$$

The covariance of the predicted state $\mathbf{P}_k^-$ is approximated with the covariance of the state propagated through a linearized system from (24):

$$\mathbf{P}_k^- = \nabla\mathbf{f}_\mathbf{x} \mathbf{P}_{k-1} \nabla\mathbf{f}_\mathbf{x}^T + \nabla\mathbf{f}_\mathbf{u} \mathbf{Q}_k \nabla\mathbf{f}_\mathbf{u}^T, \tag{30}$$

where $\nabla\mathbf{f}_\mathbf{x} = \nabla\mathbf{f}_\mathbf{x}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbb{E}\{\mathbf{v}_{k-1}\})$ is the Jacobian matrix of $\mathbf{f}$ with respect to $\mathbf{x}$, while $\nabla\mathbf{f}_\mathbf{u} = \nabla\mathbf{f}_\mathbf{u}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbb{E}\{\mathbf{v}_{k-1}\})$ is the Jacobian matrix of $\mathbf{f}(\cdot)$ with respect to control input $\mathbf{u}$. It is to be noticed that using (29) and (30) the mean and covariance are accurate only to the first-order of the corresponding Taylor series expansion Haykin (2001). If there are no new sonar readings at instant k or if they are all rejected, measurement update does not occur and the estimate mean and covariance are assigned with the predicted ones:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^-, \tag{31}$$

$$\mathbf{P}_k = \mathbf{P}_k^-. \tag{32}$$

Otherwise, measurement-update takes place where first predictions of the accepted sonar readings are collected in $\hat{\mathbf{z}}_k^-$ with ith component of it being:

$$\hat{z}_{i,k}^- = h_i(\hat{\mathbf{x}}_k^-, p_i) + E\{w_{i,k}\}. \tag{33}$$

The state estimate and its covariance in time step k are computed as follows:

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k^-), \tag{34}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \nabla \mathbf{h}_\mathbf{x})\mathbf{P}_k^-, \tag{35}$$

where $\mathbf{z}_k$ are real sonar readings, $\nabla \mathbf{h}_\mathbf{x} = \nabla \mathbf{h}_\mathbf{x}(\hat{\mathbf{x}}_k^-, \mathbb{E}\{\mathbf{w}_k\})$ is the Jacobian matrix of the measurement function with respect to the predicted state, and $\mathbf{K}_k$ is the optimal Kalman gain computed as follows:

$$\mathbf{K}_k = \mathbf{P}_k^- \nabla \mathbf{h}_\mathbf{x}^T (\nabla \mathbf{h}_\mathbf{x} \mathbf{P}_k^- \nabla \mathbf{h}_\mathbf{x}^T + \mathbf{R}_k)^{-1}. \tag{36}$$

4.3 UKF localization

The second sensor fusion based mobile robot pose tracking technique presented in this chapter uses UKF. UKF was first proposed by Julier et al. Julier & Uhlmann (1996), and further developed by Wan and van der Merwe Haykin (2001). It utilizes the unscented transformation Julier & Uhlmann (1996) that approximates the true mean and covariance of a Gaussian random variable propagated through nonlinear mapping accurate to the inclusively third order of Taylor series expansion for any mapping. Following this, UKF approximates state and output mean and covariance more accurately than EKF and thus superior operation of UKF compared to EKF is expected. UKF was already used for mobile robot localization in Ashokaraj et al. (2004) to fuse several sources of observations, and the estimates were, if necessary, corrected using interval analysis on sonar measurements. Here we use sonar measurements within UKF, without any other sensors except the encoders to capture angular velocities of the drive wheels (motion model inputs), and without any additional estimate corrections.

Means and covariances are in UKF case computed by propagating carefully chosen so called pre-sigma points through the true nonlinear mapping. Nonlinear state-update with non-additive Gaussian process noises in translation D and rotation $\Delta\Theta$ is given in (25). The measurement noise is additive and assumed to be Gaussian, see (28).

The UKF algorithm is initialized ($k = 0$) with $\hat{\mathbf{x}}_0$ and $\mathbf{P}_0$, same as the EKF. In case of non-additive process noise and additive measurement noise, state estimate vector is augmented with means of process noise $\mathbb{E}\{\mathbf{v}_{k-1}\}$ only, thus forming extended state vector $\hat{\mathbf{x}}_{k-1}^a$:

$$\hat{\mathbf{x}}_{k-1}^a = \mathbb{E}[\mathbf{x}_{k-1}^a] = \begin{bmatrix} \hat{\mathbf{x}}_{k-1}^T & \mathbb{E}\{\mathbf{v}_{k-1}\}^T \end{bmatrix}^T. \tag{37}$$

Measurement noise does not have to enter the $\hat{\mathbf{x}}_{k-1}^a$ because of additive properties Haykin (2001). This is very important from implementation point of view since the dimension of output is not known in advance because number of accepted sonar readings varies. Covariance matrix is augmented accordingly forming matrix $\mathbf{P}_{k-1}^a$:

$$\mathbf{P}_{k-1}^a = \begin{bmatrix} \mathbf{P}_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_{k-1} \end{bmatrix}. \tag{38}$$

Time-update algorithm in time instant k first requires square root of the $\mathbf{P}_{k-1}^a$ (or lower triangular Cholesky factorization), $\sqrt{\mathbf{P}_{k-1}^a}$. Obtained lower triangular matrix is scaled by the factor γ:

$$\gamma = \sqrt{L + \lambda}, \tag{39}$$

where L represents the dimension of augmented state $\mathbf{x}^a_{k-1}$ ($L = 5$ in this application), and λ is a scaling parameter computed as follows:

$$\lambda = \alpha^2(L + \kappa) - L. \tag{40}$$

Parameter α can be chosen within range $[10^{-4}, 1]$, and κ is usually set to 1. There are $2L + 1$ pre-sigma points, the first is $\hat{\mathbf{x}}^a_{k-1}$ itself, and other $2L$ are obtained by adding to or subtracting from $\hat{\mathbf{x}}^a_{k-1}$ each of L columns of $\gamma\sqrt{\mathbf{P}^a_{k-1}}$, symbolically written as:

$$\mathcal{X}^a_{k-1} = \left[\begin{array}{ccc} \hat{\mathbf{x}}^a_{k-1} & \hat{\mathbf{x}}^a_{k-1} + \gamma\sqrt{\mathbf{P}^a_{k-1}} & \hat{\mathbf{x}}^a_{k-1} - \gamma\sqrt{\mathbf{P}^a_{k-1}} \end{array} \right], \tag{41}$$

where $\mathcal{X}^a_{k-1} = [(\mathcal{X}^x_{k-1})^T \quad (\mathcal{X}^v_{k-1})^T]^T$ represents the matrix whose columns are pre-sigma points. All pre-sigma points are processed by the state-update function obtaining matrix $\mathcal{X}^x_{k|k-1}$ of predicted states for each pre-sigma point, symbolically written as:

$$\mathcal{X}^x_{k|k-1} = \mathbf{f}[\mathcal{X}^x_{k-1}, \mathbf{u}_{k-1}, \mathcal{X}^v_{k-1}]. \tag{42}$$

Prior mean is calculated as weighted sum of acquired points:

$$\hat{\mathbf{x}}^-_k = \sum_{i=0}^{2L} W^{(m)}_i \mathcal{X}^x_{i,k|k-1}, \tag{43}$$

where $\mathcal{X}^x_{i,k|k-1}$ denotes the ith column of $\mathcal{X}^x_{k|k-1}$. Weights for mean calculation $W^{(m)}_i$ are given by

$$W^{(m)}_0 = \frac{\lambda}{L + \lambda}, \tag{44}$$

$$W^{(m)}_i = \frac{1}{2(L + \lambda)}, \quad i = 1, \ldots, 2L. \tag{45}$$

Prior covariance matrix $\mathbf{P}^-_k$ is given by

$$\mathbf{P}^-_k = \sum_{i=0}^{2L} W^{(c)}_i [\mathcal{X}^x_{i,k|k-1} - \hat{\mathbf{x}}^-_k][\mathcal{X}^x_{i,k|k-1} - \hat{\mathbf{x}}^-_k]^T, \tag{46}$$

where $W^{(c)}_i$ represent the weights for covariance calculation which are given by

$$W^{(c)}_0 = \frac{\lambda}{L + \lambda} + (1 - \alpha^2 + \beta), \tag{47}$$

$$W^{(c)}_i = \frac{1}{2(L + \lambda)}, \quad i = 1, \ldots, 2L. \tag{48}$$

For Gaussian distributions $\beta = 2$ is optimal.
If there are new sonar readings available at time instant k, predicted readings of accepted sonars for each sigma-point are grouped in matrix $\mathcal{Z}_{k|k-1}$ obtained by

$$\mathcal{Z}_{k|k-1} = \mathbf{h}[\mathcal{X}^x_{k|k-1}, \mathbf{p}] + \mathbb{E}\{\mathbf{w}_k\}, \tag{49}$$

where $\mathbf{p}$ denotes the series of points in the world map predicted to be hit by sonar beams. Predicted readings $\hat{\mathbf{z}}_k^-$ are then

$$\hat{\mathbf{z}}_k^- = \sum_{i=0}^{2L} W_i^{(m)} \mathcal{Z}_{i,k|k-1}. \tag{50}$$

To prevent the sonar readings that hit near the corner of obstacles to influence on the measurement correction, since their probability distribution cannot be approximated with Gaussian, another threshold comparison were made. These problematic sonar readings are recognized with mean $\hat{z}_{i,k}^-$ that differs from $z_{i,k}$ more than the acceptation threshold amounts, and those are being discarded. Readings covariance is

$$\mathbf{P}_{\mathbf{z}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{Z}_{i,k|k-1} - \hat{\mathbf{z}}_k^-][\mathcal{Z}_{i,k|k-1} - \hat{\mathbf{z}}_k^-]^T + \mathbf{R}_k, \tag{51}$$

and state-output cross-covariance matrix is

$$\mathbf{P}_{\mathbf{x}_k \mathbf{z}_k} = \sum_{i=0}^{2L} W_i^{(c)} [\mathcal{X}_{i,k|k-1}^x - \hat{\mathbf{x}}_k^-][\mathcal{Z}_{i,k|k-1} - \hat{\mathbf{z}}_k^-]^T. \tag{52}$$

Kalman gain $\mathbf{K}_k$ is

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{x}_k \mathbf{z}_k} \mathbf{P}_{\mathbf{z}_k}^{-1}. \tag{53}$$

Posterior state covariance is finally calculated as

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k \mathbf{P}_{\mathbf{z}_k} \mathbf{K}_k^T. \tag{54}$$

The measurement correction is done as in (34).

5. Monocular vision based localization

In this section, we consider the problem of mobile robot pose estimation using only visual information from a single camera and odometry readings. Focus is on building complex environmental models, fast online rendering and real-time complex and noisy image segmentation. The 3D model of the mobile robot's environment is built using a professional freeware computer graphics tool named Blender and pre-stored in the memory of the robot's on-board computer. Estimation of the mobile robot pose as a stochastic variable is done by correspondences of image lines, extracted using Random Window Randomized Hough Transform line detection algorithm, and model lines, predicted using odometry readings and 3D environment model. The camera model and ray tracing algorithm are also described. Developed algorithms are also experimentally tested using a Pioneer 2DX mobile robot.

5.1 Scene modelling and rendering

Referential model of the environment was built using *Blender*, where vertices, edges (lines) and faces (planes) were used for model notation. An edge is defined with two vertices and a face with three or four vertices. The drawn model is one object in which all vertices, edges and faces are listed. For illustration, in Fig. 11, 3D model of the hallway in which our mobile robot moves is shown. Although the environment model is quite complex, we achieved online execution of the localization algorithms by applying fast rendering. Namely, in each

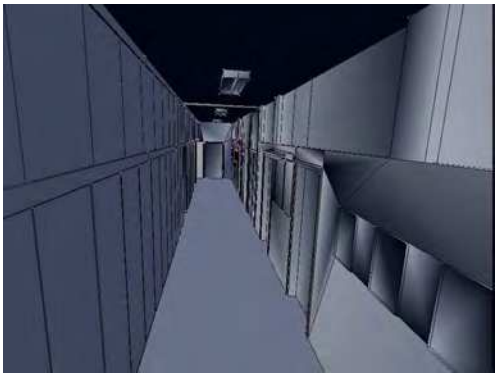
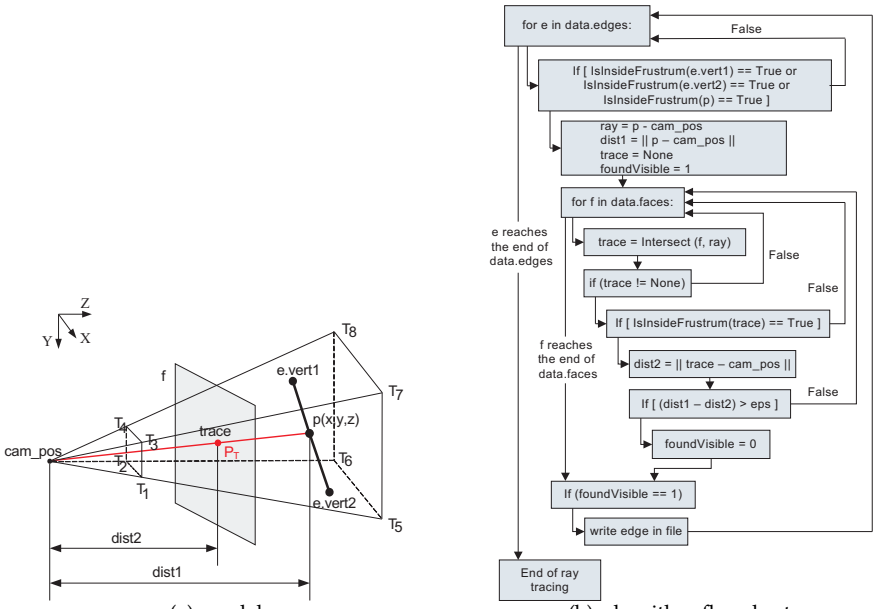


Fig. 11. Hallway 3D model

step we render only the small region enclosed with the camera frustrum and then apply a ray tracing algorithm to solve the occlusion problems.
The above described notation of the scene model, enables us to implement a ray tracing algorithm. The algorithm is organized in two "for" loops, as shown in Fig. 12(b), where the algorithm flowchart is depicted. The first (outer) loop goes over the edge list and the second (inner) loop goes over the face list. The outer loop starts with the function IsInsideFrustrum (point3D), which examines whether observed points are located inside the camera frustrum and discards those that are not in it. Then, for a point p in the frustrum, where p is the point in the middle of the edge determined with two vertices, e.vert1 and e.vert2, as shown in Fig. 12(a), the direction of the vector ray is defined with point p and camera pose (cam_pos).
The inner loop starts with choosing a plane f from the list of faces, and then the function Intersect (f, vector) returns intersection point P_T between the given plane f and direction vector as an output value, or *None* if the intersection doesn't exist. Visible edges are checked by comparing distances from the camera pose to the the point p (dist1) and to intersection point P_T (dist2), see Fig. 12(a). If these two distances do not match, the checked model edge (line) is invisible, and therefore not used in later matching procedure.
Notice the incompleteness of rendering because only edges whose middle point is visible will be rendered visible. That does not affect the accuracy of the later matching algorithm for partially visible model lines because it is done in Hough space where a line is represented with a single point regarding its length. The rendering could produce only smaller number of partially visible lines, but in this case it is not important because there are still enough lines for estimating mobile robot's pose while gaining faster algorithm execution.

5.2 Image segmentation

Mobile robot self-localization requires matching of edge segments in the current camera image and edge segments in the environment model seen from the expected mobile robot pose. In previous section we described line extraction from the environment model, and below we describe the line extraction in the camera image (image segmentation). Image segmentation is done by the Canny edge detector and RWRHT line detector algorithm described in Kälviäinen et al. (1994). The RWRHT is based on Randomized Hough Transformation (RHT), which selects n pixels from the edge image by random sampling to solve n parameters of



(a) model (b) algorithm flowchart

Fig. 12. Ray tracing

a curve, and then accumulates only one cell in parameter space. Detected curves are those whose accumulator cell is greater then predefined threshold. The RWRHT is an extension of the RHT on complex and noisy images that applies RHT to a limited neighborhood of edge pixels. The benefit is the reduction of the computational power and memory resources. The pseudo-code of the RWHT is written in Algorithm 2.

5.3 Mobile robot pose estimation

Once the correspondences have been established between image lines and model lines seen from the current mobile robot pose, we could update the mobile robot pose by applying an appropriate estimation technique. In most cases, linearized system and measurement equations and Gaussian noise in states and measurements are satisfactory approximations. Therefore, we apply Extended Kalman Filter (EKF) Welch & Bishop (2000), which is an optimal estimator under the above assumptions.

The state vector that is to be estimated is the mobile robot pose $\mathbf{p}$. Introducing uncertainty in the equations (1), (2) and (3) as the zero mean Gaussian additive noise, the state equations are obtained:

$$\mathbf{p_{n+1}} = \mathbf{f}\left[\mathbf{p_n}, v(n), \omega(n)\right] + w(n), \tag{55}$$

where $w \sim \mathcal{N}(0, Q)$.

Algorithm 2 The Random Window RHT Algorithm

1: $D \Leftarrow$ all edge points in binary edge picture
2: $d_i \Leftarrow$ randomly selected point from set D
3: $m \Leftarrow$ randomly selected window size where $m_{min} \leq m \leq m_{max}$
4: $W \Leftarrow$ pixel data set of $m \times m$ neighborhood of the point d_i
5: **repeat**
6: RHT algorithm on the set W
7: **until** maximum R times
8: **if** accumulator cell $\geq$ threshold **then**
9: corresponding parameters are the parameters of detected curve
10: **else**
11: goto 2
12: **end if**

Measurement is a set $\mathcal{S}$ of pairs "model line - image line":

$$\begin{aligned} \mathcal{S} = \{\{m, i\} | m &= \text{visible model line}, \\ i &= \text{image line - perspective projection of m}\}. \end{aligned} \tag{56}$$

The straight line in the world coordinate system which passes through the point (x_0, y_0, z_0) and has direction coefficients (a, b, c) is given by:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} u + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}, \; u \in \Re. \tag{57}$$

The straight line in image plane is given by

$$x \, cos\gamma + y \, sin\gamma = \rho, \quad \rho \geq 0, \quad \gamma \in [0, 2\pi], \tag{58}$$

where ρ and γ are the Hough space parameters of the line.
Let by applying perspective projection transformation P to a 3D model line we obtain 2D straight line m and let its pair line i in the image be defined with the Hough space parameters (ρ, γ). The projection of the 3D point (x_0, y_0, z_0) lies on the image line i and direction coefficients of m and i lines are the same if the following conditions are fulfilled:

$$z_1 = W \begin{bmatrix} \rho\, cos\gamma \\ \rho\, sin\gamma \\ 1 \end{bmatrix} - P \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} = 0, \tag{59}$$

$$z_2 = \begin{bmatrix} -sin\gamma \\ cos\gamma \\ 0 \end{bmatrix} - \frac{P \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix}}{\left\| P \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} \right\|} = 0, \tag{60}$$

$$\mathbf{z} = [z_1, z_2]^\top . \tag{61}$$

Equations (59) and (60) are measurement equations and are used to correct mobile robot pose with each correct match. Uncertainty in mobile robot pose means uncertainty in model lines and is propagated to the Hough space in the same way as in Kosaka & Kak (1992), so we will not go into details but only explain the main concept. For differentials of the equation (15) we obtain

$$\delta\mathbf{X} = \begin{bmatrix} \delta X \\ \delta Y \end{bmatrix} = J_{\mathcal{P}}(\mathbf{M},\, \hat{\mathbf{p}})\, \delta\mathbf{p}\,, \tag{62}$$

where $J_{\mathcal{P}}(\hat{\mathbf{p}})$ is Jacobian of perspective projection of the end point of 3D model line M taken at expected values of random vector $\mathbf{p}$. Notice that from this point on, we are using first order Taylor approximations of nonlinear transformations. Covariance matrix related to pixel coordinates of a single model line point is given by:

$$\Sigma_X = E[\delta\mathbf{X}\delta\mathbf{X}^\top] = J_{\mathcal{P}}(\mathbf{M},\, \hat{\mathbf{p}})\; \Sigma_p\; J_{\mathcal{P}}(\mathbf{M},\, \hat{\mathbf{p}})^\top, \tag{63}$$

where Σ_p is covariance matrix of mobile robot pose $\mathbf{p}$. So, at this moment, we have determined uncertainty convex hull in which the probability of finding corresponding image line is the highest. Furthermore, we can apply Hough transform $\mathcal{H}$ to that line segment, which would lead to point representation $\hat{h} = (\rho, \gamma)$ of the line on which the segment coincides with elliptical uncertainty region defined by the Mahalanobious distance and covariance matrix. If $J_{\mathcal{H}}$ denotes Jacobian of the Hough transform $\mathcal{H}$ with respect to variables ρ and γ we can write:

$$\begin{bmatrix} \delta\rho \\ \delta\gamma \end{bmatrix} = J_{\mathcal{H}} \begin{bmatrix} \delta X_1 \\ \delta Y_1 \\ \delta X_2 \\ \delta Y_2 \end{bmatrix}, \tag{64}$$

$$\Sigma_{\rho\gamma} = J_{\mathcal{H}} \begin{bmatrix} J_{\mathcal{P}}(\mathbf{M_1},\, \hat{\mathbf{p}}) & 0 \\ 0 & J_{\mathcal{P}}(\mathbf{M_2},\, \hat{\mathbf{p}}) \end{bmatrix} \Sigma_p \begin{bmatrix} J_{\mathcal{P}}(\mathbf{M_1},\, \hat{\mathbf{p}}) & 0 \\ 0 & J_{\mathcal{P}}(\mathbf{M_2},\, \hat{\mathbf{p}}) \end{bmatrix}^\top J_{\mathcal{H}}^\top. \tag{65}$$

We limit the search for image lines to uncertainty region in the Hough space $\subset \Re^2$ determined by the constraint:

$$(\mathbf{h} - \hat{\mathbf{h}})\Sigma_{\rho\gamma}^{-1}(\mathbf{h} - \hat{\mathbf{h}})^T \leq 2.$$

This rises a matching problem if one model line has more then one image candidate, but the problem was solved in Kosaka & Kak (1992) and Aider et al. (2005).
There are also quantization error, noise in camera image and error in edge detection and image segmentation which have been approximated by Gaussian variable $\xi \sim \mathcal{N}(0, V)$ and included in the EKF equations as the measurement noise. Finally, the equations of the implemented EKF are:
a priori update:

$$\hat{\mathbf{p}}_{\mathbf{n+1}} = f\left[\hat{\mathbf{p}}_n, v(n), \omega(n)\right], \tag{66}$$

$$\hat{\mathbf{z}}_{\mathbf{n}} = \mathbf{z}\left[\hat{\mathbf{p}}_{\mathbf{n}}, \hat{\mathbf{h}}(i)_{i\to m}\right], \tag{67}$$

$$\Sigma_{p_{n+1|n}} = A\,\Sigma_{p_n}\,A^\top + Q, \quad A = \partial\mathbf{f}/\partial\mathbf{p}|_{\mathbf{p}=\hat{\mathbf{p}}}, \tag{68}$$

a posteriori update:

$$K_{n+1} = \Sigma_{p_{n+1|n}}\,H^\top\left[H\,\Sigma_{p_{n+1|n}}\,H^\top + R\right], \tag{69}$$

$$\mathbf{p_{n+1}} = \hat{\mathbf{p}}_{\mathbf{n+1}} + K_{n+1}(\mathbf{z} - \hat{\mathbf{z}}_{\mathbf{n}}), \tag{70}$$

$$\Sigma_{p_{n+1}} = (I - K_{n+1}\,H)\Sigma_{p_{n+1|n}}, \tag{71}$$

where $H = \frac{\partial \mathbf{z}}{\partial \mathbf{p}}|_{\mathbf{p}=\hat{\mathbf{p}}}$ and $R = \frac{\partial \mathbf{z}}{\partial \mathbf{h}}|_{\mathbf{h}=\hat{\mathbf{h}}} \cdot V \cdot \frac{\partial \mathbf{z}}{\partial \mathbf{h}}^\top|_{\mathbf{h}=\hat{\mathbf{h}}}$.

6. Experimental results

This section presents obtained experimental results including description of experimental setup and experiment conditions. First are presented results obtained using sonar sensors and after that results obtained using monocular-vision. Section ends with comments on obtained results.

6.1 Experimental setup description

Experiments are performed using a Pioneer 2DX mobile robot from MobileRobots. Its configuration is presented in Fig. 13 only that in localization experiments monocular camera was used insted of depicted stereo one. Used sensors are encoders for odometry, sonars, mono-camera and a laser range finder. Laser range finder was used only for a comparison purpose as a sensor that enables a more accurate localization than the sonars or mono-camera. It is combined with a Monte-Carlo algorithm Konolige (1999) implemented as standard localization solution in the mobile robot control software. Used camera for monocular-vision localization is a SONY EVI-D31 pan-tilt-zoom analog camera. Laser sensor is a SICK LMS-200, and sonars are Polaroid 6500 sensors.
Experimental environment including trajectory traversed by the mobile robot is presented in Fig. 14. Global coordinate system is depicted on the left side. It's a hallway with several door niches. Mobile robot movement started in one corridor end and ended when it reached other corridor end. Trajectory length is approximately 20 [m] and is generated using a gradient based algorithm described in Konolige (2000). Obstacle avoidance was also active during all experiments.

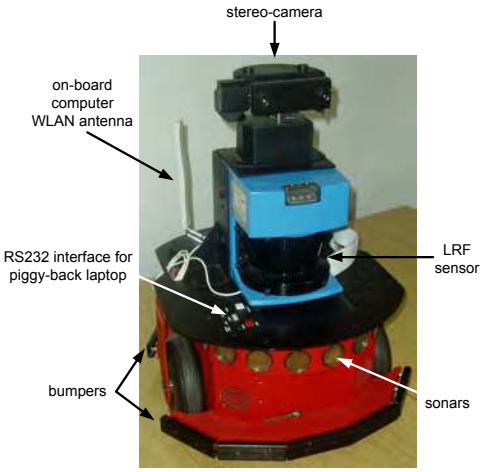


Fig. 13. Pioneer 2DX mobile robot

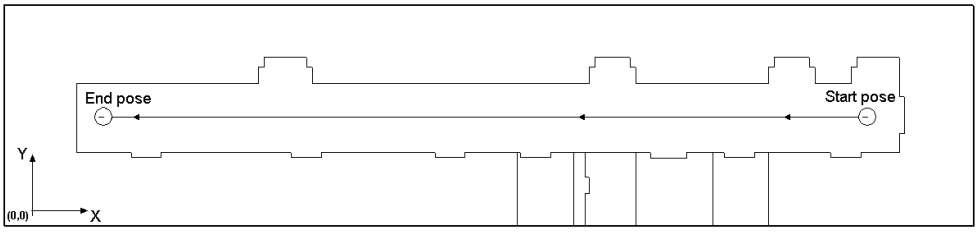


Fig. 14. Mobile robot trajectory for experimental evaluation.

Initial robot's pose and its covariance was in all experiments set to (orientation given in radians):

$$\mathbf{p_0} = \left[\begin{array}{ccc} 210 \text{ dm} & 9.25 \text{ dm} & \pi \end{array} \right]^\top,$$

$$\Sigma_{\mathbf{p}_0} = \left[\begin{array}{ccc} 0.3000 & 0 & 0 \\ 0 & 0.3000 & 0 \\ 0 & 0 & 0.0080 \end{array} \right],$$

which means about 0.55 $[dm]$ standard deviation in p_x and p_y and about 5 $[^\circ]$ standard deviation in robot orientation. In every experiment mobile robot start pose was manually set according to marks on the hallway floor, so given initial pose covariance was set to cover start pose setting error margins. Experiments show that implemented self-localization algorithms were able to cope with that starting pose error margins.
During all experiments relevant measurements data were collected and saved for obtained localization quality evaluation. Saved data included mobile robot drive wheel speeds, sampling period, sonar range measurements, camera images, evaluated self-localization algorithm estimated pose and Monte Carlo based localization results. Pose obtained using Monte Carlo

algorithm and laser range finder sensor was then used as the more accurate i.e. exact mobile robot pose for comparison.
If localization was only done by odometry, results would be like shown on Fig. 15, i.e. pose error would monotonically increase over time. It is obvious that mobile robot cannot perform its tasks without any pose correction.

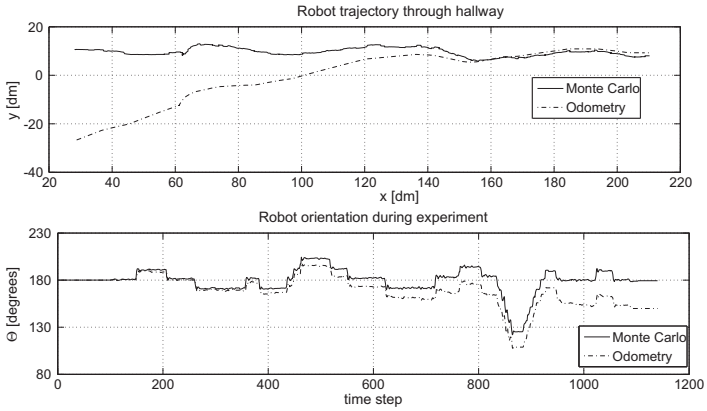


Fig. 15. Robot trajectory method comparison: solid - Monte Carlo localization, dash-dot - odometry

6.2 Sonar localization results

Figures 16 and 17 present results obtained using sonar sensors. Solid line denotes Monte Carlo localization with laser range finder and doted line denotes sonar sensor based localization results. Both figures consist of two parts. Upper part presents mobile robot trajectory i.e. its position and lower part presents mobile robot orientation change.
As mentioned before two sonar sensor based localization approaches were implemented. EKF based results are presented in Fig. 16 and UKF based results are presented in Fig. 17. Both implementations use calibrated odometry as the motion model to increase localization accuracy. An occupancy grid model is used for sonar sensor measurement prediction. Whence occupancy grid size is 100 $[mm]$ x 100 $[mm]$, localization accuracy is expected to be in this range.
Figures given in this section show that sonar sensors can be effectively used for self-localization with accuracy in range of used occupancy model grid size. It can be noticed that EKF ends with a pose estimation with bigger pose corrections. This feature arises from the first order linearization done be the EKF. In case of the UKF corrections are of smaller value as expected. More accurate pose estimation of the UKF can be proved by computing pose error values on the whole traversed path. Pose error is computed as difference between estimated trajectory and referent Monte Carlo pose. With the EKF maximal position error is 3.2 $[dm]$, and maximal orientation error is 6.8 $[^\circ]$, while with the UKF their values are 2.5 $[dm]$, and 3.7 $[^\circ]$. These values are important when self-localization algorithms are used for longer trajectories. A bigger maximal pose error indicates a greater probability that mobile robot will loose its pose indicating a necessary global pose correction.

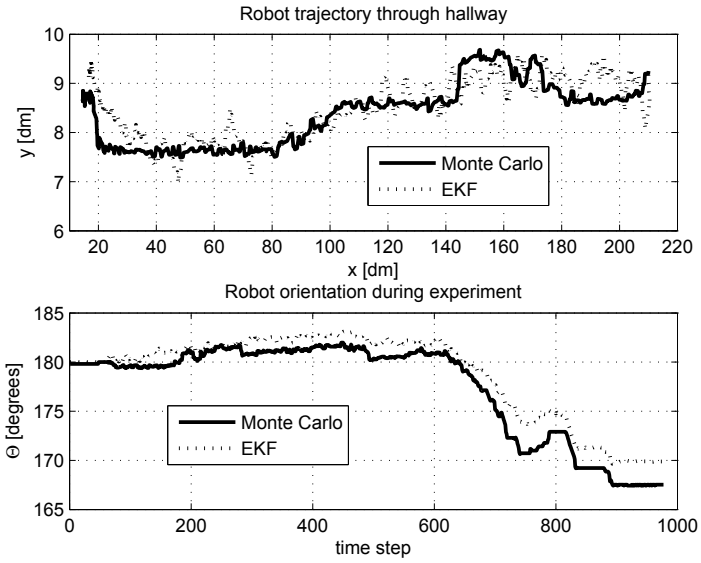


Fig. 16. Robot trajectory method comparison: solid - Monte Carlo localization, dots - EKF method.

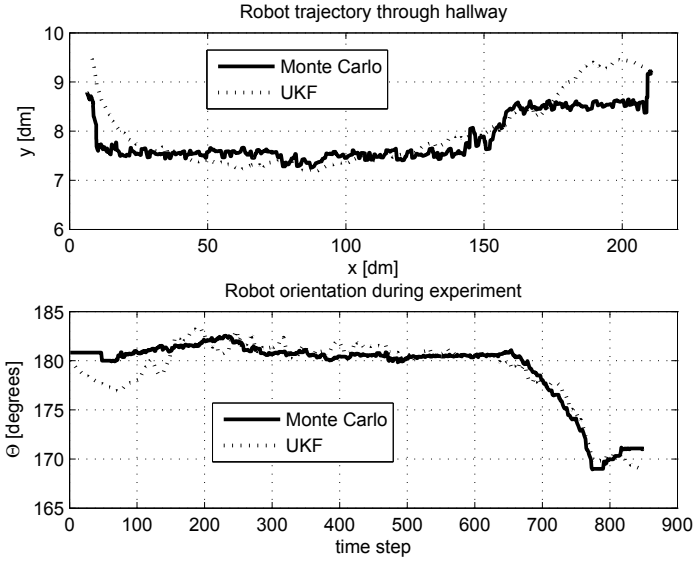


Fig. 17. Robot trajectory method comparison: solid - Monte Carlo localization, dots - UKF method.

6.3 Monocular-vision localization results

Similar as in the sonar localization experiment the mobile robot was given navigational commands to drive along a hallway and to collect odometry data and images from camera fixed to it at multiples of the discretization time. Figure 18 shows line segments superimposed to the camera view. Very good robustness to change of illumination and noise in camera image can be noticed. Figure 19 shows rendered hallway model superimposed to the camera view, before any correction was done. After image acquisition and model rendering, the off-line optimal matching of rendered image lines and lines extracted from the camera image was done. Obtained pairs and rendered model from corrected camera pose are shown in Fig. 20. Updated mobile robot start pose and its covariance was (orientation given in radians):

$$\mathbf{p_0} = \begin{bmatrix} 209.946 \text{ dm} & 9.2888 \text{ dm} & 3.1279 \end{bmatrix}^\top$$

Fig. 18. Superposition of camera view and line segments extracted by RWRHT method

Fig. 19. Superposition of camera view and rendered model before correction

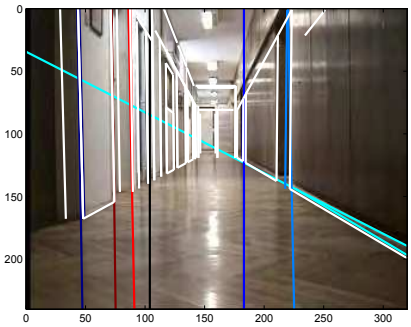


Fig. 20. Superposition of camera view and rendered model after correction. Line pairs that were matched and used as measurement are drawn with different colors for each pair.

$$\Sigma_{\mathbf{p_0}} = \begin{bmatrix} 0.2810 & -0.0332 & -0.0002 \\ -0.0332 & 0.2026 & -0.0034 \\ -0.0002 & -0.0034 & 0.0001 \end{bmatrix}$$

Complete trajectory compared to the Monte-Carlo trajectory is shown in Fig. 21. Almost identical results are obtained in orientation and little shift exists in position.

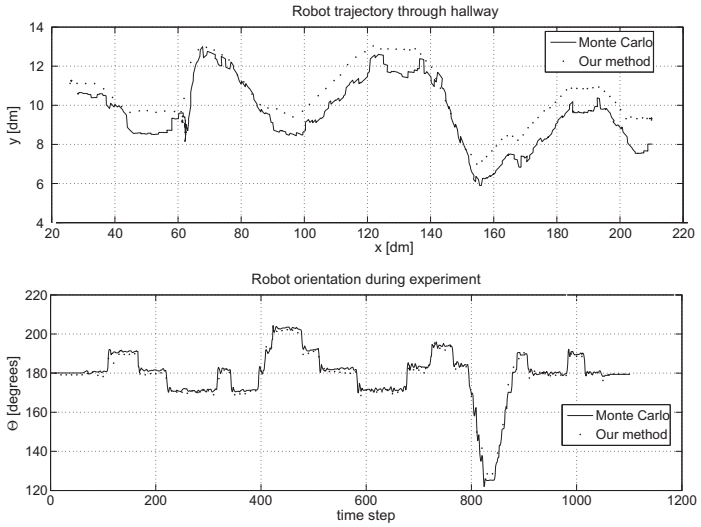


Fig. 21. Robot trajectory method comparison: solid - Monte Carlo localization, dots - our method

7. Conclusion

Monotonous position error growth is inherent characteristic of every mobile robot navigational system based solely on proprioceptive sensors. In order to deal with various sources of uncertainties in mobile robot localization it is necessary to establish a representative model of its internal states and environment and use perceptive sensors in the pose estimation. In this chapter we have demonstrated those properties on a differential drive mobile robot by localizing it in a 2D environment by using sonar ring as the perceptive sensor and in a 3D environment by using a mono camera as the perceptive sensor. In both cases we have applied nonlinear Kalman filtering for pose estimation and have compared results with the Monte Carlo localization based on a laser range finder, which is much more accurate sensor than sonars and cameras. Achieved localization accuracies with sonar ring and with mono camera are comparable to those obtained by the laser range finder and Monte Carlo localization. The applied calibration of mobile robot kinematic model also contributed to the increased accuracy.

8. References

*** (2000). *Optimization Toolbox For Use With Matlab User's Guide*, The MathWorks Inc.

*** (2007). ActivMedia robotics interface for application (ARIA): Overview, ActivMedia Robotics, LLC.

Aider, O. A., Hoppenot, P. & Colle, E. (2005). A model-based method for indoor mobile robot localization using monocular vision and straight-line correspondences, *Robotics and Autonomous Systems* **52**.

Arras, K. O., Tomatis, N., Jensen, B. & Siegwart, R. (2001). Multisensor on-the-fly localization: Precision and reliability for applications, *Robotics and Autonomous Systems* **34**.

Ashokaraj, I., Tsourdos, A., Silson, P. & White, B. (2004). Mobile robot localisation and navigation using multi-sensor fusion via interval analysis and ukf, *Proceedings of the 2004 Towards Autonomous Robotic Systems (TAROS), University of Essex,Colchester,UK* .

Borenstein, J., Everett, B. & Feng, L. (1996a). *Navigating Mobile Robots: Systems and Techniques.*, A. K. Peters, Ltd., Wellesley, MA, ISBN 1-56881-058-X.

Borenstein, J., Everett, H. R. & Feng, L. (1996b). *Where am I? Sensors and Methods for Mobile Robot Positioning*, University of Michigan, Ann Arbor, MI 48109.

Borenstein, J. & Feng, L. (1996). Measurement and correction of systematic odometry errors in mobile robots, *IEEE Transactions in Robotics and Automation* **12**(2).

Brezak, M., Petrović, I. & Ivanjko, E. (2008). Robust and accurate global vision system for real time tracking of multiple mobile robots, *Robotics and Autonomous Systems* **3**(56): 213–230.

Guilherme, N. D. & Avinash, C. K. (2002). Vision for mobile robot navigation: A survey, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(2): 237–267.

Haykin, S. (2001). *Kalman Filtering and Neural Networks*, John Wiley and Sons, chapter Ch. 7. The Unscented Kalman Filter.

Ivanjko, E., Komšić, I. & Petrović, I. (2007). Simple off-line odometry calibration of differential drive mobile robots, *Proceedings of 16th International Workshop on Robotics in Alpe-Adria-Danube Region*, Ljubljana, Slovenia, pp. 164–169.

Ivanjko, E., Petrović, I. & Perić, N. (2003). An approach to odometry calibration of differential drive mobile robots, *Proc. International Conference on Electrical Drives and Power Electronics EDPE'03*, The High Tatras, Slovakia, pp. 519–523.

Ivanjko, E., Petrović, I. & Vašak, M. (2004). Sonar-based pose tracking of indoor mobile robots, *AUTOMATIKA - časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije* **45**(3-4): 145–154.

Jeon, S. H. & Kim, B. K. (1999). Monocular-based position determination for indoor navigation of mobile robots, *IASTED Intl. Conf. on Control and Applications*, Banff, pp. 408–413.

Julier, S. J. & Uhlmann, J. K. (1996). A general method for approximating nonlinear transformations of probability distributions, *Technical report*, Dept. of Engineering Science, University of Oxford.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems, *Transactions of the ASME, Journal of Basic Engineering* **82**: 35–45.

Kälviäinen, H., Hirvonen, P., Xu, L. & Oja, E. (1994). Comparisons of Probabilistic and Non-probabilistic Hough Transforms, *Proc. of the 3rd European Conf. on Computer Vision*, Stockholm, Sweeden, pp. 351–360.

Konolige, K. (1999). Markov localization using correlation, *International Joint Conference on Artificial Intelligence*, Stockholm, Sweden.

Konolige, K. (2000). A gradient method for realtime robot control, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)*, Kagawa University, Takamatsu, Japan.

Kosaka, A. & Kak, A. C. (1992). Fast Vision-Guided Mobile Robot Navigation Using Model-Based Reasoning and Prediction of Uncertainties, *CVIPG: Image Understanding* **56**(3): 271–329.

Lee, D. (1996). *The Map-Building and Exploration Strategies of a Simple Sonar-Equipped Robot*, Cambridge University Press, Trumpington Street, Cambridge CB2 1RP.

Li, X. J., So, A. & Tso, S. K. (2002). CAD-Vision-Range-Based Self-Localization for Mobile Robot Using One landmark, *Journal of Intelligent and Robotic Systems* **35**.

Moravec, H. P. & Elfes, A. (1985). High resolution maps from wide angle sonar, *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, St. Louis, USA, pp. 116–121.

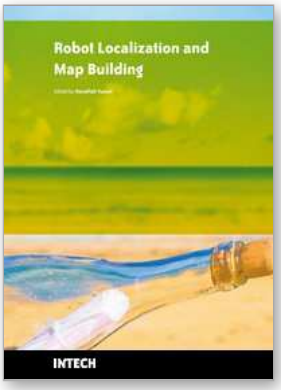
Neira, J., Ribeiro, M. I. & Tardós, J. D. (1997). Mobile Robot Localization and Map Building using Monocular Vision, *5th Int. Symp. on Intelligent Robotic Systems*, Stockholm, Sweden, pp. 275–284.

Tsai, R. (1987). A versatile camera calibration technique for high accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, *IEEE Journal of Robotics and Automation* **3**(4): 323–344.

Welch, G. & Bishop, G. (2000). An introduction to the Kalman filter, *Technical Report TR 95-041*, University of North Carolina at Chapel Hill, NC.

Wijk, O. (2001). *Triangulation Based Fusion of Sonar Data with Application in Mobile Robot Mapping and Localization*, PhD thesis, Royal Institute of Technology (KTH) Sweden, SE-100 44 Stockholm, Sweden.

www.blender3d.org (1995). *Blender Foundation* .



Robot Localization and Map Building

Edited by Hanafiah Yussof

ISBN 978-953-7619-83-1
Hard cover, 578 pages
Publisher InTech
Published online 01, March, 2010
Published in print edition March, 2010

Localization and mapping are the essence of successful navigation in mobile platform technology. Localization is a fundamental task in order to achieve high levels of autonomy in robot navigation and robustness in vehicle positioning. Robot localization and mapping is commonly related to cartography, combining science, technique and computation to build a trajectory map that reality can be modelled in ways that communicate spatial information effectively. This book describes comprehensive introduction, theories and applications related to localization, positioning and map building in mobile robot and autonomous vehicle platforms. It is organized in twenty seven chapters. Each chapter is rich with different degrees of details and approaches, supported by unique and actual resources that make it possible for readers to explore and learn the up to date knowledge in robot navigation technology. Understanding the theory and principles described in this book requires a multidisciplinary background of robotics, nonlinear system, sensor network, network engineering, computer science, physics, etc.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821