

Autonomous map building by mobile robots

Stephan ten Hagen Ben Kröse

RWCP, Autonomous Learning Functions Lab. SNN
Computer Science Institute, Faculty of Science
University of Amsterdam, The Netherlands

Abstract

In this paper we introduce a novel self-localization and map building approach. The new part is that we do not assume to have a correct motion model of the mobile robot. Our approach will improve the incorrect motion model online. We tested our approach in a simulation experiment.

1 Introduction

A useful mobile robot must know where it is with respect to some global model of the environment. The “where” usually refers to the state of the robot defined in it’s 3D configuration space: position and orientation. It is well known that the use of the motion model of the robot (based on wheel encoders) results in error on the long run because of wheel slip and model inaccuracies. Therefore information is needed from sensors observing the external world. Nearly all localization methods extract features or a partial model from the sensor data and compare that with the environment model in order to estimate the robot’s state. However, global environment models are not always available and the robot has to learn the environment. Here we encounter the problem that (unless a supervised training set is used as in [6]), the robot is not sure about its position while building the map. A number approaches has been presented dealing with the ‘SLAM’ (Simultaneous Localization and Map Building) problem [7, 2, 9, 8].

In this paper we introduce a novel framework for autonomous self localization and map building. New is that we want to correct the error in the motion model online using the estimated state based on the observations. We will assume that initially the motion model will give the most accurate estimates. After a while the estimations based on sensors can be better, if they can be related to observation made in the past when the estimates from the motion model were still accurate.

Our idea consist of three different steps. First the state estimates from the motion model are used to obtain a model that estimates the state based on the observations. Then these state estimates are used to reconstruct the past trajectory. Finally the new motion model is approximated based on the reconstructed trajectory. We will start by describing the notations and framework and present some solutions from literature. Then we will introduce our approach and test this in a simulation experiment.

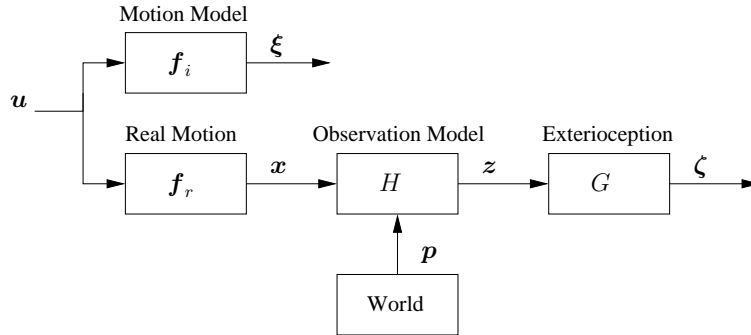


Figure 1. Schematic overview of robot localization. When control action \mathbf{u} is applied the robot’s real state \mathbf{x} and the estimated state from the motion model $\boldsymbol{\xi}$ change. The observation model maps the real state and the parameters \mathbf{p} of the environment to an observation \mathbf{z} . Based on \mathbf{z} the state is estimated resulting in $\boldsymbol{\zeta}$.

2 Simultaneous Localization and Map Building

Figure 1 shows the components of a robot localization task. The robot has an internal motion model f_i which gives an estimate of the new state $\boldsymbol{\xi}_{k+1}$ given the action \mathbf{u}_k and the old state $\boldsymbol{\xi}_k$. This model is often inaccurate and the deviation from the *real* state \mathbf{x} (with $\mathbf{x}^T = [x \ y \ \phi]$) will accumulate with time. The robot’s sensors produce a sensory signal \mathbf{z} , which is a function of the real state \mathbf{x} and the parameters \mathbf{p} of the world. The mapping G is the inverse sensor model: it gives an estimate $\boldsymbol{\zeta}$ of the state of the robot from the observation. If the observation model H is known, the uncertainty in $\boldsymbol{\xi}$ can be reduced using the observations by a *Kalman* filter. Often the H is not known and must be estimated online.

2.1 Kalman filter approaches for SLAM

Early approaches for SLAM were based on incremental parameter estimation techniques as the Kalman filter. The environment was represented as a parametric model, where the parameters \mathbf{p}_i indicated for example characteristics of line segments or beacon positions. Based on (re-)observations of such geometric features with, both the parameters of the features and the state of the robot could be estimated [2]. Later it was shown that the relation between the vehicle state and a geometric feature could not be propagated independently and the full covariance matrix has to be maintained [1]. Recently [4] it was proven that even the full-covariance SLAM algorithm always yields an inconsistent map. The characteristics of the Kalman filter approaches is that the motion model has to be known and these approaches also dependent on a good initialization.

2.2 Consistent Pose Estimation approaches

Another approach, the ‘Consistent Pose Estimation’ method [8, 5], is not based on an incremental estimation procedure but keeps sensory observations over a

longer period to avoid inconsistencies. Based on pairs of observations, the *relative* positions of the robot are derived. From the network of relative positions the absolute positions are derived by minimizing the mismatch between the aligned sensor data. This method does not need a process model, but if such a model is available it can be incorporated in the framework as an alternative for the matching of sensor data. Range scans as sensor data makes it possible to obtain a relative accurate estimate of the relative positions [8, 3].

3 Our approach

In the Kalman SLAM approach the parameters of the observation model are estimated simultaneous with the robot’s state. The Kalman filter operates on an augmented state vector consisting of the robot’s state ξ and the parameter vector p . The motion model is assumed to be known and correct. In our approach we do not make this assumption. We improve the motion model, using the observations from the sensors by first estimating the (inverse) observation model G and then use this model to update the motion model.

The key idea to our approach is that we assume the robot to arrive at locations it has visited in the past. In these locations the state estimation ξ may be very inaccurate, but the observation z made at this location will be similar to those made in the past. If an approximation of G is based on “old” estimates ξ that are still relatively accurate, then this model will give a more accurate state estimation than the motion model. Based on the more accurate estimate of the final state we can reconstruct a more accurate ‘past trajectory’. We can apply system identification on the reconstructed trajectory to come up with a motion model that is more in accordance with the motion model of the real robot.

3.1 The mapping G

The mapping G maps the observation z to the state estimate ζ :

$$\zeta_k = G(z_k). \tag{1}$$

Ideally this function should be the inverse function of H , which we assume to be unknown. Rather than trying to estimate the parameters of H and compute the inverse, we take a function approximator to represent G . To obtain the function G we need a “supervised” train set containing observations and state estimates and we also have to choose an approximator.

Initially only ξ is available to obtain the approximation. We will therefore create a data set containing all values of ξ together with the corresponding observations. We use lazy learning, so at each time step the approximation of G is made based on the available examples in the data set. In this paper we use a local linear approximation of G . This means that the estimation ζ is given by

$$\zeta_k = \hat{G}_k z_k, \tag{2}$$

with matrix \hat{G}_k as the estimated parameters of G at time step k .

The matrix \hat{G}_k is computed using a least squares estimation based on N entries in the data set. We have to determine which elements of the data set are used for the estimation of \hat{G}_k . The estimates ξ become more and more unreliable over time. So the estimation of ζ becomes more unreliable if only the most recent estimates ξ are used to obtain \hat{G}_k . We therefore include the time in our selection criterion. We look within a radius ρ around ξ_k and select the least recent entries in the data set. These will be the “oldest” values of ξ , which are more reliable than the most recent values.

To see how this selection mechanism can improve the estimate of ζ , consider that the robot makes a loop. Initially the robot will visit new locations and the estimate will totally depend on the latest values of ξ . At the end of the loop also the initial estimates ξ will be close to ξ_k , and they will be selected to obtain \hat{G} . The error due to drift in the initial values of ξ are smaller than the error in ξ_k . The resulting estimate of ζ_k using (2) will be more accurate than ξ_k .

3.2 Trajectory reconstruction

More accurate estimates ζ do not mean that a more accurate motion model can be obtained. The reliability of the approximation of the new motion model does not depend on the quality of the estimation of the state, but on the estimated change in state. Small inaccuracies in ζ makes it hard to estimate the state change.

We will not use ζ directly for the approximation of the motion model, but use this to reconstruct the past trajectory. Assume that ζ_k is correct and represents the correct value of the *end* state. Also the initial state is known correctly, so the begin and end state of the true trajectory are known.

We can use backward dead reckoning from the end state using the inverse of the motion model, resulting in state estimates β . In the beginning estimates of ξ are most accurate and near the end the estimates of β . The reconstructed trajectory is obtained by combining the forward and backward trajectories according to:

$$\tau_i = \alpha_i \beta_i + (1 - \alpha_i) \xi_i. \quad (3)$$

The value of α is linearly increasing from $\alpha_0 = 0$ to $\alpha_k = 1$.

3.3 The motion model

The ideal motion model maps the current state estimate and the control action to the next state, so:

$$\xi_{k+1} = f_i(\xi_k, \mathbf{u}_k). \quad (4)$$

In this paper we will assume that the robot has a motion model according to:¹

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_{i,k}) = \mathbf{x}_k + \Delta \mathbf{x}_k \quad (5)$$

with

$$\Delta \mathbf{x}_k = \begin{bmatrix} \frac{v_{i,k}}{w_{i,k}} (\sin(\phi_k + Tw_{i,k}) - \sin(\phi_k)) \\ \frac{v_{i,k}}{w_{i,k}} (\cos(\phi_k) - \cos(\phi_k + Tw_{i,k})) \\ Tw_{i,k} \end{bmatrix}. \quad (6)$$

¹This is a simplified model of the motion of a Nomad Scout II.

We take this as the ideal motion model \mathbf{f}_i of the robot, where $\mathbf{u}_{i,k}^T = [v_{i,k} \ w_{i,k}]$ indicates the ideal control action. The $v_{i,k}$ is the traversal speed and the $w_{i,k}$ is the rotational speed. For the real robot the size of the wheels or the distance between the wheel can differ from the ideal model. This can lead to a difference in rotational and traversal speed. We will model this as:

$$\mathbf{u}_i = L\mathbf{u}. \quad (7)$$

In \mathbf{f}_i the L is the unity matrix while in the true robot this can be different. The task of approximating \mathbf{f}_r is reduced to estimating the correct value of L .

4 Simulation Experiment

4.1 The setup

The robot moves according to the model given in (6). For L in (7) we took

$$L = \begin{bmatrix} 1 & 0 \\ 0 & 1.1 \end{bmatrix}, \quad (8)$$

so the robot rotated 10 percent more than it should according to the motion model. The robot has a set of range scanners that are mounted at the center on top of the robot. We took only 3 range sensors to reduce the number of data points required to estimate the matrix \hat{G} . The orientation of the sensor were $-\frac{2}{3}\pi$, 0 and $\frac{2}{3}\pi$. The robot navigates in an environment of four corridors. It takes 24 steps forward and then turn in one step $\frac{1}{2}\pi$ to the left. This is repeated so that the ideal robot will drive around all corridors and then repeat the lap.

Because of L , the real robot turns too much at the corners and we had to compensate this by hand. Also we added a little noise to the control action to make sure that the true states and observations are not identical in each lap. Due to the compensation the trajectory according to the motion model turns slightly to the left, when the robot is supposed to make a forward motion. This led to errors in the state estimation based on the motion model. Figure 2(a) shows the environment with the real and estimated trajectory for the first two laps.

4.2 The estimation of ζ_k

The estimates ζ_k were made according to (2), where \hat{G}_k had 9 elements because of the 3 sensors. To estimate \hat{G}_k we used $N = 18$ entries from the data set. At each time step k we selected the entries within a distance $\rho = 13$ and used the 18 “oldest” values to determine \hat{G}_k . We chose $\rho = 13$ so that always less than 18 entries were within the radius during the initial lap. No ζ_k could be estimated, so we took ξ as estimation for ζ . In the second lap the 18 entries were combinations of entries of the first and second lap and always estimations could be made.

Figure 2(b) shows the error of ξ and ζ with respect to the true state value \mathbf{x} . We see that at time steps where the robot turns, the error in ζ is often very large. In these situations the estimates of \hat{G}_k were based on points from the first and

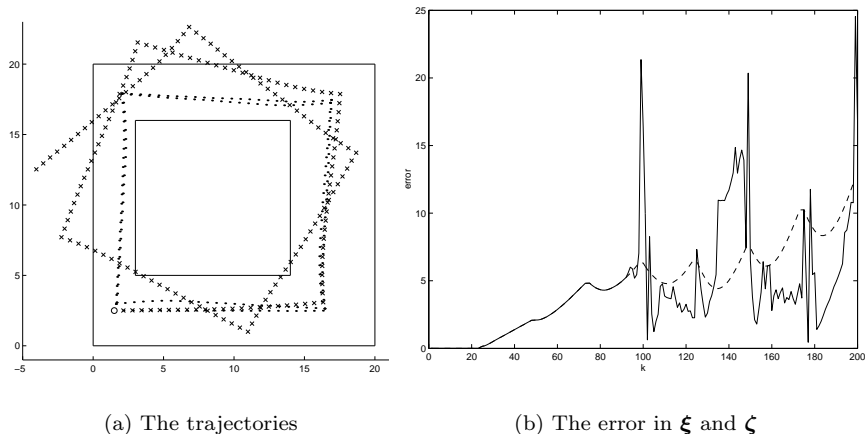


Figure 2. The true trajectories of the robot (indicated by dots) and the estimated trajectory by the motion model (indicated by crosses). The circle indicates the starting state $[1.5, 2.5, 0.0]$ and the robot initially moves to the right. The error indicates $\|\zeta_k - \mathbf{x}_k\|_2$ (solid) and $\|\xi_k - \mathbf{x}_k\|_2$ (dashed) for all time steps. In the first lap both errors are the same.

second lap, resulting in very unreliable estimates of \hat{G}_k . The high errors between $k = 130$ and $k = 150$ was due to the switching in orientation between $-\pi$ and π , which resulted in very inaccurate estimates for \hat{G}_k .

We also see that the error in ξ becomes larger at the end and that the error in ζ is often smaller than that of ξ . In the second half of the second lap many estimates ζ are closer to the true state value than the ξ in the first lap. This shows that the second lap estimates of ξ are not just replaced by those from the first lap. The estimates are improved based on the observations.

4.3 The estimation of \bar{L} using a perfect reconstruction

We tested the reconstruction and estimation of L for the situation that ζ_k has the correct value of \mathbf{x}_k at the end of the second lap. Figure 3(a) shows state estimates β computed by starting at the end. Also shown is the reconstructed trajectory τ whose “shape” resembles that of the true trajectory in figure 2(a).

The state changes in the reconstructed trajectory were used in (6) to compute the *ideal* control actions \mathbf{u}_i . Because of (7) a linear least squares estimate of L can be computed using the ideal and true sequence of control actions. This resulted in:

$$L = \begin{bmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \end{bmatrix} = \begin{bmatrix} 0.9615 & 0.0090 \\ 0.0591 & 1.0073 \end{bmatrix}. \quad (9)$$

Our reconstruction cannot compensate for scale, so we normalized L :

$$\bar{L} = \frac{1}{l_{11}}L = \begin{bmatrix} 1.0000 & 0.0093 \\ 0.0614 & 1.0476 \end{bmatrix}. \quad (10)$$

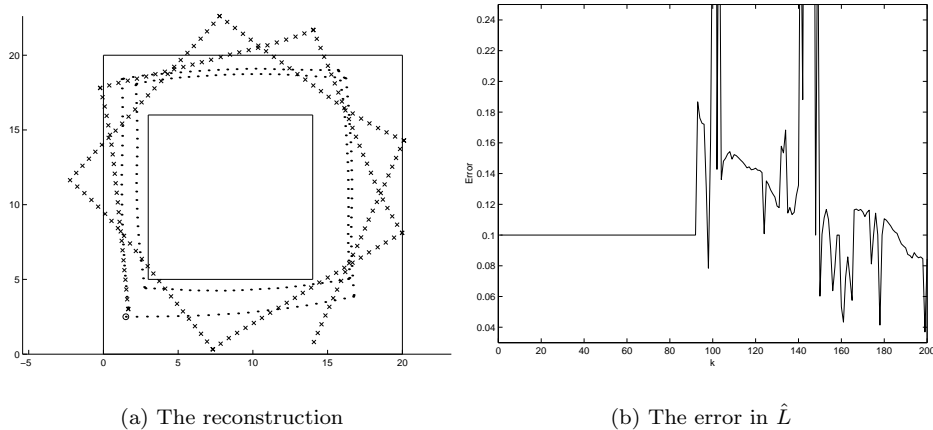


Figure 3. **a)** The backward trajectories β (indicated by the crosses) and the reconstructed trajectory (indicated by the dots). **b)** The error indicates the Euclidean distance between the \bar{L}_k and L .

We see that the value of \bar{l}_{22} does not equal 1.1. Because the begin and end of the trajectory τ resembles that of ξ and β , the resulting \bar{L} became the average estimate of the L of the ideal and true motion model. So we found $\bar{l}_{22} \approx 1.05$. When we only used values from $k = 50$ to $k = 150$ we found $\bar{L}_{22} = 1.09$, which is already closer to 1.1. We can conclude that when ζ_k is correct our approach is able to approximate the true motion model.

4.4 Estimation of \bar{L} using reconstructions based on ζ_k .

We used all values ζ_k in figure 2(b) as end point to reconstruct the past trajectory. For all these past trajectories we computed the value of \bar{L}_k . Because the length of each trajectory was different, we did not remove the first and last 50 time steps. The means that the resulting \bar{L}_k approximated the average L of the true and ideal motion model. However, in the first lap $\zeta = \xi$ and therefore τ was equal to ξ , so that the resulting \bar{L}_k was the same as the L in (8).

In figure 3(b) the error in the \bar{L}_k is shown for all time steps. In the beginning of the second lap we seen that the error is larger, even when the ζ is closer to x than ξ . This indicates that the reconstruction of the trajectory should be long enough to have effect. In the second half of the second lap we see that the error in \bar{L} becomes lower, so that \bar{L} starts to resemble the average L of the true and ideal model. So when the trajectory is long enough the motion model is improved.

5 Conclusion

In this paper we introduced a novel approach to combine self localization with map building. Unlike Kalman filter approaches, we do not assume that a perfect motion model is available. In our approach we assume that estimations made with an (possible) incorrect motion model are more accurate in the beginning than later on. By recognizing that the robot has visited a certain location before, the observation can be related to observations in the past to come to a more accurate estimation of the current state. This state estimate can be used as a starting point for backward dead reckoning using the inverse of the motion model. With this we can reconstruct the past trajectory and use this to get an improved approximation of the motion model of the robot.

References

- [1] P. Cheeseman, R.C. Smith, and M. Self. Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*, pages 167–193. Springer-Verlag, 1990.
- [2] J.L. Crowley. World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 1989.
- [3] J.S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *International Symposium on Computational Intelligence in Robotics and Automation (CIRA '99)*, 1999.
- [4] S.J. Julier and J.K. Uhlmann. A counter example to the theory of simultaneous localization and map building. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 4238–4243, 2001.
- [5] K. Konolige and K. Chou. Markov localization using correlation. In *Proc. International Joint Conference on Artificial Intelligence*, pages 1154–1159. Morgan Kaufmann, 1999.
- [6] B.J.A. Kröse, N. Vlassis, R. Bunschoten, and Y. Motomura. A probabilistic model for appearance-based robot localization. *Image and Vision Computing*, 19(6):381–391, April 2001.
- [7] J. Leonard, H. Durrant-Whyte, and I.J. Cox. Dynamic map building for an autonomous mobile robot. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 1990.
- [8] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.
- [9] S. Thrun, W. Burgard, and F. Dieter. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29–53, 1998.