in: Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005, LNCS 3697, Part II, pp. 929-934, Springer 2005

Neural Architecture for Concurrent Map Building and Localization using Adaptive Appearance Maps[*]

St. Mueller, A. Koenig, and H.-M. Gross

Department of Neuroinformatics and Cognitive Robotics
Ilmenau Technical University
98684 Ilmenau, Germany
Steffen.Mueller@tu-ilmenau.de

Abstract. This paper describes a novel omnivision-based Concurrent Map-building and Localization (CML) approach which is able to localize a mobile robot in complex and dynamic environments. The approach extends or improves known CML techniques in essential aspects. For example, a more flexible model of the environment is used to represent experienced observations. By applying an improved learning regime, observations which are not longer of importance for the localization task are actively forgotten to limit complexity. Furthermore, a generalized scheme for hypotheses fusion is presented that enables the integration of further multi-sensory position estimators.

1 Introduction

Robust self-localization plays a central role in our long-term research project PERSES (PERsonal SErvice System) which aims to develop an interactive mobile shopping assistant which can autonomously guide its user within a home store [1]. To accommodate the challenges that arise from the specifics of this scenario and the characteristics of the operation area, a regularly structured, maze-like and populated environment, we placed special emphasis on vision-based methods for robot navigation. In our previous approach [1], we have employed a static graph representation as map of the environment, which is build up manually. The nodes of the graph are labeled with visual observations extracted from omnidirectional images and corresponding position information. Given this map, localization was realized employing a Particle Filter to estimate the robot's state. The main drawback of this and other appearance-based approaches for localization published in recent years is, however, that localization is only possible in manually mapped areas. Furthermore, the learned map is only valid as far as no important modifications of the operation area occur. Therefore, we developed an alternative technique which is able to perform an omnivision-based Concurrent Map-building and Localization (CML) to overcome this drawback. Inspired

[*] This work is partially supported by TMWFK-Grant # B509-03007 to H.-M. Gross

by former approaches like [5] but especially the work of Porta and Kroese [2] and continuing our former work, we present a neural architecture (see Fig. 1), which is able to track multiple state hypotheses (position and orientation of a mobile robot) in a short-term memory (STM) using odometry data and previous state estimations, while building up a kind of long-term memory (LTM) used for associating omnidirectional views to already observed and learned states. This appearance map afterwards directly influences the tracked state hypotheses in the STM to reduce their uncertainty.

Main advantage of this approach is the advanced learning scheme used in the LTM. The network is able to actively forget information about observations that became irrelevant because of changes in the environment. This guarantees that the complexity remains limited for a given operation area and independent from working time, which is of fundamental importance for a continuous duty.

2 Neural Architecture for Probabilistic Localization

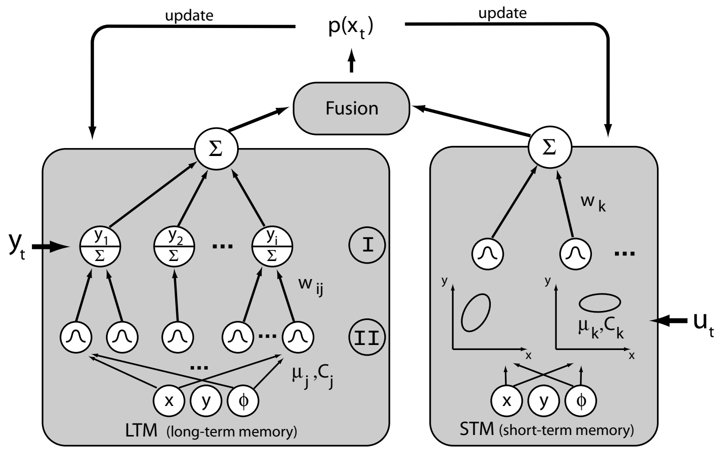

Fig. 1. Architcture of our probabilistic localization system: last known state hypothesis (position x,y and orientation ϕ of the robot) from STM (right) and a hypothesis from LTM (left) resulting from current observation $\boldsymbol{y}_t$ become merged and approximated by a Mixture of Gaussians $p(\boldsymbol{x}_t)$. Afterwards, this resulting distribution (top) is used to adapt STM again and to teach the observation-state associations in LTM. During the next step, hypotheses in STM will be updated using odometry data $\boldsymbol{u}_t$ and a motion model, then the output $p(\boldsymbol{x}_t)$ can be estimated again.

Our architecture consists of three main components, the short-term memory (STM), the long-term memory (LTM), and the fusion subsystem shown in Fig. 1. The STM is responsible for representing the distribution of possible states the robot might currently be in. By placing linear RBF-neurons in the *State Space* S (x, y, ϕ) and summing up their weighted outputs, this structure represents a Mixture of Gaussians (MoG) characterizing one hypothesis for the current state estimation. The resulting activity h^{STM} at the STM-output node can be determined as follows:

$$h^{STM}(\boldsymbol{x}_t) = \sum_k w_k \phi(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{C}_k) \tag{1}$$

whereby ϕ is a Gaussian with mean $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{C}_k$, and w_k is the weight of the respective connection to the output node. The LTM (Fig. 1, left) consists of a layer of nodes representing prototypes $\boldsymbol{y}_i$ of observations and performing a clustering of the *Observation Space*. Each node in this layer I receives the current observation $\boldsymbol{y}_t$ and a weighted sum of activity from layer II, which consists of linear RBF-neurons connected to exactly one prototype node of layer I. Experiments showed that connections to more than one prototype nodes destabilize the state estimation. While layer II nodes are representing positions in *State Space*, layer I combines them considering the similarity between the respective reference observation $\boldsymbol{y}_i$ and the current observation $\boldsymbol{y}_t$, whereby $S(\boldsymbol{y}_t, \boldsymbol{y}_i)$ is a similarity function delivering a maximum (1.0) for identical views and decreasing continuously to zero up to a minimum similarity. As a result, the activity h^{LTM} at the LTM-output node is given by:

$$h^{LTM}(\boldsymbol{x}_t|\boldsymbol{y}_t) = \sum_i (S(\boldsymbol{y}_t, \boldsymbol{y}_i) \cdot \sum_j w_{ij} \phi(\boldsymbol{x}|\boldsymbol{\mu}_j, \boldsymbol{C}_j)) \tag{2}$$

The LTM-output node integrates the activation over all reference nodes, such that the resulting output characterizes the distribution of possible states under the given observation. The sum of activation characterizes the certainty of this hypothesis resulting from more or less similarity between observation $\boldsymbol{y}_t$ and the learned prototypes. Concerning this, the output is not a true probability distribution because weights do not sum up to one.

The last component, that receives the two hypotheses h^{STM} and h^{LTM}, is responsible for their fusion. In this module, a kind of probabilistic inference takes place, which leads to a probability distribution of the robot's current state.

2.1 Fusion of Hypotheses

The fusion module has to evaluate the activity distribution of different sources of information in the *State Space*, in the case shown here of $h^{LTM}(\boldsymbol{x}_t|\boldsymbol{y}_t)$ and $h^{STM}(\boldsymbol{x}_t)$, but hypotheses from further state estimators can be integrated. To simplify the fusion process, inputs are given in form of a weighted sum of Gaussians, whereas different to a mixture probability the sum of the weights w^i needs not to be one. First, in this pool of Gaussians one has to decide which Gaussians are representing the same hypothesis. Therefore, a spatial distance criterion is applied, similar to [2] the Mahalanobis distance is employed. So the inference can realize a logic AND for all the combinations of Gaussians within a maximum spatial distance. This is done by *Covariance Intersection* similar to [2] and [4]. However, in our approach the weights w^i are explicitly considered to take the reliability of the different Gaussians into account. Gaussians that have no corresponding counterpart, are taken into account in form of a logic OR. This way, single hypothesis can be transfered into the resulting set of Gaussians, too. Final step is to normalize the weights such that the weighted sum can be interpreted as a probability distribution $p(\boldsymbol{x}_t)$. Further on, the resulting MoG can be simplified if two or more Gaussians resemble each other. This is done by approximating the overlapping Gaussians by a single one. Also components with too small weights can be removed. At this point, we want to place emphasis on the necessity of the inference realizing an AND. Without the reduction of uncertainty by means of Covariance Intersection, a convergence of the whole model cannot be forced and variances of the participating Gaussians would grow over time.

2.2 Short-term Memory (STM)

Main part for tracking the state hypotheses is the STM. After computation of the localization distribution $p(\boldsymbol{x}_t)$, the weights and parameters of the RBF nodes in the STM have to be adapted to represent the new hypothesis. This is done by transferring the weights of the MoG $p(\boldsymbol{x}_t)$ to w_k and setting up the mean values $\boldsymbol{\mu}_k$ and covariance matrices $\boldsymbol{C}_k$ according to the MoG components, while the number of nodes is adapted to the number of components in $p(\boldsymbol{x}_t)$. An other kind of STM-update takes place if a motion $\boldsymbol{u}_t$ is measured by odometry. Then a motion model is applied to each partial hypothesis represented by one RBF node. This results in new parameters $\boldsymbol{\mu}_k$ and $\boldsymbol{C}_k$. Concurrently a new visual observation $\boldsymbol{y}_t$ is captured and a new estimation of $p(\boldsymbol{x}_t)$ will be initiated.

2.3 Long-term Memory (LTM) - Adaptive Environment Model

The LTM is performing a mapping from observations $\boldsymbol{y}_t$ to a distribution of states the robot has already been in while receiving a similar observation. Unlike to our former model [1], this mapping is learned and adapted online while using it for localization. Therefore, pairs of observation $\boldsymbol{y}_t$ and related estimated state hypotheses $p(\boldsymbol{x}_t)$ serve as teach value. To speed up convergence and to reduce faulty entries in the LTM, principles similar to [2] are employed. So an update takes place only if $p(\boldsymbol{x}_t)$ is unimodal. In all other cases, the updates will be delayed until $p(\boldsymbol{x}_t)$ reaches unimodality again. Then disambiguated former positions can be reconstructed by using stored motion information (see [3]). Once given an update request, the structure and parameters of LTM are changed in three steps.

First, the clustering of *Observation Space* in layer I is updated. Therefore, if similarity of $\boldsymbol{y}_t$ to each prototype $\boldsymbol{y}_i$ falls below a threshold, a new node representing the current observation $\boldsymbol{y}_t$ is inserted. During this operation, similarities $S(\boldsymbol{y}_t, \boldsymbol{y}_i)$ of all layer I prototypes have to be computed.

Second step: In this phase, the parameters of the RBF nodes in layer II are updated. For that, first the output (merged hypotheses) $p(\boldsymbol{x}_t)$ is back-propagated to each RBF node by multiplying the weights of the MoG components by $S(\boldsymbol{y}_t, \boldsymbol{y}_i)$ according to that prototype $\boldsymbol{y}_i$ the layer II node is connected to. If this is done, a single Gaussian $\phi(\boldsymbol{x}|\boldsymbol{\mu}_t, \boldsymbol{C}_t)$ with a weight $w_t = S(\boldsymbol{y}_t, \boldsymbol{y}_i)\, w_{p(x_t)}$ is given for updating all layer II nodes that are connected to the prototype node $\boldsymbol{y}_i$.

This update is done by introducing a new RBF node, representing the new observation. Finaly, nodes with nearly similar Gaussians become merged, to reduce redundancy.

Third step: In this step the connections w_{ij} from layer II to layer I are adapted. Here, relevanceweights w_{ij} of layer II hypotheses will be increased with a learning rate β if layer I is activated by a high similarity $S(\boldsymbol{y}_t, \boldsymbol{y}_i)$ and layer II is activated by a low spatial distance to the Gaussian in $p(\boldsymbol{x}_t)$.

$$w_{ij} := \beta\, S(\boldsymbol{y}_i, \boldsymbol{y}_t)\, w_{p(x_t)} + (1 - \beta\, S(\boldsymbol{y}_i, \boldsymbol{y}_t)\, w_{p(x_t)})\, w_{ij} \tag{3}$$

To reach a stabilizing behavior (and for solving the kidnapped robot problem while building up the internal representation), on the other hand connections to inactive layer II nodes have to be reduced if the respective layer I node is activated.

$$w_{ij} := (1 - \beta\, S(\boldsymbol{y}_i, \boldsymbol{y}_t)\, w_{p(x_t)})\, w_{ij} \tag{4}$$

So long, only new information were captured and the complexity of the LTM increases continuously. But it is also necessary to delete information, because the operation area is extremely dynamic. So situations that will not be observed again can be forgotten, if there is a new observation at the same position. For this purpose, similarity $D(t, j)$ in *State Space* between $p(\boldsymbol{x}_t)$ and the Gaussian represented by each RBF node has to be evaluated. So connections from activated RBF nodes to deactivated prototype nodes in layer I will be decreased,

$$w_{ij} := (1 - f\,(D(t, j), |\boldsymbol{C}_t|))\ w_{ij} \tag{5}$$

and if these weights reach a lower bound, the respective RBF node can be deleted. If no layer II node is connected any longer to a certain prototype node in layer I, this prototype node is deleted, too. The forgetting function f decreases with growing spatial distance D and growing variance of the new Gaussian, which is contained in the determinant of its covariance matrix. Only by means of this third rule, a limitation of the number of nodes, responsible for a restricted area, can be reached.

3 Experimental Results and Conclusion

First, the algorithm was analyzed in a part of the home store with low changes and dynamic modifications. In these preliminary experiments, a mean localization error of about $0.6m$ in an area of about $25m$ by $10m$ could be reached. Observable was a localization error growing with distance to the initial position. The reason for this behavior is the erroneous odometry data used during the first lap for building the initial model. So the LTM represents correct spatial relations of the world in an internal coordinate system, which typically can be rotated to world space. Binding the model at absolute world coordinates is a general problem of this class of CML approaches.

In our desired application, the main task is not to build a model of a completely unknown area but to continuously adapt the model learned before to a changing environment, so this problem is secondary. Therefore, further long-term experiments were done in the home store. After building up an initial model similar to the first experiment, the representation in LTM was rotated and translated to fit the absolute world coordinates by means of minimizing the error between the true path and the estimation of the localization system. Afterwards the experiment was continued for several days. The result is a model of a $30m$ by $30m$ area that allows a localization with an average absolute error of less than $0.45m$, built up without any a priori information. The long-term experiments also clearly demonstrates the merits of our model. Using a model similar to the one presented in [2], the number of layer I nodes in LTM was growing continuously as long as the environment changed. The method presented here handles the situation by replacing irrelevant prototype views by new ones, finally leading to a limited number of nodes for this restricted operation area. The presented approach, thus realizes an applicable long-term localization in a continuously changing environment based on an adaptive statistical distribution with different time-scales.

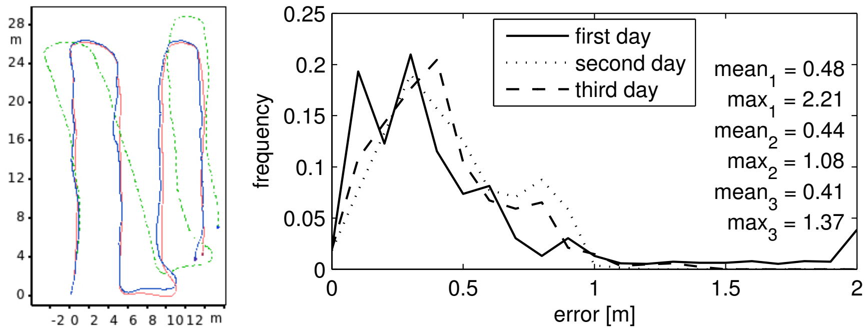

Fig. 2. Results of the long-term experiment in the home store: localization test after three days of operation (left): real path (red/grey), estimated path (blue/solid) and odometric data (green/dotted), histograms of the localization error for three trials (right) the development of means and the rising concentration on small errors visualize the convergence of the approach

References

1. H.-M. Gross, A. Koenig, Chr. Schroeter and H.-J. Boehme, "Omnivision-based Probabilistic Self-localization for a Mobile Shopping Assistant Continued", in: Proc. IEEE-IROS 2003, pp. 1505-1511
2. J. M. Porta and B. J.A. Kroese, "Appearance-based Concurrent Map Building and Localization using a Multi-Hypotheses Tracker", in: Proc. IEEE-IROS 2004, pp. 3424-3429
3. S. H. G. ten Hagen and B. J. A. Kroese, "Trajectory reconstruction for self-localization and map building", in: Proc. IEEE-ICRA 2002, pp. 1796-1801
4. J. K. Uhlmann, S. Julier and M. Csorba,"Nondivergent Simultaneous Map Building and Localization using Covariance Intersection",in Proc. of the SPIE Aerosense Conference, 3087, 4/1997
5. T. Duckett and U. Nehmzow, "Experiments in Evidence-Based Localisation for a Mobile Robot", in AISB-97, Technical Report Series UMCS-97-4-1,1996