Euromicro ´94 – Realtime-Workshop
Vaesteraas (Västerås), Sweden, June 15-17, '94

Realtime-learning on an Autonomous Mobile Robot with Neural Networks

Uwe R. Zimmer & Ewald von Puttkamer

University of Kaiserslautern - Computer Science Department - Research Group Prof. E. v. Puttkamer
P.O. Box 3049 - 67663 Kaiserslautern - Germany
Phone: …49 631 205 2624 - Fax: …49 631 205 2803 - Telex: 4 5627 unikl d
e-mail: uzimmer@informatik.uni-kl.de

The problem to be discussed here, is the usage of neural network clustering techniques on a mobile robot, in order to build qualitative topologic environment maps. This has to be done in realtime, i.e. the internal world-model has to be adapted by the flow of sensor-samples without the possibility to stop this data-flow. Our experiments are done in a simulation environment as well as on a robot, called ALICE.

1. Realtime-learning

Due to the fact that the term “Realtime-learning” is not really well defined, we have to specify it first. In our context the robot has to “learn” a representation of it's actual environment (here called “qualitative topologic map”, QTM). The process of adaptation has to be fulfilled in a fixed amount of time (here: less than the time between two sensor-samples), the sensor-sampling rate being determined by the maximum speed of the mobile system. Our robot is in the actual version able to perform a speed of 25 cm / s. So a reasonable sampling-frequency might be two complete samples of all sensors within one second. The sensors used on ALICE are passive light sensors, whiskers and simple internal motor revolution counting for dead reckoning.

2. Qualitative topologic maps

Qualitative topologic maps (QTM) are an alternative to exact geometric models of the environment (see e.g. [3] as an example of exact geometric mapping based on the data of a laser range finder). Instead of modelling the boundaries of the detected objects, only the sensor-information itself is used to build a “map of sensor-impressions” directly. This concept has already being proposed by Kuipers et al. [1], but the construction process was done by explicit rules, i.e. not by statistical techniques. Our map is constructed by special clustering techniques based on Kohonen's Self-Organizing-Maps (described in the next chapter). QTMs are able to represent significant different sensor-situations and their neighbouring relationships. Therefore similar sensor-situations (“similar” to a representative constellation found in the QTM) can be detected. So the first basic task for each autonomous mobile system can be fulfilled: “Recognize places, where you have been before (without getting exactly the same sensor-measurements)”. This task may be summarized by “Qualitative Recognition”. On the other hand it is not possible to get positions from this map for an exact recallibration of the internal position. Nevertheless it is possible to correlate the internal position within the boundaries of the granularity of the distinct represented sensor-situations.

3. Dynamic neural networks

The underlying neural network model is basically a Self-Organizing-Map in the variation by Fritzke called Growing-Cell-Structures [2] and with several adaptations and extensions described in [5]. The main idea is to use the neighbourhood connections in a dynamic neural network as a representation of topologic neighbourhood in the environment. The network-processes have to be adapted in a way to avoid not reasonable connections in the topologic interpretation, although these connections would make sense, if they are only interpreted by the processes for growing and shrinking the network.

4. Project: ALICE

The ALICE-Project consists of two major parts: The real robot and an equivalent simulation environment. With the help of the operating- and communication-system ALBATROSS [6] we are able to use the same object code on both shells, avoiding cross-compilers and other tools.

The mobile robot ALICE is a round (40 cm in diameter, 20 cm high), fully autonomous platform with an omnidirectional kinematic. 24 whisker-light-sensor pairs are distributed symmetrically at the border of the vehicle and three light-sensors are directed towards the ceiling. Also it is possible to determine a rough internal position by dead-reckoning.

The software consists of three major processes:

- *Map-building:*
  Finding an adequate internal representation of the environment as seen by the sensors. Here topology-preserving neural network structures are used as the basic concept.
- *Navigation & Pilot:*
  Finding a way (path) to a specified place in the known (mapped) area of the environment and drive along this path, until the robot reaches the target-place or detects a situation which prevents this plan.

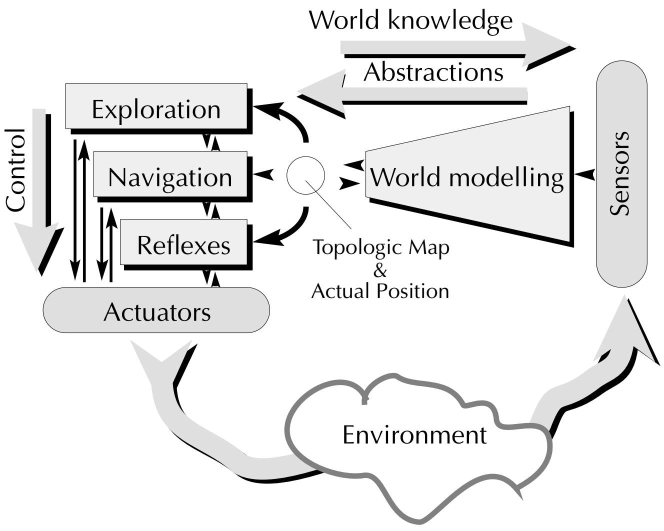

figure 1 : ALICE main-structure

Several reinforcement techniques are used to fulfil this task adaptively (see [4] for details of the navigation technique).

- *Exploration:*
  Finding moves and paths to increase the knowledge, accumulated in the map efficiently.

Although these three components may be seen as different levels of abstraction, they have to be able to operate in parallel. The process of acquiring knowledge by wandering around requires all major components.

5. Map-building

As introduced above, we build maps, which represent qualitative distinct regions in the environment. The basis elements, on which we are able to perform this task, are simple (passive) sensors, not being able to measure any distances. So what we may expect to get is a map of distinct light situations and of objects which have been detected by the touch sensors.

5-1. Structure

The robot ALICE is wandering around (controlled by an exploration algorithm or a specific navigation task) while producing “situation vectors” in equidistant time intervals. These situation vectors are calculated by concatenating the weighted rough sensor informations:

- *x, y coordinates* produced on the basis of dead reckoning. We will show later that the coordinates have to be correlated according to the actual map as to compensate drift effects.
- *24 light sensors* which are scaled according to the sensor detecting the brightest light source (in order to be independent from absolute light intensities). In specific applications the absolute amount of light intensity may be important, but we detected that the distribution of light seems to be a better criterion (regarding the map-building task).
- *24 touch sensors (whiskers)* which are not preprocessed in any way.

The weights of the several sensor kinds are approximated using the following assumption: The detection of an obstacle by a whisker should be interpreted as a new distinct sensor situation, regardless of other sensors. Therefore a relative high weight is attached to the touch sensors. The weights of the light and the position sensors depend widely on the application. If the navigation component (which uses this map) plans paths in the sense of *x*, *y*-positions, then the position data should get a high weight in order to produce a equidistant grid of nodes representing mainly different positions. On the other hand, if the navigator plans paths from one light situation to the other (e.g. a path from the “window” to the “hallway”), then a equidistant grid of positions is less important. Independent of the actual navigation task, we have to keep in mind that the position data have to be considered to a certain amount, because the connections between nodes in the graph (map) should represent topologic neighbourhoods (i.e. based on positions).

The calculated situation vector is passed to the dynamic feature-map ([2], [5]), which is a more flexible version of the common self-organizing-map introduced by Kohonen ([7], [8]). For a very short characterization, the main features are listed below:

- Unsupervised clustering
- Dynamical number of clusters
- Forgetting by time or frequency of access
- Flexibility over an infinite period of time

As mentioned in chapter 3 the connections of the dynamically constructed network (here based on triangles) are interpreted as the connections of a topologic graph, representing distinct places and their neighbourhood relationships. The connection structure will depend widely on the presentation order of the situation vectors. So it is possible to build up a “chaotic” graph (i.e. a graph with a lot of topologically not plausible connections), by presenting the input vectors stochastically distributed over a large area in the environment. It should be possible to find network adaptation functions, considering that the inputs are from different kinds of sensors, and therefore being able to construct connections which represent a “flat” and topologic “clean” graph structure in any case. Nevertheless, we will show the interpretation of the situation vector as a uniform vector only. As we will see in the next chapter one may get a well adapted structure also by using special exploration constraints.

The internal *x*, *y*-coordinates are corrected by a small fraction of the distance to the actual best matching unit (*bmu*) towards the *x*, *y*-coordinates of this bmu. The fraction must be much smaller than the adaptation factor e_{bmu} of the *bmu* itself, but higher than the drift effects, corrupting the internal position.

5-2. Simulations

The simulations are all done in one environment (figure 2), where three light sources, different kinds of objects, rough and plain surfaces, and a passage (nor symmetrical neither straight) are represented. The light sources are limited by diaphragms like that ones being used with studio spotlights. Therefore light is emitted within a certain angular range only (indicated by the white lines in figure 2).

The optimization criteria in the following simulation runs is not easy to formulate, because the usefulness of the produced topologic graph depends on the navigator or other components employing the structure for their purposes. We have successfully tested our navigator on the produced networks (shown below), but we have not yet tried to find the “best map”-criteria for our (or other) navigator(s). So we use some simple measurements to detect well adapted maps:

- *Small number of intersections:* The connections of the topologic graph should not intersect.
- *Short connections:* Only nodes being immediate neighbours should be connected.
- *Small number of nodes:* Due to the fact that the computational effort depends linear from the number of nodes, any realtime constrained task requires a map consisting of a limited number of nodes.
- *Slow “movement” of the cells:* Tasks (specially recognition tasks) employing the topologic map depend on stable structures. Or, in oth-

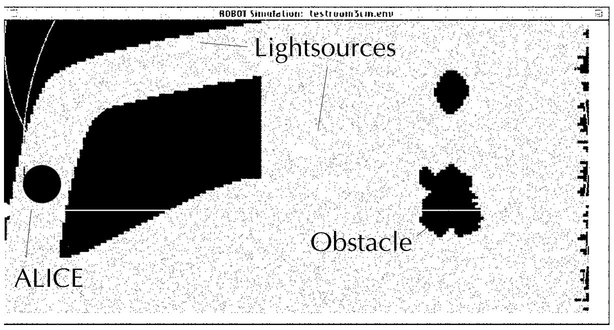

figure 2 : Test-Environment

er terms: The adaptation processes should result in similar networks after the updates.

- *Realtime Adaptation:* The adaptation process itself has to fulfil certain realtime constraints (see “Small number of nodes”). Moreover, the network structure should represent the actual environment after a reasonable amount of time.
- “*Well adapted*” *representation:* This demand depends on the tasks using the map. They have to give a certain amount of detail or generalization, which have to be represented in the map.
- Other task specific constrains …

The following groups of parameters may be altered in order to optimize the above criteria.

- *Network parameters:*
  The maximal number of nodes (the “size” of the network) may be limited. This forces the adaptation procedures to remove map regions not being used for a certain amount of steps. The required accuracy of the classification ($d_{accuracy}$) as well as the number of false classifications, until new cells are inserted ($n_{distribution}$), have to be determined. In order to get a stable but fast adapted structure, the “moving” parameters of the cells (e_{bmu}, $e_{neighbour}$, Δe_{bmu}) have to be optimized. Finally, several other parameters will not be referred here (p_{sure}, e_{stable}, starting configuration, …).
- *Sensor weights:*
  The sensor weights as introduced above, have a great influence of the kind of map being produced.
- *Exploration strategy:*
  The exploration (i.e. the order of presented situation vectors) influences widely the structure of the produced map.

The described simulation runs are evaluated on the trajectory shown in figure 3. As an exploration strategy, simple random movements are performed in a window wandering along the trajectory.

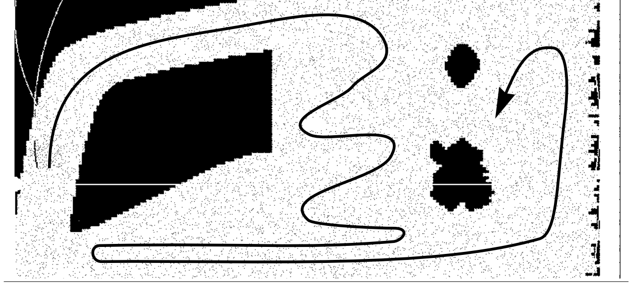

figure 3 : Test-Trajectory

In figure 4 to 6 the creation of a topologic environment map filling the test environment is shown. Notice that the nodes nearest to the obstacles represent the situations, when ALICE touches the obstacles with its whiskers, whereas nodes in open spaces represents different light situations. The total number of processed input signals was 5254. and the number of nodes in the network was 196. This run can be reproduced on other test-trajectories leading to similar results. Due to the small number of nodes in this examples, we are not forced to limit the total number of nodes. In an open (i.e. not limited) environment, it is necessary to limit the network size according to the available computing power.

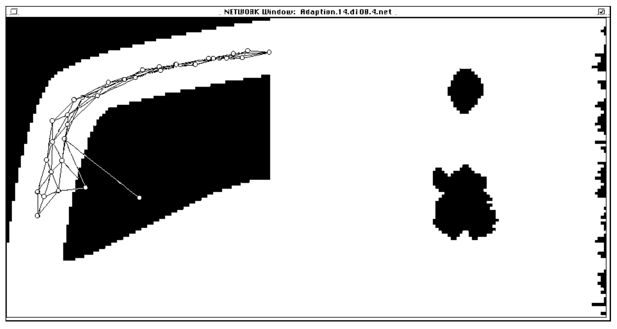

figure 4 : 1 / 3 of Test-Trajectory passed

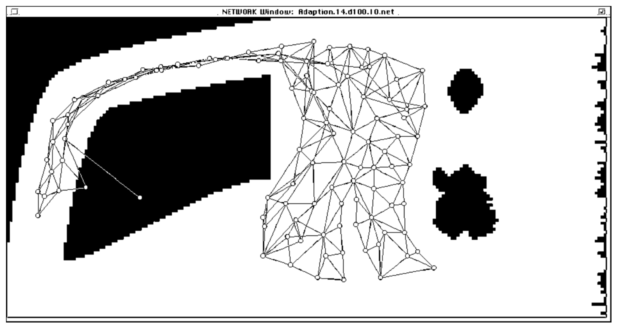

figure 5 : 2 / 3 of Test-Trajectory passed

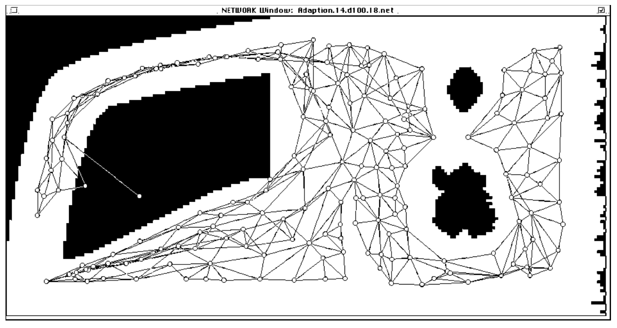

figure 6 : Full Test-Trajectory passed

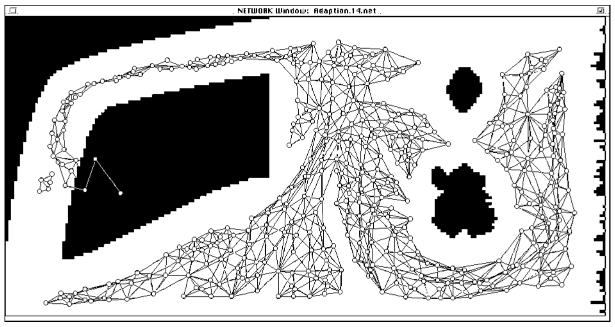

figure 7 : e_{bmu} = 0.01

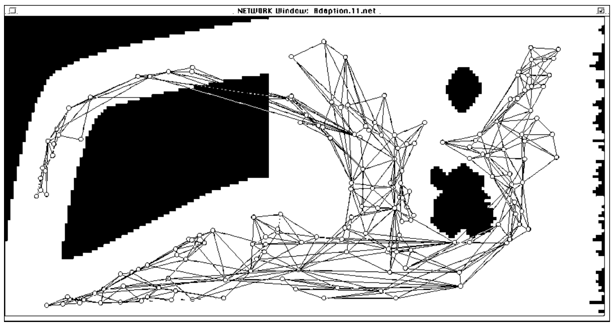

figure 8 : e_{bmu} = 0.1

One important parameter for the adaptation is e_{bmu} (and implicitly $e_{neighbour}$), which determines the adaptation speed as well as the network structure itself. In order to show two typical wrong configured networks, we set e_{bmu} to 0.01, which produces an overfitting of the environment, by inserting too much cells to fulfil the demanded accuracy (figure 7). On the other hand choosing e_{bmu} too high (0.1) results in a fast moving cell structure with chaotic connections and no useful modelling of the environment (figure 8).

The realtime aspects of the simulations might be summarized as follows: The computational effort rises linear with the number of nodes in the network (using parallel techniques it may be reduced to *O(log(n))*). In order to guarantee a certain adaptation speed, the total number of nodes must be limited. Therefore the area which might be explored without segmentation is limited by the “complexity” of the environment (which determines the necessary number of nodes) and of course by the available computing power.

6. Conclusions

We have shown by complexity approximations as well as by test-runs that the qualitative topologic maps, shown in this paper, are able to be performed under realtime constraints. Moreover they are a solid basis for a range of basic mobile robot tasks, which need not to have an exact geometric model of the environment, but which have to be performed on a stable and fast basis.

The navigation and track control task on these QTMs is quite different to exact geometric approaches. A possible solution for this task, which adapts to the actual environment as well as to internal drift effects is shown in [4].

References

[1] Benjamin J. Kuipers, Y. T. Byun
A Qualitative Approach to Robot Exploration and Map-Learning
AAAI '87 - Workshop on spatial reasoning and multi-sensor fusion

[2] Bernd Fritzke
Growing Cell Structures - A Self-organizing Network for Unsupervised and Supervised Learning
Technical Report 93-026, International Computer Science Institute, Berkeley, California

[3] Uwe R. Zimmer
Connectionist Decision Systems for a Visual Search Problem
Proc. of the IIZUKA ´94, Fukuoka, Japan, August 1-7, 1994, Invited paper

[4] Uwe R. Zimmer, Cornelia Fischer, Ewald von Puttkamer
Navigation on Topologic Feature-Maps
submitted to IIZUKA ´94, Fukuoka, Japan, August 1-7, 1994

[5] Herman Keuchel, Ewald von Puttkamer, Uwe R. Zimmer
SPIN - Learning and Forgetting Surface Classifications with Dynamic Neural Networks
Proceedings of the ICANN '93, Amsterdam, The Netherlands

[6] Ewald von Puttkamer, Christopher Wetzler, Uwe R. Zimmer
ALBATROSS - The Communication Scheme as a Key to FulFil Hard Real-Time Constraints
Proceedings of the Euromicro-Workshop on Realtime Systems '92, Athens, Greece

[7] Teuvo Kohonen
Self-Organization and Associative Memory
Springer-Verlag, Berlin Heidelberg New York Tokyo, ISBN 3-540-12165-X

[8] Teuvo Kohonen
Statistical Pattern Recognition Revisited
Advanced Neural Computers / R. Eckmiller (Editor), Elsevier Science Publishers B.V. (North-Holland), 1990