Robot Navigation using the Connectionist Navigational Map

Emil Åström* & Tom Ziemke†
Connectionist Research Group
Dept. of Computer Science
University of Skövde
Box 408, 54128 Skövde, Sweden
tom@ida.his.se‡

07/04/97

Abstract

This paper presents a number of robot navigation experiments with the Connectionist Navigational Map (CNM), originally proposed by Chrisley. The CNM is trained to map current internal state and action to sensor readings, using data collected during an exploration phase. During navigation this mapping is inverted, i.e. a desired goal state/location (but no sensory input) is mapped to the action most appropriate to reach this goal. A number of experiments in 'pointing out' goal locations and multi-step navigation on a simulated Nomad 200 robot are documented and evaluated. Results show that reasonable performance is achieved under benign conditions, but the CNM in its current form does not scale up to realistic robot navigation scenarios.

1 Introduction

The *Connectionist Navigational Map* (CNM) architecture was originally proposed by Chrisley [Chr90]. The CNM differs from most other robot navigation architectures in the sense that during navigation sensor readings are actually *ignored*. Instead a certain orientation sense and resulting navigational capacity are provided solely on basis of an internal world model, in the form of a recurrent connectionist network's internal state representations and weights, acquired during a preceding exploration phase. Hence, the CNM should be considered a possible complement to other sensor-based navigation techniques such as reactive and landmark based navigation or visual self-localization techniques.

Architecture and operation of the CNM will be introduced in the following section. A number of experiments with the CNM on a simulated Nomad 200 robot are then presented and analysed in section 3. Section 4, finally, gives a brief summary and a number of conclusions.

2 The Connectionist Navigational Map

2.1 Architecture

The CNM, in general, consists of the following three components:

- The *Predictive Map* whose basic functionality is the prediction of sensor readings. The Predictive Map is a connectionist network, consisting of the *topological (or T) mapping* from current internal state and action to the next internal state, and the *descriptive (or D) mapping* from the latter internal state to the sensor readings in the next location/time step (the CNM model uses discrete

Proc. AISB workshop on "Spatial Reasoning in Mobile Robots and Animals", Manchester 1997.
Technical Report Series, Dept. of Computer Science, Manchester University, ISSN 1361 - 6161.
Report number UMCS-97-4-1. http://www.cs.man.ac.uk/csonly/cstechrep/titles97.html

*Now with the Knowledge Technology and Methodology Group, Swedish Institute of Computer Science.
†Also with the Neural Computing Group, Dept. of Computer Science, University of Sheffield, UK.
‡All correspondence should be addressed to the second author.

time). Hence, the resulting architecture of the Predictive Map (cf. Figure 1[1]) is very similar to Elman's Simple Recurrent Network [Elm90].

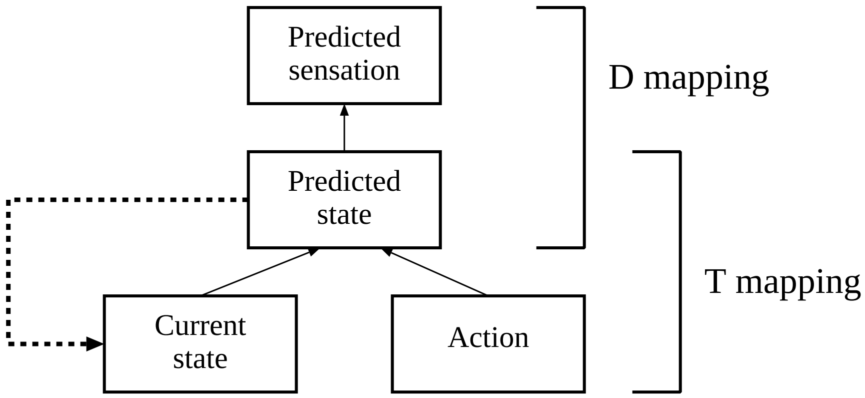

Figure 1: The Predictive Map

In the input layer of the network there are nodes both for the current state and for the action to take. A state could correspond to a unique location, but it could also refer to a 'qualitative' aspect of the world (e.g. being close to a wall). In that case several locations can have the same state vector. For the purpose of flexible and robust navigation state representations would ideally encode both 'qualitative' and 'quantitative' (e.g. the position expressed in Cartesian coordinates) properties of the corresponding locations.[2]

- The *Inverse Model* from which the system's navigational capacity is derived. The Inverse Model's basic function is to move the robot from one state/location to another nearby state/location. The naming 'Inverse Model' derives from the fact that the Inverse Model, in a sense, inverts the Predictive Map (in the case of goals being specified by sensory signatures) or the T mapping (in the case of goals being specified by state representations), since it maps current state and goal state/sensations to an appropriate action to move towards that goal.
- The *Orienting System* which is used for 'calibrating' the robot's opinion of where it is in the world. A theoretical discussion of possible implementations can be found in [Chr90] and [Åst96], the Orienting System has however neither been implemented/used in navigation experiments in the work presented here, nor in earlier work on the CNM [Chr90, Chr93, CH95].

2.2 Operation

The CNM basically works as follows:

- *Exploration*: The robot is guided on an exploration path through its environment, and sensor readings and actions (robot movements) are collected.
- *Training*: The Predictive Map is trained using the data collected during the exploration phase.
- *Navigation*: A goal state/sensation is specified, the Inverse Model determines the action that is most likely to lead towards that goal, and executes it (i.e. a particular robot movement). This process is repeated until a location sufficiently similar to the goal is reached. Note that in order for the Inverse Model to provide navigational capacity, ordered state representations throughout the navigation area are required.

[1]Note that Figure 1 is not necessarily an exact description of the network. It is possible to have hidden layers in the D mapping for example, without violating the principles discussed here.
[2]For a more detailed discussion of the issue of state representation see [Chr90] or [Åst96].

It should be noted that during navigation the CNM actually does not make use of its current sensor readings, but of its internal world model (of the relation between state representations and actions) only. Hence, the CNM should not be considered a candidate for a complete robot navigation system, but rather a component that provides a certain capacity for navigation/orientation towards a goal state/sensation in the absence of sufficient sensory input. That means the CNM could, for example, be used in a subsumption architecture (see [Bro86]) in combination with other mechanisms/behaviours providing obstacle avoidance behaviour and landmark-based navigation (note that the CNM could in fact be used to find such landmarks).

2.3 Previous work

In Chrisley's (and Holland's) own work [Chr90, Chr93, CH95] the CNM has only been used in simulated grid-worlds in which each position/location was given a rather arbitrary, binary 'sensory' signature (cf. Figure 2), and the main purpose of this work has been to investigate issues of objectivity and systematicity of representation rather than realistic robot navigation. Hence, the major goal of this study (mostly based on [Åst96]) has been to investigate in how far the CNM architecture could scale up to more realistic robot navigation scenarios, in particular the use of realistic sensors.

0001	1000	1100
1011	1010	0010
1110	1111	0010

Figure 2: 'Sensory' signatures in (part of) a grid world, as used by Chrisley and Holland [CH95].

3 Experiments

3.1 Robot Simulator

All experiments documented in this paper have been carried out using data from a simulated Nomad 200 mobile robot. Apart from odometry, laser range finder and tactile sensors not used in the experiments presented here, the Nomad 200 is equipped with the following sensors evenly spaced around the robot's cylindrical body (diameter: 18 inches)

- 16 infrared sensors (range: up to 36 inches),
- 16 sonar sensors (range: originally 6 to 255 inches, here limited to 100 inches).

The fact that the results presented here are derived solely from simulation should however not necessarily be considered a serious limitation, since the Nomad 200 simulator does actually provide rather realistic simulations. Therefore the resulting sensory signatures can be considered far more realistic than those used in the original CNM work.

3.2 Exploration/Training

The experiments documented in this paper have been carried out using the environments and exploration paths illustrated in Figure 3, which shows the robot as a black circle in the last location of the exploration path (in each of the environments), and all other locations on the exploration paths as non-solid circles.

- World 1 is a rather complex world, containing several obstacles, and the exploration path is very sparse (72 locations).[3]

[3] World 1 is larger than the other worlds, therefore the robot appears relatively smaller in Figure 3.

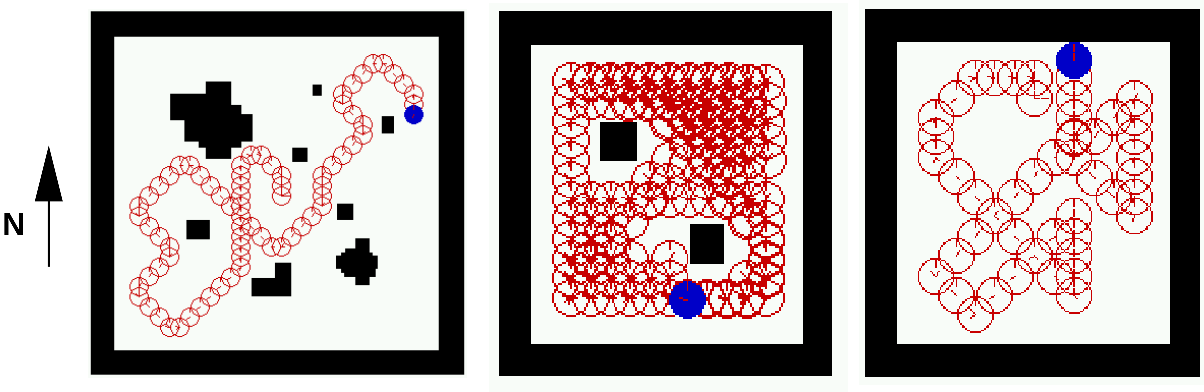

Figure 3: Worlds 1, 2, and 3.

- World 2 is smaller than world 1 and far less complex (containing only two obstacles). Moreover the exploration path consists of 343 locations, such that most locations are visited several times during exploration.
- World 3 is of the same size as world 2, but does not contain any obstacles. The exploration path is relatively sparse, containing 52 locations, very few of which are visited more than once.

As described for the original CNM above, during exploration the robot was guided on obstacle avoidance paths[4] through the respective environment (cf. Figure 3). As in earlier work on the CNM, eight possible discrete actions were used: moving north, west, south or east (N, W, S, E) for a certain distance, and combinations of these four (NW, NE, SW, SE), such that the robot moves in a grid-like fashion (slight deviations from this (cf. Figure 3) result from the fact that the simulation does take into account noise, possible wheel slippage, etc. to a certain degree).

At each location on the respective exploration path sensor readings (initially from the 16 short-range infrared sensors, in later experiments from the 16 long-range sonar sensors, cf. section 3.4) and executed actions were recorded. It should be noted that during exploration the robot's orientation was kept constant[5], such that one location would have very similar sensory signatures (i.e. the same except for variations due to noise) even if repeatedly visited from different directions. After exploration the Predictive Map was trained on the collected data, using 20 units for representation of the states.[6] The actions were encoded using a four-unit localistic representation, such that N would correspond to an action vector 1000, NW to 1100, W to 0100, etc. In these experiments all network units used sigmoidal activation functions.

3.3 Navigation

For navigation the Inverse Model was derived from the Predictive Map, by

- setting current and goal state to state representations (not sensory signatures) learned by the Predictive Map (in the case of several codes for a location visited more than once, just one of them was picked arbitrarily), and
- iteratively comparing predicted and goal state while modifying the action vector to decrease the difference between them[7] such that the most appropriate action was found, i.e. the action that, according to the T mapping, would take the robot closest to its goal.

[4]For a more detailed description of how the obstacle avoidance behaviour was 'constructed' see [Åst96].

[5]This if of course somewhat unrealistic, since on the real Nomad 200, due to wheel slippage, etc., the orientation probably could not be kept constant over a long period of time

[6]Experiments with different number of units indicate that the exact number is not critical.

[7]For a more detailed description see [Åst96]. An alternative scheme, although only applicable in the case of discrete actions, would have been to simply apply the T mapping to the possible eight actions and pick the action that results in the

Two types of experiments were carried out in order to evaluate the CNM's navigational capacities and the systematicity of the learned state representations:

- *'Pointing'*: the CNM was given a specific location on the exploration path as its current location and another one as goal, and the action found to be most appropriate to approach the goal by the Inverse Model was interpreted as the robot pointing from its current location to the given goal.
- *Multi-step navigation*, i.e. testing the robot's capacity to navigate towards a distant goal in a series of time steps at each of which the robot moves in the direction pointed to by the Inverse Model.

3.4 Results and Analysis

3.4.1 Pointing

For worlds 1 and 2 and (using short-range sensors) the results in pointing from all possible current locations (vertical axis) to all possible goal locations (horizontal axis) on the exploration path are shown in Figure 4. Two things should be noted here, and kept in mind when interpreting the results presented below:

- Due to the fact that the Inverse Model can only 'choose' between eight directions (N, NE, E, etc.) to point to, it is not possible to always point out the right direction exactly. The errors measured are absolute values, i.e. they will always lie between 0 and 180 degrees. Hence, completely untrained networks, pointing randomly, exhibit a mean error of approximately 90 degrees. That means, even with the Inverse Model performing 100 percent correctly, an average pointing error of 11.25 degrees[8] would have to be expected.
- A failure to point out a goal location in one step is not necessarily equivalent to a failure to navigate to that location in a series of steps. In fact, in the case of current and goal location being separated by an obstacle, the 'decision' not to point/move straight towards the goal will actually be required.

Figure 4 illustrates that rather poor performance is exhibited when using the short-range infrared sensors. The graphs should be interpreted as follows: The brightness of the cell at location (x, y) indicates the success of the Inverse model to give the direction to the x'th location in the exploration sequence when the robot is at the y'th location. White corresponds to an error of 0 degrees and black corresponds to an (absolute) error of 180 degrees. Note that this diagram is not plotted to spatially reflect the world in any respect, the axes only show indexes into the exploration path.

In both cases the figures are rather dark, corresponding to the rather high mean errors of 86.9 and 61.4 degrees respectively. A number of interesting observations can however be made in these diagrams:

1. The Inverse model is able to rather accurately point out the next locations in the exploration sequence. This is the explanation for the bright diagonal in both gray-scale graphs.
2. It seems more difficult to give the correct directions for locations immediately before than immediately after the current one (there are dark areas above and to the left of the above mentioned bright diagonal).
3. Both images exhibit a certain structure, i.e. they contain horizontal and vertical bars. This is more clear in world 2. Further analysis showed that the dark bars occur exactly when the robot moves from one of the two open areas of world 2 to the other, i.e. pointing works better within the open areas, and the robot gets 'confused' when moving between them.
4. The performance for the world 2 experiment is significantly better than for world 1 (mean errors of 61.4 vs. 86.9 degrees). This is not particularly surprising considering the differences between the worlds: World 2 is smaller, simpler (more globally ordered sensor readings) and explored in more detail which should make it easier to generalise the effects of actions on sensor readings.

---

state closest to the goal. The iterative scheme was used here in order keep the procedure general enough to be applicable to continuous navigation/motor control.

[8] 90 degrees divided by 8.

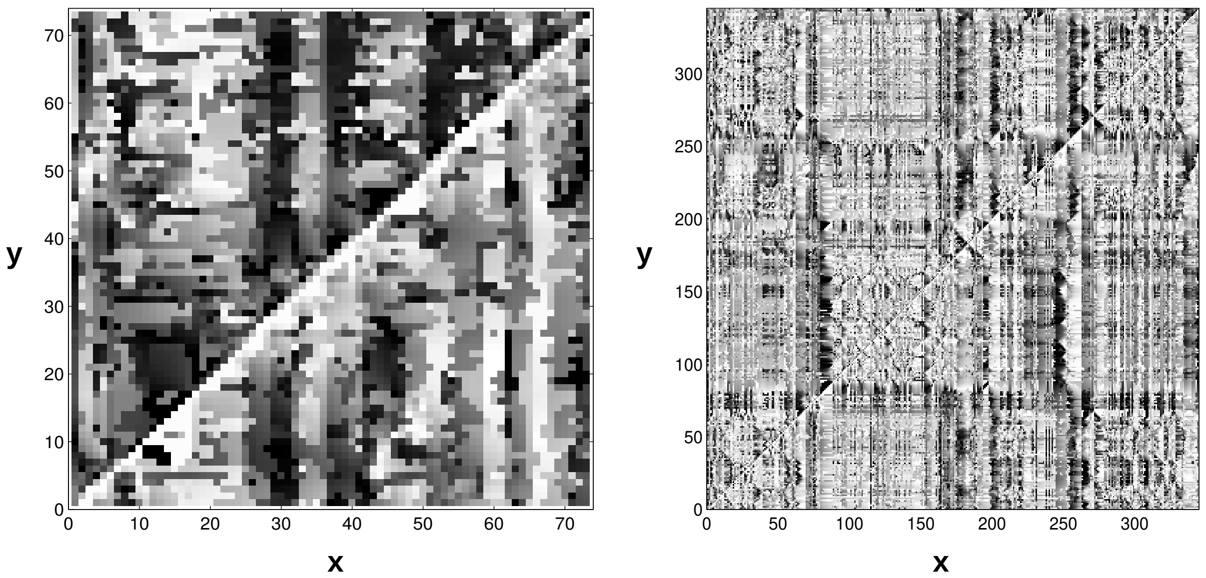

Figure 4: Pointing in world 1 and 2.

In the experiment illustrated in Figure 4 the robot used short-range sensors only. The experiment was repeated with long-range sonar sensors (the sensor range (100 inches) is such that in worlds 2 and 3 from any location at least two walls can be sensed, and in about 10% of the locations all four of them), which of course should make it easier to establish a global ordering of state representations. Accordingly, the performance now significantly increased (mean error of only 26.5 degrees, as opposed to 61.4 earlier), as shown in the left half of Figure 5.

Performance could be increased even further (not surprisingly) when repeating the experiment with the obstacles removed from the environment. The robot's 'direction pointing' performance now is very good, with a mean error of 19.6 degrees (as compared to the 'theoretically optimal' mean error of 11.25 degrees) and errors below 90 degrees almost everywhere in the navigation area, as shown in the right half of Figure 5. It is obvious that this data set gives the best pointing performance of all experiments documented here.

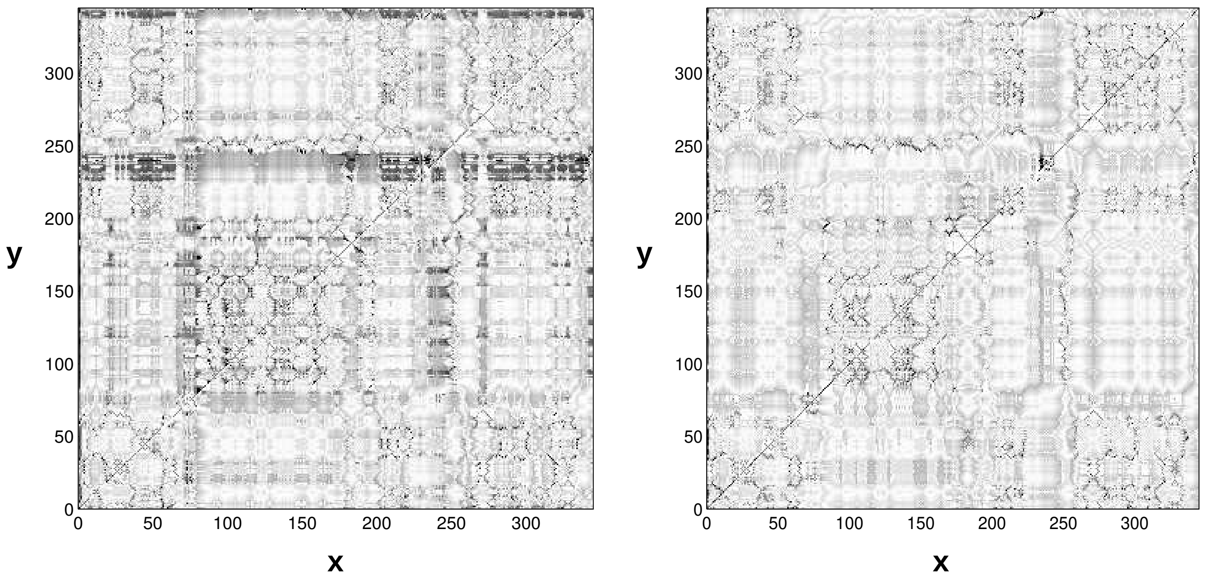

Figure 5: Pointing in world 2, using long range sensors, (a) with and (b) without obstacles. The diagram for the original experiment can be seen in Figure 4 (right half).

To evaluate the role of the exploration path the latter experiment (using long-range sensors) has been repeated in world 3, which is basically the same as world 2 without obstacles, using a sparse exploration path of 52 locations only (as compared to 343 in world 2). The mean error here is 40.5 degrees (cf. Figure 6), i.e. significantly worse than with the longer exploration path (mean error of 19.6 degrees), but within reasonable limits. The goal locations that work best are the ones at the borders of the world.

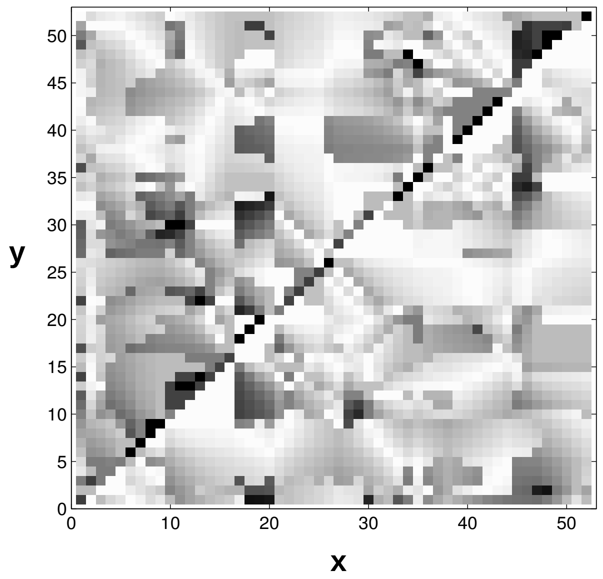

Figure 6: Pointing in world 3 (no obstacles, using long-range sensors, and a sparse exploration path).

3.4.2 Multi-step navigation

As mentioned above, robot navigation using the CNM is carried out by specifying a goal state/sensation, and repeatedly letting the Inverse Model determine the action/movement taking the robot closest to the goal (i.e. pointing) and moving one step in the direction pointed to. The last two steps are repeated until the goal (or a state sufficiently similar to it) is reached. Hence, it is clear that reasonable pointing performance is essential for navigating in a series of pointing/moving steps.

It should be noted, once again, that during navigation the CNM does not make use of actual sensor readings, but only of its internal model of the world (memorized in form of the connection weights in the Predictive Map), i.e. the robot navigates, in a sense, blindly and 'by heart'. This should be kept in mind when judging the results presented in the following.

Multi-step navigation experiments using the internal models acquired from short range sensor measurements, were in general not particularly successful, not surprisingly considering the high error levels in pointing documented above.[9] The problem is illustrated in Figure 7[10] which shows pointing performance for one of the possible goal locations (results are representative though) in world 1: In many of

[9]Furthermore it was noted that (in particular in world 2) visiting a location several times often led to the acquisition of rather diverse state representations for the same location. That is not necessarily a problem when applying the T mapping, since two state representations can perfectly well be functionally equivalent (i.e. lead to the same sensor predictions), without necessarily being close in Euclidean space (see [SJ94] for an extensive discussion of the difference between functional and Euclidean equivalence). Diverse location representations do however pose a problem when applying the Inverse Model, since different state representations will (possibly) make the Inverse Model point in different directions.

[10]This graph shows what directions the Inverse Model points to for a specific goal state (marked with a rectangle). The figure essentially shows one column of the gray-scale graph for world 1 as shown in the left half of Figure 4, but in more detail. All the elements of the exploration path are used as current locations and the locations are plotted so that their positions in the diagram reflect their positions in the world. A $\diamond$ indicates that the location was visited during the exploration path. The locations with $\diamond$'s also have lines showing the directions that were the results of the execution of the Inverse model. Some location have several lines. This is the case when the location in question was visited more than once (thus having more than one code) in which case all the codes for that location are used in the Inverse model. If the codes give different results then all the codes' directions are plotted. Coordinates are shown relative to the start location of the exploration path.

the locations the Inverse Model is able to 'point' and move towards the goal (approximately), in others however the robot is falsely led away from the goal, and this even happens in locations close to the goal (e.g. the locations southwest of the goal, cf. Figure 7). Hence, in many cases it is unlikely that the robot can find a possible path, i.e. a series of pointing steps, that takes it to the goal.

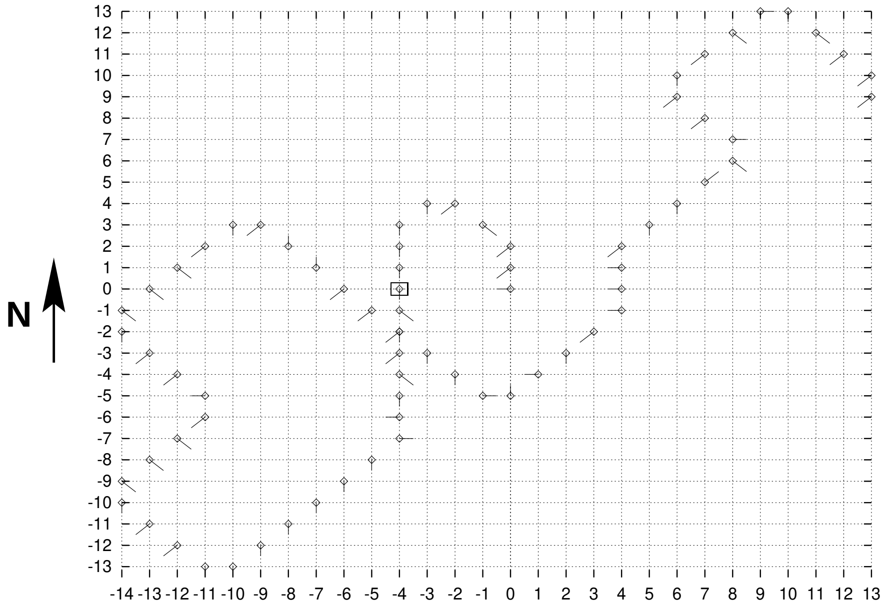

Figure 7: Pointing to one goal location (marked with a rectangle) in world 1.

With long-range sensors, however, pointing performance is rather good, as illustrated for worlds 2 and 3 in Figures 5 and 6. Figure 8 shows pointing performance for one of the goal locations in world 3. It can be seen that here the Inverse Model is able to point towards the goal more or less correctly from most of the locations. Hence, multi-step navigation in the case of long-range sensors is much easier to accomplish, and in fact (more or less) successful in the majority of cases (at least in world 2). An example of multi-step navigation in world 3 (same goal location as in Figure 8) can be seen in Figure 9.[11] Performance is typically not perfect but reasonably good. Note, for example, that in Figure 9 the robot ends up in a location two steps south of the actual goal (note also that the sensor readings in these two locations are likely to be very similar, i.e. the robot could be said to have reached its 'qualitative' goal ('find a wall to the right').

3.4.3 Further experiments

Further experiments were carried out in which the Predictive Map, in order to enforce globally ordered state representations, was trained not only to predict the sensor readings (cf. Figure 1), but also the absolute position in the environment, i.e. coordinates normalized to values between 0 and 1. Experiments were carried out both with sigmoidal activation functions for the two extra units, as well as linear ones.

Training the Predictive Map this way did in fact improve pointing performance, leading to decreases of mean error between 7 and 17 degrees in a number of experimental setups, but not all of them. In particular, results indicate that the above training scheme consistently does improve performance in the case of short-range sensors, whereas it does not necessarily when using long-range sensors (i.e. the additional training on coordinates does not help to enforce stronger global ordering than provided by the long-range sensor input anyway).

[11] The figure should be interpreted as follows: The $\diamond$'s represent the different codes for the locations. If a location was not visited during exploration then no $\diamond$ is present at that location. The levels of the $\diamond$'s are the Euclidean distances for the codes to the average code for the individual locations. The starting position is marked with a cross and the goal with a rectangle. The path of the robot is represented by two dashed lines: one at the mesh level and one where the level represents the similarity of the location codes to the average codes from exploration (or if the location was not previously visited, the distance to a 0-vector). The mesh-level path is present only to make it easier to see which way the robot actually took during navigation.

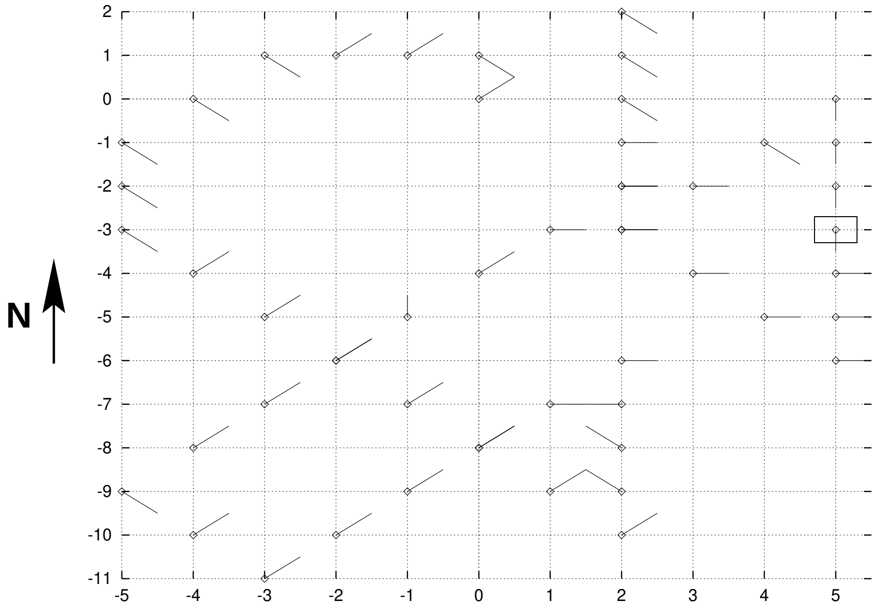

Figure 8: Pointing to one goal location (marked with a rectangle) in world 3 (no obstacles, long-range sonar sensors, and sparse exploration path).

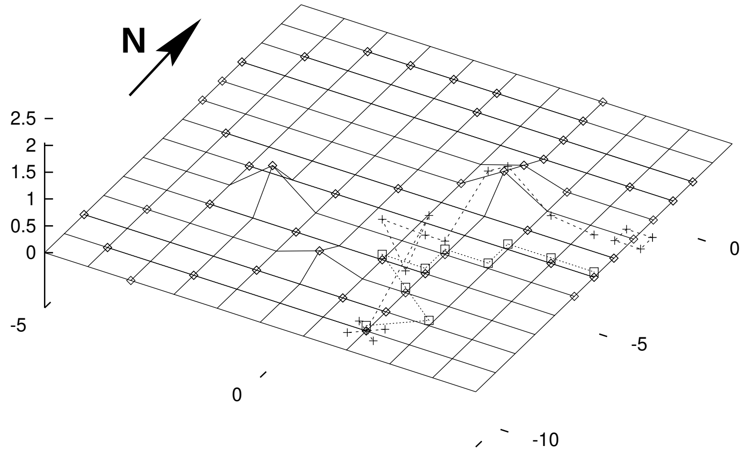

Figure 9: Multi-step navigation in world 3 (no obstacles, long-range sonar sensors, and sparse exploration path).

4 Summary and Conclusion

The CNM has been discussed in this paper as a possible complement to other robot navigation techniques. In a navigation subsumption architecture, at least in theory, the CNM could provide a certain goal orientation/navigation capacity in the absence of sufficient sensory stimuli, and be combined with other levels/behaviours such as obstacle avoidance and landmark-based navigation (cf. section 2.2).

Another aspect that makes the CNM interesting is that, given adequately globally ordered state representation the CNM, at least theoretically, could combine 'qualitative' ('find a wall on the right') and 'quantitative' ('go to location X') navigation. These aspects have however not been investigated in detail here, and much more experimental work will be required in order to evaluate this possibility.

Experimental results presented here however showed that the CNM in its current form exhibits good pointing and navigation performance (such as successful navigation through previously unseen territory) only under very benign conditions (globally ordered sensor readings, etc.) whereas it does not scale up to (more) realistic robot navigation scenarios. The following limitations of the CNM can be identified:

- State representations are strongly influenced by the current sensor readings. The CNM, however, depends strongly on ordered state representations, i.e. the state should encode a richer context (coming from which direction, etc.) than mere sensor readings.
- In order to achieve reasonable performance the exploration path cannot be too sparse, it should cover most of the world.
- The CNM has so far only been evaluated in simulated grid-like worlds, and it is not clear how it could scale up to continuous motion in realistic environments.
- Unless complemented by a sensor-based navigation technique, it is a limitation that the CNM cannot make use of its sensor readings during navigation.

Hence, the overall conclusion of the work presented here is that the CNM does, in theory, have a number of interesting features, that could be useful to complement other robot navigation techniques. Practically however, the experimental results presented here clearly show that the CNM in its current form does not exhibit the degree of reliability and robustness required in realistic robot navigation scenarios.

References

[Åst96] E. Åström. A Connectionist Architecture for High-Level Robot Navigation. Master's thesis, Department of Computer Science, University of Skövde, Sweden, 1996.

[Bro86] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 1:14–23, 1986.

[CH95] R.L. Chrisley and A. Holland. Connectionist Synthetic Epistemology: Requirements for the Development of Objectivity. In L.F. Niklasson and M.B. Bodén, editors, *Current Trends in Connectionism, Proceedings of the 1995 Swedish Conference on Connectionism*, pages 283–309. Lawrence Erlbaum Associates, 1995.

[Chr90] R.L. Chrisley. Cognitive Map Construction and Use: A Parallel Distributed Processing Approach. In D.S. Touretzky, G. Hinton, and T. Sejnowski, editors, *The Proceedings of the 1990 Connectionist Models Summer School*. Morgan Kaufmann, 1990.

[Chr93] R.L. Chrisley. Connectionism, Cognitive Maps and the Development of Objectivity. *Artificial Intelligence Review*, 7:329–354, 1993.

[Elm90] J.L. Elman. Finding Structure in Time. *Cognitive Science*, 14:179–211, 1990.

[SJ94] N.E. Sharkey and S. Jackson. An internal report for connectionists. In R. Sun and L. Bookman, editors, *Computational architectures integrating neural and symbolic processes*, pages 223–244. Kluwer Academic Press, Boston, USA, 1994.