

# Learning to understand tasks for mobile robots\*

Stephan H.G. ten Hagen  
Faculty of Science  
University of Amsterdam  
Amsterdam, The Netherlands  
Email: stephanh@science.uva.nl

Ben J.A. Kröse  
Faculty of Science  
University of Amsterdam  
Amsterdam, The Netherlands  
Email: krose@science.uva.nl

**Abstract** – *We propose a way to represent the environment by storing observations taken in that environment, together with their task-related ‘values’. This representation allows for robots being taught by human instructors based on rewards and punishments. We show that the robot is able to learn to execute different tasks. The results of training can be interpreted to gain understanding about the environment in which the robot has to operate. So instead of first modeling the environment and using this to execute the task, we first start by learning to execute the task and use this to obtain knowledge about the environment.*

**Keywords:** Mobile robot navigation, Learning systems, Human-robot interaction.

## 1 Introduction

In foreseeable future robots may become common place in every day life. People, not interested in the technical details of robotics, may purchase a robot and bring it into their homes. These people are not likely capable or willing to give a quantitative task description to the robot, like “move one meter to the left and turn 30 degrees”. It is more likely that tasks will be expressed qualitatively like, “vacuum the bedroom” or “bring me some coffee”. These task descriptions are related to elements or places that can be observed in the environment. To make future robots more user friendly, they need a way represent the environment and a more ‘natural’ way of being told what to do there. So robots should be able to recognize relevant elements or places and associate them with task descriptions.

We use a navigation task for a mobile robot to present a framework for both representing the environment and learning the task. In contrast to most approaches that use a *geometric* map, we use a database of sensory observations taken at different poses [19]. The user can easily fill the database by moving the robot around and have it store all observations. Alternatively the robot can explore the environment randomly.

---

\*0-7803-8566-7/04/\$20.00 © 2004 IEEE.

This research is supported by the Netherlands Organization for Scientific Research (NWO).

Navigation requires the ability to use the current observation to move towards the pose of a stored observation. By picking the right observations at the right time as ‘subgoals’, the robot can complete the task. The robot uses reinforcement learning to improve the selection of observations, based on the rewards and punishments received from the user. So by judging the robots behavior the user can make the robot perform the task.

Section 2 provides some background information and in section 3 we present our approach of representing the environment and learning the task. In section 4 we show in an experiment that the robot is capable of learning different tasks. Furthermore we show that the robot is capable of deriving knowledge about the environment from what it has learned.

## 2 Background

A mobile robot can perform a certain task if it knows the environment and know what to do. Navigation requires an internal representation of the environment and a way to express which ‘features’ in that environment correspond to the goal of the task.

### 2.1 Mapping and Navigation

In mobile robot navigation the state of the robot is often expressed as its pose (position and orientation) in some sort of global reference frame. This pose can be estimated from odometry in combination with external sensors, such as range sensors or vision. For this an explicit, ‘metric’ map can be used as internal representation of the environment, for example an occupancy grid, a polygonal model or positions of landmarks. A drawback of a metric map is that it does not scale well for larger environments.

An alternative ‘symbolic’ representation for space is a topological map. This consists of a graph with vertices representing places and edges representing the actions or action sequences to move from one vertex to another. Depending on what is represented by the vertices different ‘types’ of topological maps can be distinguished.

Vertices can represent distinctive places [14], that can be used for planning and reasoning about the navigation

task. It forms a hierarchy in which the task is split in low level local navigation and high level decision making. The edges represent the ability to move from one vertex to another using some local navigation strategy, like wall following or homing. A local metric map can be created for each vertex in the topological map [21], or a topological representation can be used to enhance building of metric maps [20].

Vertices can represent sensory events defined by the user [1, 21] or snapshots taken by the robot itself [6]. The panoramic snapshots in [6] form vertices of a *view graph*, that can be constructed autonomously by connecting new snapshots to the graph. In biological inspired topological maps, that are sometimes called cognitive maps, the vertices are units or neurons in a connectionist representation [22] that are called place cells because they fire when being at certain places.

Vertices can represent partitions of the state space with a specific control regime or behavior strategy. In [4] landmarks are learned by selecting ‘interesting’ features and connecting them together. Partitions, formed by the union of visibility areas of sets of landmarks, give a symbolic representation of the view that can be used for reasoning and planning. In [5] the graph itself is omitted and learned landmarks are linked to production rules telling the robot what to do when detecting landmarks given a certain task.

## 2.2 Teaching and Learning

Purposeful behavior of the robot needs, besides the representation of the environment, also a certain goal. This can be programmed into the robot. In case off mass produced robots, where each has to operate in a slightly different environment, this becomes infeasible. For each environment the user has to express what the robot has to do, and the robot has to be able to relate this to its environment representation.

The goal of a task can be represented in a topological map, when the goal coincides with one or more vertices in the graph. The robot is placed at the goal and is told the ‘name’ of the current place in the environment. An other way to associate human understandable ‘labels’ with places is by having the robot ask the humans [15]. Once goal vertices are identified the graph can be used to plan navigation to the goal from any other vertex. Values representing the distance to the goal in the graph can be associated with the vertices. The robot performs the task by selecting the adjacent vertex with the lowest value as subgoal, until the goal is reached.

The task can also be following a specific path or trajectory. In programming by demonstration [10, 3, 2] the user controls the robot and in programming by imitation [8, 7] the robot tracks an object or estimate a trajectory. In both case the robot keeps track of sensory observations along the trajectory that are relevant for executing the task. The sequence of observations

can be seen as a topological map of the task and ‘time stamps’ can be seen as the values associated with these observations. Now the task can be performed and relevant features in the observations can be detected. Note that the observations used here do not have to agree with the way the user thinks about the task.

A task can be taught by making the robot do the right thing under the right circumstances. In a ‘behavior based network’ sensory inputs can trigger the selection of the local navigation strategy and Reinforcement Learning (RL) can be applied to optimize the selection [11]. This network can be used in the context of training by example, where the trainer can be a human or an other robot [13]. In [12] the behavior of the robot is interpreted as ‘implicit communication’ or ‘communication through actions’. A human observer is capable of understanding the intentions of the robot from the exhibited behavior, because the actions carry the intentional meaning and are therefore easy to understand. These intentions of the exhibited behavior should, in our case, coincide with the task the user has in mind and are therefore easy to evaluate. Then the use of reinforcement learning allows the user to correct the observed behavior using rewards and punishments. This way of teaching is appropriate for novice users without any robot programming skills.

## 3 Our Approach

### 3.1 Mapping and Navigation

The general architecture of our approach is depicted in figure 1. The core of this architecture is a set of observations  $\{\mathbf{z}^1, \dots, \mathbf{z}^i, \dots, \mathbf{z}^N\}$  that were taken at different, *possibly unknown*, poses  $\{\mathbf{x}^1, \dots, \mathbf{x}^i, \dots, \mathbf{x}^N\}$ . These are the views (a). In section 4 we use gray scale panoramic images as observations, but other sensory observations can be used as well. The only requirement is the ability to estimate the (partial) difference in pose at which two observations were taken, given *only* the two observations. Define

$$D^{ij} = D(\mathbf{z}^i, \mathbf{z}^j) = \mathbf{x}^i - \mathbf{x}^j \quad (1)$$

as the estimated pose difference. The estimate  $D^{ij}$  can be used for navigation as an alternative to estimating poses  $\mathbf{x}^i$  and  $\mathbf{x}^j$ .

Besides the type of observation, the poses at which the two observations were made determine whether  $D$  can be estimated. For any view  $\mathbf{z}^i$  we define an area  $X^i$  around  $\mathbf{x}^i$  in which observations made can be used to estimate  $D$ . For (1) this implies that  $\mathbf{x}^i \in X^j$ . Given the sensor’s current observation  $\mathbf{z}_k$

$$D_k^i = \begin{cases} D(\mathbf{z}^i, \mathbf{z}_k) & \Rightarrow \mathbf{x}_k \in X^i \\ \emptyset & \Rightarrow \mathbf{x}_k \notin X^i \end{cases} \quad (2)$$

is computed for the entire set of views (b). The computation of  $D(\mathbf{z}^i, \mathbf{z}_k)$  does not always render reliable estimates and if it does not,  $\mathbf{x}_k$  is by definition outside  $X^i$ .

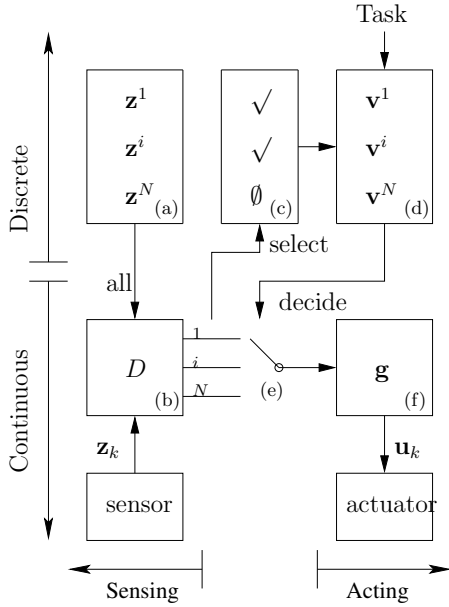


Figure 1: The proposed architecture, with (a) the views, (b) the estimation of  $D$  for each view, (c) the views for which  $D$  could be estimated, (d) the values of each view for each task, (e) the  $D$  chosen and (f) the controller using the chosen  $D$ .

Only the ‘feasible’ views for which  $\mathbf{x}_k \in X^i$  are selected (c), because at this time step  $k$  other views can not be used for navigation. From the feasible views the robot decides which one is the most appropriate for the given task. A vector  $\mathbf{v}$ , associated with each view, contains scalar values indicating the appropriateness of this view for each task. Once the decision is made about the target view (e), the corresponding  $D$  is used to compute the control action  $\mathbf{u}_k$  by the controller (f).

The bottom part of our architecture in figure 1 is a continuous perception-action cycle, that is closed by making a decision about the target view. The robot will home in to the selected view until an other target view is chosen. Since the  $D$  is defined as pose difference, it can be interpreted as the ‘state error’ with respect to the pose of the target view. Thus given the kinematic/dynamic model of the robot, the  $\mathbf{g}$  can be implemented as an error state feedback controller. This is less trivial if  $D$  is not the complete pose difference, but views  $\mathbf{z}^i$  can be a combination from different sensors that complement each other. Our robot uses panoramic images, and from only two images the scale of the pose difference cannot be estimated. It could be estimated if the views  $\mathbf{z}^i$  were containing images *and* sonar readings. An alternative solution is possible for our robot because  $D$  gives the direction of the target pose. The controller  $\mathbf{g}$  can make the robot turn in the right direction and then move forward a fixed amount. When this does not give the right performance more elaborate methods from visual servoing should be used.

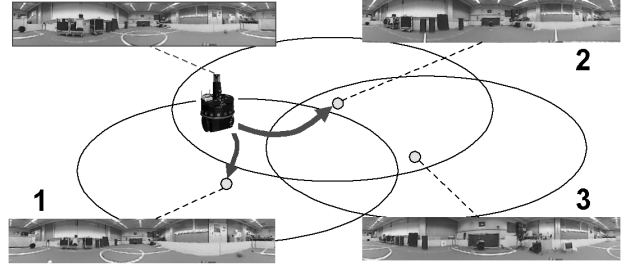


Figure 2: Navigation example: The robot with omni directional camera has a database containing three stored views  $\mathbf{z}^1$ ,  $\mathbf{z}^2$  and  $\mathbf{z}^3$ . Circles indicate poses  $\mathbf{x}^1$ ,  $\mathbf{x}^2$  and  $\mathbf{x}^3$  and ellipses the corresponding areas  $X^1$ ,  $X^2$  and  $X^3$ . The robot is in  $X^1$  and  $X^2$  and can choose between 1 and 2 as target. When choosing 2 eventually it will enter area  $X^3$  and may choose 3 as target.

The top part of our architecture can be seen as a view graph with the stored views as the vertices. An edge between vertex  $i$  and vertex  $j$  exist if  $\mathbf{x}^i \in X^j$  and  $\mathbf{x}^j \in X^i$ , because this implies that controller  $\mathbf{g}$  is capable of moving the robot from  $\mathbf{x}^i$  to  $\mathbf{x}^j$  and back. Values associated with the vertices for the given task provide the plan to reach the task’s goal. However, this plan does not describe the true trajectory of the robot. The selection (c) in figure 1 can be seen as the edges that connect the robot’s current observations  $\mathbf{z}_k$  as a new vertex to the view graph. From this vertex one of the adjacent vertices is chosen as target and the robot moves in that direction. When at the next step  $\mathbf{z}_{k+1}$  is connected to the graph the set of adjacent vertices may differ. This may lead to a different choice in target vertex so that the target chosen at time step  $k$  will not be reached. This happens in the example in figure 2 when the robot selects 3 as new target. Decisions about target views are not made at the vertices, but at the borders of the areas  $X$ . So instead of a view graph we can also see the architecture as a partition of the state space into different control regimes, given by the overlap in areas  $X$  and the values.

### 3.2 Teaching and Learning

In our architecture the tasks are encoded in the values in (d) of figure 1. These values can be manually selected but also most approaches described in section 2.2 can be applied. When the robot is placed at some arbitrary location, then this ‘new vertex’ can serve immediately as the goal state for this task. The values for views are given by the distance in the graph. Programming by demonstration is also possible by starting with zero values for each view. Now values are incremented every time step from the moment that the robot enters their area  $X$  until the trajectory finishes.

We use Reinforcement Learning (RL) [9, 17] to learn the values for each view. In this way the robot can be

taught by reward and punishment. The reinforcement  $r$  is a scalar evaluation and we use a positive  $r$  for a reward and a negative  $r$  for a punishment. The objective of RL becomes the maximization of the expected cumulative future reinforcements for all states or state/action pairs. Define the value function as:

$$V(s) = E\left\{\sum_{i=k}^K \gamma^{i-k} r_i \mid s = s_k\right\}, \quad (3)$$

with  $\gamma \in [0, 1]$  the discount factor and  $K$  the (possible infinite) final time step. The  $s$  is a discrete state from the finite set  $\mathcal{S}$ . The values for each state can be estimate using the temporal difference learning:

$$V(s_k) \leftarrow (1 - \alpha)V(s_k) + \alpha(r_k + \gamma V(s_{k+1})), \quad (4)$$

with  $\alpha \in [0, 1]$  the learning rate.

Equation (4) only applies to finite discrete state spaces but our robot state is given by the 2D continuous position. One way to deal with a continuous state space is using coarse coding [17] representation for the value function. Here a finite number of overlapping areas, called receptive field, cover the continuous state space. This is exactly the same as the areas  $X$  in figure 2. Each receptive field has it's own value. The value for the continuous state is given by the average of all receptive fields it is currently in.

Define a discrete state vector  $\mathbf{s}$  for which each element correspond with exactly one receptive field. This element equals one if the continuous state is in that field and zero otherwise. So the current state in figure 2 would be  $s_k = [1 \ 1 \ 0]$ . Instead of (4) the  $i^{\text{th}}$  element of  $\mathbf{s}$  is updated according to:

$$V(s_k^i) \leftarrow (1 - \alpha)V(s_k^i) + \alpha(r_k + \gamma V(\mathbf{s}_{k+1})), \quad (5)$$

with  $V(\mathbf{s}_{k+1}) = \frac{1}{n} \sum_{i=1}^n V(s_{k+1}^i)$  and  $n$  the number of ones in  $\mathbf{s}_{k+1}$ . Because more elements of  $\mathbf{s}$  are updated at the same time, learning is much faster compared to updating only one state as in (4).

The selection of the view with the highest value give the greedy policy for the current estimated value function. During learning alternative target view have to be tried by adding exploration. In [19] the  $\epsilon$ -greedy policy [16] was used, so with probability  $\epsilon$  the observation with the highest value was selected and with probability  $1 - \epsilon$  some other observation from the feasible subset. Based on the results in [19] we found an exploration strategy more appropriate for our approach. We call it the  $\epsilon$ -rejection policy. Each feasible target observation has a probability of  $\epsilon$  of being rejected as target. From the remaining set the observation with the highest value will be selected. Empirically this results in shorter learning times. An additional advantage is that the sensory uncertainty can be interpreted as adding extra exploration, when assuming that all feasible target observations have an equal probability of being missed because of noise.

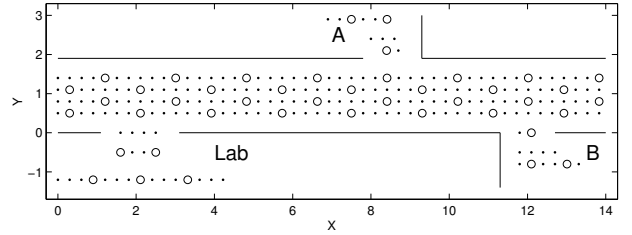


Figure 3: The data set. All dots and circles indicate positions where images were taken. The circles were used as the database of stored images. Lab indicates our lab and A and B are office rooms.

When using RL the view graph is not used and all values are a consequence of the interaction with the (human) teacher. When multiple tasks are learned the values are stored in the vectors  $\mathbf{v}$ . So each view has a set of values indicating how good it is to approach this view given a task. The highest values for a given task should show what the goal looks like. So not only does the robot learn to perform the task, it also learns to associate the observations with the task ‘label’ given by the teacher.

## 4 Experiment

### 4.1 Setup

An experiment was carried out using real data and an emulated navigation process. We used a Nomad Scout equipped with an omni-directional camera to generate a data set in our corridor and three rooms. We placed the robot at a dense grid and stored the images together with their positions. Figure 3 depicts our corridor with the positions where images were taken. A subset of these images were used as the set of stored views  $\{\mathbf{z}^1, \dots, \mathbf{z}^{43}\}$ . The rest of the images were used to emulate the motion of the robot during training, so that we could guarantee that each task was learned under identical conditions.

The robot had to learn three different tasks independently. The destination of these tasks were:

- a Room A
- b Room B
- c The Lab

A task was executed correctly if the robot navigated into the destination room. Eventually the robot should be able to do this from any position in figure 3.

We implemented the architecture from figure 1 as follows:

- z** The observation were 120 by 720 gray scale panoramic images obtained from the omni-directional camera.
- D** For the  $D$ , matching features in two panoramic images were used to robustly estimate the epipolar geometry. The  $D$  follows from the decomposition of the essential matrix. For details see [18].

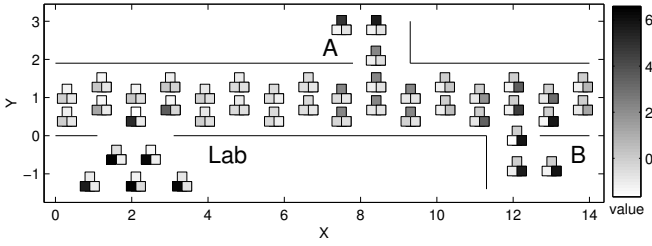


Figure 4: The resulting values  $\mathbf{v}^i$  for each stored observation plotted at their poses. At each pose there are three squares whose gray scale indicates the values. The top square corresponds to task “Room A”, the bottom left to “Robot Lab” and the bottom right to “Room B”.

- u In this experiment the motion of the robot is constrained to the grid positions in figure 3. The robot always moved to a neighboring grid position that is closest to the position of the target image.
- v All values were initialize at a random value between  $-10^{-7}$  and  $10^{-7}$ .

The learning parameters:

- We started with learning rate  $\alpha = 0.9$  a discount factor  $\gamma = 0.9$  and exploration  $\epsilon = 0.99$ .
- The robot learned each task for 50 epoch of 100 steps.
- After each epoch  $\alpha$  and  $\epsilon$  were reduced by multiplying it with 0.95.

We made sure the robot was rewarded and punished consistently by emulating a human teacher.

- The robot was punished with  $r = -1$  for being in a room not corresponding with the destination of the task. Also the robot was punished for moving away from the door of the destination room.
- The robot was rewarded with  $r = 3$  for being in the destination room.

## 4.2 Result

The result after training is shown in figure 4. The values for the task with destination A is indicated by the top square for each pose. In room A they have a high value (very dark), in front of that room the values are lower (gray) and further away they are very low (white). The same can be said for destination Lab (bottom left square) and room B (bottom right square). This indicates that the robot is capable of executing the three different tasks.

Figure 4 could only be plotted because we measured the pose at which each image was taken. The robot did not have access to the pose information. All it knew for each image was  $\mathbf{v}$  containing the three values corresponding to the three tasks. The values are plotted as points in in figure 5. Navigation is based on selecting that target images that has the highest value for the task. So executing a task corresponds to moving away

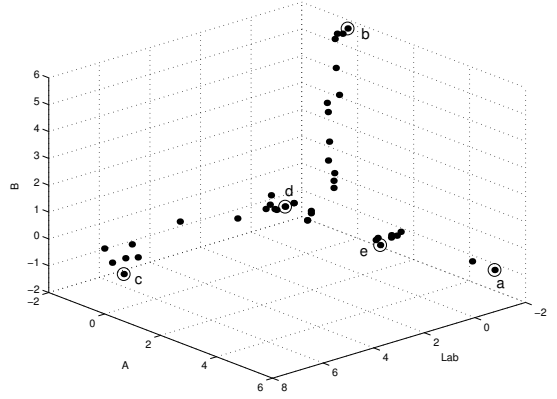


Figure 5: The values  $\mathbf{v} \in \mathbb{R}^3$  for all images plotted as dots. Each axis corresponds to the value of one task. A circle is drawn around five of them and the labels correspond to the images in figure 6.



Figure 6: The panoramic images corresponding to the labels in figure 5.

from the origin in figure 5. When the robot reaches the images with the highest value, the task is completed.

In figure 5 five points are selected (indicated with the circles). They are:

- a The highest value for destination A
- b The highest value for destination B
- c The highest value for destination Lab
- d The closest to the origin
- e Between highest value for room A and the origin

Figure 6 shows the corresponding images. The images in figure 6(a) and 6(c) show room A and the Lab and the image in figure 6(b) was taken in the door opening of room B. The image in figure 6(d) was taken somewhere in the corridor and the image in figure 6(e) in the corridor in front of room A. This indicates that once the robot is capable of executing a certain task, the values can be used to associate perceptions with the destination of the task.

Figure 5 shows that most points are placed along the axis. There is no point somewhere on a straight line from point a to point c (average values for both tasks). This suggests that there is no image that is good as target for going from room A to the Lab or visa versa. The only way to go from point a to points c is by reducing the value corresponding to destination room A. This will lead to a next target images close to point d, suggesting that the only way from room A to the lab is through the corridor. This indicates that values can also provide a clue about the structure of the environment.

## 5 Conclusion

In this paper we presented a mobile robot navigation method that uses a database of stored observations as representation of the environment. We have shown that it was capable of learning to move to different destinations based on rewards and punishments provided by a teacher. Also we have shown that results of learning can be interpreted to provide some information about the environment.

## References

- [1] P. Althaus and H.I. Christensen. Behaviour coordination in structured environments. *Advanced Robotics*, 17(7):657–674, 2003.
- [2] C.S. Anderson, S.D. Jones, and J.L. Crowley. Appearance based processes for visual navigation. In *Proc. of the 5th Int. Symposium on Intelligent Robotic Systems (SIRS)*, pages 227–236, 1997.
- [3] C.G. Atkeson and S. Schaal. Robot learning from demonstration. In *Proc. of the 4th Int. Conf. on Machine Learning (ICML)*, pages 12–20, 1997.
- [4] C. Balkenius. Spatial learning with perceptually grounded representations. *Robotics and Autonomous Systems*, 25:165–175, 1998.
- [5] J.Y. Donnart and J.A. Meyer. Spatial exploration, map learning and self-positioning with MonaLisa. In *From Animals to Animats 4*, 1996.
- [6] M.O. Franz, B. Schölkopf, P. Georg, H.P. Mallot, and Bülthoff. Learning view graphs for robot navigation. *Autonomous Robots*, 5:111–125, 1998.
- [7] P. Gaussier, S. Moga, J.P. Banquet, and M. Quoy. From perception-action loops to imitation processes: A bottom-up approach of learning by imitation. In *Socially Intelligent Agents, AAI Fall Symposium*, pages 49–54, 1997.
- [8] G. Hayes and J. Demiris. A robot controller using learning by imitation. In *Proc. of the 2nd Int. Symposium on Intelligent Robotic Systems (SIRS)*, Grenoble, France, 1994.
- [9] L.P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 1996.
- [10] V. Klingspor, J. Demiris, and M. Kaiser. Human-robot-communication and machine learning. *Applied Artificial Intelligence*, 11(7/8), 1997.
- [11] M.J. Mataric. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 1997.
- [12] M. Niolescu and M. J. Mataric. Learning and interacting in human-robot domains. *IEEE transaction on Systems, Man and Cybernetic*, 2001.
- [13] M. Niolescu and M.J. Mataric. Learning task representations from experienced demonstrations. In *Proc. of the AAI Fall Symposium on Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems*, 2001.
- [14] D. Pierce and B.J. Kuipers. Learning to explore and build maps. In *National Conference on Artificial Intelligence*, pages 1264–1271, 1994.
- [15] L. Saebra Lopes and Q.H. Wang. Towards grounded human-robot communication. In *Proc. IEEE Int. Workshop on Robot and Human Interactive Communication*, pages 312–318, 2002.
- [16] R.S. Sutton. Generalizing in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems (NIPS8)*, 1996.
- [17] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
- [18] S.H.G. ten Hagen. Good features to map. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1512–1517, 2003.
- [19] S.H.G. ten Hagen and B.J.A. Kröse. Learning to navigate using a lazy map. In *Proc. of the 11th Int. Conf. on Advanced Robotics (ICAR)*, pages 299–304, 2003.
- [20] S. Thrun, J.-S. Gutmann, D. Fox, W. Burgard, and B. Kuipers. Integrating topological and metric maps for mobile robot navigation: A statistical approach. In *AAAI/IAAI*, pages 989–995, 1998.
- [21] N. Tomatis, I. Nourbakhsh, and R. Siegwart. Hybrid simultaneous localization and map building: Closing the loop with multi-hypotheses tracking. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 2002.
- [22] O. Trullier and J. Meyer. Animat navigation using a cognitive graph. In *From Animals to Animats 5*, pages 213–222, 1998.