Spatial Information Extraction for Cognitive Mapping with a Mobile Robot

Jochen Schmidt[1], Chee K. Wong[1], and Wai K. Yeap[2]

[1] Centre for Artificial Intelligence Research
Auckland University of Technology, Auckland, New Zealand
[2] Department of Artificial Intelligence, University of Malaya,
Kuala Lumpur, Malaysia

Abstract. When animals (including humans) first explore a new environment, what they remember is fragmentary knowledge about the places visited. Yet, they have to use such fragmentary knowledge to find their way home. Humans naturally use more powerful heuristics while lower animals have shown to develop a variety of methods that tend to utilize two key pieces of information, namely distance and orientation information. Their methods differ depending on how they sense their environment. Could a mobile robot be used to investigate the nature of such a process, commonly referred to in the psychological literature as cognitive mapping? What might be computed in the initial explorations and how is the resulting "cognitive map" be used for localization? In this paper, we present an approach using a mobile robot to generate a "cognitive map", the main focus being on experiments conducted in large spaces that the robot cannot apprehend at once due to the very limited range of its sensors. The robot computes a "cognitive map" and uses distance and orientation information for localization.

1 Introduction

Since Tolman [1] suggested that animals (including humans) create a representation of the environment in their minds and referred to it as a "cognitive map", many psychological experiments have been conducted to study the nature of cognitive maps (see [2] for a review). Some of the important characteristics of cognitive maps highlighted by these studies include distorted information about distances and directions, landmarks, places, and paths, a hierarchical organization as well as multiple frames of reference. Many models of cognitive maps have also been proposed and one idea appears to be most prominent, namely that the map begins with some form of a network of "place representations". Several computational theories of cognitive mapping have been published since then, including the works of Poucet [3], Chown, Kaplan and Kortenkamp [4], Kuipers [5], and Yeap and Jefferies [6].

More recently, researchers began to use mobile robots to test ideas about cognitive mapping as opposed to robot mapping. In robot mapping, one is concerned with the development of efficient algorithms for the robot, with its particular

sensors, to simultaneously localize and map its environment (SLAM). For some examples of recent work in this area see [7] and the references therein. The maps acquired via this paradigm are precise. By "precise", we refer to the fact that every surface the robot encounters is remembered and its position is known with a certain degree of accuracy. The robot also knows its position in the environment. In contrast, the mapping process used by humans (and animals), referred to as cognitive mapping, produces an imprecise map initially, which later turns into a representation laden with one's own interpretations and experiences of the world. The map produced in such a process is known as a cognitive map.

For example, Kuipers and his team have been experimenting with robots to find ways to compute his Spatial Semantic Hierarchy from the ground up [8]. Both the gateway construct in the PLAN model of cognitive mapping [4] and the use of exits in the Absolute Space Representation (ASR) model of cognitive mapping [6] were tested on a mobile robot: refer to [9] for the former and to [10] for the latter. Also, ideas about cognitive mapping based upon neurological findings were being tested using mobile robots. Examples of such work include [11, 12]. However, many of these attempts produced algorithms that were more an inspiration from observations about cognitive mapping than a test-bed for theories of cognitive mapping. These researchers were concerned on their robots successfully mapping their environments. Hence, instead of investigating cognitive mapping, they ended up trying to solve the robot mapping problem.

Our goal in this paper differs. Different animals compute cognitive maps using different sensors, and therefore our robot should be treated as a kind of animal with its own peculiar sensing capabilities. For unknown reasons, humans do not remember a precise map after one or two visits to a new environment. We assume animals do not too, and so neither should our robot. To investigate our robot's cognitive mapping process, it is thus best to have our robot compute an imprecise map first and then investigate animal-like strategies for finding its way home using such a map. It is argued that the behavior of such a robot might shed light on cognitive mapping. Refer to [13] for a review on biological navigation approaches implemented on robots.

To do so, we use a robot equipped with sonar sensors to compute a description of each local space visited. The robot's "cognitive map"[3] is thus a network of such local spaces. Following the theory of cognitive mapping as presented in [6], we refer to each local spaces computed as an Absolute Space Representation (ASR). With sonar sensors, the description of each ASR computed (or more precisely, the shape computed) is not accurate enough to allow its identification on its return journey, even more so in large environments. As lower animals (especially rats) have been observed to use distance and direction information encoded in their cognitive map to find their way [14], we implemented similar strategies for the robot.

[3] by definition, a robot cannot have a cognitive map. However, rather than being verbose and say "using a robot to simulate computing a cognitive map", we will simply say the robot computes a cognitive map.

The algorithms presented here are based on our previous work on robot and cognitive mapping [15–17]. They have been considerably extended to be able to cope with "large" spaces (for a robot with sonar sensors), and new experimental results are shown here. The spaces considered are too large for the robot to be apprehended at once due to the limited range of its sonar sensors.

Section 2 describes the way our robot computes its "cognitive map". Section 3 presents the two strategies that our robot uses to localize itself in the environment. Section 4 shows the results of our experiments, the main focus being on large spaces.

2 Generating an Absolute Space Representation

When exploring the environment for the first time, the robot creates a "cognitive map" of its environment, i. e., a network of local spaces visited, namely ASRs. The process of generating this topological space representation from sonar information gathered while mapping the environment is described in the following.

For mapping, we use a mobile robot equipped with eight sonar sensors and an odometer. The robot acquires sonar readings while moving on a "straight" line (we are not concerned about drift compensation or correction) until it runs into an obstacle. At this point an obstacle avoidance algorithm is used, after which the robot can wander straight on again. A single one of these straight movements will be called *robot path* throughout this paper. The starting point of the algorithm is a geometric map that contains the robot movement path as well as linear[4] approximations of surfaces generated from the original range data. The goal is to split the map into distinct regions, e. g., corridors and rooms. One of the main problems at this stage is that the range of the sonar sensors (for our robot about 4m) is not sufficient for apprehending large rooms at once with range data acquired from a single position. Therefore, the robot has to travel through the environment, which results in distorted maps due to odometry errors. The algorithm presented here is capable of handling these problems, as will be shown in the experiments' section.

Splitting is done along the robot movement path, using an objective function that computes the quality of a region, based on local metric features derived from the geometric map, such as the average room width (corridors are long and narrow compared to rooms) and overall direction (e. g., a corridor is separated from another one by a sharp bend in the wall). No fixed thresholds are used, so the algorithm can balance the influence of the separate criteria used as needed.

2.1 Split and Merge

The basis of the ASR generating algorithm is the well-known split and merge method [18–20], which originated in pattern recognition. A classic application

[4] this is not mandatory; any approximation will be fine as long as the total area of a region can be computed

of this algorithm is finding piecewise linear approximations of contour points that have been detected in an image. A variety of other applications has been proposed, including segmentation of image regions given a homogeneity criterion, e. g., with respect to color or texture [20]. The split and merge algorithm is the core part of the process resulting in a topological ASR representation of the environment. Therefore, it is described here, based on the classic version for contour approximation.

The input data of split and merge is a sorted set of (contour) points, which is to be approximated. A parametric family of functions $\mathcal{F}$ (e. g., lines) to be used has to be chosen, as well as a metric for computing the residual error ϵ of the resulting approximation (usually root mean square error), or, when used for regions, a homogeneity or quality criterion. The algorithm results in a piecewise approximation of the original points, where every single residual error is below a given threshold θ_s. The single steps of the algorithm are [20]:

1. Start with an initial set of points $\mathcal{P}^0$ consisting of n_0 parts $\mathcal{P}_0^0, \ldots, \mathcal{P}_{n_0-1}^0$. Each part $\mathcal{P}_i^0$ is approximated by a function from $\mathcal{F}$. Compute the initial residual error ϵ_i^0 for each part of $\mathcal{P}^0$.
2. Split each $\mathcal{P}_i^k$ where $\epsilon_i^k > \theta_\mathrm{s}$ into two parts $\mathcal{P}_j^{k+1}$ and $\mathcal{P}_{j+1}^{k+1}$, compute the approximation and residuals ϵ_j^{k+1}, ϵ_{j+1}^{k+1}. Repeat until $\epsilon_i^k \leq \theta_\mathrm{s}\ \forall i$
3. Merge two adjacent parts $\mathcal{P}_i^k$, $\mathcal{P}_{i+1}^k$ into one new part $\mathcal{P}_j^{k+1}$ if $\epsilon_j^{k+1} \leq \theta_\mathrm{s}$. Repeat until merging is not possible any more.
4. Shift the split point shared by two adjacent parts $\mathcal{P}_i^k$, $\mathcal{P}_{i+1}^k$ to left and right while leaving the overall number of parts fixed. Keep the split that reduces the overall error, repeat until no further changes occur.

2.2 Region Splitting of the Map

Before a region split and merge algorithm on the geometric map can be applied, it is necessary to create an initial split of the map. The easiest way to do so is to treat the whole map as a single large region defined by the start and end points of the journey. More sophisticated initializations can be used as well, e. g., based solely on the robot movement without taking into account range data [15].

After the initialization step, the actual division of the map into distinct regions is performed based on a split and merge that uses a residual error function $h(\mathcal{P}_i, \mathcal{P}_j)$ which compares two regions $\mathcal{P}_i$ and $\mathcal{P}_j$ and computes the homogeneity of the two regions (low values of $h(\mathcal{P}_i, \mathcal{P}_j)$ means homogeneous, high values very inhomogeneous). This function is used during the split phase for deciding whether a region $\mathcal{P}_i^k$ will be split again at a given position into two new regions $\mathcal{P}_j^{k+1}$ and $\mathcal{P}_{j+1}^{k+1}$, and in the merge (or shift) phase to determine whether two adjacent regions can be merged (or the splitting point be shifted). When the homogeneity is above a given threshold θ_r, the region will be split again.

The basic idea is to use the average width of a region in the map as a criterion for splitting, as a width change resembles a changing environment, e. g., a transition from a corridor to a big room. The homogeneity (residual)

function used is:

$$h(\mathcal{P}_i, \mathcal{P}_j) = \frac{\max\{f_\mathrm{w}(\mathcal{P}_i), f_\mathrm{w}(\mathcal{P}_j)\}}{\min\{f_\mathrm{w}(\mathcal{P}_i), f_\mathrm{w}(\mathcal{P}_j)\}} + s_\mathrm{r} r(\mathcal{P}_i, \mathcal{P}_j)\,, \tag{1}$$

where $f_\mathrm{w}(\mathcal{P}_i)$ is the average width of region $\mathcal{P}_i$, and $r(\mathcal{P}_i, \mathcal{P}_j)$ is a regularization term that takes care of additional constraints during splitting. The factor s_r controls the influence of $r(\mathcal{P}_i, \mathcal{P}_j)$.

Obviously, the average width is given by $f_\mathrm{w}(\mathcal{P}_i) = \frac{A_{\mathcal{P}_i}}{l_{\mathcal{P}_i}}$, where $A_{\mathcal{P}_i}$ is the area of region $\mathcal{P}_i$, and $l_{\mathcal{P}_i}$ is its length. The definition of the length of a region in particular is not always obvious, but can be handled using the robot movement paths, which are part of each region. The length $l_{\mathcal{P}_i}$ is then defined by the length of the line connecting the start point of the first robot path of a region and the end point of the last path of the region, i. e., the line connecting the exits of an ASR, which the robot used while travelling through the environment. This is an approximation of a region's length having the advantage that disturbance caused by zig-zag movement of the robot during mapping does not affect the end result.

For area computation, the gaps contained in the map have to be taken into account, either by closing all gaps, or by using a fixed maximum distance for gaps. Closing a gap is a good approach if it originated from missing sensor data, but may distort the splitting result when the gap is an actual part of the environment. Closing it would make the region appear smaller than it actually is. Our implementation uses a combination of methods: small gaps are closed in a pre-processing step, large ones are treated as distant surfaces.

Depending on how gaps are handled, the algorithm possibly creates a large number of very small regions. This is where the regularization term $r(\mathcal{P}_i, \mathcal{P}_j)$ comes in: it ensures that regions do not get too small. It penalizes small regions but still allows to create them if the overall quality is very good. We use a sigmoid function centered at n, which is the desired minimum size of a region:

$$r(\mathcal{P}_i, \mathcal{P}_j) = \frac{1}{1 + \exp\left(-\frac{\min\{A_{\mathcal{P}_i}, A_{\mathcal{P}_j}\}}{A_\mathrm{max}} + n\right)} - 1\,. \tag{2}$$

This function can assume values values between -1 and 0. The exponent is basically the ratio of the area of the smaller one of two adjacent regions to the maximum area A_max of the smallest possible region that the algorithm is still allowed to create. Thus, the smallest ratio is 1. It increases when the region gets larger.

The regularization term only has an influence when a region is already small, making it less likely to be split again. As the sigmoid reaches 0 asymptotically, it has virtually no influence when a region is large. The overall influence of the regularization can be controlled by the factor s_r in (1). It is given by $s_\mathrm{r} = s\theta_\mathrm{r}$, where $0 \leq s \leq 1$ is set manually and defines the percentage of the threshold θ_r that is to be used as a weight. θ_r is the threshold introduced earlier, which determines that a region is to be split into two when the first region is θ_r times larger than the second one.

3 Localization Strategies

In this section we describe the strategies used for localization based on ASR information, which is extracted from two sources, namely a map that has been generated on the outward journey, i. e., while the robot was exploring the environment, and a second map which is being generated during the homeward journey. A data fusion algorithm is applied for merging localization information computed by separate simple strategies with varying reliability. Each strategy by itself may be not sufficient for the robot to localize itself in the environment, but the combined result is. The fusion is based on the *Democratic Integration* technique proposed by Triesch and von der Malsburg [21, 22]. Originally it was developed for the sensor data fusion in computer vision, and uses images as input data. The method has been extended and embedded into a probabilistic framework in [23, 24], still within the area of machine vision. We have amended the original approach such that instead of images, information extracted from the ASRs is used. A main advantage of the fusion approach is that the extension is straightforward, i. e., more localization strategies can be added easily.

Two different strategies for localization of the robot with respect to the original map generated on its way to the current position are presented in the following. Each method computes a local confidence map that contains a confidence value between zero and one for each ASR of the original map. Note that these confidence values are not probabilities, and they do not sum up to one; the interval has been chosen for convenience, and different intervals can be used as desired. Refer to [21, 22] for further details regarding this matter.

As the "cognitive maps" generated are a topological representation enhanced by metric information, there is no need to correct them for odometry drift. It is important to keep this in mind when developing new localization strategies: it necessitates the use of local information only, i. e., information extracted from adjacent ASRs, or information that can be computed without having to worry about any negative influence originating from odometry errors.

3.1 Distance

Just as humans have a rough notion of how far they walked starting at a certain location, so should a robot. Note that we are not talking about exact measurements, but rather about whether the robot has travelled, say, 5m or 10m. Also, we do not use the actual distance travelled as provided by odometry, because the robot often moves in a zig-zag fashion rather than straight, which would result in quite different distances for each journey through the same space. Particularly for large spaces, the odometry readings also depend highly on which path the robot actually took when crossing empty space, and whether it has explored the space wandering around multiple times before exiting. Our proposed solution for this problem is to use distance information obtained from the "cognitive maps" computed during outward and homeward journeys. These maps have been split into distinct ASRs, and the length of each ASR can be computed, which is defined by the distance between the entrance and the exit the robot used when

passing through (cf. Sect. 2.2). In the maps shown, e. g., in Fig. 1, start and end points of an ASR are depicted by dark dots (split points) located on a set of connected lines representing the path the robot took. The zig-zag movement of the robot in between two splits is clearly visible, and can be quite different from the line connecting start and end points, in particular if an ASR has been generated from a large empty space. The strategy for computing a local confidence map that can later on be used for data fusion, is to compare the distance d travelled when returning home, measured in ASR lengths taken from the intermediate map computed on the return journey, to the ASR lengths taken from the original map computed during the mapping process.

The local confidence map $\boldsymbol{c}_{\mathrm{Dist}} \in \mathbb{R}^N$ (N being the total number of ASRs in the original map) is computed as follows: Each ASR's confidence is dependent on the overall distance d travelled on the return journey; the closer an ASR is to this distance from the origin, the more likely it is the one the robot is in currently. To model the confidences for each ASR we use a Gaussian, the horizontal axis being the distance travelled in mm, centered at the current overall distance travelled d. Its standard deviation σ is dependent on the distance travelled, and was chosen as $\sigma = 0.05d$. Note that although a Gaussian is used here, we do not try to model a probability density function, but rather make use of the bell-shape it provides. There are a number of reasons making it most suitable for our purpose: It allows for a smooth transition between ASRs, and the width can be easily adjusted by altering the standard deviation. This is necessary as the overall distance travelled gets more and more unreliable (due to slippage and drift) the further the robot travels.

The confidence value for a particular ASR is determined by sampling the Gaussian at the position given by the accumulated (ASR-)distances from the origin (i. e., where the robot started the homeward journey) to the end of this ASR. After a value for each ASR is computed, the local confidence map $\boldsymbol{c}_{\mathrm{Dist}}$ is normalized to the interval $[0; 1]$.

3.2 Relative Orientation

The second method for computing local confidence maps containing estimates of the robot's position with respect to the original map is based on using relative orientation information between adjacent ASRs. During its journey, the robot enters an ASR at one location and exits at a different one, usually including zig-zag movements in between. We define the direction of an ASR as the direction of the line connecting the entrance and exit points. As before, direction information varies every time the robot travels through the environment, but the overall shape between adjacent ASRs is relatively stable. Therefore, we propose to use angles between ASR directions as a local measure of the current position of the robot. Note that this information is pretty much useless on its own, because the same angles (i. e., direction changes) can be found in different locations of the environment. However, combining this strategy with others can help to decide between position estimates that would otherwise be indistinguishable.

Firstly, all angles $\alpha_1, \ldots, \alpha_{N-1}$ between adjacent ASRs in the original map are computed. This can be done offline, as this map is fixed during the homeward journey. In the re-mapping process while returning home, new ASRs are computed in the new map based on data gathered while the robot travels. Using the direction information contained in this map, the angle β between the current ASR and the previous one can be computed. Comparing this angle to all angles of the original map gives a clue (or multiple clues) for the current location of the robot.

The comparison of angles is done by computing the difference angle between the current angle β obtained from the newly generated map and all angles α_i of the original map. This difference angle is then mapped linearly to the interval $[0; 1]$:

$$\boldsymbol{c}_{\mathrm{Dir}\,i} = -\frac{1}{\pi}|\alpha_i - \beta| + 1, \quad i = 1, \ldots, N-1 \quad . \tag{3}$$

Another obvious choice would be to use the cosine of the difference angle instead of the linear mapping. However, this would "compress" confidence values for similar angles, which is not a desired effect.

The mapping given by (3) results in high values for similar angles and low values for dissimilar ones. The confidence map computed this way can already be used for further processing. Since the overall reliability of the relative orientation strategy as described above is rather low compared to the confidence values from other methods (in this case using distance information), we currently reduce the confidence values by a constant factor. As the data fusion method presented in the next section is capable of adjusting the relative weights of the different localization strategies, this is non-critical, because it will change automatically over time anyway, depending on the reliability of the other methods used.

3.3 Fusion of Strategies

The separate local confidence maps are merged into a single global one based on the Democratic Integration method proposed in [21, 22]. Fusion itself is done by computing a weighted sum of all local confidence maps, which is straightforward and not a new concept. However, Democratic Integration allows for the weights to be adjusted dynamically and automatically over time, dependent on the reliabilities of the local map.

Given M local confidence maps $\boldsymbol{c}_{\mathrm{l}i}(t)$ at time t generated using different strategies, the global map $\boldsymbol{c}_{\mathrm{g}}(t)$ is computed as:

$$\boldsymbol{c}_{\mathrm{g}}(t) = \sum_{i=0}^{M-1} w_i(t)\boldsymbol{c}_{\mathrm{l}i}(t) \quad , \tag{4}$$

where $w_i(t)$ are weighting factors that add up to one. In this paper, $M = 2$, namely the local confidence maps based on distance ($\boldsymbol{c}_{\mathrm{Dist}}$) and relative orientation ($\boldsymbol{c}_{\mathrm{Dir}}$).

An estimate of the current position of the robot with respect to the original map can now be computed by determining the largest confidence value in $\boldsymbol{c}_{\mathrm{g}}(t)$.

Its position b in $\boldsymbol{c}_{\mathrm{g}}(t)$ is the index of the ASR that the robot believes it is in. The confidence value c_{g_b} at that index gives an impression about how reliable the position estimate is in absolute terms, while comparing it to other ASRs shows the relative reliability.

In order to update the weighting factors, the local confidence maps have to be normalized first. The normalized map $\boldsymbol{c}'_{\mathrm{l}_i}(t)$ is given by:

$$\boldsymbol{c}'_{\mathrm{l}_i}(t) = \frac{1}{N}\boldsymbol{c}_{\mathrm{l}i}(t) \quad . \tag{5}$$

Recall that N is the total number of ASRs in the original map. The idea when updating the weights is that local confidence maps that provide very reliable data get higher weights than those which are unreliable. Different ways for determining the quality of each local confidence map are presented in [22]. We use the normalized local confidence values at index b, which has been determined from the global confidence map as described above, i. e., the quality $q_i(t)$ of each local map $\boldsymbol{c}_{\mathrm{l}i}(t)$ is given by $c'_{\mathrm{l}b}(t)$. Normalized qualities $q'_i(t)$ are computed by:

$$q'_i(t) = \frac{q_i(t)}{\sum_{j=0}^{M-1} q_j(t)} \quad . \tag{6}$$

The new weighting factors $w_i(t+1)$ can now be computed from the old ones:

$$w_i(t+1) = w_i(t) + \frac{1}{t+1}\left(q'_i(t) - w_i(t)\right) \quad . \tag{7}$$

This is a recursive formulation of the average over all qualities from time zero to t. Using this update equation and the normalization of the qualities in (6) ensures that the sum of the weights equals one at all times.

4 Experimental Results

For experimental evaluation we have used a Pioneer 3 robot from MobileRobots Inc (formerly Activmedia Robotics), equipped with eight sonar sensors and an odometer. Using sonar sensors only instead of, say, a laser range finder was a deliberate choice, as the restrictions imposed by the limited range of sonar sensors allow us to better test the cognitive mapping algorithms and the behaviour of the robot in large spaces. The robot must first explore and generate a representation of the environment, i. e., a "cognitive map". This is called the "outward journey" further on, the map is called "original map". At some point the robot is stopped and turned around. On its way back home (the "homeward journey") it remaps the environment and computes a new "cognitive map", which is used in conjunction with the original map for localization. Computation of the "cognitive map" and localization is done each time the robot stops, which is normally due to an obstacle in its way. Note that we turn off the robot and change its position slightly before it is allowed to start the remapping process. Therefore, the maps generated during the outward and homeward journeys are recorded in different

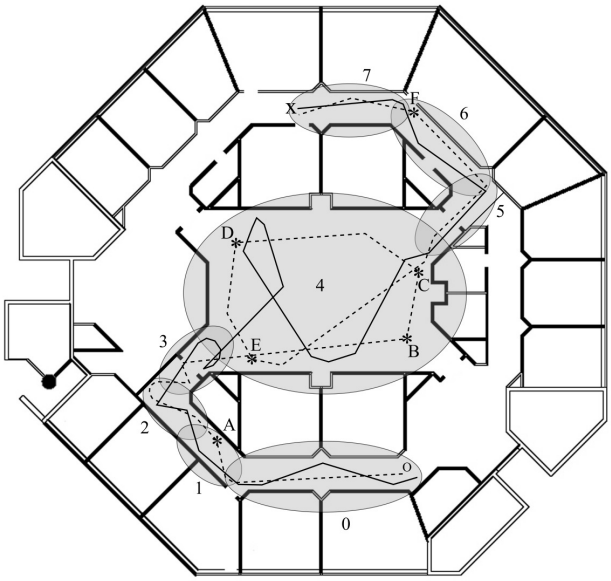
global coordinate systems, and the robot has no way of aligning them. In the following we show experimental results for generating the spatial representation of the environment based on the split and merge method presented in Sect. 2, as well as experiments on how the robot uses this inexact "cognitive map" for localization on its way back home using the localization method from Sect. 3. The parameters used for computing the maps were the same for all runs, namely $\theta_{\mathrm{r}} = 1.7$ and $s = 0.2$.

The range of the sonar sensors is about 4m; consequently, spaces larger than the sonar range cannot be apprehended with a single scan, but the robot has to move in order to build a spatial representation of the environment. Remember that the "cognitive map" is a metric-topological representation rather than a purely metric one, and the data acquired during mapping is not corrected for odometry errors. This is one of the main advantages of choosing a cognitive mapping approach over more traditional robot mapping techniques like SLAM. Obviously odometry errors will be visible in the generated maps, but as long the algorithm is able to generate a single ASR as opposed to multiple ones from the data acquired, say, in a large room, this does not pose a problem.

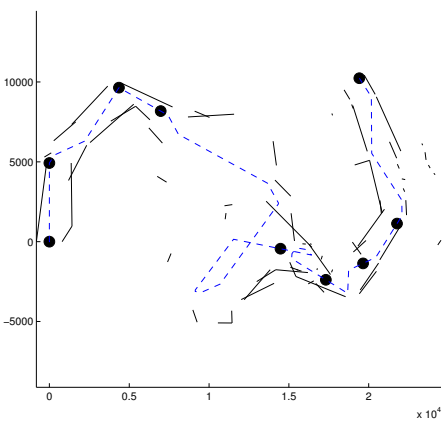
We conducted various experiments in an office environment that contains a large room having dimensions of approximately 9m × 15m. In particular when the robot is close to the center of this room, it does not obtain any sensor readings. Depending on the angle of movement, this may even happen at positions farther from the room center. Two experiments have been selected for a detailed discussion in this paper. Each experiment consists of the original map generated on the outward journey, and maps computed on the way back home as well as confidence maps used for localization. The experiments are labelled *Experiment 1* and *Experiment 2*, respectively.

Figures 1(a) and 2(a) show the layout of the environment used, the large room being in the center of the building. The paths that the robot took during mapping (solid line) and going home (dashed line) are both shown as well. For the mapping stage, the robot started at the location marked by 'X' and was stopped at a random position, turned around and started the homeward journey nearby (marked by 'O'). The return journey stopped when the robot believed that it reached home, or, more precisely, the ASR that contains the start location 'X'. The ASRs generated during the outward journey are indicated by ellipses and numbered from zero starting at the *last* ASR (so that during the homeward journey ASRs are visited in ascending order).

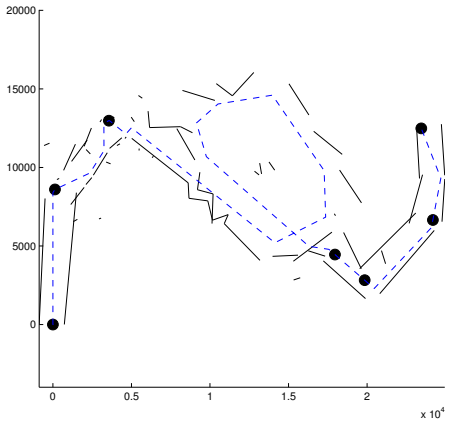
The "cognitive map" generated during the outward journey in *Experiment 1* is shown in Fig. 1(b), the map computed on the homeward journey in Fig. 1(c); likewise for *Experiment 2* in Figs. 2(b) and 2(c). Note that the paths the robot took during outward and homeward journey are quite different, particularly when it maps the big room. All units are given in millimeters, black dots indicate the split points between ASRs. Due to the re-initialization of the robot before returning home, the starting point for both, outward and homeward journeys, is the origin of the coordinate system, which also results in the map depicted in Fig. 1(c) to be upside-down with respect to the map in Fig. 1(b). When comparing



(a) Actual floor plan Experiment 1

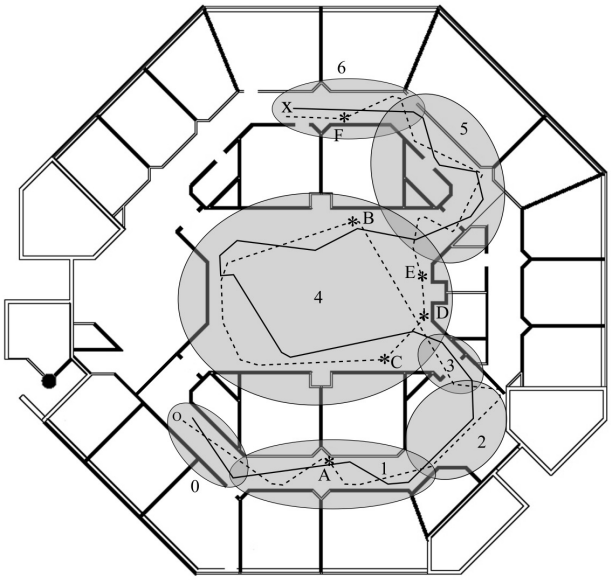


(b) Map outward journey Exp. 1

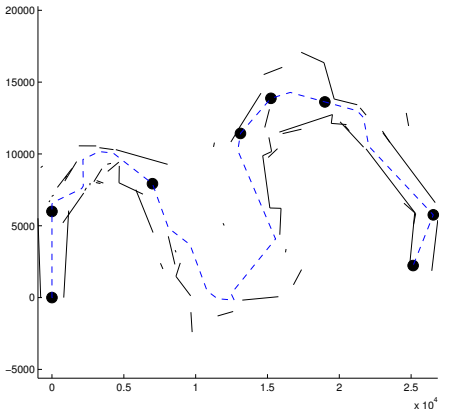


(c) Map homeward journey Exp. 1

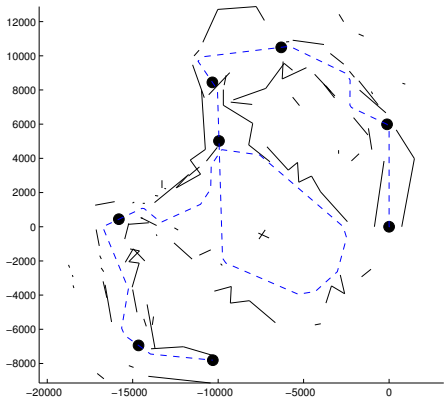
Fig. 1. Experiment 1: (a) Map showing the actual layout of the building. The path the robot took during the outward journey is shown as a solid line, with 'X' marking the starting location. The dashed line visualizes the path for the homeward journey and 'O' marks its starting point. ASRs generated during the outward journey are indicated by ellipses. The labels A - F indicate positions during the homeward journey referred to in the text and in Fig. 3. (b) Map generated during outward journey ("original map"). (c) Map generated during homeward journey.



(a) Actual floor plan Experiment 2



(b) Map outward journey Exp. 2



(c) Map homeward journey Exp. 2

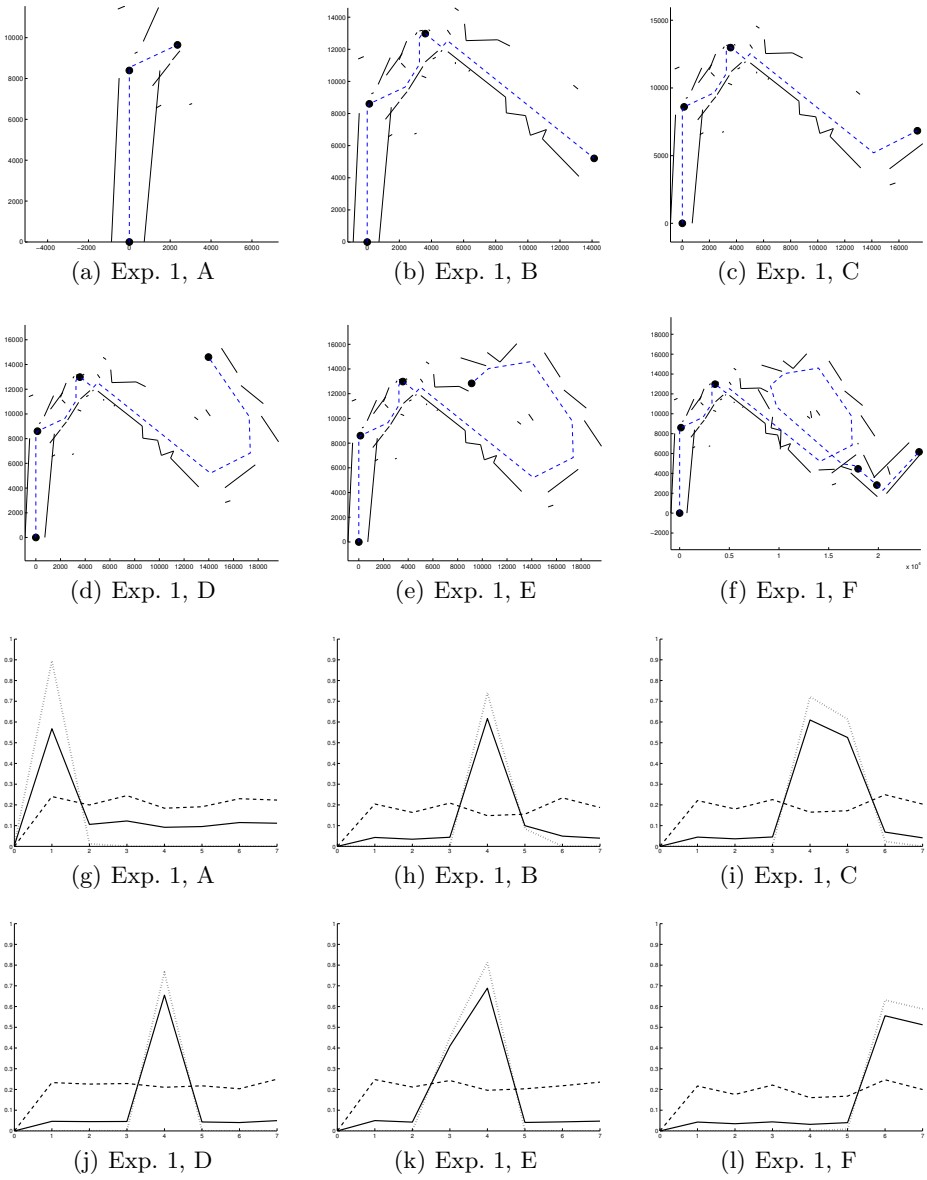
Fig. 2. Experiment 2: (a) Map showing the actual layout of the building. The path the robot took during the outward journey is shown as a solid line, with 'X' marking the starting location. The dashed line visualizes the path for the homeward journey and 'O' marks its starting point. ASRs generated during the outward journey are indicated by ellipses. The labels A - F indicate positions during the homeward journey referred to in the text and in Fig. 4. (b) Map generated during outward journey ("original map"). (c) Map generated during homeward journey.

the maps generated during both journeys, it becomes obvious that different representations may be computed, in particular a different number of ASRs, and splits generated at different locations. This is due to sensory inaccuracies, but it does not pose a problem for the localization method we use. The position of the robot is always given with respect to the ASR representation of the original outward journey map. What can be clearly seen in all maps generated in both experiments is that the algorithm is capable of representing the large room as a single ASR, independent of the path that the robot took while mapping. The split points are nicely located near the exits of the room, as desired. Apart from that, the algorithm separates corridors from bigger rooms (e. g., ASR 1 and 2 in Fig. 2), and corridors from other corridors, most of the times at locations where a human would do so as well. Although robots and humans generating cognitive maps may not necessarily come up with alike representations due to the "hardware" being quite different, this is a desired result as we try to simulate a human cognitive mapping process.

We will now take a closer look at the (obviously incomplete) maps and localization information generated at intermediate stops during the homeward journey. For this purpose we have selected graphs computed at six positions on the homeward journey, which allow for interesting insights into how the localization performs. These positions are marked by '*' and labelled A - F in Fig. 1(a) (*Experiment 1*) and 2(a) (*Experiment 2*).

Figures 3(a) to 3(f) show intermediate maps generated at positions A - F in *Experiment 1* (cf. Fig. 1(a)). The corresponding confidence maps are depicted in Figs. 3(g) to 3(l). In the confidence graphs, the light dotted line shows the ASR estimate using the ASR length information (distance method) and the dark dashed line depicts the ASR estimate using the angles between ASRs (relative orientation method). The solid line is the overall estimate after fusion. The horizontal axis corresponds to the index of the ASR, the vertical axis to the confidence, which can have values between zero and one.

The confidence graph at position A in Fig. 3(g) shows a peak for the overall confidence at ASR 1, signifying the robot is very confident of being in this particular ASR. The same is true for the graph in Fig. 3(h) corresponding to position B, where the robot has moved far into the big room in the center being ASR 4. It can be seen that position C is close to the border between ASRs 4 and 5, which is reflected by high confidence values for both ASRs in the confidence map in Fig. 3(i), where the robot gets more and more unsure about whether it is still in the big room or whether it has entered the next region yet (ASR 5). As it moves further away from the exit and back into the room, reaching position D it is very confident again that it is still in ASR 4, not having actually exited the big room (see Fig. 3(j)). Position E is close to where the robot has entered the room, coming through ASR 3. Again, this can be observed in the confidences plotted in Fig. 3(k), which shows that the robot is still quite confident of being in ASR 4, but where the value for ASR 3 has increased considerably, i. e., the robot "knows" that it is close to where it was when it entered the room. Finally, the



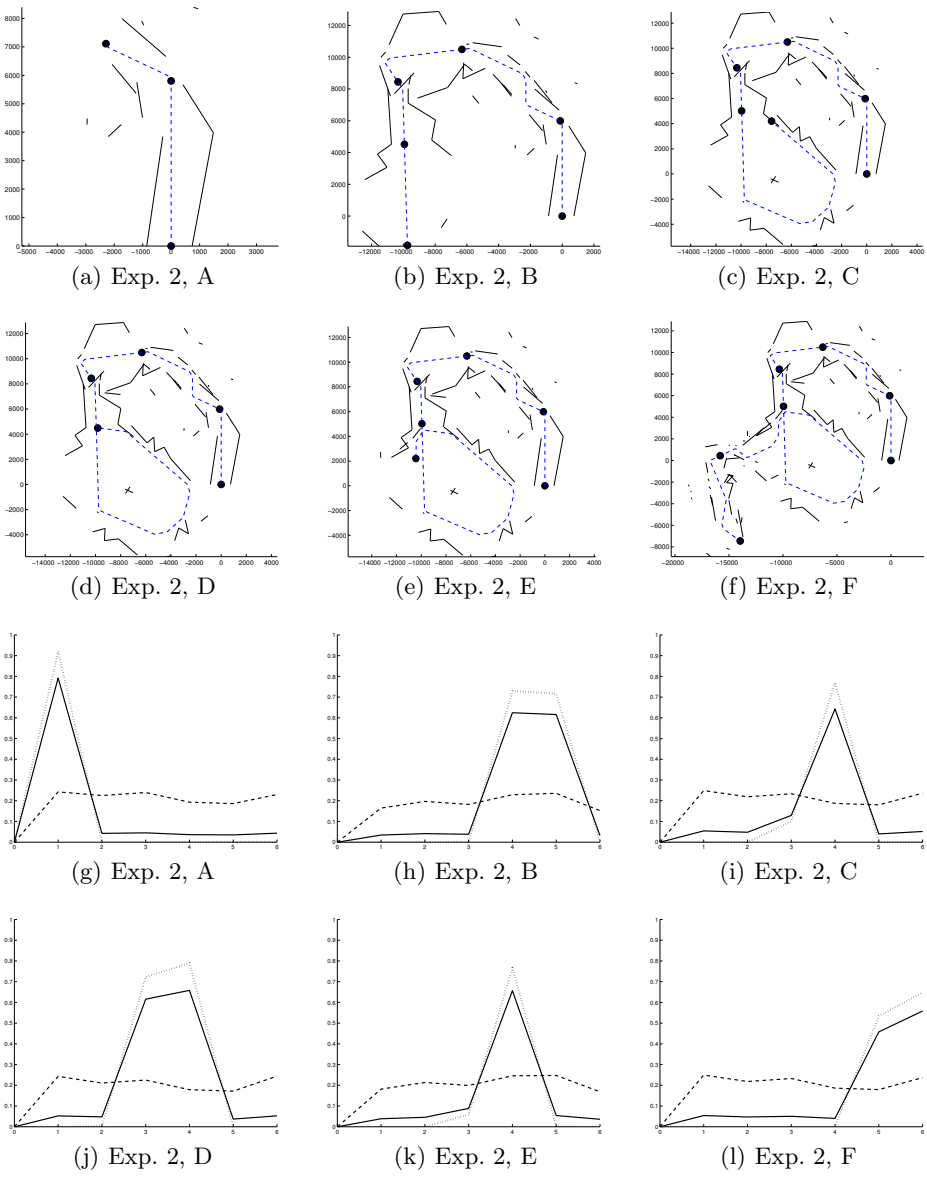
(a) Exp. 1, A (b) Exp. 1, B (c) Exp. 1, C

(d) Exp. 1, D (e) Exp. 1, E (f) Exp. 1, F

(g) Exp. 1, A (h) Exp. 1, B (i) Exp. 1, C

(j) Exp. 1, D (k) Exp. 1, E (l) Exp. 1, F

Fig. 3. Experiment 1: (a) - (f) Maps at intermediate stops A - F (cf. Fig. 1(a)); (g) - (l) Confidence maps corresponding to stops A - F: distance (light dotted), relative orientation (dark dashed), and overall confidence (solid). Horizontal axis: ASR number; vertical axis: confidence (0 to 1).



(a) Exp. 2, A (b) Exp. 2, B (c) Exp. 2, C

(d) Exp. 2, D (e) Exp. 2, E (f) Exp. 2, F

(g) Exp. 2, A (h) Exp. 2, B (i) Exp. 2, C

(j) Exp. 2, D (k) Exp. 2, E (l) Exp. 2, F

Fig. 4. Experiment 2: (a) - (f) Maps at intermediate stops A - F (cf. Fig. 2(a)); (g) - (l) Confidence maps corresponding to stops A - F: distance (light dotted), relative orientation (dark dashed), and overall confidence (solid). Horizontal axis: ASR number; vertical axis: confidence (0 to 1).

graph in Fig. 3(l) was generated at position F, which is at the border between ASRs 6 and 7, reflected by high confidences for both.

Figures 4(a) to 4(f) show intermediate maps generated at positions A - F in *Experiment 2* ((cf. Fig. 2(a))), and the corresponding confidence maps in Figs. 4(g) to 4(l). As in the previous experiment, when reaching position A the robot is very confident about being in ASR 1, indicated by the distinct peak in the graph in Fig. 4(g). At position B it has entered the big room being ASR 4. As can be seen in Fig. 4(h) it is not sure about whether it is still in ASR 4, or whether it has reached the next ASR already, the confidences for both being equally high, with a slight bias towards ASR 4. Taking into account odometry inaccuracies and the information that is available to the robot using the present localization methods, this behaviour makes perfect sense. When it reaches position C after travelling through the big room in a loop, the robot is highly confident (Fig. 4(i)) that it is still in ASR 4 (which is true for the positions in between as well, which are not shown). Being close to where it has entered the room, the confidence for ASR 3 starts to increase at this location. This effect becomes more prominent in Fig. 4(j), when the robot is at position D, which is at the border between ASRs 3 and 4. Again, it is unsure about which ASR it is in, meaning that it can infer that it actually is at the border between two regions, near the entrance to the room. Moving away from the border to position E (Fig. 4(k)), it is again confident about being in ASR 4. At the last position F, the robot is very close to its home position in ASR 6 already, which is reflected in the confidence values shown in Fig. 4(l).

The results show that the method proposed provides a consistent approach for creating and using an inexact "cognitive map" to allow a mobile robot to localize itself. It does not provide the exact pose of the robot in the environment, but rather an approximation, which we believe is sufficient for navigation and new exploration.

5 Conclusion

We have presented algorithms for creating and utilizing a "cognitive map" on a mobile robot equipped with sonar sensors and an odometer. As the range of the sonars is only a few meters, the robot cannot apprehend large spaces at once. The main focus in this paper was on showing that a "cognitive map" can be computed robustly for these spaces, and how the robot localizes itself in these spaces.

The algorithm for creating the "cognitive map" is based on the split and merge method. It divides a given metric map into distinct regions (namely ASRs), thus creating a topological representation on top of the metric one. Features of the ASRs thus computed are then used as part of the localization strategy, which can fuse information from different sources, resulting in a confidence map that tells the robot where its current location in the environment supposedly is. Currently, we exploit basic features only, namely ASR-distance travelled and relative orientation between adjacent ASRs. The purpose of these

two simple is strategies is mainly to show how fusion and localization work with our approach. They can easily be replaced or augmented by more sophisticated methods: As long as each method computes a confidence value for each ASR, the integration into the fusion approach is straightforward. Nevertheless, even the basic strategies presented here are powerful enough to allow for a localization that should be accurate enough for many applications.

Much has been discussed with respect to the use of distance information in cognitive mapping. For example, numerous experiments with chickens and pigeons have shown that they are able to use both absolute and relative distance in their search for food (e.g., [25]). Experiments with bees and ants have shown that they can perform internal calculations of the distance and direction travelled to perform path integration (e.g., [26] for a general discussion). Most of these experiments were concerned with the actual distance travelled and how the individual species deal with the errors in their measurements, as do most work on robot mapping to date. Using our robot, we have shown another way of using distance information, namely ASR-distance travelled as opposed to actual distance travelled.

ASR-distance is obtained from the shape of the ASR computed. In the past, there has been scant evidence that humans/animals do pay attention to the shape of each local environment (or, in our terminology, ASR) very early on in their initial exploration of a new environment. However, the debate has now intensified and this is especially true in the animal literature where the problem is commonly referred to as geometry in animal spatial behavior [27]. In many of these experiments, a relocation task utilizing a box-shaped environment is used, and the principal axes of the environment appear to be most useful. Our work here emphasized yet another possibility, namely using a straight line distance between exits of interests in an ASR.

References

1. Tolman, E.C.: Cognitive Maps in Rats and Men. Psychological Review **55**(4) (1948) 189–208
2. Golledge, R.G.: Human wayfinding and cognitive maps. In Golledge, R.G., ed.: Wayfinding Behavior: Cognitive Mapping and Other Spatial Processes. John Hopkins University Press, Baltimore (1999)
3. Poucet, B.: Spatial cognitive maps in animals: New hypotheses on their structure and neural mechanisms. Psychological Review **100** (1993) 163–182
4. Chown, E., Kaplan, S., Kortenkamp, D.: Prototypes, location, and associative networks (PLAN): Towards a unified theory of cognitive mapping. Cognitive Science **19**(1) (1995) 1–51
5. Kuipers, B.: The spatial semantic hierarchy. Artificial Intelligence **119** (2000) 191–233
6. Yeap, W.K., Jefferies, M.E.: Computing a Representation of the Local Environment. Artificial Intelligence **107**(2) (1999) 265–301
7. Estrada, C., Neira, J., Tardos, J.D.: Hierarchical SLAM: Real-Time Accurate Mapping of Large Environments. IEEE Trans. on Robotics **21**(4) (2005) 588–596

8. Piers, D.M., Kuipers, B.J.: Map learning with uninterpreted sensors and effectors. Artificial Intelligence **92** (1997) 169–227
9. Kortenkamp, D.: Cognitive Maps for Mobile Robots: A Representation for Mapping and Navigation. PhD thesis, University of Michigan (1993)
10. Jefferies, M., Weng, W., Baker, J.T., Mayo, M.: Using context to solve the correspondence problem in simultaneous localisation and mapping. In: Proc. Pacific Rim Int. Conf. on Artificial Intelligence. Volume 3157 of Lecture Notes in AI. (2004) 664–672
11. Gaussier, P., Revel, A., Banquet, J.P., Babeau, V.: From view cells and place cells to cognitive map learning: processing stages of the hippocampal system. Biol. Cybern. **86** (2002) 15–28
12. Hafner, V.V.: Cognitive maps in rats and robots. Adaptive Behavior **13**(2) (2005) 87–96
13. Franz, M., Mallot, H.: Biomimetic Robot Navigation. Robotics and Autonomous Systems **30** (2000) 133–153
14. Gallistel, C.R.: Animal Cognition: The Representation of Space, Time and Number. Annual Review of Psychology **40** (1989) 155 – 189
15. Schmidt, J., Wong, C.K., Yeap, W.K.: A Split & Merge Approach to Metric-Topological Map-Building. In: Int. Conf. on Pattern Recognition (ICPR). Volume 3., Hong Kong (2006) 1069–1072
16. Schmidt, J., Wong, C.K., Yeap, W.K.: Mapping and Localisation with Sparse Range Data. In Mukhopadhyay, S., Gupta, G.S., eds.: Proceedings of the Third International Conference on Autonomous Robots and Agents (ICARA), Palmerston North, New Zealand (2006) 497–502
17. Wong, C.K., Schmidt, J., Yeap, W.K.: Using a Mobile Robot for Cognitive Mapping. In: International Joint Conference on Artificial Intelligence (IJCAI), Hyderabad, India (January 2007) 2243–2248
18. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. John Wiley & Sons, New York, USA (1973)
19. Pavlidis, T., Horowitz, S.L.: Segmentation of Plane Curves. IEEE Trans. on Computers **C-23** (1974) 860 – 870
20. Niemann, H.: Pattern Analysis and Understanding. 2nd edn. Volume 4 of Springer Series in Information Sciences. Springer, Berlin (1990)
21. Triesch, J.: Vision-Based Robotic Gesture Recognition. Shaker Verlag, Aachen (1999)
22. Triesch, J., von der Malsburg, C.: Democratic Integration: Self-Organized Integration of Adaptive Cues. Neural Computation **13**(9) (2001) 2049–2074
23. Denzler, J., Zobel, M., Triesch, J.: Probabilistic Integration of Cues From Multiple Cameras. In Würtz, R., ed.: Dynamic Perception. Aka, Berlin (2002) 309–314
24. Kähler, O., Denzler, J., Triesch, J.: Hierarchical Sensor Data Fusion by Probabilistic Cue Integration for Robust 3-D Object Tracking. In: IEEE Southwest Symp. on Image Analysis and Interpretation, Nevada (2004) 216–220
25. Cheng, K., M. L. Spetch, D.M.K., Bingman, V.P.: Small-scale Spatial Cognition in Pigeons. Behavioural Processes **72** (2006) 115–127
26. Cornell, E.H., Heth, C.D.: Memories of travel: Dead reckoning within the cognitive map. In Allen, G., ed.: Human spatial memory: Remembering where. Lawrence Erlbaum Associates, Mahwah, NJ (2004) 191–215
27. Cheng, K., Newcombe, N.S.: Is there a geometric module for spatial orientation? Squaring theory and evidence. Psychonomic Bull Rev **12** (2005) 1–23