J. of Mult.-Valued Logic & Soft Computing, Vol. 9, pp. 195-220
Reprints available directly from the publisher
Photocopying permitted by license only

A Multi-Agent Approach to Fuzzy Landmark-Based Navigation

D. BUSQUETS, C. SIERRA AND R. LÓPEZ DE MÀNTARAS

Artificial Intelligence Research Institute (IIIA), Spanish Council for Scientific Research (CSIC), Campus UAB, 08193 Bellaterra, Barcelona, Spain

Received January 20, 2002; In final form September 1, 2002

(Recommended for publication by Andrew Adamatzky and Anthony Pipe)

This work explores the use of bidding mechanisms to coordinate the actions requested by a group of agents in charge of achieving the task of guiding a root towards a specified target in an unknown environment. This approach is based on a fuzzy approach to landmark-based navigation.

Keywords: fuzzy arithmetics, qualitative navigation, multiagent architecture, bidding mechanissms

1. INTRODUCTION

Outdoor navigation in unknown environments is still a difficult open problem in the field of robotics. Existing approaches assume that an appropriately detailed and accurate metric map can be obtained through sensing the environment. However, most of these approaches rely on odometry sensors, which can be very imprecise and lead to many errors (e.g. errors due to the wheels or legs slipping, which is quite common in outdoor environments). Our approach considers using only visual information. The robot must be equipped with a vision system capable of recognising visual salient objects (i.e. landmarks), and use them for mapping and navigation tasks. The

*e-mail: didac@iiia.csic.es

specific scenario that we are studying assumes that there is a target landmark that the robot is able to recognize visually. The target is visible from the robot's initial location, but it may subsequently be occluded by intervening objects. The challenge for the robot is to acquire enough information about the environment (locations of other landmarks and obstacles) so that it can move along a path from the starting location to the target.

But even landmark-based navigation approaches assume unrealistically accurate distance and direction information between the robot and the landmarks. We propose a fuzzy set based approach to landmark-based navigation in outdoor environments that assumes only very rough vision estimation of the distances and, therefore, does not rely on GPS information. Our approach is implemented by means of a multiagent architecture.

The main goal of our research is to design a robust vision-based navigation system for unstructured unknown environments. In particular, we want to provide the robot with orientation sense, similar to that found in humans or animals. This orientation sense will be realized by the use of landmark-based navigation, topological mapping and qualitative computation of landmark locations.

The navigation system uses a camera for landmark identification and recognition and has to compete with other systems for the control of the camera. This control is achieved through a bidding mechanism (see Section 4).

The map of the environment is represented by a labelled graph whose nodes represent triangular shaped regions delimited by groups of three non-collinear landmarks and whose arcs represent the adjacency between regions, that is, if two regions share two landmarks, the corresponding nodes are connected by an arc. The arcs are labelled with costs that reflect the easiness of the path between the two corresponding regions. A blocked path would have an infinite cost whereas a flat, hard paved path would have a cost close to zero. Of course these costs can only be assigned after the robot has moved (or tried to move) along the path connecting the two regions. Therefore, the map is built while the robot is moving towards the target. The only a priori assumption is that the target is visible from the initial robot location. Of course, the target can be lost during the navigation and is then when the navigation system will need to compute its location with respect to a set of previously seen landmarks whose spatial relation with the target is qualitatively computed both in terms of fuzzy distance and direction.

The navigation algorithm is part of a bigger robotics project. Another partner in the project is building a six legged robot with an on board cam-

era. The goal of the project is to have a completely autonomous robot able to navigate in outdoor unknown environments (to accomplish tasks such as planetary exploration). A human operator will select a target using the visual information received from the robot's camera (here is where we need the assumption of the target visibility), and the robot will have to reach it without any intervention from the operator. When available, we will test our approach with the real robot. Right now, the navigation system is tested over a simulation of an outdoor environment composed of elements such as buildings, trees, rivers, etc.

2. RELATED WORK

Mapping for robot navigation dates back to the famous SRI Shakey robot [14]. Recent research on modeling unknown environments is based on two main approaches: occupancy grid-based (or metric), proposed among others by Elfes [4] and Moravec [13], and topological such as those proposed by Chatila [3], Kuipers and Byun [9], Mataric [12] and Kortenkamp [8] among others. Cells in an occupancy grid contain information about the presence or not of an obstacle. The position of the robot is incrementally computed using odometry and information from sensors. One problem with this approach is that it requires an accurate odometry to disambiguate among positions that look similar. Besides, grids are computationally very expensive, specially in large outdoor environments, because the resolution of the grid (cell's size) cannot be too large. Contrary to grid-based representations, topological representations are computationally cheaper. They use graphs to represent the environment. Each node corresponds to an environment feature or landmark and arcs represent direct paths between them. In our work, however, nodes are regions defined by groups of three landmarks and they are connected by arcs if the regions are adjacent. This graph is incrementally built while the robot is moving within the environment. Topological approaches compute the position of the robot relative to the known landmarks, therefore they have difficulties in determining if two places that look similar are or not the same place unless a robust enough landmark recognition system is in place. Landmark recognition is a very active field of research in vision and very promising results are being obtained [11]. In this work we assume that the vision system can recognize landmarks. Thrun [20] combines grid-based and topological representations in his work on learning maps for indoor navigation. This is indeed a good idea for indoor environments but for large-

scale outdoor environments may not be worth the computational effort of maintaining a grid representation under a topological one. In [19] he uses a probabilistic approach for map learning and localization to cope with the uncertainty in sensor readings and robot motion. However, this work is also well applied in indoor environments.

The incremental map building approach presented here is based on previous work by Prescott [15] that proposed a network model that used barycentric coordinates, also called beta-coefficients by Zipser [21], to compute the spatial relations between landmarks for robot navigation. By matching a perceived landmark with the network, the robot can find its way to a target provided it is represented in the network. Prescott's approach is quantitative whereas our approach uses a fuzzy extension of the beta-coefficient coding system in order to work with fuzzy qualitative information about distances and directions. Levitt and Lawton [10] also proposed a qualitative approach to the navigation problem but that assumes an unrealistically accurate distance and direction information between the robot and the landmarks. Another qualitative method for robot navigation was proposed by Escrig and Toledo [5], using constraint logic. However, they assume that the robot has some a priori knowledge of the spatial relationship between the landmarks, whereas our system builds these relationships while exploring the environment.

Regarding the related work on multi-agent architectures, Liscano et al [6] use an activity-based blackboard consisting of two hierarchical layers for strategic and reactive reasoning. A blackboard database keeps track of the state of the world and a set of activities to perform the navigation. Arbitration between competing activities is accomplished by a set of rules that decides which activity takes control of the robot and resolves conflicts. Other hierarchical centralized architectures similar to that of Liscano et al are those of Stentz [17] to drive CMU's Navlab and Isik [7] among others.

Our approach is completely decentralized which means that the broadcast of information is not hierarchical. This approach is easier to program and is more flexible and extensible than centralized approaches. Arkin [1] also emphasized the importance of a nonhierarchical broadcast of information. Furthermore, we propose a model for cooperation and competition between activities based on a simple bidding mechanism. A similar model was proposed by Rosenblatt [16] in the CMU's DAMN project. A set of modules cooperated to control a robot's path by voting for various possible actions, and an arbiter decided which was the action to be performed. However the set of actions was pre-defined, while in our system each agent can bid for

any action it wants to perform. Moreover, in the experiments carried out with this system (DAMN), the navigation system used a grid-based map and did not use landmark based navigation at all. Sun and Sessions [18] have also proposed an approach for developing multi-agent reinforcement learning systems that uses a bidding process to segment sequences (and distribute the sequences among agents) in sequential decision tasks, their goal being to facilitate the learning of the overall task based on reinforcements received during task execution.

3. MAP REPRESENTATION

For map representation and wayfinding, we will use the model proposed by Prescott in [15]. The principles underlying this model are inspired by studies of animal (including man) navigation and wayfinding behaviour. The model, called the *beta-coefficient system*, is based on the relative positions of landmarks in order to estimate the location of a target.

We will firstly describe how this method works when the robot is able to have exact information about its environment, and then we will explain how we have adapted it to work with imprecise information.

3.1. Beta-Coefficient System

Let us assume that three landmarks and a target (which is also a landmark) have already been seen (e.g., landmarks *A*, *B* and *C* and target *T* from viewpoint *V*, as shown in Figure 1) by the system. Then seeing only the three landmarks, but not the target, from another viewpoint (e.g., *V*′), the system is able to compute the position of the target.

The only calculation needed to do this is

$$\beta = \mathrm{X}^{-1}\,\mathrm{X}_{\mathrm{T}}$$

where $X = [X_A X_B X_C]$ with $X_i = (x_i, y_i, 1)^{\mathrm{T}}$; $i \in \{A, B, C, T\}$, are the homogeneous coordinates of object i from the viewpoint V. The resultant vector, β, is called the β-vector of landmarks A, B, C and T. This relation is unique and invariant for any viewpoint, with the only restriction for the landmarks to be distinct and non collinear.

The target's new location from viewpoint V' is computed as

$$X'_{\mathrm{T}} = X' \beta$$

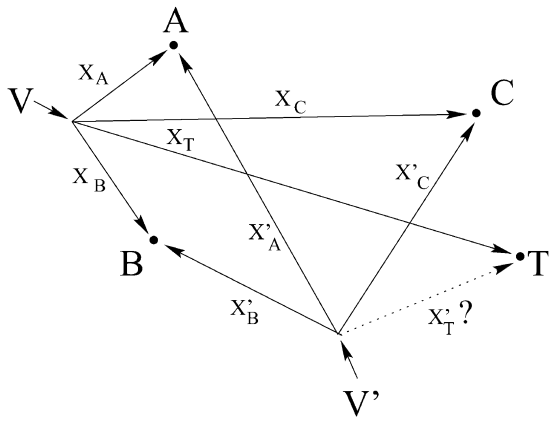

FIGURE 1
Possible landmark configuration and points of view.

This method can be implemented with a two-layered network. Each layer contains a collection of units, which can be connected to units of the other layer. The lowest layer units are *object-units*, and represent the landmarks the robot has seen. Each time the robot recognizes a new landmark, a new object-unit is created. The units of the highest layer are *beta-units* and there is one for each β-vector computed. When the robot has four landmarks in its viewframe, it selects one of them to be the target, a new beta-unit is created, and the β-vector for the landmarks is calculated. This beta-unit will be connected to the three object-units associated with the landmarks (as incoming connections) and to the object-unit associated with the target landmark (as an outgoing connection). Thus, a beta-unit will always have four connections, while an object-unit will have as many connections as the number of beta-units it contributes to. An example of the network can be seen in Figure 2b. In this figure there are six object-units and three beta-units. The notation ABC/D is understood as the beta-unit that computes the location of landmark D when the locations of landmarks A, B and C are known.

This network has a propagation system that permits computation of the location of the non-visible landmarks. The system works as follows: when the robot sees a group of landmarks, it activates (sets the value) of the associated object-units with the egocentric locations of these landmarks. When an object-unit is activated, it propagates its location to the beta-units it is connected to. On the other hand, when a beta-unit receives the location of its three incoming object-units, it gets active and computes the location of the

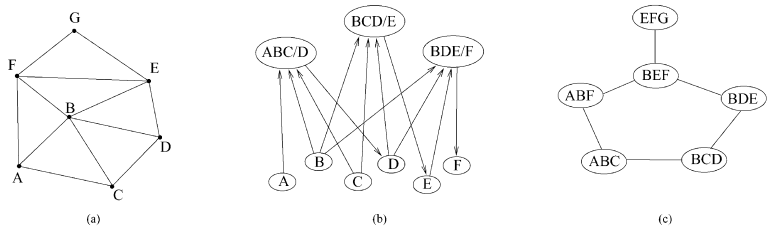

FIGURE 2
(a) Landmark configuration (b) Associated network (partial view) and (c) Associated topological map.

target it encodes using its β-vector, and propagates the result to the object-unit representing the target. Thus, an activation of a beta-unit will activate an object-unit that can activate another beta-unit, and so on. For example, in the network of Figure 2b, if landmarks A, B and C are visible, their object-units will be activated and this will activate the beta-unit ABC/D, computing the location of D, which will activate BCD/E, activating E, and causing BDE/F also to be activated. Knowing the location of only three landmarks (A, B and C), the network has computed the location of three more landmarks that were not visible (D, E and F). This propagation system makes the network compute all possible landmarks locations. Obviously, if a beta-unit needs the location of a landmark that is neither in the current view nor activated through other beta-units, it will not become active.

The network created by object and beta units is implicitly defining an adjacency graph of the topology of the landmarks. It can be converted to a graph where the nodes represent regions (delimited by a group of three landmarks), and the arcs represent paths. These arcs can have an associated cost, representing how difficult it is to move from one region to another. The arcs and their costs are created and updated as the robot explores the environment. An example of how the topology is encoded in a graph is shown in Figure 2c.

This topological graph will be used when planning routes to the target. Sometimes, when the position of the target is known, the easiest thing to do is to move in a straight line towards it, but sometimes it is not (the route can be blocked, the cost too high...).

With the topological graph, a route to the target can be computed. This route will consist of a sequence of regions through which the robot will have to go.

3.2 Fuzzifying the System

However, this method assumes that the robot will have the exact position of every landmark, in order to create the beta-units and use the network. But this is not the case here. The vision system of our robot will provide us with inexact information about the locations of landmarks. To work with this uncertain information we will need to use a multi-valued logic, such as fuzzy logic.

3.2.1 Fuzzy numbers and fuzzy operations

A fuzzy number can be thought of as a weighted interval of real numbers, where each point of the interval has a degree of membership, ranging from 0 to 1 [2]. The higher this degree, the higher the confidence that a given point belongs to the fuzzy number. The function $F_A(x)$, called membership function, gives us the degree of membership for x in the fuzzy number A.

Before defining arithmetic with fuzzy numbers, we have to introduce the concept of α-cut. The α-cut ($\alpha \in [0, 1]$) of a fuzzy number A, is the interval $\{A\}_\alpha = [a_1, a_2]$ such that $F_A(x) >= \alpha$, $\forall x \in [a_1, a_2]$.

Let A and B be fuzzy numbers, and $\{A\}_\alpha$ and $\{B\}_\alpha$ α-cuts. The fuzzy arithmetic operations are defined as follows,

$$\begin{aligned}
\{A + B\}_\alpha &= \{A\}_\alpha \oplus \{B\}_\alpha \; \forall\alpha \\
\{A - B\}_\alpha &= \{A\}_\alpha \ominus \{B\}_\alpha \; \forall\alpha \\
\{A \times B\}_\alpha &= \{A\}_\alpha \otimes \{B\}_\alpha \; \forall\alpha \\
\{A \div B\}_\alpha &= \{A\}_\alpha \oslash \{B\}_\alpha \; \forall\alpha
\end{aligned}$$

where the operations $\oplus, \ominus, \otimes$ and $\oslash$ are interval operators and are defined as follows

$$\begin{aligned}
&[a_1, a_2] \oplus [b_1, b_2] = [a_1 + b_1, a_2 + b_2] \\
&[a_1, a_2] \ominus [b_1, b_2] = [a_1 - b_2, a_2 - b_1] \\
&[a_1, a_2] \otimes [b_1, b_2] = [min(a_1b_1, a_1b_2, a_2b_1, a_2b_2), max(a_1b_1, a_1b_2, a_2b_1, a_2b_2)] \\
&[a_1, a_2] \oslash [b_1, b_2] = [a_1, a_2] \otimes [\tfrac{1}{b_2}, \tfrac{1}{b_1}], \; 0 \notin [b_1, b_2]
\end{aligned}$$

3.2.2 Fuzzy beta-coefficient system

To use the beta-coefficient system with fuzzy numbers, we simply perform the calculations using the fuzzy operators defined above. However, because of the nature of fuzzy operators, some landmark configurations may not be feasible (because of the division by 0) and alternative landmarks might be needed.

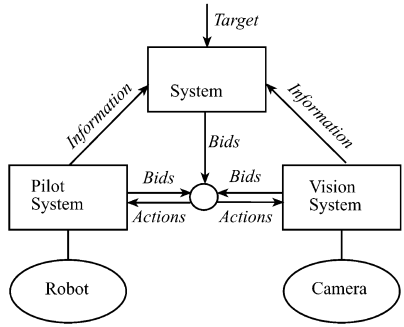

FIGURE 3
Robot architecture.

When using the network to compute the position of a landmark, we obtain a fuzzy polar coordinate (r, ϕ), where r and ϕ are fuzzy numbers, giving us qualitative information about its location.

4. ROBOT ARCHITECTURE

Navigation, as the general activity of leading a robot to a target destination, is naturally intermingled with other low-level activities of the robot such as actual leg coordination or obstacle avoidance and high-level activities such as landmark identification. All these activities *cooperate* and *compete*. They cooperate because they need one another in order to fulfill their tasks. For instance, the navigation system may need to identify the known landmarks in a particular area of the environment or to find new ones; or, the vision system may need the pilot to move in a particular direction to change the point of view on a landmark. These activities compete for the use of the most important resource, the camera. For instance, the pilot may need the camera to have a close view in front of a leg to safely avoid and obstacle, the navigation system may need sometimes to look behind in order to position the robot by seeing known landmarks, or the vision system may need to look to the one side of the robot to track an already identified landmark.

We propose a model for cooperation and competition based on a simple mechanism: *bidding*. We can see each of the activities (services), from an engineering point of view, as a system (represented as a square in Figure 3), that is, systems require and offer services to one another. The model works as follows: each system provides a number of services and waits for bids for

them. Each system generates bids for the services offered by the other systems according to the internal expected utility associated with the provisioning of such service. Each system decides which service to engage in based on the active bids that have been received.

This modular view conforms an extensible architecture. To extend this architecture with a new capability we would just have to plug in a new system. Not only that, it also permits us to have a modular view of each one of the systems.

4.1 Pilot System

In this work we do not focus on the algorithmics of the pilot system. We assume the pilot is able to safely command the motors that control the robot legs to move in a given direction. When the sensors detect problems: a leg is blocked, or slips, for instance, the pilot bids for the control of the camera to inspect the surroundings of the robot in order to perform a local planning to avoid the obstacle. This system is being currently developed in parallel by another partner in the project (IRI[1]).

4.2 Vision System

We also do not focus on the vision system description, although we build upon the results for landmark identification obtained by [11]. The vision system is able to recognise new landmarks in the vision field of the camera and is also able to identify previously recognised landmarks. It will also provide information about the movement performed by the robot.

The algorithmics of the vision system may need a complementary view in order to perform a correct landmark identification. In those cases, it will bid for the control of the pilot to momentarily divert the robot to take a new perspective on a landmark. This system is also being developed in parallel by another partner in the project.

4.3 Navigation System

This is the system where we have focused our work. We have used the modular view underpinning the overall robot architecture in the design of the navigation system. Again, the overall activity of leading the robot to the target destination is decomposed into a set of simple *agents* that bid for services provided by other robot systems. This system has a communication agent that gathers the different bids and determines which one to select at

[1] Institut de Robòtica i Informàtica Industrial, http://www.iri.csic.es

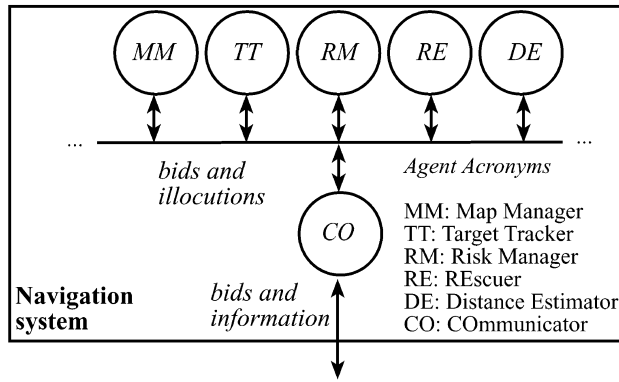

FIGURE 4
Multiagent view of the navigation system.

any given time. Thus, the navigation system is defined to be a multi-agent system where each agent is competent in a particular task. The coordination between the agents is made through a common representation of the map. Agents consult the map and suggest changes to it. Other robot systems provide information about the environment — position of landmarks, obstacles, difficulty of terrain, and so on — which is also used to update the map.

The local decisions of agents take the form of bids for services and are combined into a group decision: which set of compatible services to require, and hence gives us a handle on the difficult combinatorial problem of deciding *what to do next*. In the next section we describe in detail the society of agents that models the navigation process.

The multiagent navigation system is depicted in Figure 4.

5. THE GROUP OF BIDDING AGENTS

In the model reported in this work we present a group of five agents that take care of different tasks that, when coordinated through the bidding mechanism, provide the overall desired behaviour of leading the robot to a target landmark. The tasks are:

- to *keep the target located* with minimum uncertainty,
- to *keep the risk* of losing the target low,
- to *recover* from blocked situations,
- to *keep the error distance* to landmarks low, and

- to *keep the information on the map* consistent and up-to-date.

An agent has been designed to fulfill each one of these goals, plus a communicator agent, which will be responsible for communication of the navigation system with the other robot systems.

The services — actions — that agents can bid for, in order to fulfill their tasks are:

- Move (*direction*), instructs the pilot system to move the robot in a particular *direction*,
- Look_for_target (*angle, error*), instructs the vision system to look for the target that can be found in the area at angle error radians from the current body orientation, and,
- Identify_landmarks (*number, area*), instructs the vision system to identify a certain number of landmarks in a particular area represented as an angle arc relative to current heading.

Finally, agents may request information one another with respect to the different capabilities they have. For instance, any agent in the society may request the agent responsible of keeping the uncertainty low, which is the current level of uncertainty, or request from the map manager whether the target is currently visible or not. When describing the algorithm schemas these speech acts will appear as expressions in a KQML-style type of language.

Agents have a hybrid architecture. We will use the following construct to model the reactive component of an agent:

On *condition* **do** *action*

Whenever the *condition* holds (typically an illocution arriving to the agent) the *action* is executed immediately. Agents will refer to themselves by the special symbol “self.” When referring to all the agents of the society, they will use the symbol “all.”

The code schemas of the agents can be found in Section 5.7.

5.1 Map Manager

This agent is responsible for maintaining the information of the explored environment as a map. It also maintains the information of the current viewframe. As explained in Section 3, a map is a graph where each node cor-

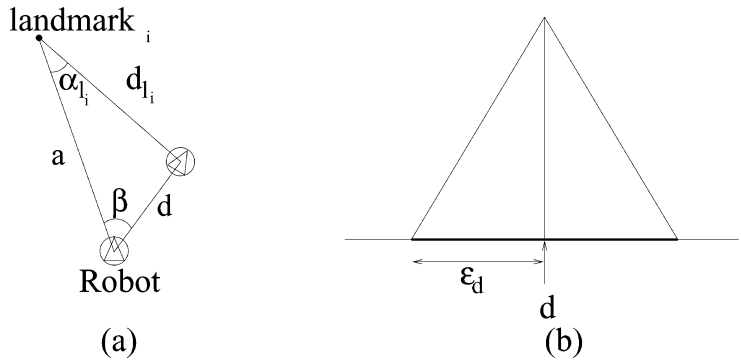

FIGURE 5
(a) Robot movement and angle variation and (b) Fuzzy distance and error

responds to a group of three landmarks and where arcs are labelled with a "passability" cost. An arc labelled with an infinite cost represents a non-passable section of the terrain — for instance, an obstacle, a wall, a river and so on. The activity of this agent consists of processing the information associated with the incoming viewframes — expanding the graph and possibly changing the beta-vectors, and asynchronously changing arcs' cost labels when informed by other agents or by other robot systems.

Each time a new viewframe arrives, it allows for computation of the difference in angle from the last viewframe for each landmark. This angle is used to determine the distance to the landmark, d_{l_i} , and also the distance error for that landmark, ε_{l_i}, in the following way:

$$d_{l_i} = d\frac{\sin\beta}{\sin\alpha_{l_i}} \qquad \epsilon_{l_i} = \frac{\epsilon_d}{d}\frac{\sin\beta}{\sin\alpha_{l_i}}$$

where ε_d is half of the support of the fuzzy number representing the distance in the direction of the movement of the robot, d is the most confident value for the distance, α_{l_i} is the difference in angle for $landmark_i$ from the last frame and the current frame and β is the angle of the movement performed by the robot with respect to the $landmark_i$ (see Figure 5).

This agent will use the fuzzy beta-coefficient system described in Section 3 to answer questions about landmark and target positions, coming from the *target tracker* and the distance estimator. It will activate the network with the information of the current viewframe, and the propagation system will compute the positions of the non visible landmarks.

The network and the topological graph will be created as new information about landmarks is received. One of the tasks of this agent is to decide which landmark configurations (a configuration is a group of 4 landmarks, with one of them being selected as the target) have to be incorporated in the network (i.e. create beta-units for them). As there can be many possible configurations (the number of combinations of 4 landmarks multiplied by 4—any landmark can be the target landmark—that is, for m landmarks in the current viewframe, there are $\binom{m}{4} \cdot 4$ configurations), and it is not feasible to store them all, some selection criterion must be used. The criterion currently used depends on the quality of the configuration (as will be explained below) and the size of the region associated with the configuration: the quality must be above a threshold, in order to avoid poor configurations, which would add a high error in the computations; and the size of the region should be the smallest possible (it is better to have the space divided in small regions). From the configurations meeting the criterion, the first n (this number should depend on the computing power of the computer used by the robot; right now we have $n = 5$) will be incorporated in the network.
It also will be important to consider the quality of the landmarks in the current viewframe, if required by the *risk manager*. This quality will be a function of the collinearity of the landmarks. Having a set S of landmarks, their quality is computed as:

$$q_s = \max\{1 - Col(S') | S' \subseteq S, |S'| = 3\}$$

where

$$Col(S') = 1 - \frac{\alpha\beta\gamma}{(\frac{\pi}{3})^3},$$

and α, β and γ are the three angles of the triangle formed by the landmarks in S'. The best quality is associated to equilater triangles, where $\alpha = \beta = \gamma = \pi/3$, and hence their collinearity is 0. When one of the angles is 0, landmarks would be maximally collinear and $Col(S') = 1$.

Another responsibility of this agent is to compute diverting of targets when asked for by the *rescuer*, possibly backtracking from the current situation. To do this it uses the map where all path costs and previous navigation decisions are recorded.

5.2 Target Tracker

The goal of this agent is to keep the target located all times. Ideally, the target should be always within the current view of the camera. If it is not, the uncertainty associated to its location is computed by this agent using the map and the current view. Actions of other systems are requested to keep the uncertainty as low as possible.

We model uncertainty as a function of the angle arc, ε_a, from the robot's current position, where the target is thought to be located. When we are sure of the position of the target we have a crisp direction and hence, $\varepsilon_a = 0$ and the uncertainty is 0. When we are completely lost, any direction can be correct, $\varepsilon_a = 2\pi$, and the uncertainty level is 1. Thus, the uncertainty level is computed as:

$$U_a = (\tfrac{\varepsilon_\alpha}{2\pi})^\beta$$

where β gives a particular increasing shape to the uncertainty function. If β is much smaller than 1, we are going to increase uncertainty very quickly as the imprecision in angle grows. For β values well over 1, uncertainty will grow very slowly until the error angle gets very big.

The actions required by this agent are to move towards the target and to look at the place where the target is assumed to be. Bids for moving towards the target start at a value κ_1 and decrease polynomially (depending on a parameter α) to 0 when the uncertainty increases ($bid\_move(U_a) = \kappa_1 (1 - U_a^{1/\alpha})$). Bids for looking at the target follow a sinusoidal increasing from 0, reaching a maximum of κ_2 for uncertainty equal to 0.5 and then decreasing again until 0 ($bid\_look(U_a) = \kappa_2 \sin(\pi U_a)$). This is so because there is no need to look at the target when the uncertainty is very low, and it does not make sense to bid high to look at the target when the uncertainty is already too high. The graphs of the bidding functions are shown in Figure 6.

5.3 Distance Estimator

The goal of this agent is to keep the distance error to the neighbouring landmarks as low as possible. This agent will play a very important role at the beginning of the navigation. When analysing the first viewframe to obtain the initial landmarks, the error in distance will be maximal, there will be no reference view to obtain an initial estimation of the distance to the target. This agent will generate high bids to move, preferably orthogonally, with respect to the line connecting the robot and the target in order to get another view on it and establish an initial estimation of the distances to the

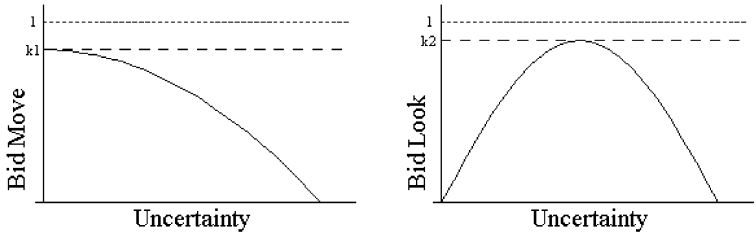

FIGURE 6
Target Tracker's bidding functions

visible landmarks. Similarly, when a target switch is produced (by the intervention of the rescuer) this agent may become relevant again if the distance value to the new selected target is very imprecise. Again, the same process will have as consequence a decrease in the new target distance error.

We model distance uncertainty as the size of the support of the fuzzy number modeling distance. As already mentioned in Subsection 5.1, we will note ε_{l_i} the imprecision error in distance to $landmark_i$ and ε_t the imprecision error to the current target. Thus, the uncertainty in distance to the target can be modeled as $U_d = 1 - \frac{1}{e^{\kappa \varepsilon_t}}$ where κ is a parameter that changes the shape of U_d; high values of κ will give faster increasing shapes. At the beginning of a run the *distance* is the fuzzy number $[0, \infty]$, $\varepsilon_t = +\infty$ and hence $U_d = 1$. The graphic of the bidding function is shown in Figure 7.

This agent will be relevant when this value is very high. Its action will be to bid to move the robot in an orthogonal direction using as its bid, the value of U_d .

The single action required by this agent is to move orthogonally to the line connecting the robot and the target (*a* in Figure 5).

This agent will also be responsible for deciding (up to a certainty degree ϕ) whether the robot is *at target*. It will consider that the robot has reached the target if the upper bound of the α-cut of level ϕ of the fuzzy number representing the distance to the target is less than δ times the body size of the robot.

5.4 Risk Manager

The goal of this agent is to keep the risk of losing the target as low as possible. The way to satisfy this goal is by keeping a reasonable number of landmarks, as non collinear as possible, in the surroundings of the robot. The less landmarks we keep around the more risky is our current exploration and the

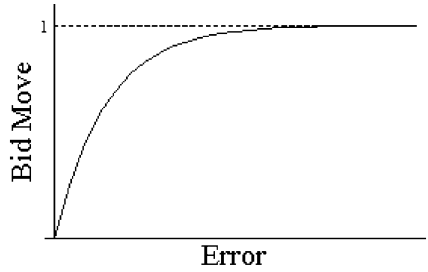

FIGURE 7
Distance Estimator's move bidding function

higher the probability of losing the target. Also, the more collinear the landmarks the higher the error in the location of the target will be and thus the higher the uncertainty in its location.

We will model the risk as a function that combines several variables: 1) the number of landmarks ahead (elements in set *A*), 2) the number of landmarks around (elements in set *B*), and 3) their "quality" (q_A and q_B). *A* and *B* are the sets of landmarks ahead and around of the robot, respectively. We understand by "quality" how collinear they are. A lowest risk of 0 will be assessed when we have at least four visible landmarks in the direction of the movement with the minimum collinearity between them. A highest risk of 1 is given to the situation when there are neither landmarks ahead nor around us.

$$R = 1 - \min(1, q_A(\tfrac{|A|}{4})^{\gamma_A} + q_B(\tfrac{|B|}{4})^{\gamma_B})$$

The values γ_A and γ_B determine the relative importance of the type of landmarks. Having landmarks ahead should be privileged somehow, so normally $\gamma_A > \gamma_B$.

Given that the robot has little to do in order to increase the quality of the landmarks, the only way to decrease the risk level is by increasing the number of landmarks ahead and around. We more heavily weight having landmarks ahead by bidding $\gamma_r R$ for that action and $\gamma_r R^2$ (which is obviously smaller that $\gamma_r R$) for the action of identifying landmarks around the robot, where γ_r is a parameter to control the shape of the bid function. The graphs of the bidding functions are shown in Figure 8.

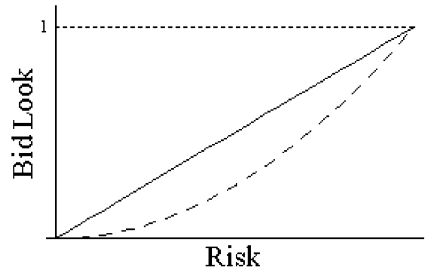

FIGURE 8
Risk Manager's look bidding functions (look ahead -solid line- and look behind -dashed line-)

5.5 Rescuer

The goal of the rescuer agent is to rescue the robot from problematic situations. These situations may occur due to three reasons. First, the pilot can lead the robot to a position with an obstacle ahead. Second, the uncertainty of the location of the target is too high, over a threshold $\overline{U_a}$. Finally, we can be in a very risky place, that is a place where we can get easily lost, a value of R over a threshold $\overline{R}$. If any of these situations occur, the rescuer agent asks the map manager for a diverting target. The algorithm uses a stack where the different diverting targets are stacked.

5.6 Communicator

The multiagent system implementing the navigation algorithm communicates with the remaining robot systems through the communicator agent. This agent receives bids for services from the other agents. The services required may be conflicting or not. For instance, an agent that requires the camera to look behind and another that requires it to identify a new landmark on the right, end up with conflicting service biddings, that cannot be fulfilled at the same time. On the other hand, an agent requiring the robot to move forward, and an agent requiring the camera to look behind might be perfectly non-conflicting. The communicator agent combines the bids for the different services, determines the service with highest bid for each group of conflicting services and outputs service bids accordingly.

We note the set of services, or actions, as A. If two actions cannot be performed at the same time, we say that they conflict, and we note it as $Conflict(a_i, a_j)$. A bid is a pair of the form (a_i, b) such that $a_i \in A$ and $b \in [0, 1]$. The set of active bids in a given moment is a set of bids received from the remaining agents in the multiagent system since the last decision taken

by the communicator agent. The communicator generates as output to the other systems a set of feasible bids.

Given a set of active biddings *B*, a feasible bidding is a set *F* of bids {(a_1, b_1),…, (a_i, b_i),…, (a_n, b_n)} such that for all a_i and a_j, $a_i \neq a_j$, *Conflict*(a_i, a_j) is false and $b_i = \vee \{b_j | (a_i, b_j) \in B\}$. The combination function $\vee$ being any form of disjunctive operator, such as *max*. We note by *B** the set of feasible biddings of *B*.

Given a set of bids, the agent will select the feasible bidding associated with those that maximises the welfare of the society. Understanding the value in a bid as a measure of the expected utility of the action in that bid to a particular agent, we use the sum of the bids as the function to maximise. Thus, the actual output is:

$$F = arg\ max\{\sum_{i=1}^{j} b_i | \{(a_1, b_1), \ldots, (a_j, b_j)\} \in B^*\}$$

5.7 Agents code schemas

In this section we present the code schemas for the agents *Map Manager, Target Tracker, Distance Estimator, Risk Manager* and *Rescuer*. The schemas have some parameters, such as the target that has to be reached, its initial heading, and some other particular parameters for each agent that define its behaviour (bidding functions parameters, thresholds and so on).

Agent MM(initial_target)=
  Begin
    target = initial_target
    On inform(CO,self,current_view(CV,movement)) **do**
      update_map(CV,movement)
    On ask(X,self,position-landmark(L)?) **do**
      ⟨angle,ϵ_α,d,ϵ_d⟩ = *compute_landmark_position(L)*
      inform(self,X,position-landmark(L,angle,ϵ_α,d,ϵ_d))
    On ask(X,self,landmarks?) **do**
      $\langle |A|, |B|, q_A, q_B \rangle$ = *compute_landmarks_quality*
      inform(self,X,landmarks($|A|, |B|, q_A, q_B$)
    On ask(X,self,diverting-target?) **do**
      T = *compute_diverting_target*
      inform(self,X,diverting-target(T))
    On inform(RE,self,target(T)) **do** target = T
  end

Agent TT(initial_target, initial_angle, $\alpha, \beta, \kappa_1, \kappa_2$)=
Begin
$\langle$target,angle,$\epsilon_\alpha\rangle$ = $\langle$initial_target,initial_angle,0$\rangle$
Repeat
ask(self,MM,position-landmark(target)?)
When inform(MM,self,position-landmark(target, angle,ϵ_α,dist,ϵ_{dist})) **do**
$U_a = (\frac{\epsilon_\alpha}{2\pi})^\beta$
inform(self,all,uncertainty(U_a))
inform(self,CO,
$\{(\text{Move}(angle), \kappa_1(1-(U_a^{1/\alpha}))),$
$(\text{Look\_for\_target}(angle, \epsilon_\alpha),$
$\kappa_2 \sin(\pi U_a))\}$)
endwhen
Until inform(DE,self,at_target(initial_target))
On inform(RE,self,target(T)) **do** target = T
end

Agent DE(initial_target, κ, ϕ, δ)=
Begin
target = initial_target
d = $[0, \infty]$
$\epsilon_t = +\infty$
$U_d = 1$
Repeat
ask(self,MM,position-landmark(target)?)
[min,max]=$\{d\}_\phi$
at_target = max $\leq \delta$*bodyshape
If at_target **then** inform(self,all,at_target(target))
When inform(MM,self, position-landmark(target,angle,ϵ_α,dist,ϵ_t)) **do**
$U_d = 1 - \frac{1}{e^{\kappa\epsilon_t}}$
d = dist
inform(self,CO, $\{(\text{Move}(angle + \frac{\pi}{2}), U_d)\}$)
endwhen
Until inform(self,self,at_target(initial_target))
On inform(RE,self,target(T)) **do** target = T
end

Agent RM(initial_target, $\gamma_A, \gamma_B, \gamma_r$)=
Begin
target = initial_target
R = 0
Repeat
ask(self,MM,landmarks?)
When inform(MM,self,
landmarks($|A|$,$|B|$,q_A,q_B) **do**
$R = 1 - \min(1, q_A(\frac{|A|}{4})^{\gamma_A} +$
$q_B(\frac{|B|}{4})^{\gamma_B})$
inform(self,all,risk(R))
If $|A| < 4$ **then** inform(self,CO,
$\{(\text{Identify\_landmarks}(4, ahead),$
$\gamma_r R)\})$
If $|B| < 4$ **then** inform(self,CO,
$\{(\text{Identify\_landmarks}(4, around),$
$\gamma_r R^2)\})$
endwhen
Until inform(DE,self,at_target(initial_target))
On inform(RE,self,target(T)) **do** target = T
end

Agent RE(initial_target, $\overline{U}_a, \overline{R}$)=
Begin
stack = push(emptystack,initial_target)
Repeat
When inform(CO,self,Blocked)
or (inform(TT, self, uncertainty(U_a)) **and** $U_a > \overline{U}_a$)
or (inform(RM, self, risk(R)) **and** $R > R$) **do**
ask(self,MM,diverting_target?)
endwhen
Until inform(DE,self,at_target(initial_target))

On inform(DE,self,at_target(T)) **do**
pop(stack)
if not empty(stack) **then**
T := top(stack)
inform(self, all, target(T))
On inform(MM,self,diverting_target(T)) **do**
push(stack, T)
inform(self, all, target(T))
end

6. EXPERIMENTATION

We have implemented the agents of the navigation system and tested the algorithm on the Webots[2] simulator. Since we do not have the real robot yet, we also simulate the pilot and vision systems. Each agent is executed as an independent thread, and they use shared memory to simulate messages passing.

Figure 9 shows a navigation run. It shows the path followed by the robot from a starting point to a target landmark. The environment is composed of a set of landmarks (shown as circles), a river (the thick traversing line) with a couple of bridges, and some fences and other obstacles.

This example shows how the system is able to avoid an obstacle encountered while trying to reach the target, and continue its way to the target after the obstacle has been passed. The target landmark is landmark number 10 (at the left-hand side of the world). At the very begining the agent DE bids very high in order to estimate the distance to the target. The agent TT bids for moving towards the target, but the bid of DE is higher and the robot moves orthogonally with respect to the target. When the distance is estimated, the bids of DE are very low, and therefore the bid of TT wins and the robot starts going towards the target. However, the robot encounters an obstacle and the pilot takes control of the movement in order to avoid it. When the obstacle has been totally avoided, the pilot lets the navigation system control the robot again, and it moves towards the target. This situation is repeated a couple of times until the robot finally reaches the target.

In the second example (Figure 10) we see how the Navigation system computes a diverting path for reaching the target when it loses it. In this environment, the grey polygons are occluding obstacles, and the green ones are non-occluding ones. At point A, it sees the target and starts going towards it. However, at point B, it detects an obstacle, so the pilot forces the robot to turn. When it reaches point C, it cannot see the target anymore, as it is behind an occluding obstacle. At this point, a diverting target is computed (in this case, landmark 30 is selected). The robot starts going towards this diverting target. Once reached (point D), a new diverting target is computed (landmark 38 is selected), and the robot goes towards it. At point E, after reaching the current diverting target, a new one is computed (landmark 12), which is reached at point F. From this point, it sees the initial target again, and goes straight towards it. In Figure 11 the map generated while

[2] Cyberbotics, http://www.cyberbotics.com

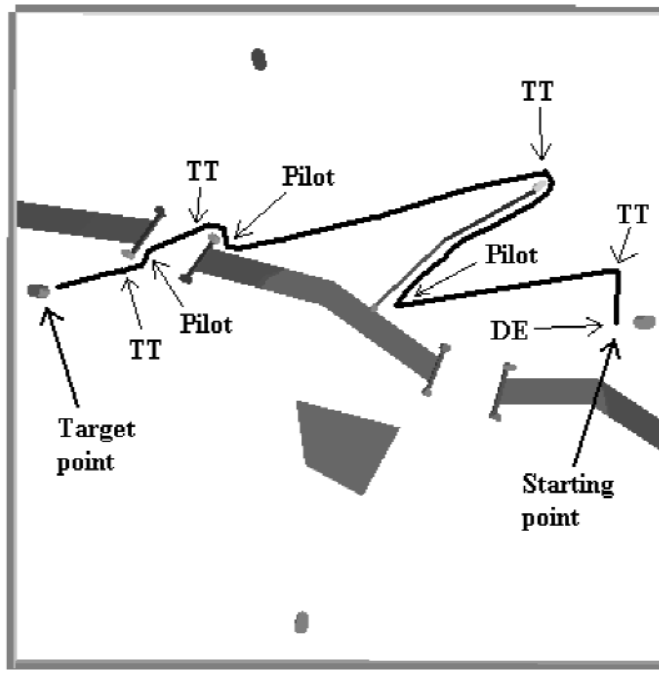

FIGURE 9
Robot's path from starting point to the target. See Color Plate I at back of issue.

reaching the target is shown. Although, internally, the Navigation system stores the map as a graph, here, for clarity, we show the triangular regions corresponding to the nodes of this graph.

7. CONCLUDING REMARKS AND FUTURE WORK

This work presents a new approach to robot navigation based on the combination of fuzzy representation and multiagent coordination based on a bidding mechanism. The implementation is finished on top of a simulator and we are currently working on the phase of experimentation. Experimentation is being carried out along three dimensions: number of environments (one or several), number of points in each environment at which to start a run (one or several), and number of samplings of robot parameters per starting point (one or several). This gives eight increasingly more complex experimental scenarios for which the results will be given as path length averages com-

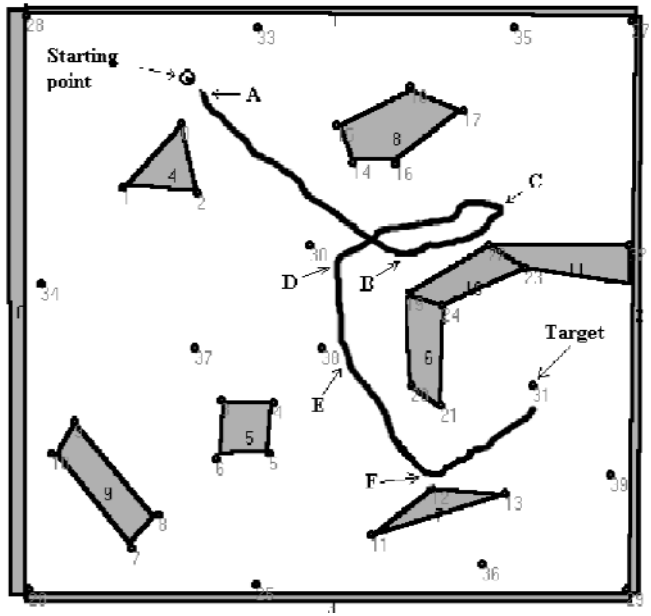

FIGURE 10
Computing diverting targets. See Color Plate II at back of issue.

pared either with optimal results or with human obtained results.

The goal of the experimentation is to tune the different parameters of the agents, which define the overall behaviour of the robot through the combination of the individual behaviours of each agent, in order to achieve the best behaviour of the robot in any environment. Of course, it can be the case that such a "best robot" does not exist, and it would then be necessary to distinguish between different types of environments, so we would end up with a set of robot configurations, each one useful in a concrete situation. These configurations could be used to set up the robot for a certain task *a priori*, or the robot could dynamically change its configuration while it is performing its task, depending on the situations it encounters. For this latter case, we plan to develop a new agent that would be responsible for recognising the different situations and deciding which configuration to use. We will use genetic algorithms to carry out this tuning.

We are also using reinforcement learning techniques in order to learn policies for individual agents. In particular, we are trying to have the system learn to use the camera only when it is completely necessary, as it is an expensive and high demanded resource.

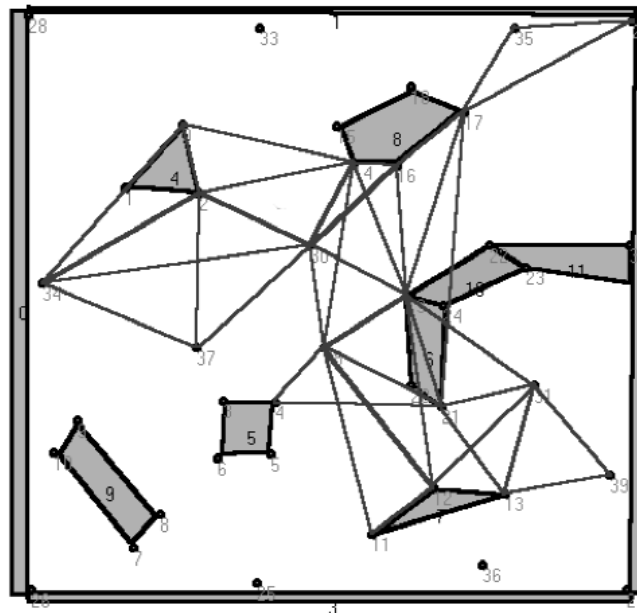

FIGURE 11
Associated map. See Color Plate III at back of issue.

Once the simulation results are satisfactory enough, and the real robot is available, we plan to test the navigation algorithm in real environments.

Acknowledgments

This research has been supported by MCYT Project DPI 2000-1352-C02 and CIRIT project CeRTAP. Dídac Busquets holds a CIRIT doctoral scholarship 2000FI-00191. We acknowledge the discussions held with Thomas Dietterich during his sabbatical stay at IIIA, at the early stage of this research work.

REFERENCES

[1] Arkin, R.C. (1989). Motor schema-based mobile robot navigation. *Int. J. Robotics Research*, *8*(4), 92–112.

[2] Bojadziev, G. and Bojadziev, M. (1995). Fuzzy sets, fuzzy logic, applications, Advances in Fuzzy Systems, volume 5. World Scientific.

[3] Chatila, R. (1982). Path planning and environment learning in a mobile robot system. In

Proceedings of the 1982 European Conference on Artificial Intelligence (ECAI-82).

[4] Elfes, A. (1987). Sonar-based real-world mapping and navigation. *IEEE J. Robotics and Automation*, *3*(3), 249–265.

[5] Teresa Escrig, M. and Toledo, F. (2000). Autonomous robot navigation using human spatial concepts. *Int. Journal of Intelligent Systems*, *15*, 165–196.

[6] Liscano, R. et al. (1995). Using a blackboard to integrate multiple activities and achieve strategic reasoning for mobile-robot navigation. *IEEE Expert*, *10*(2), 24–36.

[7] Isik, C. and Meystel, A.M. (1988). Pilot level of a hierarchical controller for an unmanned mobile robot. *IEEE J. Robotics and Automation*, *4*(3), 242–255.

[8] Kortenkamp, D.M. (1993). Cognitive maps for mobile robots: A representation for mapping and navigation. Ph.D. thesis, University of Michigan, Computer Science and Engineering Department, Michigan.

[9] Kuipers, B.J. and Byun, Y.-T. (1988). A robust qualitative method for spatial learning in unknown environments. In *Proceedings AAAI-88*, Menlo Park, CA, AAAI Press/MIT Press.

[10] Levitt, T.S. and Lawton, D.T. (1990). Qualitative navigation for mobile robots. *Artificial Intelligence Journal*, *44*, 305–360.

[11] Martinez, E. and Torras, C. (2000). Qualitative vision for the guidance of legged robots in unstructured environments. *Pattern Recognition*, In press.

[12] Mataric, M.J. (1991). Navigating with a rat brain: a neurobiologically-inspired model for robot spatial representation. In J.-A. Meyer and S.W. Wilson, editors, *From Animals to Animats*, Cambridge, MA: MIT Press.

[13] Moravec, H.P. (1988). Sensor fusion in certainty grids for mobile robots. *AI Magazine*, *9*(2), 61–74.

[14] Nilsson, N.J. (1969). A mobile automaton: An application of AI techniques. In *Proceedings of the 1969 International Joint Conference on Artificial Intelligence*.

[15] Prescott, T.J. (1996). Spatial representation for navigation in animats. Adaptive Behavior, *4*(2), 85–125.

[16] Rosenblatt, J. (1995). Damn: A distributed architecture for mobile navigation. In *Proceedings of the 1995 AAAI SpringSymposium on Lessons Learned from Implemented Software Architectures for Physical Agents*. AAAI Press, March.

[17] Stentz, A. (1990). The codger system for mobile robot navigation. In C.E. Thorpe, editor, *Vision and Navigation, the Carnegie Mellon Navlab*, pages 187–201, Boston: Kluwer Academic Pub.

[18] Sun, R. and Sessions, C. (1999). Bidding in reinforcement learning: A paradigmfor multi-agent systems. In J.P. Müller O. Etzioni and J.M. Bradshaw, editors, *Proceedings 3d Annual Conference on Autonomous Agents*, pages 344–345, Seattle.

[19] Thrun, S. (1998). Bayesian landmark learning for mobile robot navigation. *Machine Learning*, *33*(1), 41–76.

[20] Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence Journal*, *99*(2), 21–72.

[21] Zipser, D. (1986). Biologically plausible models of placerecognition and place location. In J.L. McClelland and D.E. Rumelhart, editors, *Parallel Distributed Processing: Explorations in the Micro-Structure of Cognition*, Vol. 2, pages 432–470, Cambridge: MA. Bradford Books.