



A Multiagent Approach to Qualitative Landmark-Based Navigation

DÍDAC BUSQUETS, CARLES SIERRA AND RAMON LÓPEZ DE MÀNTARAS

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

didac@iiia.csic.es

sierra@iiia.csic.es

mantaras@iiia.csic.es

Abstract. In this paper we present a multiagent system for landmark-based navigation in unknown environments. We propose a bidding mechanism to coordinate the actions requested by the different agents. The navigation system has been tested on a real robot on indoor unstructured environments.

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

1. Introduction

Robotic systems for navigating through unknown environments are a focus of research in many application areas including, among others, spacecraft (rovers for Mars and the Moon) and search and rescue robotics. These systems have to perform very different tasks, from looking for rocks, picking them up and analyzing them, to assessing damages or looking for survivors after a natural disaster or accident has happened. However, they all share two key characteristics: first, they have to achieve their goals *autonomously*, and second, they have to *navigate* in unknown environments.

The first key point in these applications, *autonomy*, arises from the impossibility of always having a human operator controlling the robotic system. Although the ideal situation would be to have an expert operator controlling the robot, the necessary conditions cannot always be met. These conditions are usually related to the communication between the operator and the robot. In many situations it is very difficult to guarantee that the communication link will be robust, in terms of speed and availability. A clear example is found on planetary exploration missions. A major problem in such missions is the distance between the robot and the control station (usually located on the Earth); the time of sending an order to the robot and having it executed can be in the order of minutes. In the case a

fast reaction were needed (changing the trajectory of the robot, selecting a new scientific target that might be more relevant to the mission, etc.), this time would not be acceptable at all. Another disadvantage of depending on external agents (be it a human or any other device—e.g., a GPS device for localization) is that the robot can get blocked if any of these agents providing basic information for accomplishing the task crashes. This would leave the robot with no means to continue with its mission. Therefore, all the decisions should be taken on-board, without having to exchange commands or information with any external agent, or at least, make this exchange minimal, such as sending only information about task initialization (e.g., target selection, task description, etc.).

The other important point for such applications is *navigation*. The robot must be able to start in an unknown location and navigate to a desired target. Navigation in unknown unstructured environments is still a difficult open problem in the field of robotics. The first difficulty of such an environment is that there is no a priori knowledge about it, and therefore a map can only be built while exploring. Secondly, unstructured environments are characterized, precisely, by the lack of structure among the different objects of the world. This is usually the case for outdoor environments. On the other hand, in structured environments, such as offices, buildings, etc. many suppositions about their

structure can be done. For instance, walls and corridors are straight, they are usually orthogonal, most of the doors have the same size, etc. These characteristics are very helpful when building a map of the environment, as its structure can be inferred without the need of sensing the whole environment. Contrarily, in unstructured environments such suppositions do not hold, so the robot can only rely on the information it is able to gather from its sensors. This makes the task of map building and navigating even more difficult.

This research work is part of a larger robotics project. Another partner (IRI¹) in the project is building a six legged robot with on board cameras for outdoor landmark recognition. The goal of the project is to have a completely autonomous robot capable of navigating in outdoor unknown environments. A human operator will select a target using the visual information received from the robot's camera, and the robot will have to reach it without any intervention of the operator. The robot could also have an image or description of the target, so the human intervention would not even be needed for selecting the target. A first milestone for achieving the final goal of the project is to develop a navigation system for indoor unknown unstructured environments for a wheeled robot. Moreover, the environments of this first stage are composed of easily recognizable landmarks, since the vision system for outdoors is not yet available.

We propose a robot architecture to accomplish this first milestone. The approach used and the results obtained are the subject of this paper. The robot architecture is an instantiation of a general bidding coordination architecture that is proposed in this paper. The concrete bidding architecture for landmark-based navigation that we have developed is composed of three systems: the *Pilot* system, the *Vision* system and the *Navigation* system. Each system competes for the two available resources: motion control (direction of movement) and camera control (direction of gaze). The three systems have the following responsibilities. The *Pilot* is responsible for all motions of the robot. It selects these motions in order to carry out commands from the *Navigation* system and, independently, to avoid obstacles. The *Vision* system is responsible for identifying and tracking landmarks (including the target landmark). Finally, the *Navigation* system is responsible for choosing higher-level decisions in order to move the robot to a specified target. This requires requesting the *Vision* system to identify and track landmarks (in order to build a map of the environment) and requesting

the *Pilot* to move the robot in various directions in order to reach the goal position or some intermediate target.

From the brief description of the robot architecture given above, it can be observed that the three systems must *cooperate* and *compete*. They must cooperate because they need one another in order to achieve the overall task of reaching the target position. But at the same time they are competing for motion and camera control.

The *Navigation* system is implemented as a multi-agent system, where each agent is competent in a specific task. Depending on its responsibilities and the information received from other agents, each agent proposes which action the *Navigation* system should take. Again, we find that the agents must cooperate, since an isolated agent is not capable of moving the robot to the target, but they also compete, because different agents want to perform different actions.

The problem of cooperation and competition between different agents is very common in robotics, and *Behavior-based Robotics* (Arkin, 1998) addresses exactly this issue. This approach to robotic systems deals with coordinating, or arbitrating, different behaviors sending requests for actions, usually incompatible with each other, to a robot. The role of the coordination is to select a single action command that will be sent to the robot. When this action is a selection of one of the behaviors' requests, we talk about *competitive* coordination, whereas if the action is a mix of several behaviors' requests, we talk about *cooperative* coordination. In our architecture, each agent plays the role of a behavior, and there is an additional agent playing the role of coordinator.

For both the overall robot system and the *Navigation* system, we propose the use of a new competitive coordination system based on a *bidding mechanism*. In the overall robot system, the *Navigation* and the *Pilot* systems generate bids for the services offered by the *Pilot* and *Vision* systems. These services are to move the robot toward a given direction, and to move the camera and identify the landmarks found its view-field, respectively. The service actually executed by each system depends on the winning bid at each point in time. Similarly, in the *Navigation* system, each agent bids for the action it wants the robot to perform. These bids are sent to a special agent that gathers all bids and determines the winning action. The selected action is then sent as the *Navigation* system's bid for the services of the *Vision* and *Pilot* systems.

To navigate in an unknown environment, the robot must build a map. 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, due to the wheels or legs slipping, and lead to many errors that grow as the robot moves.

Our approach considers using only visual information. The advantage of using visual information is that it is independent from any past action the robot may have performed, which is not the case for odometry. The robot must be equipped with a robust vision system capable of recognizing landmarks, and use them for mapping and navigation tasks. The 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 (so that the human operator can select it), 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 vision-based navigation approaches assume unrealistically accurate distance and direction information between the robot and the landmarks. We propose a fuzzy set based approach that assumes only very rough vision estimation of the distances and, therefore, does not rely on any localization device.

The main goal of our research is to design a robust vision-based navigation system for unknown unstructured environments. In particular, we want to provide the robot with orientation sense, similar to that found in humans or animals. The rationale of the orientation sense is that the robot does not need to know what is the exact route from its starting point to the target's location, but it uses landmarks as references to find its way to the target. To make a parallelism with humans, when giving directions for going somewhere in our city, no one gives exact distances, turning angles, etc., but gives approximate distances, and more important, reference points (distinctive places such as buildings, squares, etc.) that help us getting to the destination. In our approach, this orientation sense is realized by the use of *landmark-based navigation, topological mapping* and *qualitative computation* of landmark locations.

The map of the environment is represented by a labeled graph whose nodes, instead of representing isolated landmarks, 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 of the graph are labeled 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. Since the map is not given, but built while the robot moves, these costs can only be assigned after the robot has moved (or tried to move) along the path connecting the two regions. Although the adjacency of nodes in our graph also represents adjacency of topological places, the arcs contain only cost information, not instructions on how to get from one place to another. But, actually, this information is not missing, it is implicit in the adjacency of regions. Given that two nodes are adjacent only if their regions share two landmarks, it is clear that to go from one region to another the robot has to cross the edge formed by the two common landmarks, unless there is a long obstacle blocking this path.

Although this topological map would be sufficient for carrying out navigational tasks, we also provide the robot with the capability of storing the spatial relationships among landmarks. To realize this capability we have extended Prescott's beta-coefficients system (Prescott, 1996). Prescott's model stores the relationships among the landmarks in the environment. The location of a landmark is encoded based on the relative locations (headings and distances) of three other landmarks. This relationship is unique and invariant to viewpoint. Once this relationship has been stored, the location of each landmark can be computed from the locations of the three landmarks encoding it, no matter where the robot is located, as long as the robot can compute the heading and distance to each of the three landmarks. As the robot explores the environment, it stores the relationships among the landmarks it sees. This creates a network of relationships among the landmarks in the environment. If this network is sufficiently-richly connected, then it provides a computational map of the environment. Given the headings and distances to a subset of currently-visible landmarks, the network allows to compute the locations of all of the remaining landmarks, even if they are currently not visible from the robot. Prescott assumed that the robot could have the exact distances and headings to the landmarks, but as we mentioned previously, we need to deal with the imprecision of the real world. To deal with it, we have extended the model using fuzzy numbers and fuzzy

arithmetics. With this extension, if the target is ever lost during the navigation, the robot will 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.

We have implemented the overall architecture and the Navigation system and first tested it over a simulator. After obtaining promising results on simulation, we have implemented the algorithm on a wheeled robot and tested it on real environments.

2. Related Work

Mapping for robot navigation dates back to the famous SRI Shakey robot Nilsson (1969). Recent research on modeling unknown environments is based on two main approaches: occupancy grid-based (or metric), proposed among others by Elfes (1987) and Moravec (1988), and topological such as those proposed by Chatila (1982), Kuipers and Byun (1988), Mataric (1991) and Kortenkamp (1993) among others. Cells in a 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. Contrarily 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 to determine 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 (Martinez and Torras, 2001). In this work we have developed a robust vision system to recognize simple landmarks. Thrun (1998) combines grid-based and topological representations in his work on learning

maps for navigation in indoor structured environments. 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 Thrun (1998) he uses a probabilistic approach for map learning and localization to cope with the uncertainty in sensor readings and robot motion, which makes it very difficult to disambiguate among positions that look similar. In our case we do not have to deal with this ambiguity, since the Vision system is robust enough to correctly identify the landmarks. Thus, there is no uncertainty about the presence of a landmark. However, there is some imprecision about its location, which we deal with using fuzzy techniques.

The incremental map building approach presented here is based on previous work by Prescott (1996) that proposed a network model that used barycentric coordinates, also called beta-coefficients by Zipser (1986), 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 (1990) also proposed a qualitative approach to the navigation problem but they also assume 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 (2000), using constraint logic. However, they assume that the robot has some a priori knowledge of the spatial relationship of the landmarks, whereas our system builds these relationships while exploring the environment.

Regarding the research on control architectures, in the last years it has been mainly focused on Behavior-based architectures (Arkin, 1998). The most representative of such architectures are Brook's subsumption architecture (Brooks, 1986), Maes' action selection (Maes, 1989) and Arkin's motor schema (Arkin, 1989). Since then, many other architectures have been proposed. Liscano et al. (1995) 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 (1990) to drive CMU's Navlab, and Isik and Meystel (1988) 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 (1989) also emphasized the importance of a non-hierarchical 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 (1995) 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 at all landmark based navigation. Also at CMU, the FIRE project (Dias and Stentz, 2001) uses a market-oriented approach to model the co-operation of a team of robots. Sun and Sessions (1999) have also proposed an approach for developing a multiagent reinforcement learning system that uses a bidding mechanism to learn complex tasks. The bidding is used to decide which agent gets the control of the learning process. The agents bid according to the expected reward that would receive if they were given the control. Thus, although they are competing for the control, they also cooperate, since they seek to maximize the overall system reward.

3. Map Representation

As already mentioned, the task the robot has to perform is to navigate through an unknown unstructured environment and reach a target landmark specified by a human operator. This task is not easy to solve, since it has to be carried out in a complex environment, and the target can be occluded by other objects. Purely reactive robotic systems would have problems on trying to accomplish this task, since they do not build any model of the environment. If the target were lost, it would be difficult to recover its visibility and continue the navigation towards it. For this reason, we thought that the robot should build a map of the environment in order to navigate through it. The information stored in the map must permit the robot to compute its location,

the location of the target, and how to get to this target. Although the objective of this research is to develop a navigation system for indoor environments, we have used a map representation that also works outdoors, since this is the next milestone of the project in which we are involved. Thus, instead of using a grid-based approach, the most widely used approach for indoor environments, we have used a topological one, most appropriate also for outdoors.

Our approach is based on the model proposed by Prescott (1996). The principles underlying this model are inspired by studies of animal and human navigation and wayfinding behavior. This model, called *beta-coefficient system*, does not only deal with how to represent the environment as a map, but also adds a mechanism for computing the location of landmarks when they are not visible, based on the relative positions of the landmarks. This mechanism is what we have used to provide the robot with orientation sense, since it captures the relationship among different places of the environment. The robot makes use of this orientation sense to compute the location of the target when it is occluded by other objects or obstacles.

In this section we firstly describe how Prescott's model works when the robot is able to have exact information about its environment, and then we explain how we have extended it to work with imprecise information. We also describe the method used for dividing the environment into appropriate topological regions, and finally how the topological map is used to navigate through the environment.

3.1. Beta-Coefficient System

The idea behind Prescott's model is to encode the location of a landmark (which we refer to as target—not to confuse with the target or goal of the Navigation system) with respect to the location of three other landmarks. Having seen three landmarks and a target from a viewpoint (e.g., landmarks A, B and C and target T from viewpoint V , in Fig. 1), the system is able to compute the target's position when seeing again the three landmarks, but not the target, from another viewpoint (e.g., V'). A vector, called the β -vector of landmarks A, B, C and T, is computed as

$$\beta = X^{-1} X_T \quad (1)$$

where $X = [X_A X_B X_C]$ and $X_i = (x_i, y_i, 1)^T$, are the homogeneous coordinates of object i , $i \in \{A, B, C, T\}$,

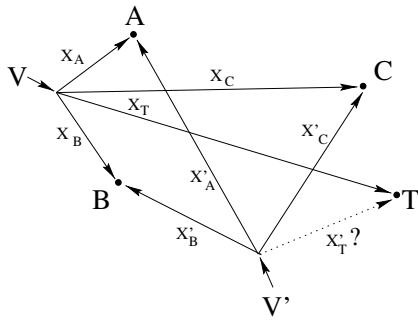


Figure 1. Possible landmark configuration and points of view. Landmarks A, B, C and T are visible from viewpoint V. Only landmarks A, B and C are visible from viewpoint V'.

from viewpoint V. This relation is unique and invariant for any viewpoint if landmarks are distinct and non collinear. The target's location from viewpoint V' is computed as

$$X'_T = X'\beta \quad (2)$$

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 participates in. An example of the network can be seen in Fig. 2(b). 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 to compute the location of non-visible landmarks. It 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 connected to it. 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 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 Fig. 2(b), 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. In this case, 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 get active.

This propagation system adds robustness to the computation of non-visible landmarks, since a landmark can be the target of several beta-units at the same time. Because of imprecision in the perception on landmark locations, the estimates of the location of a target using different beta-units are not always equal. When

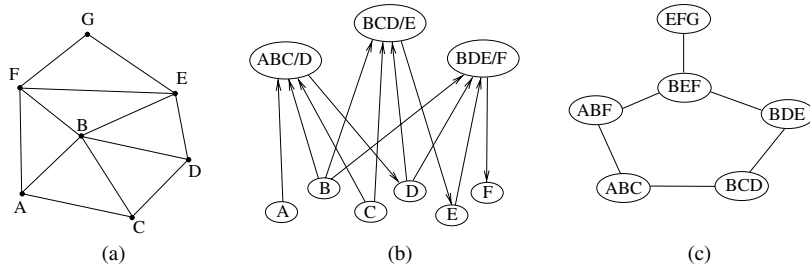


Figure 2. (a) Set of landmarks (b) associated network (partial view) and (c) associated topological map.

this happens, the different location estimates must be combined. Prescott uses the size of the β -vector as the criterion to select one among them. A beta-unit with a smaller β -vector is more precise than those with larger β -vectors (see Prescott (1996) for a detailed discussion on how to compute the estimate error from the size of the β -vector). The propagation system does not only propagate location estimates, but also the size of the largest β -vector that has been used to compute each estimate. When a new location estimate arrives to an object-unit, its location is substituted with the new one if the size of the largest β -vector used is smaller than that used for the last location estimate received.

The network created by object and beta units can be converted into a graph where the nodes represent triangular shaped 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. Although the arcs are created immediately when adding a new node to the graph, the costs can only be assigned after the robot has moved (or tried to move) along the path connecting the two regions. In the case the path is blocked by an obstacle, the arc is assigned an infinite cost, representing that it is not possible to go from one region to the other. This graph is a topological map, and we call its nodes *topological units*. An example of how the topology is encoded in a graph is shown in Fig. 2(c).

This topological map is 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 map, a route to the target can be computed. In Section 3.4 a detailed explanation on how to compute routes to the target is given.

3.2. Extending Prescott's System: Moving to Fuzzy

The beta-coefficient system, as described by Prescott, assumes that the robot can compute the position of the landmarks with small errors, in order to create the beta-units and use the network. But this is never the case: the Vision system provides the robot with inexact information about the location of landmarks. To work with this imprecise information we use fuzzy numbers.

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 (Bojadziev and Bojadziev, 1995). The higher this degree, the higher the confidence that the 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 the 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 $F_A(x) \geq \alpha, \forall x \in [a_1, a_2]$.

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

$$\begin{aligned} A + B = C, & \quad \text{s.t. } \{C\}_\alpha = \{A\}_\alpha \oplus \{B\}_\alpha \quad \forall \alpha \\ A - B = C, & \quad \text{s.t. } \{C\}_\alpha = \{A\}_\alpha \ominus \{B\}_\alpha \quad \forall \alpha \\ A \times B = C, & \quad \text{s.t. } \{C\}_\alpha = \{A\}_\alpha \otimes \{B\}_\alpha \quad \forall \alpha \\ A \div B = C, & \quad \text{s.t. } \{C\}_\alpha = \{A\}_\alpha \oslash \{B\}_\alpha \quad \forall \alpha \end{aligned}$$

where the operations \oplus , \ominus , \otimes and \oslash are performed on intervals and are defined as

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

3.2.2. Fuzzy Beta-Coefficient System. To use the beta-coefficient system with fuzzy numbers, we simply perform the calculations described in the previous section using the fuzzy operators defined above. However, because of the nature of fuzzy operators, some landmark configurations may not be feasible (the matrix inversion used for computing the β -vector—Eq. (1)—may produce a division by 0), so not all configurations can be stored on the network.

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.

Another difference with Prescott's model is the criterion used to select among different estimated locations for the same landmark. In our extended system, instead of looking at the size of the β -vectors, we use the imprecision of the estimated location itself. The imprecision of a landmark location, $I(l)$, is computed by combining

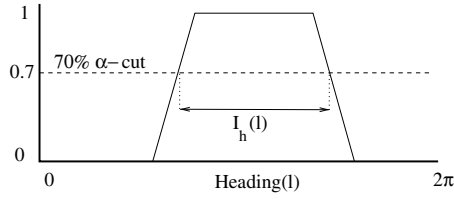


Figure 3. Computation of the imprecision of the heading toward landmark l as a fuzzy number.

the imprecision in the heading and in the distance as follows. $I_h(l)$ is the imprecision in heading, and it is defined by taking the interval corresponding to the 70% α -cut of the fuzzy number representing the heading to the landmark (see Fig. 3). This imprecision is normalized dividing it by its maximum value of 2π . Similarly, $I_d(l)$ is the imprecision in distance, and it is defined as the 70% α -cut of the fuzzy number representing the distance. It is normalized by applying the hyperbolic tangent function, which maps it into the $[0, 1]$ interval. Finally, the two imprecisions are combined according to:

$$I(l) = \lambda \cdot \tanh(\beta \cdot I_d(l)) + (1 - \lambda) \cdot \frac{I_h(l)}{2\pi} \quad (3)$$

where λ weighs the relative importance of the two imprecisions, and β controls how quickly the transformed I_d approaches 1. In our experiments, we set $\beta = 1$ and $\lambda = 0.2$. When an object-unit receives a new location estimate, it computes the imprecision of this estimate, compares it with the imprecision of the current location estimate, and keeps the least imprecise location.

3.3. Building the Map

In Section 3.1 we mentioned that when the robot has four landmarks in its viewframe, it creates a new beta-unit for them. However, with four landmarks, there are four candidates to be the target of the beta-unit. Moreover, if the robot has more than four landmarks in the viewframe, there are many possible beta-units to be created. More precisely, if there are n visible landmarks, there are $\binom{n}{4} \cdot 4$ candidates for being new beta-units. However, it is not feasible to store them all, firstly because of the huge number of combinations, and secondly, and more important, because some configurations are better than others. Thus, some selection criterion must be used.

Before describing the criterion we have used, we explain how the obstacles are represented in the map. We

differentiate two types of obstacles: *point* obstacles and *linear* obstacles. *Point* obstacles are those the robot can easily avoid by slightly modifying its trajectory, since they do not completely block the path. In our indoor environment such obstacles are boxes and bricks. In outdoor environments they could be small rocks, trees, etc. These obstacles do not affect the global navigation, as the Pilot can tackle them alone, so the Navigation system does not take them into account and they are not stored in the map. On the other hand, *linear* obstacles are long obstacles that completely block the path of the robot. They can also be avoided by the Pilot, but the trajectory has to be drastically modified. In our indoor environment we use several bricks to form these obstacles. In an outdoor environment these obstacles could be fences, walls, groups of rocks, etc. Since these obstacles do highly affect the navigation task, they have to be represented in the map, so that they are taken into account when computing routes to the target. The information about these obstacles is stored on the arcs of the topological map. An arc is labeled with an infinite cost to indicate that there is an obstacle between the two regions connected by the arc. Notice that with this representation we can only represent those obstacles placed along the line connecting two landmarks. Although in our experiments we have designed the environments so that they satisfy this condition, the system would also work if it were not satisfied. However, in this latter case, the Navigation system could not take all the obstacles into account, and thus, its performance would be worse. The arcs' labels are updated whenever the Pilot system informs about the presence of an obstacle between two landmarks.

Going back to the selection criterion, given a set of landmarks, for which their location is known, we seek to obtain a set of triangular regions with the following constraints:

- *Low collinearity*: the collinearity of a region is computed as

$$\text{Col}(R) = 1 - \frac{\alpha\beta\gamma}{\left(\frac{\pi}{3}\right)^3} \quad (4)$$

where α , β and γ are the three angles of the triangular region. The best quality is associated to equilateral triangles, where $\alpha = \beta = \gamma = \frac{\pi}{3}$, and hence their collinearity is 0. When one of the angles is 0, landmarks would be maximally collinear and $\text{Col}(R) = 1$. The highest the collinearity, the highest the error on the computation of the β -vector

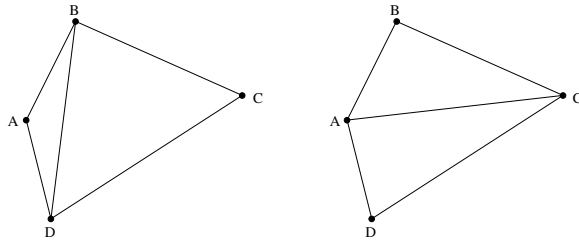


Figure 4. Left: bad set of regions; region ABD is too collinear. Right: good set of regions.

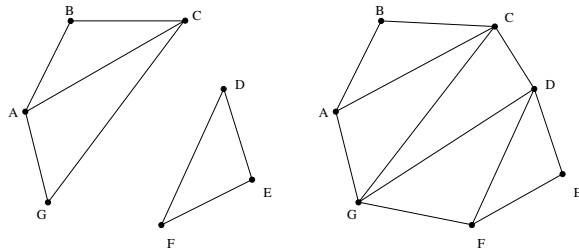


Figure 5. Left: bad set of regions; there are two disconnected components. Right: good set of regions.

and landmark locations (see Prescott (1996) for a detailed explanation). Therefore, the regions with lowest collinearity are preferred. For example, in Fig. 4 the two regions on the right are preferred over the two on the left, since the region ABD is too collinear.

- **Connectivity:** the set of regions must be converted into a graph with a single component, so that there is a path between any two nodes of the graph. In Fig. 5, the set of regions on the left is not acceptable, since there are two disconnected components, whereas in the set on the right all the regions are connected.
- **Convex hull covering:** the regions must cover the convex hull of the set of landmarks, so that the environment is represented completely, with no unrepresented regions. In Fig. 6, the set on the left is not acceptable, since the region DFG is not represented.

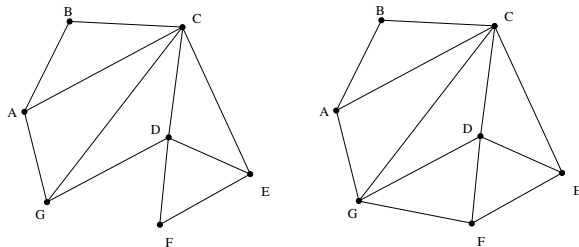


Figure 6. Left: bad set of regions; region DFG is missing. Right: good set of regions.

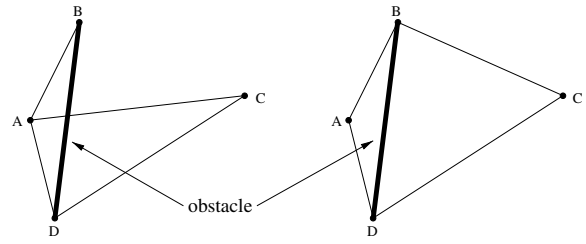


Figure 7. Left: bad set of regions; the obstacle between landmarks B and D is inside the region ACD. Right: good set of regions.

- **Non overlapping:** the regions should not overlap with each other. If this were the case, the robot could be in more than one region at the same time, which could cause some problems when computing routes to the target. For instance, if the robot were in the overlapping area of the two regions, it would make no sense to order the robot to move from one region to the other, since it would already be inside both regions, and the order would not have any effect. Moreover, if one of the overlapping edges is an obstacle, the path from one side of the adjacent region to the other side would be blocked, which is obviously a bad representation of the environment, since the robot must be able to move around the whole space of a region. In Fig. 7, the set of regions on the left is a bad set, since part of the obstacle between landmarks B and D lies inside the region ACD. In this case, the associated graph would have two nodes, ABD and ACD, which would be connected, so the robot would think that it can move from region ABD to region ACD, but it would find the path blocked because of the obstacle.
- **Keep obstacles:** if an edge of a region is marked as an obstacle, this edge must be kept in the map, even if it causes to keep high collinear regions. The obstacle edges are the only ones that cannot be removed from the map. If we did so, the information about the location of obstacles would be lost and would not be taken into account when computing routes to the target.

To compute the optimal set of regions for a given set of landmarks, we have developed an incremental algorithm that treats landmarks one by one to update the map. However, the algorithm only starts working when the location of at least four landmarks are known, since this is the number of landmarks needed to create a beta-unit. With these four landmarks, the mapping algorithm computes the best set of regions according

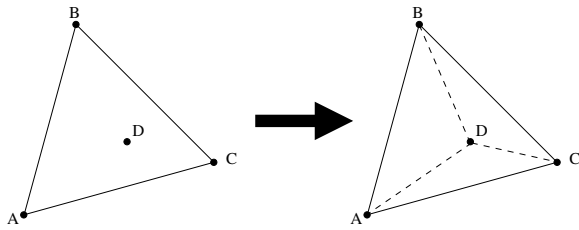


Figure 8. Adding a new landmark (D) located inside an existing region (ABC) resulting in the substitution of the original region for three new regions (ABD, ACD, BCD).

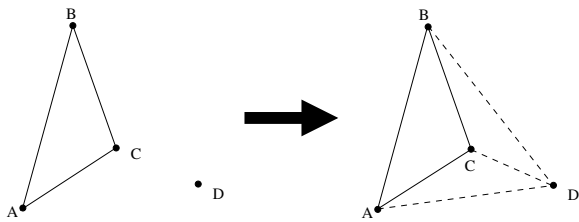


Figure 9. Adding a new landmark (D) located outside any existing region resulting in the addition of two new regions (ACD, BCD).

to the constraints given above. Then, the rest of visible landmarks, if any, are added one by one to the already built map. When adding a new landmark to the map, two situations can happen: (1) the landmark is inside an already existing region, or (2) the landmark is outside any region. In the first case, the region containing the new landmark is substituted by three new regions (see Fig. 8). In the second case, all the possible new regions are created (see Fig. 9). No matter the situation of the landmark, once the new regions have been created, the algorithm checks if the resulting map is still optimal. This optimization consists on analyzing each pair of adjacent regions and checking if their configuration is the optimal according to the constraints. If it finds that some regions could be changed so that a better config-

uration is obtained, it does so. An example of this step by step updating is shown in Fig. 10.

Once the set of regions is computed, new beta and topological units can be created. For each new region a beta-unit is created for each region adjacent to it, taking the three landmarks of the first region as the encoding landmarks, and the the landmark of the second region that is not in the first one as the target. In other words, for each pair of adjacent regions, two “twin” beta-units are created. An example can clarify this explanation: with the regions ABC and ACD shown on the right in Fig. 4, the beta-units ABC/D and ACD/B would be created. One topological unit is also created for each new region, and the graph is updated according to the adjacency of regions. Initially, the arcs are labelled with a default cost of 1, and they are changed to ∞ whenever an obstacle is detected. The topological units corresponding to regions that are not used anymore are removed from the graph. However, beta-units are never removed, since they add robustness to the system, as in Section 3.1.

This triangulation algorithm needs the location of the landmarks be known (either recognized by the Vision system or computed by the beta-coefficient system). However, not all landmark locations can always be known. The algorithm only takes into account those landmarks whose locations are known. This causes that the five constraints explained above are satisfied only for the located landmarks. When one of the unlocated landmarks is seen or computed, some constraints might become unsatisfied. Whenever any constraint is broken, the map is rebuild in order to satisfy again all the constraints. This constraint break can also be caused by the fuzziness of the locations. Because of the imprecision of the locations, the map can suddenly be breaking some of the constraints. To avoid having an inconsistent map, every once in a while the satisfaction of the constraints is checked, and, if needed, the map is rebuilt.

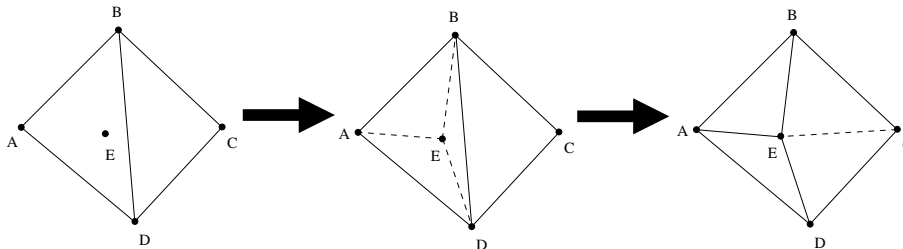


Figure 10. Adding a new landmark (E) into a map with two regions (ABD and BCD): first, region ABD is substituted for three new regions (ABE, ADE, BDE); after that, optimization for regions BCD and BDE is performed and they are substituted for the new regions BCE and CDE.

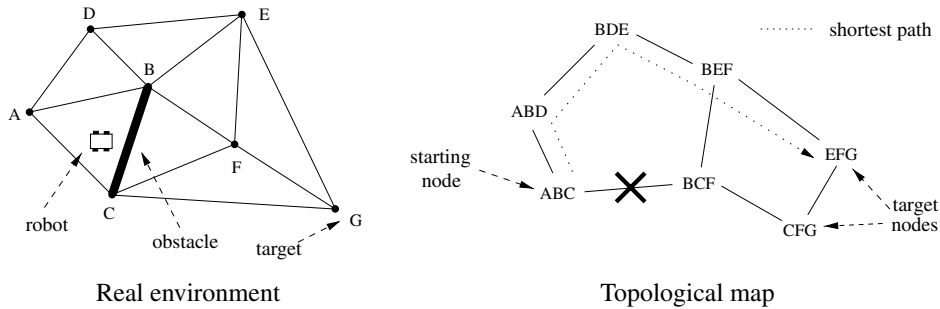


Figure 11. Diverting target computation.

3.4. Navigating Through the Environment

The beta-coefficient system described above provides the means for computing the location of a target even if it is not visible. This is very useful if the robot is navigating in an environment with a high density of landmarks and obstacles that occlude the target. In this case, the robot is able to go towards the target by seeing other landmarks. However, in some cases the obstacles might be blocking the direct path to the target. In this case, knowing the location of the target is not enough and an alternative route to reach it must be computed using the topological map.

Although a route consists of a sequence of regions the robot should navigate through in order to reach the target, only the first region is taken into account. The reason for doing so is that since the environment is never fully known, the robot cannot commit to a given route because it might encounter new landmarks and obstacles that would change the shape of the map, and possibly, the route to the target. Therefore, hereafter, instead of talking about routes, we will talk about diverting targets. A diverting target can be: (1) an *edge* between two landmarks, which the robot has to cross in order to go from one region to another, or (2) a *single landmark* to which the robot has to approach.

When the system is asked for a diverting target in order to reach another target, it first finds out in which region the robot is currently located, using the information about the landmarks whose location is known. This region will be the starting node on the topological map. The shortest path from this node to any of the nodes containing the target landmark (a landmark can be component of several topological regions) is computed. The edge connecting the current region with the next one on the shortest path will be the diverting target. The edge is given as a pair of landmarks, one that has

to be left on the left hand side of the robot and another to be left on the right hand side, so the robot knows which way the edge has to be crossed. An example is shown in Fig. 11. In this case, the robot is in region ABC, the target is G, and the shortest path to the target would be $\{ABC, ABD, BDE, BEF, EFG\}$. Thus, the diverting target would be the edge AB.

However, it could happen that there is no such shortest path. The cases in which such path does not exist are the following:

- The robot is not in any topological region.
- The cost of the shortest path is infinite. This means that the path is blocked by an obstacle, so it is not a valid path.
- The target is not found in any topological region.

To solve the first two cases, the map has to be enlarged with virtual regions through which the robot can navigate. The idea is to let the robot move in an unknown area outside the map. The virtual regions are built by placing some virtual landmarks around the existing map, and creating the appropriate regions using the same algorithm as described in the previous section. An example of these virtual regions is depicted in Fig. 12. To force the robot using regions of the original map, the arcs connecting virtual regions are labeled with a high cost (though not infinite), so that they are used only if it is absolutely necessary. With this enlarged map, the shortest path is computed again. However, it can be that the edge to be crossed contains one virtual landmark. In this case, the edge cannot be given as the diverting target, since the virtual landmarks do not exist on the real environment and cannot be tracked. In this situation, the direction to the middle point of the edge is computed and given as the diverting target. We assume that there is always some free space around

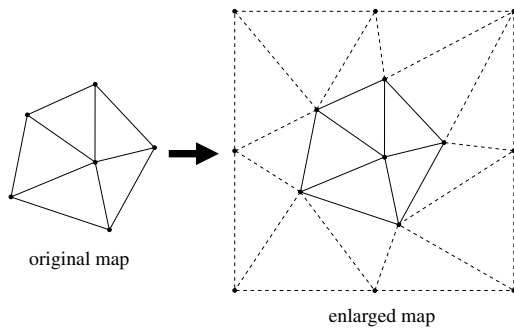


Figure 12. Enlarging the map with virtual regions (dotted lines).

the explored area, so that the regions created with the virtual landmarks can be traversed.

In the case the target is not in any topological region, there is no way to compute which should be the next region to visit, since there is no destination node. When this happens, the diverting target is set to any of the visible landmarks, hoping that in the way to this diverting target, the map is updated and the target for which a diverting target has been computed is incorporated into it.

4. Robot Architecture

Navigation, as the general task of leading a robot to a target destination, is naturally intermingled with other low-level tasks such as obstacle avoidance, and high-level tasks such as landmark identification. We can see each of the tasks, from an engineering point of view, as a system, that is, systems require and offer services one another. These systems need to *cooperate*, since they need one another in order to achieve the overall task of reaching the target. However, they also *compete* for controlling the available actuators of the robot. To exemplify this cooperation and competition, imagine a robot controlled by three systems, the Pilot system, the Vision system and the Navigation system. Actually, these three systems compose the architecture we have used to control our robot, which will be described in detail in the following section. Regarding the cooperation, the Navigation system needs the Vision system to recognize the known landmarks in a particular area of the environment or to find new ones, and it also needs the Pilot system to move the robot towards the target location. Regarding the competition, the Navigation system may need the robot to move towards the target, while the Pilot system may need to change the robot's trajectory to safely avoid an obstacle. Moreover,

the Pilot may need the camera to check whether there is any obstacle ahead and, at the same time, the Navigation system may need to look behind to position the robot by recognizing known landmarks. Thus, some coordination mechanism is needed in order to handle this interaction among the different systems. The mechanism has to let the systems use the available resources in such a way that the combination of these interactions results in the robot reaching its destination.

We propose a general architecture for managing this cooperation and competition. We differentiate two types of systems: *executive systems* and *deliberative systems*. *Executive systems* have access to the sensors and actuators of the robot. These systems offer services for using the actuators to the rest of the systems and also provide information gathered from the sensors. On the other hand, *deliberative systems* take higher-level decisions and require the services offered by the executive systems in order to carry out the task assigned to the robot. Despite this distinction, the architecture is not hierarchical, and the coordination is made at a single level involving all the systems. The services offered by the executive systems are not only available to the deliberative systems; they are also available to the executive systems themselves. Actually, an executive system must compete with the rest of the systems even for the services it is offering. The systems (no matter their type) can exchange information between them (be it sensory information or any other information they could have—e.g., map of the environment). The architecture is depicted in Fig. 13.

The coordination is based on a simple mechanism: *bidding*. Each system (executive and deliberative) generates bids for the services offered by executive

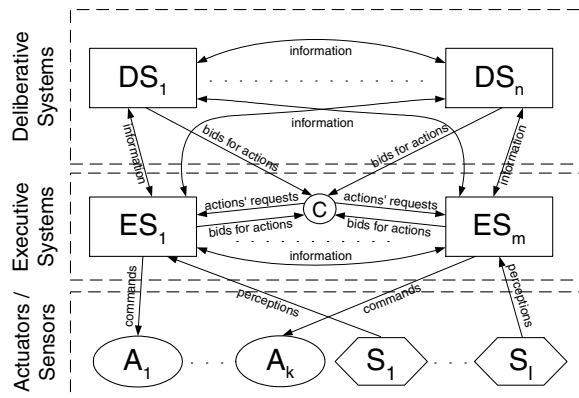


Figure 13. General bidding coordination architecture.

systems, according to the internal expected utility associated to the provisioning of such service. A coordinator receives these biddings and decides which service has to engage each of the executive systems. Executive systems are not obliged to bid, some of them can be only offering services to the rest of systems. However, deliberative systems always bid, since this is the only way in which they can have their decisions executed.

Although we use the term “bidding”, there is no economic connotation as in an auction. That is, systems do not have any amount of money to spend on the bids, nor there is any reward or good given to the winning system. We use it as a way to represent the urgency of a system on having a service engaged. The bids are in the range $[0, 1]$, with high bids meaning that the system really thinks that the service is the most appropriate to be engaged at that moment, and with low bids meaning that it has no urgency on having the service engaged.

This bidding mechanism is a competitive coordination mechanism, since the action executed by each system is the consequence of a request of one of the systems, not a combination of several requests for actions made by different systems, as would be in a cooperative mechanism.

This modular view conforms an extensible architecture. To extend this architecture with a new capability we would just have to plug in one or more new systems, eventually adding new sensors or actuators, and eventually changing the bidding functions of the existing systems. Not only that, it also permits to recursively have a modular view of each one of the systems, as will be soon seen in the design of our Navigation system. Moreover, this architecture is not thought only for navigation purposes since its generality permits to use it for any task that could be assigned to a robotic system.

For our specific robot navigation problem, we have instantiated the general architecture described above (see Fig. 14). It has two executive systems, the *Pilot* and *Vision* systems, and one deliberative system, the *Navigation* system. Each system has the following responsibilities. The Pilot is responsible for all motions of the robot, avoiding obstacles if necessary. The Vision system is responsible for identifying and tracking landmarks (including the target landmark). Finally, the Navigation system is responsible for taking higher-level decisions in order to move the robot to a specified target. The robot has two actuators: the *wheels’ motors*, used by the Pilot system, and the *camera’ motor*, used by the Vision system. The available sensors are the wheels’ encoders and bumpers, which provide

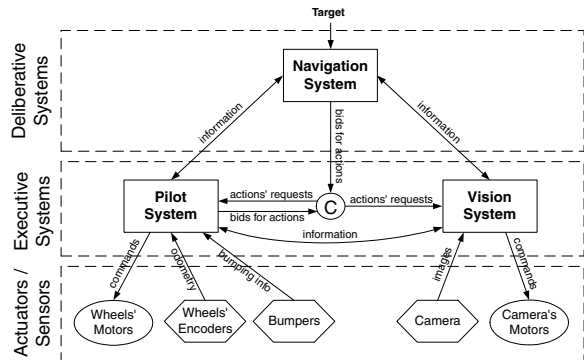


Figure 14. Specific robot architecture.

odometric and *bumping* information to the Pilot, and the *images* obtained by the camera, used by the Vision system to identify landmarks. The Pilot system offers the service of moving the robot in a given direction, and the Vision system offers the service of moving the camera and identifying the landmarks found within a given area. The bidding systems are the Pilot and the Navigation system, while the Vision system does not bid for any service.

In the next sections we describe each of the three systems of the robot architecture, focusing on the Navigation system, the main subject of this research.

4.1. Pilot System

The Pilot is able to safely command the motors that control the robot to move in a given direction. It bids for motion control to avoid obstacles, and also for the control of the camera to look forward in order to detect possible obstacles.

For obstacle avoidance, it uses the information coming from the Vision system and the information stored on the Visual Memory (described in the next section), applying an obstacle growing technique. The obstacles are grown a given size to define forbidden areas occupied by the obstacles. The obstacles are represented as points (for landmarks and simple obstacles) and lines (for linear obstacles between landmarks), which, after growing them, become circles and rounded rectangles, respectively. In our case, the growing size is the diameter of the robot. An example of how the obstacles are grown is shown in Fig. 15. The Pilot uses a simple obstacle avoidance algorithm. It checks whether the robot is about to enter any of the forbidden areas associated to the obstacles. If the robot is in such situation, the Pilot

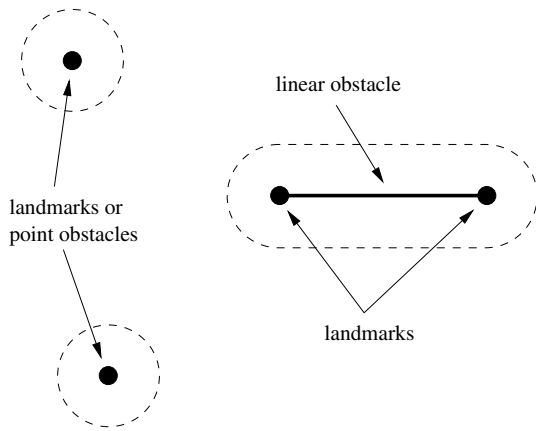


Figure 15. Growing obstacles. Points and solid lines are the obstacles; dotted lines show grown obstacles.

bids to modify the trajectory in order to avoid the obstacle. The modified trajectory is tangential to the grown obstacle to be avoided. Since obstacle avoidance is of maximal importance, the bid should be higher than the other systems. However, it should not be set to the highest possible value, 1, so that there is the possibility of adding a new system that overrides the Pilot (e.g., a teleoperation system). If the robot is in a safe area, the Pilot does not bid at all.

Regarding the bids for camera control, it is based on a function that increases the bid depending on the distance traveled since the last time the robot looked forward:

$$bid(look(ahead)) = \left(\frac{dist_since_last_look}{max_dist_not_looking} \right)^{exp} \quad (5)$$

where $max_dist_not_looking$ is the maximum distance allowed to travel without looking ahead, and exp defines the increasing shape of the bidding function.

The Pilot also informs the Navigation system and the Visual Memory about any obstacle it detects. Whenever it detects a single obstacle (i.e., it bumps into it), it stores the obstacle's location in the Visual Memory, and checks whether it can be part of a larger linear obstacle. Such linear obstacles are detected when a series of single obstacles have been detected along the line connecting two landmarks and the distance between these obstacles is below a given threshold. If this is the case, the Pilot informs the Navigation system about the presence of a blocking obstacle between two landmarks.

4.2. Vision System

The Vision system is able to identify new landmarks in the vision field of the camera and is also able to recognize previously identified landmarks. This system does not bid for any of the available services. A detailed description of the vision system developed to recognize indoor landmarks is given in Section 6.

The Vision system is simple but robust enough to correctly identify the landmarks. Thus, there is no uncertainty about the presence of a given landmark. However, there is some imprecision about its location, since the Vision system only gives approximate distance and angular information. To deal with this imprecision we use the fuzzy techniques described in Section 3.2.

The goal of this research is to develop a vision-based navigation system that does not use any specialized localization device (e.g., GPS) nor odometric information. However, we found that it was very restricting for the Navigation system to use only the visual information available after processing each viewframe. Firstly, because it is very difficult to have more than three landmarks on the view field, since it is very narrow, and the beta-coefficient system needs to have at least four visible landmarks in order to create a new β -unit. But even if four landmarks were in the view field, they would probably be highly collinear, which is not a good configuration for creating β -units. Secondly, it was a very unrealistic behavior to completely forget the landmarks that were not in the view field, even though they had been recently seen. We thought that adding the ability of remembering what has been previously seen would improve the behavior of the robot. Moreover, as it has already been mentioned, we want the robot to imitate the navigational behavior of humans and other animals, and we certainly have the ability of remembering what has been recently seen. A short term memory, called *Visual Memory*, implements this ability, and it is part of the Vision system.

4.2.1. Visual Memory. The Visual Memory stores landmarks and detected obstacles, with their location constantly updated using odometric information. To deal with the imprecision in odometry we use, again, a fuzzy approach. The odometric information coming from the robot is indeed fuzzy information about its motion, used to recompute the location of the objects stored in the Visual Memory. The imprecision of this motion is higher when the robot turns, and lower if it moves straight.

As the robot moves, the imprecision on these locations grows unless the landmarks are recognized again by the Vision system (which obviously reduces their location's imprecision). When the imprecision about the location of a landmark reaches a given upper threshold, the landmark is removed from the Visual Memory. The idea behind this being that the Visual Memory only remembers those landmarks whose location is precise enough.

The information stored in the Visual Memory is treated by the Navigation system in the same way as the information coming from the Vision system. The only difference is that the Visual Memory will be more imprecise than the Vision system. The Pilot system also uses this information to avoid colliding with remembered obstacles and landmarks.

4.3. Navigation System

This research has been mainly motivated by this system. We have used the modular view inspiring the overall robot architecture in the design of the Navigation system. The overall activity of leading the robot to the target destination is decomposed into a set of simple tasks. Working with simple tasks instead of using a single large module carrying out the whole navigation process is the basis of *Behavior-based robotics*. The idea is to divide the overall behavior of the robot into simpler behaviors, each one with its own goal, acting in parallel. These simpler tasks are much easier to build and debug than a larger module, since we only have to focus on separately solving smaller problems. Moreover, it permits to incrementally increase the complexity of the robotic system, that is, adding new capabilities, by simply adding new behaviors, without having to modify already existing code.

The Navigation system is defined to be a multiagent system where each agent is competent in one of these tasks (see Fig. 16). These agents must cooperate, since an isolated agent is not capable of moving the robot to the target, but they also compete, because different agents may want to perform conflicting actions. Again, we use the bidding mechanism to coordinate the agents. Each agent bids for services provided by other robot systems (Pilot and Vision systems), and an additional agent, the communication agent, gathers the different biddings and determines which one to select at any given time. This agent is also the responsible of all the communication between the Navigation system

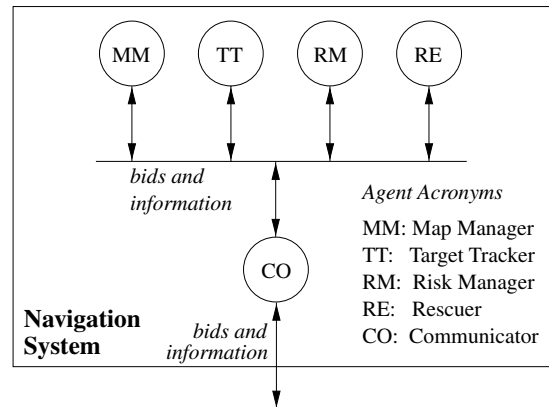


Figure 16. Multiagent view of the navigation system.

and the other systems of the robot. The coordination between the agents is also made through a common representation of the map. Agents consult the map and the Pilot and Vision systems provide information about the environment—position of landmarks, obstacles—which is used to update it.

The local decisions of the 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.

5. The Group of Bidding Agents

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

- to *keep the information on the map* consistent and up-to-date,
- to *keep the target located* with minimum imprecision and *move towards it*,
- to *keep the risk* of losing the target low,
- to *recover* from blocked situations.

Four agents have been designed to fulfill each one of these goals (*Map Manager*, *Target Tracker*, *Risk Manager* and *Rescuer*, respectively), plus a *communicator* agent that is the responsible for communicating the

Navigation system with the other robot systems (Pilot and Vision).

The actions that agents can bid for are:

- Move (*direction*), instructs the Pilot system to move the robot in a particular *direction*,
- Stop, instructs the Pilot system to stop the robot,
- Look (*angle*), instructs the Vision system to identify all the possible landmarks that can be found in the area at *angle* radians from the current body orientation.

Finally, agents may ask one another with respect to the different knowledge they have. For instance, any agent in the society may request from the *Map Manager* to compute the location of the target or a diverting target. Agents may also broadcast messages to the rest of the agents in the society. For example, the *Rescuer* informs about the target to be reached, and the *Target Tracker* informs about the imprecision on the target's location.

In the next sections we describe each one of the agents.

5.1. *Map Manager*

This agent is responsible for maintaining the information of the explored environment in the topological map. The activity of this agent consists on processing the information associated with the incoming viewframes—expanding the graph, creating β -vectors, and asynchronously changing arcs' cost labels when informed by other robot systems. This agent uses the fuzzy beta-coefficient system described in Section 3 to build the map and answer questions about landmark positions.

The *Map Manager* is also the responsible for computing the quality of the set of landmarks in the current viewframe, when required by the *Risk Manager*. This quality is a function of the collinearity of the landmarks. Having a set S of landmarks, their quality is computed as: $q_s = \max\{1 - \text{Col}(S') \mid S' \subseteq S, |S'| = 3\}$ where $\text{Col}(S')$ is computed using the Eq. (4).

This agent also computes diverting targets when asked for by the *Rescuer*. To do so, it uses the topological map, where all path costs are recorded, to compute which should be the next region to visit in order to reach the target. A description of the computation of diverting targets was already given in Section 3.

5.2. *Target Tracker*

The goal of this agent is to keep the target located at any time and move towards it. Ideally, the target should be always within the view field of the camera. If it is not, the imprecision associated to its location is computed by this agent using the information of the map. Actions of other systems are required to keep the imprecision as low as possible.

We model the imprecision as a function on the size of the angle arc, ϵ_θ , from the robot's current position, where the target is thought to be located. When the robot is sure of the position of the target (because it is on the current view field of the camera) we have a crisp direction and, hence, $\epsilon_\theta = 0$ and the imprecision is 0. If the target's location is obtained from the Visual Memory or computed by the *Map Manager*, ϵ_θ is computed as the size of the interval corresponding to the 70% α -cut of the fuzzy number representing the heading to the landmark. When the robot is completely lost, any direction can be correct, $\epsilon_\theta = 2\pi$, and the imprecision level is 1. Thus, the imprecision level is computed as:

$$I_a = \left(\frac{\epsilon_\theta}{2\pi} \right)^\beta \quad (6)$$

where β gives a particular increasing shape to the imprecision function. If β is much smaller than 1, the imprecision increases quickly as the imprecision in angle grows. For β values well over 1, imprecision 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 towards the place where the target is assumed to be. The bids for moving towards the target start at a value $\kappa_1 (\leq 1)$ and decrease polynomially to 0, depending on a parameter α . The rationale of this is that when the imprecision about the target location is low, this agent is confident about the target's position and therefore bids high to move towards it. As the imprecision increases, this confidence decreases and so does the bid. Bids for looking at the target increase from 0 to a maximum of $\kappa_2 (\leq 1)$ and then decrease again to 0. The rationale being that when the imprecision is low there is no urgency in looking to the target, since its location is known with high precision. This urgency starts to increase as the imprecision increases. When the imprecision reaches a level in which the agent has no confidence on the target location, it starts decreasing the bid so as to give the opportunity to other agents to win the bid. The equations involved

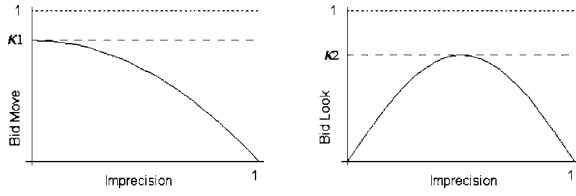


Figure 17. *Target Tracker's* bidding functions.

are:

$$bid(move(\theta)) = \kappa_1 (1 - I_a^{1/\alpha}) \quad (7)$$

$$bid(look(\theta)) = \kappa_2 \sin(\pi I_a) \quad (8)$$

where α controls how rapidly the moving bids decrease, and θ is the crisp angle where the target is thought to be. The bidding functions are shown in Fig. 17.

This agent is constantly asking the *Map Manager* for the location of the target. When it receives an answer (obtaining θ and ϵ_θ), it computes the imprecision and informs the rest of the agents about it. If the *Target Tracker* is not informed about the target's location within a given time limit, it sets the imprecision level to 1.

The behavior described above is applied when the goal is to reach a single landmark. However, as mentioned in Section 3.4, the goal can also be to cross the edge connecting two landmarks (if the *Rescuer* has set it as the diverting target). In this latter case this agent is constantly asking for the location of the two landmarks (thus, obtaining θ and ϵ_θ for each landmark) and computes their associated imprecision. The highest imprecision is used as I_a for computing the bidding values for moving and looking actions. It is also used to decide to where the camera should look at; it looks in the direction of the landmark with highest imprecision. Regarding the motion action, the agent bids to move in the direction of the angle between the two landmarks.

The *Target Tracker* is also the responsible for deciding whether the robot is *at target*. If the target is a single landmark it considers that the robot has reached the target if the upper bound of the α -cut of level ϕ of the fuzzy number modeling the distance to the target is less than δ times the body size of the robot. The parameters ϕ and δ can be tuned to modify the accuracy of the agent. In the case of the target being an edge (between landmarks L_l and L_r), it checks whether the robot is on the adequate side of line connecting the two landmarks. If the robot is on the left of the directed line

through L_l and L_r , it is on the adequate side, that is, the edge has been crossed. If it is on the right of the line, it means that the robot has not still crossed the edge.

5.3. Risk Manager

The goal of this agent is to keep the risk of losing the target as low as possible. While the *Target Tracker's* goal is to locate the target by maintaining it in the camera's view field, this agent tries to keep a reasonable amount of known landmarks, as non collinear as possible, in the surroundings of the robot. The rationale is to have as many visible landmarks as possible so that the *Map Manager* is able to compute the location of the target using the beta-coefficient system when it is not visible nor in the Visual Memory. The less landmarks around whose locations are known, the more risky is the current situation and the higher the probability of losing the target and getting lost. Also, the more collinear the landmarks the higher the error in the location of the target, and thus, the higher the imprecision on its location.

We model the risk as a function that combines: (1) the number of landmarks ahead (elements in set A), (2) the number of landmarks around (elements in set B), and (3) their "collinearity quality" (q_A and q_B). As we have described, these qualities are computed by the *Map Manager*. A minimum risk of 0 is assessed when there are at least six minimally collinear visible landmarks in the direction of the movement. Although the location of only three landmarks are needed in order to use the beta-coefficient system, we want to have additional landmarks around the robot whose locations are known, so that there are more chances to compute the target's location. A maximum risk of 1 is assessed when there are no landmarks ahead nor around:

$$R = 1 - \min\left(1, q_A \left(\frac{|A|}{6}\right)^{\gamma_A} + q_B \left(\frac{|B|}{6}\right)^{\gamma_B}\right) \quad (9)$$

The values γ_A and γ_B determine the relative importance of the situation of landmarks (ahead or around).

Given that the robot cannot decrease the collinearity of the visible landmarks, the only way to decrease the risk level is by increasing the number of landmarks ahead and around. Having more landmarks, besides increasing $|A|$ or $|B|$, also helps on possibly increasing the qualities q_A and q_B .

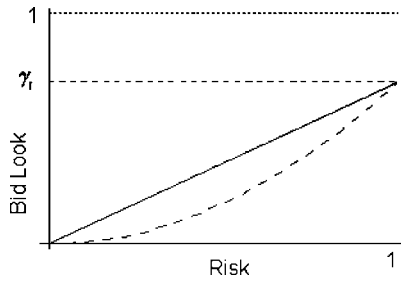


Figure 18. Risk Manager's look bidding functions (look ahead -solid line- and look behind -dashed line-).

We privilege the fact of having landmarks ahead by bidding

$$\text{bid}\left(\text{look}\left(\text{random}\left(\left[-\frac{\pi}{4}, +\frac{\pi}{4}\right]\right)\right)\right) = \gamma_r \cdot R \quad (10)$$

for the action of looking at a random direction in front of the robot and trying to identify the landmarks in that area, if $|A| < 6$, and

$$\text{bid}\left(\text{look}\left(\text{random}\left(\left[+\frac{\pi}{4}, +\frac{7\pi}{4}\right]\right)\right)\right) = \gamma_r \cdot R^2 \quad (11)$$

(which is obviously smaller than $\gamma_r \cdot R$) for the action of looking at a random direction around the robot and trying to identify landmarks, if $|B| < 6$, where γ_r is a parameter to control the maximum value of the bidding function. The bidding functions are shown in Fig. 18.

The behavior of this agent also helps the *Map Manager* build the map when the robot is in an unexplored area. Since it bids for looking for landmarks when there are not many visible, its bids will be high, and thus new landmarks (if there are landmarks, obviously) will be identified and the map will be updated.

5.4. Rescuer

The goal of the *Rescuer* agent is to rescue the robot from problematic situations. These situations may happen due to two reasons. First, the Pilot can lead the robot to a position with a long obstacle ahead that cannot be easily avoided. Second, the imprecision of the location of the target may be too high (over a threshold \bar{I}_a).

If the robot gets blocked, this agent asks the *Map Manager* to compute a diverting target, and informs the

rest of the agents about the new target. If the diverting target computed by the *Map Manager* is just a direction (this means that the robot should cross an edge containing a virtual landmark, as explained in Section 3.4), the *Rescuer* bids for turning the robot in the given direction. In order to have the robot moving in this direction for a short period of time, it sets the target to be a landmark that does not exist. However, the rest of the agents do not know that it does not exist, therefore, they behave as if it was an existing landmark. Thus, the *Map Manager* will not be able to compute its location when asked for by the *Target Tracker*. This latter agent, after asking several times for the location of the target and not receiving any answer, will set the imprecision level to 1, which will cause the *Rescuer* to get active again. The rationale of this “trick” is that during the time the robot has been moving, it will have probably (and hopefully) recognized more landmarks so that the *Map Manager* can compute a better diverting target. Finally, in the case the *Map Manager* fails to compute a diverting target, the *Rescuer* bids for making the robot turn around (a random angle in $\pi \pm \frac{\pi}{6}$), hoping again that with the new direction it detects landmarks that help computing the location of the target or a diverting target. In case the current diverting target cannot be reached, this agent will ask for a new diverting target for the initial target.

On the other hand, if the imprecision of the target's location is too high, the agent bids for stopping the motion and starting a visual scan around the robot, trying to detect as many landmarks as possible. The scan will stop when the imprecision of the location of the target has decreased to an acceptable level, either because it has been recognized by the Vision system or because its location has been computed by the *Map Manager* using other landmarks' locations. Since in this situation no obstacle has been detected, the *Rescuer* assumes that the path to the target is not blocked, so there will not be any target change. However, if at the end of the scanning the imprecision level is still too high, it will ask for a diverting target.

This agent also performs a visual scan at the very beginning when the initial target is given, in order to detect some landmarks and start building the map before the robot begins moving to the target. Only after the scan is completed, this agent will inform the other agents which is the target to be reached.

The bidding values for the actions required by this agent are constant (parameter ω) and should be higher than those of the other agents ($\omega > \max(\kappa_1, \kappa_2, \gamma_r)$),

since it is absolutely necessary to execute the actions in order to continue the navigation to the target.

5.5. Communicator

The multiagent system implementing the navigation algorithm communicates with the remaining robot systems through the *Communicator* agent. This agent receives the information about the visible landmarks and obstacles detected, which is passed to the appropriate agents (*Map Manager* and *Rescuer*). This agent also receives bids for actions from the other agents and is responsible for determining which one to select and send as the Navigation system's bid. The actions required may be conflicting or not. For instance, an agent requiring the camera to look behind and another requiring it to identify a new landmark on the right, bid for conflicting actions, that is, actions that cannot be fulfilled at the same time. On the contrary, an agent requiring the robot to move forward, and an agent requiring the camera to look behind might be perfectly non-conflicting. It can be easily seen that the conflicts occur when the actions require the use of the same resource (robot motion or camera control). Thus, the request for actions will be separately treated depending on the resource required: Move and Stop actions on one side, and Look actions on the other. The *Communicator* agent receives the bids for the two different types of actions, and selects the moving action with the highest bid and the looking action with the highest bid. The resulting two action-bid pairs are sent to the Pilot and Vision system, respectively.

As already mentioned, the bidding mechanism implements a competitive coordination mechanism. This mechanism has problems with selfish agents. The problem arises when there is one (or more) agents that always bids very high so that it wins all the bids, thus, not letting the other agents having their actions executed. In this case, there is no coordination at all between the agents, and it is very difficult, if not impossible, to achieve the goal of reaching the target destination. For instance, if we set the *Target Tracker* to bid always higher than the Pilot system, the robot would not be able to avoid any obstacle, and would get stuck if any was encountered. To avoid such problem, the agents and systems should bid rationally, that is, bidding high only when the action is found to be the most appropriate for the current situation, and bidding low when it is not clear that the action will help, giving the op-

portunity to other agents to win the bid. Thus, special attention must be paid when designing the agents and their bidding functions.

To solve this problem we could use a more economic view of the bidding mechanism, assigning a limited credit to each agent, and allowing them to bid only if they had enough credit. With this new system there should also have to be a way to reward the agents. If not, they would run out of credit after some time and no agent would be able to bid. However, we face the credit assignment problem, that is, deciding when to give a reward and which agent or set of agents deserve to receive it. This problem is very common in multiagent learning systems, especially in Reinforcement Learning, and there is not a general solution for it. Each system uses an ad hoc solution for the task being learned. Other possible solutions would be to have a mechanism to evaluate the bidding of each agent, assigning them succeeding or failing bids, or some measure of trust, in order to take or not take into account their opinions. However, we would have again the credit assignment problem. Thus, in the multiagent system reported in this paper we have designed the agents so that they bid rationally, leaving the exploration of these evaluation mechanisms as a line of future research.

6. The Robot Platform

6.1. The Robot

The robot used in the experimentation is an ActiveMedia Pioneer 2 AT. It is a 4-wheel drive all-terrain robot, equipped with a pan and tilt unit with two B&W cameras. It is also equipped with front and rear bumpers for collision detection. The dimensions of the robot are $50 \times 50 \times 26$ (in cm, length \times width \times height). The field of view of the cameras are of 45 degrees, and the pan/tilt unit can pan from +150 (left) to -150 (right) degrees, and tilt from -90 (down) to +90 (up) degrees. Some pictures of the robot are shown in Fig. 19.

Although the final objective of the project we are involved in is to have a completely autonomous robot, we are currently working with off-board control and vision processing, for it is easier for programming and debugging our algorithms. We use a wireless Ethernet to communicate with the robot, and the cameras images are sent through a video sender (see Fig. 20). Once the algorithm will be completely tested, we will only need

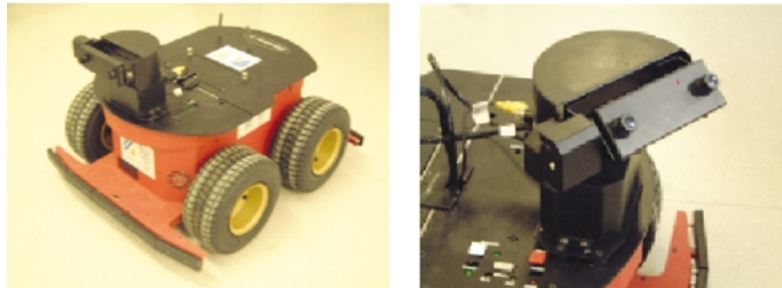


Figure 19. Left: pioneer 2 AT. Right: detail of the pan and tilt unit with two B&W cameras.

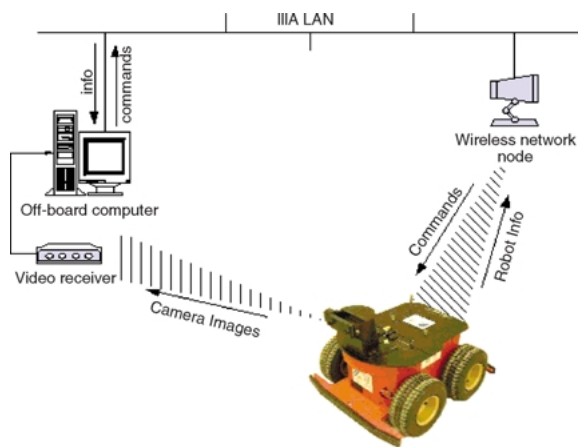


Figure 20. Communication with the robot.

to put it into the on-board computer of the robot to make it fully autonomous.

The experimentation has been carried out in an indoor unstructured (not office-like) environment, with easily recognizable and controlled landmarks and obstacles. The environment is an area of about 50 m², containing ten landmarks plus the target and a few non visible obstacles. A difficulty in real environments is the vision system, as it is highly sensitive to changes in the illumination, which makes it very hard to detect objects. Therefore, we have developed a simple and robust vision system that recognizes bar-coded landmarks. Moreover, the simplicity of the landmarks permits us to easily configure different environments by changing their location.

6.2. Vision

Since we do not focus our research on the vision system of the robot, we did not intend to develop a vision

system capable of recognizing any kind of object, but just a very simple type of landmark. The simplest type we thought of was bar-codes.

Landmark labels have a common part of five vertical black bars, to indicate that it is a landmark, and at the right side of the bars, a vertical binary codification with black and white squares. The binary code is composed of five squares (black meaning 1, white meaning 0), so we have 32 different codes. However, codes 0 and 31 are not used, as they give many problems when trying to identify them, so we have a total of 30 different codes, which is enough for our environment. We have used boxes with the same landmark label on their four sides so the Vision system is able to detect the landmarks from any perspective. The labels are printed on DIN A4 papers, and the dimensions of the boxes are 30 × 30 × 40 (length × width × height), having the labels at the highest part of them. Examples of such landmarks are shown in Fig. 21.

The algorithm for recognizing these landmarks is based on the fact that the pattern of a series of alternated black and white bars of equal width is very unusual. First of all, the image is binarized, since it is in gray scale, and the algorithm needs to have pure black and white images. A *close* operation is also applied. This operation is useful for removing noise from the image. Once the binarization and the close operations are done, the algorithm starts scanning the image line by line, looking for the pattern of black and white bars. When it finds such a pattern, it scans vertically the binary code to identify which landmark has been detected. Depending on the lighting, a landmark can be detected using a binarization threshold, but not detected for other thresholds. Thus, this scanning process is done several times with different thresholds. Once the whole image has been processed with all the thresholds values, the information of all detected landmarks is sent to the Navigation system.

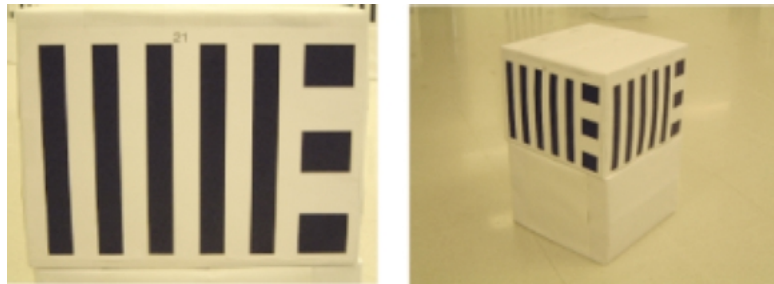


Figure 21. Left: landmark label 21 (code = 10101 = 21). Right: one of the boxes with multiple landmark labels.

Although the robot is equipped with two cameras, we are now processing only the images of one of them, as we have not yet implemented the stereo vision algorithm that would use the images from both cameras to compute the distance to the detected landmarks. However, we simulate that we already have this stereo vision algorithm. To do so, we have designed the landmarks so that all of them have the same size. This way, knowing the height of the bars (in the image) of a landmark, the distance from the robot to that landmark can be computed. The heading is taken as the angle to the central point of the label.

Since the quality of the cameras is not very good, the vision system has some problems for recognizing landmarks that are far from the robot. To have a robust recognition system, we have set that it only informs about the landmarks that are within a distance of 3 meters around the robot. However, even if a landmark is in this “visible area”, the Vision system sometimes misidentifies it. To solve this problem, we force that a landmark has to be recognized several times with the same code before informing about its detection. With all these provisions, landmarks are always identified, therefore there is no uncertainty about the presence of landmarks, although there is imprecision about their exact location.

The fact of the Vision system being only capable of recognizing landmarks not further than 3 meters from the robot, together with the assumption of the initial visibility of the target, restricts the possible environments on which we can experiment. In order to be able to test the Navigation System on more interesting (larger) environments, we have a special landmark that is considered as the target and can be seen from 7–8 meters. This landmark is of the same type as the rest, but its size is doubled, and when computing the distance from the robot to it, this is taken into account.

7. Navigation Experiments

The goal of the experiments is to check whether the Navigation system is able to perform well in different types of environments, and whether the design of each agent is appropriate for obtaining a good overall performance of the Navigation system. The different parameters of the agents’ bidding functions have been previously tuned by running an evolutionary approach on a simulator (Ambastha et al., 2002). The main two variables that describe the complexity of a scenario are:

- *Density of landmarks*: the less landmarks in the scenario, the more risk to get lost, because if the density of landmarks is high, some will always be visible and the Navigation system will be able to compute the location of the target.
- *Density of obstacles*: if the density of obstacles is low, the less chances that the path from the starting point to the target be blocked, and if blocked, there are more chances that the obstacle can be easily avoided, so the Navigation system may not need to compute diverting targets. Contrarily, in a scenario with many obstacles the robot is forced to change direction very often, which might cause increasing the imprecision of the location of landmarks and losing the vision of the target, and then the Navigation System may need to compute a diverting target to reach the initial one.

7.1. Experimentation Results

We describe the experimentation carried out in four scenarios of increasing complexity (see Fig. 22). For each scenario, there is a brief discussion on the results. The relevant statistics of the experiments are shown in Table 1.

We impose the restriction that the landmarks and obstacles in the environment be static. If that were not the

Table 1. Results of experimentation.

Scenario class	Success rate (%)	#d.t.	Winning moving bids (%)			Winning looking bids (%)			
			TT	RE	PS	TT	RM	RE	PS
1	100	0	100	0	0	0	54	0	46
2	100	0	79	0	21	0	66	0	34
3	85	0	78	0	22	0	56	0	44
4	85	0:25 1:55 2:20	71	1	28	1	52	0	47

TT: *Target Tracker*; RM: *Risk Manager*; RE: *Rescuer*; PS: *Pilot System*.



Figure 22. Top left: one of the obstacles used in the environments. Top right: scenario 1. Middle left: scenario 2. Middle right: scenario 3. Bottom left and right: scenario 4.

case, the computed relation among landmarks would be inconsistent, and thus the β -vectors computation would not be valid at all.

In each of these scenarios we have defined several starting points. We have run 20 trials for each starting

point and stored the following statistics:

- Success/failure rate
- Distribution of winning bids among the agents
- Number of diverting targets

7.1.1. Scenario 1. Single Landmark. *Description:* scenario with just one landmark and no obstacles. The task is to reach the landmark.

Results: in this scenario the robot behavior was, as expected, to go directly to the target in a straight line. The *Target Tracker* won 100% of the moving actions it bid for. The *Rescuer* did not bid because it never reached its activation levels. Similarly, as there were no obstacles, the *Pilot* did not have to bid for changing the robot's trajectory. Regarding the looking actions, the *Risk Manager* and the *Pilot* won a similar number of bids. The target was precisely located all the time, so the looking bids of the *Target Tracker* were very low.

7.1.2. Scenario 2. Single Landmark and Small Obstacles. *Description:* scenario with just one landmark and some small obstacles between the robot and the landmark. The small obstacles are not visible and can only be detected by bumping into them. The task is to reach the landmark, avoiding the obstacles detected by the bumpers.

Results: the robot did always reach the target. The winning bids for looking actions were distributed among the *Risk Manager* and the *Pilot*. The *Target Tracker* did not win any of the bids because the imprecision of the target's location was not high enough. As in the previous scenario, the *Rescuer* did not have to intervene at any point. Regarding the moving actions, only the *Pilot* and *Target Tracker* won bids: the *Pilot* when an obstacle was encountered and avoided, and the *Target Tracker* when the path to the target was free.

7.1.3. Scenario 3. Several Landmarks. *Description:* scenario with many landmarks and with no obstacles apart from the landmarks themselves. In order to have an interesting scenario, we placed the target landmark label higher so that it was visible from the starting point, even if there were other landmarks in the view line from the robot to the target. If we had not done so, since the target has to be initially visible, the path from the starting point to the target would always have been clear, which actually corresponds to the first scenario. This change caused that the robot needed to move the camera up and down to be able to have the target landmark in its view field. Thus, we had to change the looking actions in order to incorporate the tilt angle. The task is to reach one of the landmarks (set as target), eventually avoiding others along the way and build a map of the environment.

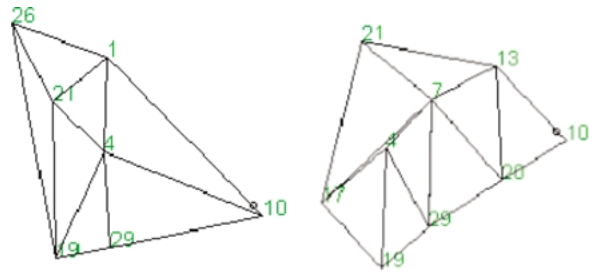


Figure 23. Maps of 2 different scenarios of scenario class 3.

Results: the behavior of the robot in this scenario was very similar to the one exhibited in the previous one. However, it reached the target 85% of the trials. In 15% of the trials it failed because the error on the location of the target made it suppose it was at the target location when it was really not there yet. Some examples of maps built in scenarios of this class during the trials are shown in Fig. 23. In these maps, numbers represent landmarks the robot has seen, and the triangular regions correspond to *topological units* of the *Map Manager's* topological map.

7.1.4. Scenario 4. Several Landmarks and Obstacles. *Description:* in this scenario there are also some non visible long obstacles between some landmarks that completely block the shortest path from the starting point to the target landmark. The task is to reach the target landmark avoiding obstacles and building a map of the environment and using it to compute diverting targets.

Results: the robot did successfully encode the obstacles on the topological map and used it to compute diverting targets. In 55% of the trials only one diverting target was computed in order to avoid a long obstacle blocking the path; the rest of the obstacles were avoided by the *Pilot* system, with no need to compute more diverting targets. In 20% of the trials, however, it was needed to compute another diverting target, since the *Pilot* found the path blocked again by a long obstacle. On the other hand, in 25% of the trials the *Pilot* was able to avoid the obstacles itself, with no need to inform the Navigation system about the path being blocked. Bids for moving actions were distributed very similarly as in the two previous scenarios. The only difference is that the *Rescuer* also won some bids (actually, it only wins one bid for stopping the robot each time it asks for a diverting target). Regarding bids for looking actions, now the *Target Tracker* also won a few bids to look towards the target to decrease its location's imprecision. Be it

for these actions or because the scenario was not complex enough, the imprecision was never high enough so that the *Rescuer* had to bid for looking actions. Again, some of the trials failed because of the error on the target's location.

8. Concluding Remarks and Future Work

In this paper we have presented a novel approach for robot navigation based on the combination of landmark-based navigation, fuzzy distances and angles representation and multiagent coordination based on a bidding mechanism. The objective of our research was to have a robust navigation system with orientation sense for unstructured environments using visual information.

To achieve such objective we have focused our efforts on two main threads: control architectures for autonomous robots, and navigation and mapping methods.

Regarding the control architecture, we have proposed a bidding mechanism for coordinating a group of systems (and agents) that control a robot. This mechanism can be used at different levels of the control architecture. In our case, we have used it to coordinate two of the systems of the robot (Navigation and Pilot systems) and also to coordinate the agents that compose the Navigation system itself. Moreover, the multiagent view of the Navigation system could also be applied to other systems, having a multiagent Pilot or a multiagent Vision system. Using this bidding mechanism, the action actually being executed by the robot is the most urgent one at each point in time, and thus, if the agents bid rationally, the dynamics of the biddings leads the robot to execute the necessary actions in order to reach a given target. An advantage of using such mechanism is that there is no need to create a hierarchy, such as in the subsumption architecture, but it is dynamically changing depending on the specific situation of the robot and the characteristics of the environment. A second advantage is that its modular view conforms an extensible architecture. To extend this architecture with a new capability we would just have to plug in a new system (or agent).

Regarding the mapping and navigation system, we have extended the work presented by Prescott (1996), so that it can be used with fuzzy information about the locations of landmarks in the environment. This is of great importance when working with real robots, as it is impossible to avoid dealing with the imprecision of

real world environments. Together with this extension, we have also developed methods that permit computing diverting targets, needed by the robot when there is no clear path to the goal.

We have obtained successful results on real experimentation showing that the mapping system is capable of building a map of an unknown environment and use this information to move the robot from a starting point to a given target. The experimentation also showed that the bidding mechanism we designed for controlling the robot produces the overall behavior of executing the proper action at each moment in order to reach the target.

The next step on our research is to move the experimentation to more complex scenarios, including outdoor environments. The main difficulty of doing so is the availability of a vision system for outdoors, which we do not have at this moment. However, we think that the successful results obtained in indoor unstructured environments could be quite easily obtained outdoors, since neither the navigation method nor the control architecture are dramatically affected by the differences of indoor/outdoor environments.

Acknowledgments

This research has been partially supported by the Spanish MICYT project ARGOS (DPI2000-1352-C02-02), CICYT Project number TAP97-1209 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.

Note

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

References

- Ambastha, M., Busquets, D., López de Mántaras, R., and Sierra, C. 2002. Evolving a multiagent system for landmark-based robot navigation. Technical report, IIIA.
- Arkin, R.C. 1989. Motor schema-based mobile robot navigation. *Int. J. Robotics Research*, 8(4):92–112.
- Arkin, R.C. 1998. *Behaviour-Based Robotics*. MIT Press.
- Bojadziev, G. and Bojadziev, M. 1995. *Fuzzy Sets, Fuzzy Logic, Applications*, vol. 5 of *Advances in Fuzzy Systems*. World Scientific.

- Brooks, R. 1986. A robust layered control system for a mobile robot. *IEEE J. Robotics and Automation*, RA-2(1):14–23.
- 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)*.
- Dias, M.B. and Stentz, A. 2001. A market approach to multirobot coordination. Technical report, Robotics Institute, Carnegie Mellon University.
- Elfes, A. 1987. Sonar-based real-world mapping and navigation. *IEEE J. Robotics and Automation*, 3(3):249–265.
- Escrig, M.T. and Toledo, F. 2000. Autonomous robot navigation using human spatial concepts. *Int. Journal of Intelligent Systems*, 15:165–196.
- 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.
- 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.
- Kuipers, B. and Byun, Y.-T. 1988. A robust qualitative method for spatial learning in unknown environments. In *Proceedings AAAI-88*, AAAI Press/MIT Press, Menlo Park, CA.
- Levitt, T.S. and Lawton, D.T. 1990. Qualitative navigation for mobile robots. *Artificial Intelligence Journal*, 44:305–360.
- 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.
- Maes, P. 1989. The dynamics of action selection. In *Proceedings of IJCAI'89*, pp. 991–997.
- Martinez, E. and Torras, C. 2001. Qualitative vision for the guidance of legged robots in unstructured environments. *Pattern Recognition*, 34(8):1585–1599.
- Mataric, M.J. 1991. Navigating with a rat brain: A neurobiologically-inspired model for robot spatial representation. In *From Animals to Animats*, J.-A. Meyer and S.W. Wilson (Eds.), MIT Press: Cambridge, MA.
- Moravec, H.P. 1988. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74.
- Nilsson, N.J. 1969. A mobile automaton: An application of AI techniques. In *Proceedings of the 1969 International Joint Conference on Artificial Intelligence*.
- Prescott, T.J. 1996. Spatial representation for navigation in animats. *Adaptive Behavior*, 4(2):85–125.
- Rosenblatt, J. 1995. Damn: A distributed architecture for mobile navigation. In *Proc. of the 1995 AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents*, AAAI Press.
- Stentz, A. 1990. The codger system for mobile robot navigation. In *Vision and Navigation, the Carnegie Mellon Navlab*, C.E. Thorpe (Ed.), Kluwer Academic Pub: Boston, pp. 187–201.
- Sun, R. and Sessions, C. 1999. Bidding in reinforcement learning: A paradigm for multi-agent systems. In *Proceedings 3rd Annual Conference on Autonomous Agents*, O. Etzioni, J.P. Müller, and J.M. Bradshaw (Eds.), Seattle, pp. 344–345.
- Thrun, S. 1998. Bayesian landmark learning for mobile robot navigation. *Machine Learning*, 33(1):41–76.
- Thrun, S. 1998. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence Journal*, 99(2):21–72.

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



Dídac Busquets is a Ph.D. student in the Artificial Intelligence Research Institute (IIIA) of the Spanish Council for Scientific Research (CSIC). In 1999 he received the M.Sc. degree in Computer Science from the Technical University of Catalonia (UPC). His main research interests are the application of Artificial Intelligence to Robotics, focusing on qualitative navigation, map building and multi-agent systems for control architectures. He has other publications related to this paper in international scientific journals, conferences and workshops.



CV Carles Sierra is a Research Full Professor at the Artificial Intelligence Research Institute of the Spanish Council for Scientific Research (CSIC). He received the M.S. in Computer Science (1986) and the Ph. D. in Computer Science (1989) from the Technical University of Catalonia, Barcelona, Spain. He was on sabbatical at Queen Mary and West field College in London in 96/97. His current research interests include: Formal methods, computational reflection, multi-agent systems, uncertainty, robotics, and applications of AI to medicine. Recently, he has been particularly active in the area of electronic commerce. He has participated in around thirty research projects funded by the European Commission and the Spanish Government, and has published more than fifty papers in specialised scientific journals, and a hundred of papers in Scientific conferences and workshops. He is member of the program committees of around a dozen of conferences and workshops per year, and is a member of the editorial board of the journal *Autonomous Agents and Multi-Agent Systems*.



Ramon López de Màntaras is Research Full Professor and Deputy Director of the Artificial Intelligence Research Institute of the Spanish Higher Council for Scientific Research. He received a Ph.D. in Applied Physics (Automatic Control) in 1977 from the University of Toulouse (France), a M.Sc. in Engineering (Computer Science) from the University of California at Berkeley and a Ph.D. in Computer Science in 1981 from the Technical University of Catalonia. He has held a Visiting Professor appointment at the University of

Paris VI in 1986. From 1981 to 1985 he taught at the Computer Science Department of the Technical University of Catalonia. He is a member of the editorial board of several international journals and former Editor-in-Chief of *Artificial Intelligence Communications*. He is the recipient, among other awards, of the “City of Barcelona” Research Prize in 1982, the “European Artificial Intelligence Research Paper Award” of ECCAI in 1987, the “Swets & Zeitlinger Distinguished Paper Award” of the International Computer Music Association in 1997, and the “Best Paper Award” of the 5th International Conference on Case-Based Reasoning in 2003. He is also an ECCAI Fellow. He has been Academic Coordinator of the European Network of Excellence in Machine Learning. Presently working on Case-Based Reasoning, on Autonomous Robots Qualitative Navigation, and on AI applications to Music. He is author or co-author of over 180 scientific papers and of the book “Approximate Reasoning Models” published in the Ellis Horwood Artificial Intelligence Series.