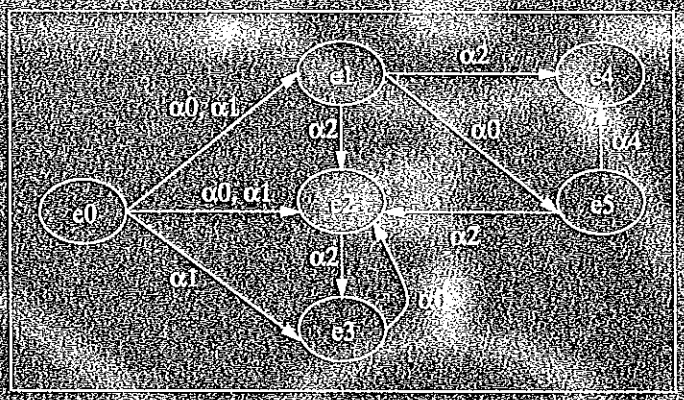
Cristiano Castelfranchi
Yves Lespérance (Eds.)

Intelligent Agents VII

Agent Theories Architectures and Languages



Springer

Lecture Notes in
Artificial Intelligence 1986
Subseries of Lecture Notes in Computer Science

Cristiano Castelfranchi
Yves Lespérance (Eds.)

Intelligent Agents VII

Agent Theories Architectures and Languages

7th International Workshop, ATAL 2000
Boston, MA, USA, July 2000
Proceedings

Springer

Multiagent Bidding Mechanisms for Robot Qualitative Navigation

Carles Sierra, Ramon López de Màntaras, and Dídac Busquets

Artificial Intelligence Research Institute (IIIA),
Spanish Council for Scientific Research (CSIC)
Campus UAB, 08193 Bellaterra, Barcelona, Spain
{sierra, mantaras, didac}@iiia.csic.es

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

1 Introduction

Navigating in outdoor unknown environments is a difficult open problem in robotics. Existing approaches assume that an appropriately detailed and accurate metric map can be obtained through sensing the environment. Even landmark-based navigation approaches assume unrealistically (except if a GPS system is available) accurate distance and direction information between the robot and the landmarks (see Section 6). In this work we propose a fuzzy set based approach to landmark-based navigation in outdoor environments that assumes only very rough vision estimation of the distances and therefore does not rely on GPS information. The motivation being to test the feasibility of animal-like qualitative navigation in machines. Our approach is implemented by means of a multiagent architecture.

The navigation system uses a camera for landmark identification and recognition and has to compete with other systems for the control of the camera. This control is achieved through a bidding mechanism (see Section 3).

Another partner in the project is building a six legged robot with an on board camera. When available, we will test our approach with the real robot. Right now, the navigation system is tested over a simulation of an outdoor environment composed of elements such as buildings, trees, rivers, etc.

The map of the environment is represented by a labelled graph whose nodes represent triangular shaped regions delimited by groups of three non-collinear landmarks and whose arcs represent the adjacency between regions, that is, if two regions share two landmarks, the corresponding nodes are connected by an arc. The arcs are labelled with costs that reflect the easiness of the path between the two corresponding regions. A blocked path would have an infinite cost whereas a flat, hard paved path would have a cost close to zero. Of course these costs can only be assigned after the robot has moved (or tried to move) along the path connecting the two regions. Therefore, the map is built

C. Castelfranchi, Y. Lespérance (Eds.): Intelligent Agents VII, LNAI 1986, pp. 198–212, 2001.

while the robot is moving towards the target. The only a priori assumption is that the target is visible from the initial robot location. Of course, the target can be lost during the navigation and is when the navigation system will need to compute its location with respect to a set of previously seen landmarks whose spatial relation with the target is qualitatively computed both in terms of fuzzy distance and direction.

This paper is structured as follows. Section 2 discusses the map representation, Section 3 the multiagent architecture and bidding mechanism for cooperation and competition among the agents. In Section 4 we describe each individual agent. Section 5 contains an example and Section 6 is devoted to relevant related work. Finally, the paper is concluded in Section 7.

2 Map Representation

For map representation and wayfinding, we will use the model proposed by Prescott in [15]. This model is based on the relative positions of landmarks in order to estimate the location of a target. The method is named the *beta-coefficient system*.

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

2.1 Beta-Coefficient System

Having seen three landmarks and a target (which is also a landmark) from a viewpoint (i.e., landmarks A, B and C and target T from viewpoint V, as shown in Figure 1), then seeing only the three landmarks, but not the target, from another viewpoint (i.e., V'), the system is able to compute the position of the target. The only calculation needed to do this is

$$\beta = X^{-1} X_T$$

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

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

$$X'_T = X'\beta$$

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,

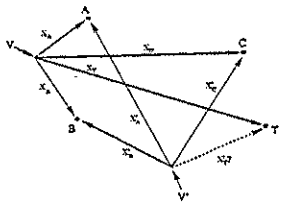


Fig. 1. Possible landmark configuration and points of view.

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 to. An example of the network can be seen in Figure 2b. In this figure there are six object-units and three beta-units. The notation ABC/D is understood as the beta-unit that computes the location of landmark D when knowing the locations of landmarks A, B and C.

This network has a propagation system that permits to compute the location of the non-visible landmarks. The system works as follows: when the robot sees a group of landmarks, it activates (sets the value) of the associated object-units with the egocentric locations of these landmarks. When an object-unit is activated, it propagates its location to the beta-units 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 Figure 2b, if landmarks A, B and C are visible, their object-units will be activated and this will activate the beta-unit ABC/D, computing the location of D, which will activate BCD/E, activating E, and causing BDE/F also to be activated. Knowing the location of only three landmarks, the network has computed the location of three more landmarks that were not visible. 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.

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

This topological graph will be useful when planning routes to the target. Sometimes, when the position of the target is known, the easiest thing to do is to move in a straight line towards it, but sometimes it is not (the route can be blocked, the cost too high...). With the topological graph, a route to the target can be computed. This route will consist in a sequence of regions through which the robot will have to go.

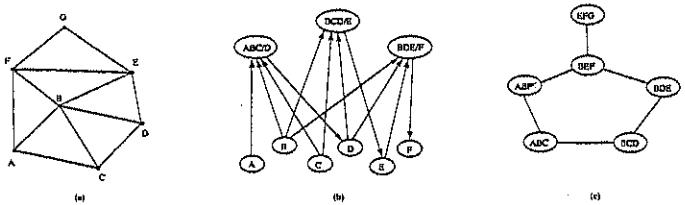


Fig. 2. (a) Landmark configuration (b) Associated network (partial view) and (c) Associated topological map.

2.2 Moving to Fuzzy

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

To use the beta-coefficient system with fuzzy numbers, we simply perform the calculations using the fuzzy operators as defined in [2]. However, because of the nature of fuzzy operators, some landmark configurations may not be feasible (because of the division by 0) and alternative landmarks might be needed.

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.

3 Robot Architecture

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

We propose a model for co-operation and competition based on a simple mechanism: *bidding*. We can see each of the activities (services), from an engineering point of view, as a system (represented as a square in Figure 3), that is, systems require and offer services one another. The model works as follows, each system generates bids for the services offered by the other systems according to the internal expected utility associated to the

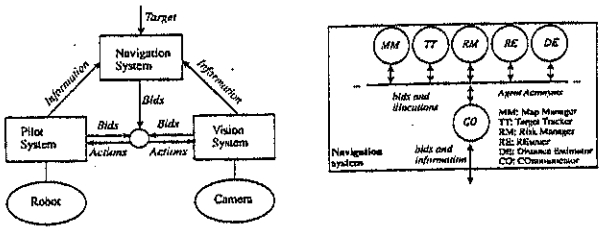


Fig. 3. Robot architecture and multiagent view of the navigation system.

provisioning of such service. Each system provides a number of services and waits for bids for them. Each system decides which service to engage on based on the active bids that have been received.

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

3.1 Pilot System

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

3.2 Vision System

We neither focus on the vision system description, although we build upon the results for landmark identification obtained by [11]. The vision system is able to recognise new landmarks in the vision field of the camera and is also able to identify previously recognised landmarks. It will also provide information about the movement performed by the robot. The algorithmics of the vision system may need a complementary view in order to perform a correct landmark identification. In those cases, it will bid for the control of the pilot to momentaneously divert the robot to take a new perspective on a landmark. This system is also being developed in parallel by another partner in the project.

3.3 Navigation System

We have used the modular view inspiring the overall robot architecture in the design of the navigation system. Again, the overall activity of leading the robot to the target

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

destination is decomposed into a set of simple *agents* that bid for services provided by other robot systems. This system has a communication agent that gathers the different biddings and determines which one to select at any given time. Thus, the navigation system is defined to be a multiagent system where each agent is competent in a particular task. The co-ordination between the agents is made through a common representation of the map. Agents consult the map and suggest changes to it. Other robot systems provide information about the environment —position of landmarks, obstacles, difficulty of terrain, ...— which is also used to update the map.

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

4 The Group of Bidding Agents

In the model reported in this paper we present a group of five agents that take care of different tasks that, when co-ordinated through the bidding mechanism, provide the overall desired behaviour of leading the robot to a target landmark. The tasks are: to *keep the target located* with minimum uncertainty, to *keep the risk* of losing the target low, to *recover* from blocked situations, to *keep the error distance* to landmarks low, and to *keep the information on the map* consistent and up-to-date.

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

The services —actions— that agents can bid for, in order to fulfill their tasks are: Move(*direction*), instructs the pilot system to move the robot in a particular *direction*; Look_for_target(*angle*, *error*), instructs the vision system to look for the target that can be found in the area at $angle \pm error$ radians from the current body orientation; and, Identify_landmarks(*number*, *area*), instructs the vision system to identify a certain number of landmarks in a particular area represented as an angle arc.

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

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

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

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

The code schemas of the agents can be found in Table 1.

4.1 Map Manager

This agent is responsible of maintaining the information of the explored environment as a map. It also maintains the information of the current view frame. As explained in Section 2, a map is a graph where each node corresponds to a group of three landmarks and where arcs are labelled with a passability cost. An arc labelled with an infinite cost represents a non-passable section of the terrain —for instance, an obstacle, a wall, a river ... The activity of this agent consists on processing the information associated with the incoming view frames – expanding the graph and possibly changing the beta-vectors, and asynchronously changing arcs' cost labels when informed by other agents or by other robot systems.

Each time a new view frame arrives, it permits to compute the difference in angle from the last view frame for each landmark. This angle is used to determine the distance to the landmark, d_{L_i}, and also the overall distance error, ϵ_{L_i}, in the following way:

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

where ϵ_d is half of the support of the fuzzy number representing the distance in the direction of the movement of the robot, d is the most confident value for the distance, α_{L_i} is the difference in angle for $landmark_i$ from the last frame and the current frame and β is the actual angle of the movement performed by the robot (see Figure 4).

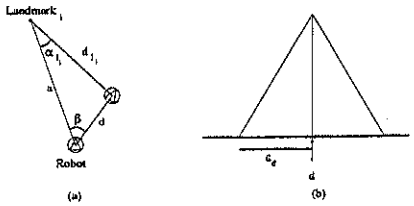


Fig. 4. (a) Robot movement and angle variation and (b) Fuzzy distance and error

This agent will use the fuzzy beta-coefficient system described in Section 2 to answer questions about landmark and target positions, coming from the *target tracker* and the *distance estimator*. It will activate the network with the information of the current view frame, and the propagation system will compute the positions of the non visible landmarks.

It will also be the responsible of computing the quality of the landmarks in the current view frame, when required by the *risk manager*. This quality will be a function of the collinearity of the landmarks. Having a set S of landmarks, their quality is computed as: $q_s = \max\{1 - Col(S') | S' \subseteq S, |S| = 3\}$ where $Col(S') = 1 - \frac{\alpha\beta\gamma}{(\frac{\pi}{3})^3}$, and α, β and γ are the three angles of the triangle formed by the landmarks in S'. The best quality is associated to equilater triangles, where $\alpha = \beta = \gamma = \frac{\pi}{3}$, and hence their collinearity is 0. When one of the angles is 0, landmarks would be maximally collinear and $Col(S') = 1$.

Another responsibility of this agent is to compute diverting targets when asked for by the *rescuer*, possibly backtracking from the current situation. To do this it uses the map where all path costs and previous navigation decisions are recorded. We have explored two ways of computing a new target. With the first method, we select the target that makes the diverting path shorter. With the second, we select the target that minimizes the direction change of the robot.

4.2 Target Tracker

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

We model uncertainty as a function on the angle arc, ϵ_α, from the robot's current position, where the target is thought to be located. When we are sure of the position of the target we have a crisp direction and hence, $\epsilon_\alpha = 0$ and the uncertainty is 0. When we are completely lost, any direction can be correct, $\epsilon_\alpha = 2\pi$, and the uncertainty level is 1. Thus, the uncertainty level is computed as: $U_a = (\frac{\epsilon_\alpha}{2\pi})^\beta$, where β gives a particular increasing shape to the uncertainty function. If β is much smaller than 1, we are going to increase uncertainty very quickly as the imprecision in angle grows. For β values well over 1, uncertainty will grow very slowly until the error angle gets very big.

The actions required by this agent are to move towards the target and to look at the place where the target is assumed to be. Bids for moving towards the target start at a value κ_1 and decrease polinomically (depending on a parameter α) to 0 when the uncertainty increases. Bids for looking at the target follow a sinusoidal increasing from 0, reaching a maximum of κ_2 for uncertainty equal to 0,5 and then decreasing again until 0. This is so because there is no need to look at the target when the uncertainty is very low, and it does not make sense to bid high to look at the target when the uncertainty is already too high.

4.3 Distance Estimator

The goal of this agent is to keep the distance error to the neighboring landmarks as low as possible. This agent will play a very important role at the beginning of the navigation. When analysing the first view frame to obtain the initial landmarks, the error in distance will be maximal, there will be no reference view to obtain an initial estimation of the distance to the target. This agent will generate high bids to move, preferably orthogonally, with respect to the line connecting the robot and the target in order to get another view on it and establish an initial estimation of the distances to the visible landmarks. Similarly, when a target switch is produced (by the intervention of the rescuer) this agent may become relevant again if the distance value to the new selected target is very imprecise. Again, the same process will have as consequence a decrease in the new target distance error.

We model distance uncertainty as the size of the support of the fuzzy number modeling distance. As already mentioned in Subsection 4.1, we will note ϵ_{L_i} the imprecision error in distance to $landmark_i$ and ϵ_t the imprecision error to the current target. Thus,

the uncertainty in distance to the target can be modeled as $U_d = 1 - \frac{1}{e^{\kappa \epsilon_t}}$, where κ is a parameter that changes the shape of U_d; high values of κ will give faster increasing shapes. At the beginning of a run the *distance* is the fuzzy number $[0, \infty]$, $\epsilon_t = +\infty$ and hence $U_d = 1$.

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

The single action required by this agent is to move orthogonally to the line connecting the robot and the target (a in Figure 4).

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

4.4 Risk Manager

The goal of this agent is to keep the risk of losing the target as low as possible. The way to satisfy this goal is by keeping a reasonable amount of landmarks, as non collinear as possible, in the surroundings of the robot. The less landmarks we keep around the more risky is our current exploration and the higher the probabilities of losing the target. Also, the more collinear are the landmarks the higher is the error in the location of the target and thus the higher the uncertainty on its location.

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

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

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

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

4.5 Rescuer

The goal of the rescuer agent is to rescue the robot from problematic situations. These situations may happen due to three reasons. First, the pilot can lead the robot to a position with an obstacle ahead. Second, the uncertainty of the location of the target is too high, over a threshold $\overline{U}_a$. Finally, we can be at a very risky place, that is a place where we

can get easily lost, a value of R over a threshold $\overline{R}$. If any of these situations happen, the rescuer agent asks the map manager for a diverting target. The algorithm uses a stack where the different diverting targets are stacked.

4.6 Communicator

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

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

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

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

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

5 A Navigation Example

We are currently implementing the agents of the navigation system and testing the algorithm on the Webots[2] simulator. Since we do not have the real robot yet, we also simulate the pilot and vision systems.

The system is being implemented in Youtoo[3] connected to the Webots simulator. Each agent is executed as an independent thread, and they use shared memory to simulate messages passing.

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

[3] A multi-threaded dialect of Lisp, http://www.maths.bath.ac.uk/~jap/ak1/youtoo

Table 1. Agents code schemas

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

```
Agent TT(α, β, κ1, κ2, initial_target, initial_angle)=
  Begin
    ⟨target,angle,ε_α⟩ = ⟨initial_target,initial_angle,0⟩
    Repeat
      ask(self,MM,position-landmark(target)?)
      When inform(MM,self,position-landmark(target,
            angle,ε_α,dist,ε_dist)) do
        U_a = (ε_α / (2π))^β
        inform(self,all,uncertainty(U_a))
        inform(self,CO,
          {(Move(angle), κ1(1 − (U_a^(1/α)))),
          (Look_for_target(angle, ε_α),
            κ2 sin (πU_a))})
      endwhen
    Until inform(DE,self,at_target(initial_target))
    On inform(RE,self,target(T)) do target = T
  end
```

```
Agent DE(initial_target, κ, φ)=
  Begin
    target = initial_target
    d = [0, ∞]
    ε_t = +∞
    U_d = 1
    Repeat
      ask(self,MM,position-landmark(target)?)
      [min,max]={d}_φ
      at_target = max ≤ 3*bodyshape
      If at_target then inform(self,all,at_target(target))
      When inform(MM,self,
            position-landmark(target,angle,ε_α,dist,ε_t)) do
        U_d = 1 − 1/(e^(κ ε_t))
        d = dist
        inform(self,CO, {(Move(angle + π/2), U_d)})
      endwhen
    Until inform(self,self,at_target(initial_target))
    On inform(RE,self,target(T)) do target = T
  end
```

```
Agent RM(γ_A, γ_B, γ_r, initial_target)=
  Begin
    target = initial_target
    R = 0
    Repeat
      ask(self,MM,landmarks?)
      When inform(MM,self,
            landmarks(|A|,|B|,q_A,q_B) do
        R = 1 − min(1, q_A(|A|/4)^γA + q_B(|B|/4)^γB)
        inform(self,all,risk(R))
        If |A| < 4 then inform(self,CO,
          {(Identify_landmarks(4, ahead),
            γ_r R)})
        If |B| < 4 then inform(self,CO,
          {(Identify_landmarks(4, around),
            γ_r R^2)})
      endwhen
    Until inform(DE,self,at_target(initial_target))
    On inform(RE,self,target(T)) do target = T
  end
```

```
Agent RE(Ū_a, R̄, initial_target)=
  Begin
    stack = push(emptystack,initial_target)
    Repeat
      When inform(CO,self,Blocked) or (inform(TT, self, uncertainty(U_a)) and U_a > Ū_a)
        or (inform(RM, self, risk(R)) and R > R̄) do
          ask(self,MM,diverting_target?)
      endwhen
    Until inform(DE,self,at_target(initial_target))

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

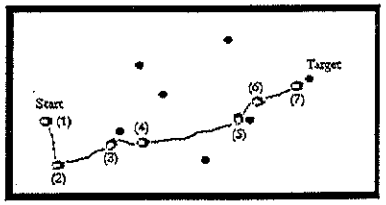



Fig. 5. Robot's path from starting point to the target

We will explain a navigation run as shown in Figure 5. It shows the path followed by the robot from a starting point to a target landmark. The environment is very simple, it is composed only of landmarks (drawn as black circles), and the obstacles are the landmarks themselves.

When the robot starts its search, in point (1), it sees the target, but as there is no distance estimation, the agent DE bids very high to move orthogonally with respect to the target. The agent TT will also bid high to move towards the target, but DE wins and the robot turns. When the robot reaches point (2) the angle to the target has varied enough to compute its distance, so the bids of DE to move in that direction decrease. Since the robot is looking at the target, there is no uncertainty about its location, so TT will bid high to move towards it. As bids of DE are low, TT wins and the robot starts moving to the target. In point (3) the pilot detects an obstacle so it takes the control to avoid it. Once the obstacle is avoided, in point (4), TT's bid wins again and the robot continues moving to the target. The same situation is encountered in point (5). There, the pilot takes the control for a while, and TT resumes it in point (6). The robot finally reaches the target in point (7).

6 Related Work

Mapping for robot navigation dates back to the famous SRI Shakey robot [14]. Recent research on modeling unknown environments is based on two main approaches: occupancy grid-based (or metric), proposed among others by Elfes [4] and Moravec [13], and topological such as those proposed by Chatila [3], Kuipers and Byun [9], Mataric [12] and Kortenkamp [8] among others. Cells in 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 [11]. In this work we assume that the vision system can recognize landmarks. Thrun [19] combines grid-based and topological representations in his work on learning maps for indoor navigation. This is indeed a good idea for indoor environments but for large-scale outdoor environments may not be worth the computational effort of maintaining a grid representation under a topological one.

The incremental map building approach presented here is based on previous work by Prescott [15] that proposed a network model that used barycentric coordinates, also called beta-coefficients by Zipser [20], to compute the spatial relations between landmarks for robot navigation. By matching a perceived landmark with the network, the robot can find its way to a target provided it is represented in the network. Prescott's approach is quantitative whereas our approach uses a fuzzy extension of the beta-coefficient coding system in order to work with fuzzy qualitative information about distances and directions. Levitt and Lawton [10] also proposed a qualitative approach to the navigation problem but assumes an unrealistically accurate distance and direction information between the robot and the landmarks. Another qualitative method for robot navigation was proposed by Escrig and Toledo [5], using constraint logic. However, they assume that the robot has some a priori knowledge of the spatial relationship of the landmarks, whereas our system builds these relationships while exploring the environment.

Regarding the related work on multi-agent architectures, Liscano et al [6] use an activity-based blackboard consisting of two hierarchical layers for strategic and reactive reasoning. A blackboard database keeps track of the state of the world and a set of activities to perform the navigation. Arbitration between competing activities is accomplished by a set of rules that decides which activity takes control of the robot and resolves conflicts. Other hierarchical centralized architectures similar to that of Liscano et al are those of Stentz [17] to drive CMU's Navlab and Isik [7] among others. Our approach is completely decentralized which means that the broadcast of information is not hierarchical. This approach is easier to program and is more flexible and extensible than centralized approaches. Arkin [1] also emphasized the importance of a nonhierarchical broadcast of information. Furthermore, we propose a model for cooperation and competition between activities based on a simple bidding mechanism. A similar model was proposed by Rosenblatt [16] in the CMU's DAMN project. A set of modules cooperated to control a robot's path by voting for various possible actions, and an arbiter decided which was the action to be performed. However the set of actions was pre-defined, while in our system each agent can bid for any action it wants to perform. Moreover, in the experiments carried out with this system (DAMN), the navigation system used a grid-based map and did not use at all landmark based navigation. Sun and Sessions [18] have also proposed an approach for developing multi-agent reinforcement learning systems that uses a bidding process to segment sequences (and divided up among agents) in sequential decision tasks, their goal being to facilitate the learning of the overall task based on reinforcements received during task execution.

7 Concluding Remarks and Future Work

This paper presents a new approach to robot navigation based on the combination of fuzzy representation and multiagent coordination based on a bidding mechanism. The implementation is finished on top of a simulator and we are starting the phase of experimentation.

Experimentation will be carried out along three dimensions: number of environments (one or several), number of points in each environment where to start a run (one or several), and number of samplings of robot parameters per starting point (one or several). This gives eight increasingly more complex experimental scenarios for which the results will be given as path length averages compared either with optimal results or with human obtained results.

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

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

New methods of computing the quality of the landmarks will also be developed. One of these methods takes into account how close are the landmarks from being lost from the current view and how far has the robot moved since the last view frame ahead was taken. For instance, the closer the landmarks to the margins of the frame corresponding to the front view, the less quality associated to them.

We will also explore more complex types of interaction, such as negotiation, between the different agents of the navigation system.

Acknowledgments. This research has been supported by CICYT Project number TAP97-1209 and CIRIT project CeRTAP. Dídac Busquets enjoys the CIRIT doctoral scholarship 2000FI-00191. We acknowledge the discussions held with Thomas Dietterich during his sabbatical stay at IIIA, at the early stage of this research work.

References

1. R.C. Arkin. Motor schema-based mobile robot navigation. *Int. J. Robotics research*, 8(4):92–112, 1989.
2. G. Bojadziev and M. Bojadziev. *Fuzzy sets, fuzzy logic, applications*, volume 5 of *Advances in Fuzzy Systems*. World Scientific, 1995.
3. R. Chatila. Path planning and environment learning in a mobile robot system. In *Proceedings of the 1982 European Conference on Artificial Intelligence (ECAI-82)*, 1982.

4. A. Elfes. Sonar-based real-world mapping and navigation. *IEEE J. Robotics and Automation*, 3(3):249–265, 1987.
5. M. Teresa Escrig and F. Toledo. Autonomous robot navigation using human spatial concepts. *Int. Journal of Intelligent Systems*, 15:165–196, 2000.
6. R. Liscano et al. Using a blackboard to integrate multiple activities and achieve strategic reasoning for mobile-robot navigation. *IEEE Expert*, 10(2):24–36, 1995.
7. C. Isik and A.M. Meystel. Pilot level of a hierarchical controller for an unmanned mobile robot. *IEEE J. Robotics and Automation*, 4(3):242–255, 1988.
8. D. M. Kortenkamp. *Cognitive maps for mobile robots: A representation for mapping and navigation*. PhD thesis, University of Michigan, Computer Science and Engineering Department, Michigan, 1993.
9. B.J. Kuipers and Y.-T. Byun. A robust qualitative method for spatial learning in unknown environments. In *Proceedings AAAI-88*, Menlo Park, CA, 1988. AAAI Press/MIT Press.
10. T.S. Levitt and D.T. Lawton. Qualitative navigation for mobile robots. *Artificial Intelligence Journal*, 44:305–360, 1990.
11. E. Martinez and C. Torras. Qualitative vision for the guidance of legged robots in unstructured environments. *Pattern Recognition*, In press, 2000.
12. M.J. Mataric. Navigating with a rat brain: a neurobiologically-inspired model for robot spatial representation. In J.-A. Meyer and S.W. Wilson, editors, *From Animals to Animats*, Cambridge, MA, 1991. MIT Press.
13. H.P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74, 1988.
14. N.J. Nilsson. A mobile automaton: An application of AI techniques. In *Proceedings of the 1969 International Joint Conference on Artificial Intelligence*, 1969.
15. T.J. Prescott. Spatial representation for navigation in animats. *Adaptive Behavior*, 4(2):85–125, 1996.
16. Julio Rosenblatt. Damn: A distributed architecture for mobile navigation. In *Proceedings of the 1995 AAAI SpringSymposium on Lessons Learned from Implemented Software Architectures for Physical Agents*. AAAI Press, March 1995.
17. A. Stentz. The codger system for mobile robot navigation. In C.E. Thorpe, editor, *Vision and Navigation, the Carnegie Mellon Navlab*, pages 187–201, Boston, 1990. Kluwer Academic Pub.
18. R. Sun and C. Sessions. Bidding in reinforcement learning: A paradigm for multi-agent systems. In J.P. Müller O. Etzioni and J.M. Bradshaw, editors, *Proceedings 3d Annual Conference on Autonomous Agents*, pages 344–345, Seattle, 1999.
19. S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence Journal*, 99(2):21–72, 1998.
20. D. Zipser. Biologically plausible models of placerecognition and place location. In J.L. McClelland and D.E. Rumelhart, editors, *Parallel Distributed Processing: Explorations in the Micro-Structure of Cognition*, Vol. 2, pages 432–470, Cambridge, MA, 1986. Bradford Books.

Performance of Coordinating Concurrent Hierarchical Planning Agents Using Summary Information*

Bradley J. Clement and Edmund H. Durfee

Artificial Intelligence Laboratory, University of Michigan
1101 Beal Avenue, Ann Arbor, MI 48109-2110, USA
+1-734-764-2138
{bradc, durfee}@umich.edu

Abstract. Recent research has provided methods for coordinating the individually formed concurrent hierarchical plans (CHiPs) of a group of agents in a shared environment. A reasonable criticism of this technique is that the summary information can grow exponentially as it is propagated up a plan hierarchy. This paper analyzes the complexity of the coordination problem to show that in spite of this exponential growth, coordinating CHiPs at higher levels is still exponentially cheaper than at lower levels. In addition, this paper offers heuristics, including "fewest threats first" (FTF) and "expand most threats first" (EMTF), that take advantage of summary information to smartly direct the search for a global plan. Experiments show that for a particular domain these heuristics greatly improve the search for the optimal global plan compared to a "fewest alternatives first" (FAF) heuristic that has been successful in Hierarchical Task Network (HTN) Planning.

1 Introduction

In a shared environment with limited resources, agents may have enough information about the environment to individually plan courses of action but may not be able to anticipate how the actions of others will interfere with accomplishing their goals. Prior techniques have enabled such agents to cooperatively seek merges of individual plans that will accomplish all of their goals if possible [7]. This is done by identifying conflicts and adding synchronization actions to the plans to avoid conflicts. Agents can also interleave planning and merging, such that they propose next-step extensions to their current plans and reconcile conflicts before considering extensions for subsequent steps. By formulating extensions in terms of constraints rather than specific actions, a "least commitment" policy can be retained [5]. In addition, recent research has provided these agents with tools to coordinate their hierarchical plans resulting in more flexible abstract solutions that allow the agents to choose refinements of their actions during execution that can withstand some amount of failure and uncertainty [3]. In addition to adding ordering constraints, agents may need to eliminate choices of subplans for accomplishing subgoals. In order to reason about abstract plans to identify and resolve conflicts, information about how the abstract plans must or may be refined into lower level actions

* This work was supported by DARPA (F30602-98-2-0142).

C. Castelfranchi, Y. Lespérance (Eds.): Intelligent Agents VII, LNAI 1986, pp. 213–227, 2001.

acia
ASSOCIACIÓ CATALANA D'INTELLIGÈNCIA ARTIFICIAL

BUTLLETÍ
de l'A C I A

P.U.P. 2.000 PTS.

Tardor 2000, número 22

3r Congrés Català d'Intel·ligència Artificial

VILANOVA I LA GELTRÚ, 5-6-7 D'OCTUBRE DE 2000

PERIODICITAT TRIMESTRAL

Multiagent Bidding Mechanisms for Robot Qualitative Navigation

Carles Sierra, Ramon López de Màntaras and Dídac Busquets
Artificial Intelligence Research Institute (IIIA),
Spanish Council for Scientific Research (CSIC)
Campus UAB, 08193 Bellaterra, Barcelona, Spain
{sierra, mantaras, didac}@iiia.csic.es

Abstract

This paper explores the use of bidding mechanisms to coordinate the actions requested by a group of agents in charge of achieving the task of guiding a robot towards a specified target in an unknown environment. This approach is based on a qualitative (fuzzy) approach to landmark-based navigation.
Keywords: robotic and perception, multiagent systems, fuzzy arithmetics

1 Introduction

Navigating in outdoor unknown environments is a difficult open problem in robotics. Existing approaches assume that an appropriately detailed and accurate metric map can be obtained through sensing the environment. Even landmark-based navigation approaches assume unrealistically (except if a GPS system is available) accurate distance and direction information between the robot and the landmarks (see Section 6). In this work we propose a fuzzy set based approach to landmark-based navigation in outdoor environments that assumes only very rough vision estimation of the distances and therefore does not rely on GPS information. The motivation being to test the feasibility of animal-like qualitative navigation in machines. Our approach is implemented by means of a multiagent architecture.

The navigation system uses a camera for landmark identification and recognition and has to compete with other systems for the control of the camera. This control is achieved through a bidding mechanism (see Section 3).

Another partner in the project is building a six legged robot with an on board camera. When available, we will test our approach with the real robot. Right now, the navigation system is tested over a simulation of an outdoor environment composed of elements such as buildings, trees, rivers, etc.

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

This paper is structured as follows. Section 2 discusses the map representation, Section 3 the multiagent architecture and bidding mechanism for cooperation and competition among the agents. In Section 4 we describe each individual agent. Section 5 contains an example and Section 6 is devoted to relevant related work. Finally, the paper is concluded in Section 7.

2 Map representation

For map representation and wayfinding, we will use the model proposed by Prescott in [15]. This model is based

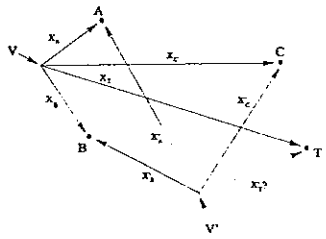


Figure 1: Possible landmark configuration and points of view.

on the relative positions of landmarks in order to estimate the location of a target. The method is named the *beta-coefficient system*.

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

2.1 Beta-coefficient system

Having seen three landmarks and a target (which is also a landmark) from a viewpoint (i.e. landmarks A, B and C and target T from viewpoint V, as shown in Figure 1), then seeing only the three landmarks, but not the target, from another viewpoint (i.e. V'), the system is able to compute the position of the target. The only calculation needed to do this is

$$\beta = X^{-1} X_T$$

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

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

$$X'_T = X'\beta$$

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 to. An example of the network can be seen in Figure 2b. In this figure there are six object-units and three beta-units. The notation ABC/D is understood as the beta-unit that computes the location of landmark D when knowing the locations of landmarks A, B and C.

This network has a propagation system that permits to compute the location of the non-visible landmarks. The system works as follows: when the robot sees a group of landmarks, it activates (sets the value) of the associated object-units with the egocentric locations of these landmarks. When an object-unit is activated, it propagates its location to the beta-units 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 Figure 2b, if landmarks A, B and C are visible, their object-units will be activated and this will activate the beta-unit ABC/D, computing the location of D, which will activate BCD/E, activating E, and causing BDE/F also to be activated. Knowing the location of only three landmarks, the network has computed the location of three more landmarks that were not visible. 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.

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

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

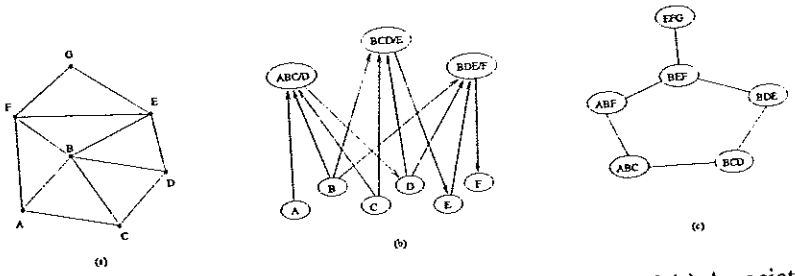


Figure 2: (a) Landmark configuration (b) Associated network (partial view) and (c) Associated topological map.

This route will consist in a sequence of regions through which the robot will have to go.

2.2 Moving to fuzzy

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

To use the beta-coefficient system with fuzzy numbers, we simply perform the calculations using the fuzzy operators as defined in [2]. However, because of the nature of fuzzy operators, some landmark configurations may not be feasible (because of the division by 0) and alternative landmarks might be needed.

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.

3 Robot Architecture

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

We propose a model for co-operation and competition based on a simple mechanism: *bidding*. We can see each of the activities (services), from an engineering point of view, as a system (represented as a square in Figure 3), that is, systems require and offer services one another. The model works as follows, each system generates bids for the services offered by the other systems according to the internal expected utility associated to the provisioning of such service. Each system provides a number of services and waits for bids for them. Each system decides which service to engage on based on the active bids that have been received.

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

3.1 Pilot system

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

3.2 Vision system

We neither focus on the vision system description, although we build upon the results for landmark identification obtained by [11]. The vision system is able

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

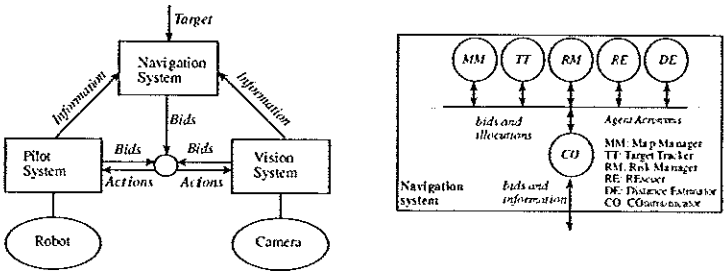


Figure 3: Robot architecture and multiagent view of the navigation system.

recognise new landmarks in the vision field of the camera and is also able to identify previously recognised landmarks. It will also provide information about the movement performed by the robot. The algorithmics of the vision system may need a complementary view in order to perform a correct landmark identification. In those cases, it will bid for the control of the pilot to momentaneously divert the robot to take a new perspective on a landmark. This system is also being developed in parallel by another partner in the project.

3.3 Navigation system

We have used the modular view inspiring the overall robot architecture in the design of the navigation system. Again, the overall activity of leading the robot to the target destination is decomposed into a set of simple *agents* that bid for services provided by other robot systems. This system has a communication agent that gathers the different biddings and determines which one to select at any given time. Thus, the navigation system is defined to be a multiagent system where each agent is competent in a particular task. The co-ordination between the agents is made through a common representation of the map. Agents consult the map and suggest changes to it. Other robot systems provide information about the environment —position of landmarks, obstacles, difficulty of terrain, ...— which is also used to update the map.

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

4 The group of bidding agents

In the model reported in this paper we present a group of five agents that take care of different tasks that, when coordinated through the bidding mechanism, provide the overall desired behaviour of leading the robot to a target landmark. The tasks are: to *keep the target located* with minimum uncertainty, to *keep the risk* of losing the target low, to *recover* from blocked situations, to *keep the error distance* to landmarks low, and to *keep the information on the map* consistent and up-to-date.

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

The services —actions— that agents can bid for, in order to fulfill their tasks are: Move(*direction*), instructs the pilot system to move the robot in a particular *direction*; Look_for_target(*angle*, *error*), instructs the vision system to look for the target that can be found in the area at $angle \pm error$ radians from the current body orientation; and, Identify_landmarks(*number*, *area*), instructs the vision system to identify a certain number of landmarks in a particular area represented as an angle arc.

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

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

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

Whenever the *condition* holds (typically an illocution arriving to the agent) the *action* is executed immediately. Agents will refer to themselves by the special

symbol 'self'. When referring to all the agents of the society, they will use the symbol 'all'.

The code schemas of the agents can be found in Table 1.

4.1 Map manager

This agent is responsible of maintaining the information of the explored environment as a map. It also maintains the information of the current view frame. As explained in Section 2, a map is a graph where each node corresponds to a group of three landmarks and where arcs are labelled with a passability cost. An arc labelled with an infinite cost represents a non-passable section of the terrain —for instance, an obstacle, a wall, a river ...The activity of this agent consists on processing the information associated with the incoming view frames – expanding the graph and possibly changing the beta-vectors, and asynchronously changing arcs' cost labels when informed by other agents or by other robot systems.

Each time a new view frame arrives, it permits to compute the difference in angle from the last view frame for each landmark. This angle is used to determine the distance to the landmark, d_{l_i}, and also the overall distance error, ϵ_{l_i}, in the following way:

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

where ϵ_d is half of the support of the fuzzy number representing the distance in the direction of the movement of the robot, d is the most confident value for the distance, α_{l_i} is the difference in angle for $landmark_i$ from the last frame and the current frame and β is the actual angle of the movement performed by the robot (see figure 4).

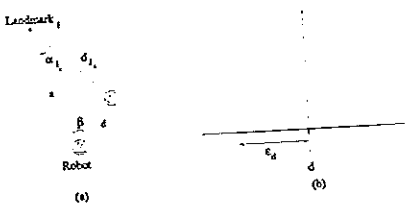


Figure 4: (a) Robot movement and angle variation and (b) Fuzzy distance and error

This agent will use the fuzzy beta-coefficient system described in Section 2 to answer questions about landmark and target positions, coming from the *target tracker* and the *distance estimator*. It will activate the network with the information of the current view frame, and the propagation system will compute the positions of the non visible landmarks.

It will also be the responsible of computing the quality of the landmarks in the current view frame, when required by the *risk manager*. This quality will be a function of the collinearity of the landmarks. Having a set S of landmarks, their quality is computed as: $q_s = \max\{1 - Col(S')|S' \subseteq S| S| = 3\}$ where $Col(S') = 1 - \frac{\alpha\beta\gamma}{(\frac{\pi}{3})^3}$, and α, β and γ are the three angles of the triangle formed by the landmarks in S'. The best quality is associated to equilater triangles, where $\alpha = \beta = \gamma = \frac{\pi}{3}$, and hence their collinearity is 0. When one of the angles is 0, landmarks would be maximally collinear and $Col(S') = 1$.

Another responsibility of this agent is to compute diverting targets when asked for by the *rescuer*, possibly backtracking from the current situation. To do this it uses the map where all path costs and previous navigation decisions are recorded. We have explored two ways of computing a new target. With the first method, we select the target that makes the diverting path shorter. With the second, we select the target that minimizes the direction change of the robot.

4.2 Target tracker

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

We model uncertainty as a function on the angle arc, ϵ_α, from the robot's current position, where the target is thought to be located. When we are sure of the position of the target we have a crisp direction and hence, $\epsilon_\alpha = 0$ and the uncertainty is 0. When we are completely lost, any direction can be correct, $\epsilon_\alpha = 2\pi$, and the uncertainty level is 1. Thus, the uncertainty level is computed as: $U_\alpha = (\frac{\epsilon_\alpha}{2\pi})^\beta$, where β gives a particular increasing shape to the uncertainty function. If β is much smaller than 1, we are going to increase uncertainty very quickly as the imprecision in angle grows. For β values wel over 1, uncertainty will grow very slowly until the erro angle gets very big.

The actions required by this agent are to move to wards the target and to look at the place where the ta get is assumed to be. Bids for moving towards the ta get start at a value κ_1 and decrease polinomically (d pending on a parameter α) to 0 when the uncertain increases. Bids for looking at the target follow a sig soidal increasing from 0, reaching a maximum of κ_2

uncertainty equal to 0,5 and then decreasing again until 0. This is so because there is no need to look at the target when the uncertainty is very low, and it does not make sense to bid high to look at the target when the uncertainty is already too high.

4.3 Distance estimator

The goal of this agent is to keep the distance error to the neighboring landmarks as low as possible. This agent will play a very important role at the beginning of the navigation. When analysing the first view frame to obtain the initial landmarks, the error in distance will be maximal, there will be no reference view to obtain an initial estimation of the distance to the target. This agent will generate high bids to move, preferably orthogonally, with respect to the line connecting the robot and the target in order to get another view on it and establish an initial estimation of the distances to the visible landmarks. Similarly, when a target switch is produced (by the intervention of the rescuer) this agent may become relevant again if the distance value to the new selected target is very imprecise. Again, the same process will have as consequence a decrease in the new target distance error.

We model distance uncertainty as the size of the support of the fuzzy number modeling distance. As already mentioned in Subsection 4.1, we will note ϵ_{l_i} the imprecision error in distance to $landmark_i$ and ϵ_t the imprecision error to the current target. Thus, the uncertainty in distance to the target can be modeled as $U_d = 1 - \frac{1}{e^{\kappa\epsilon_t}}$, where κ is a parameter that changes the shape of U_d; high values of κ will give faster increasing shapes. At the beginning of a run the *distance* is the fuzzy number $[0, \infty]$, $\epsilon_t = +\infty$ and hence $U_d = 1$.

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

The single action required by this agent is to move orthogonally to the line connecting the robot and the target (a in figure 4).

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

4.4 Risk manager

The goal of this agent is to keep the risk of losing the target as low as possible. The way to satisfy this goal is by keeping a reasonable amount of landmarks, as non collinear as possible, in the surroundings of the robot. The less landmarks we keep around the more risky is our current exploration and the higher the probabilities of losing the target. Also, the more collinear are the landmarks the higher is the error in the location of the target and thus the higher the uncertainty on its location.

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

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

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

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

4.5 Rescuer

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

4.6 Communicator

The multiagent system implementing the navigation algorithm communicates with the remaining robot sys-

tems through the communicator agent. This agent receives bids for services from the other agents. The services required may be conflicting or not. For instance, an agent that requires the camera to look behind and another that requires it to identify a new landmark on the right end up with conflicting service biddings, that is services 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. The communicator agent combines the bids for the different services, determines the service with highest bid for each group of conflicting services and outputs service bids accordingly.

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

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

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

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

5 A navigation example

We are currently implementing the agents of the navigation system and testing the algorithm on the Webots[2] simulator. Since we do not have the real robot yet, we also simulate the pilot and vision systems.

The system is being implemented in Youtoo[3] connected to the Webots simulator. Each agent is executed as an independent thread, and they use shared memory to simulate messages passing.

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

[3] A multi-threaded dialect of Lisp, http://www.maths.bath.ac.uk/~jap/ak1/youtoo

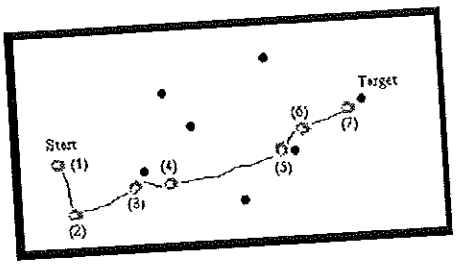


Figure 5: Robot's path from starting point to the target

We will explain a navigation run as shown in Figure 5. It shows the path followed by the robot from a starting point to a target landmark. The environment is very simple, it is composed only of landmarks (drawn as black circles), and the obstacles are the landmarks themselves.

When the robot starts its search, in point (1), it sees the target, but as there is no distance estimation, the agent DE (Distance Estimator) bids very high to move orthogonally with respect to the target. The agent TT (Target Tracker) will also bid high to move towards the target, but DE wins and the robot turns. When the robot reaches point (2) the angle to the target has varied enough to compute its distance, so the bids of DE to move in that direction decrease. Since the robot is looking at the target, there is no uncertainty about its location, so TT will bid high to move towards it. As bids of DE are low, TT wins and the robot starts moving to the target. In point (3) the pilot detects an obstacle so it takes the control to avoid it. Once the obstacle is avoided, in point (4), TT's bid wins again and the robot continues moving to the target. The same situation is encountered in point (5). There, the pilot takes the control for a while, and TT resumes it in point (6). The robot finally reaches the target in point (7).

6 Related work

Mapping for robot navigation dates back to the famous SRI Shakey robot [14]. Recent research on modeling unknown environments is based on two main approaches: occupancy grid-based (or metric), proposed among others by Elfes [4] and Moravec [13], and topological such as those proposed by Chatila [3], Kuipers and Byun [9], Mataric [12] and Kortenkamp [8] among others. Cells in 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 di

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

Agent TT(α, β, κ_1, κ_2, initial_target, initial_angle)=
Begin
$\langle$target,angle,$\epsilon_\alpha\rangle$ = $\langle$initial_target,initial_angle,0$\rangle$
Repeat
ask(self,MM,position-landmark(target)?)
When inform(MM,self,position-landmark(target, angle,ϵ_α,dist,ϵ_{dist})) **do**
$U_a = (\frac{\epsilon_\alpha}{2\pi})^\beta$
inform(self,all,uncertainty(U_a))
inform(self,CO,
{(Move($angle$), $\kappa_1(1-(U_a^{1/\alpha}))$),
(Look_for_target($angle$, ϵ_α),
$\kappa_2 \sin(\pi U_a)$)})
endwhen
Until inform(DE,self,at_target(initial_target))
On inform(RE,self,target(T)) **do** target = T
end

Agent DE(initial_target, κ, ϕ)=
Begin
target = initial_target
d = $[0, \infty]$
$\epsilon_t = +\infty$
$U_d = 1$
Repeat
ask(self,MM,position-landmark(target)?)
[min,max]=$\{d\}_\phi$
at_target = max $\leq$ 3*bodyshape
If at_target **then** inform(self,all,at_target(target))
When inform(MM,self,
position-landmark(target,angle,ϵ_α,dist,ϵ_t)) **do**
$U_d = 1 - \frac{1}{e^{\kappa\epsilon_t}}$
d = dist
inform(self,CO, {(Move($angle + \frac{\pi}{2}$), U_d)})
endwhen
Until inform(self,self,at_target(initial_target))
On inform(RE,self,target(T)) **do** target = T
end

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

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

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

Table 1: Agents code schemas

ambiguate 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 [11]. In this work we assume that the vision system can recognize landmarks. Thrun [19] combines grid-based and topological representations in his work on learning maps for indoor navigation. This is indeed a good idea for indoor environments but for large-scale outdoor environments may not be worth the computational effort of maintaining a grid representation under a topological one.

The incremental map building approach presented here is based on previous work by Prescott [15] that proposed a network model that used barycentric coordinates, also called beta-coefficients by Zipser [20], to compute the spatial relations between landmarks for robot navigation. By matching a perceived landmark with the network, the robot can find its way to a target provided it is represented in the network. Prescott's approach is quantitative whereas our approach uses a fuzzy extension of the beta-coefficient coding system in order to work with fuzzy qualitative information about distances and directions. Levitt and Lawton [10] also proposed a qualitative approach to the navigation problem but assumes an unrealistically accurate distance and direction information between the robot and the landmarks. Another qualitative method for robot navigation was proposed by Escrig and Toledo [5], using constraint logic. However, they assume that the robot has some a priori knowledge of the spatial relationship of the landmarks, whereas our system builds these relationships while exploring the environment.

Regarding the related work on multi-agent architectures, Liscano et al [6] use an activity-based blackboard consisting of two hierarchical layers for strategic and reactive reasoning. A blackboard database keeps track of the state of the world and a set of activities to perform the navigation. Arbitration between competing activities is accomplished by a set of rules that decides which activity takes control of the robot and resolves conflicts. Other hierarchical centralized architectures similar to that of Liscano et al are those of Stentz [17] to drive CMU's Navlab and Isik [7] among others. Our approach is completely decentralized which means that the broadcast of information is not hierarchical. This approach is easier to program and is more flexible and extensible than centralized approaches. Arkin [1] also emphasized the importance of a nonhierarchical broadcast of information. Furthermore, we propose a model for cooperation and competition between activities based on a simple bidding mechanism. A similar model was proposed by Rosenblatt [16] in the CMU's DAMN project. A set of modules cooperated to control a robot's path by voting for various possible actions, and an arbiter decided which was the action to be performed. However the set of actions was pre-defined, while in our system each agent can bid for any action it wants to perform. Moreover, in the experiments carried out with this system (DAMN), the navigation system used a grid-based map and did not use at all landmark based navigation. Sun and Sessions [18] have also proposed an approach for developing multi-agent reinforcement learning systems that uses a bidding process to segment sequences (and divided up among agents) in sequential decision tasks, their goal being to facilitate the learning of the overall task based on reinforcements received during task execution.

7 Concluding remarks and future work

This paper presents a new approach to robot navigation based on the combination of fuzzy representation and multiagent coordination based on a bidding mechanism. The implementation is finished on top of a simulator and we are starting the phase of experimentation.

Experimentation will be carried out along three dimensions: number of environments (one or several), number of points in each environment where to start a run (one or several), and number of samplings of robot parameters per starting point (one or several). This gives eight increasingly more complex experimental scenarios for which the results will be given as path length averages compared either with optimal results or with human obtained results.

The goal of the experimentation will be to tune the different parameters of the agents, which define the

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

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

New methods of computing the quality of the landmarks will also be developed. One of these methods takes into account how close are the landmarks from being lost from the current view and how far has the robot moved since the last view frame ahead was taken. For instance, the closer the landmarks to the margins of the frame corresponding to the front view, the less quality associated to them.

We will also explore more complex types of interaction, such as negotiation, between the different agents of the navigation system.

8 Acknowledgments

This research has been supported by CICYT Project number TAP97-1209 and CIRIT project CeRTAP. Dídac Busquets enjoys the CIRIT doctoral scholarship 2000FI-00191. We acknowledge the discussions held with Thomas Dietterich during his sabbatical stay at IIIA, at the early stage of this research work.

References

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

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

[3] R. Chatila. Path planning and environment learning in a mobile robot system. In *Proceedings of the 1982 European Conference on Artificial Intelligence (ECAI-82)*, 1982.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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