P. Maes, M. Mataric, J.A. Meyer, J. Pollack, and S.W. Wilson (Eds.). Cambridge, MA: The MIT Press/Bradford Books.

Spatial exploration, map learning, and self-positioning with MonaLysa

Jean-Yves Donnart and Jean-Arcady Meyer
AnimatLab
Ecole Normale Supérieure
46, rue d'Ulm 75230 Paris Cedex 05, France
(donnart@wotan.ens.fr - meyer@wotan.ens.fr)

Abstract

This paper describes how the MonaLysa control architecture implements a route-following navigation strategy. Two procedures that allow map building and self-positioning are described, and experimental results are provided that demonstrate that such procedures are robust with respect to noise. This approach is compared to others with similar objectives, and directions for future work are outlined.

1 Introduction

In robotics or animat research, traditional navigation methods that use internal geometrical representations of the environment (Latombe, 1991) are confronted with various implementation difficulties, due to memory and time requirements, as well as sensory and motor errors (Nehmzow, 1995). To overcome these difficulties, several researchers (Chatila and Laumond, 1985; Kuipers and Byun, 1991; Mataric, 1992; Nehmzow, 1995) have advocated the use of various types of topological models to represent the connectivity of the environment, and several such models have been devised that aim to mimic known nervous architectures or behavioral capacities in animals (Muller et al., 1991; Mataric, 1992; Schmajuk and Thieme, 1992; Penna and Wu, 1993; Bachelder and Waxman, 1994; Nehmzow, 1995; Scholkopf and Mallot, 1995). Basically, these topological models endow an animat or a robot with cognitive abilities that make possible to " recognize " the place it is situated in and to " know " that, if it performs a given move in a given direction, it will arrive in another " known " place. In other words, such models belong to the category of so-called *world models*, and they encode a variety of *Stimulus-Response-Stimulus* (S-R-S) information (Riolo, 1991; Roitblat, 1994). At the functional level, they allow the animat or the robot to navigate according to a *route-following* strategy (Gallistel, 1990) and to plan a trajectory from a given starting place to a given goal place, provided that such a trajectory can pass through already known places and involves already experienced moves from place to place. They also facilitate place recognition because such a task may take into account, not only the various features that characterize a given place, but also the specific moves that lead to that place, or depart from it, together with the specific places that are thus connected to it. Moreover, additional information can also be taken into account and facilitate the place-recognition task. This is, for instance, the case with Mataric's robot (Mataric, 1992), that uses the metric information - like distances and orientations - provided by specialized sensors. This is also the case with Kuipers and Byun's animat (Kuipers and Byun, 1991) that uses the information provided by the control architecture that governs its successive moves.

This paper is primarily concerned with the place-recognition and self-positioning functionalities. It describes how it has been possible to add to the basic exploratory abilities of an animat endowed with the MonaLysa control architecture (Donnart and Meyer, 1994; Donnart and Meyer, 1996) the faculties of building a so-called *cognitive map* (Tolman, 1948; O'Keefe and Nadel, 1978; Kuipers, 1982; Thinus-Blanc, 1988; Gallistel, 1990; Poucet, 1993) of its environment, and of locating itself despite the lack of precision in its sensors and actuators. It is structured as follows: Section 2 summarizes the characteristics of MonaLysa that have already been described elsewhere. Section 3 describes how it detects *landmarks* and builds a spatial representation of the environment. Section 4 describes the *self-positioning* procedure. Experimental results are shown in Section 5. The following section discusses the advantages and drawbacks of this approach and ends with perspectives for future work.

2 The MonaLysa architecture.

In the present application, the MonaLysa architecture enables an animat to explore a two-dimensional environment and to navigate from one place to another despite the various obstacles that the environment may contain. The animat is equipped with proximate sensors that keep it informed of the presence or absence of any obstacle in front of it, 90° to its right, or 90° to its left. It is also able to estimate the direction of a goal to be reached in each

of the eight sectors of the space surrounding it. Lastly, it is capable of moving straight ahead, 90° to its right, or 90° to its left. This architecture, which relies upon a hierarchical *classifier system*, is organized into 8 modules - a reactive module, a planning module, a context manager, an auto-analysis module, an internal reinforcement module, a mapping module, a spatial information processor and a place recognition module (Figure 1).

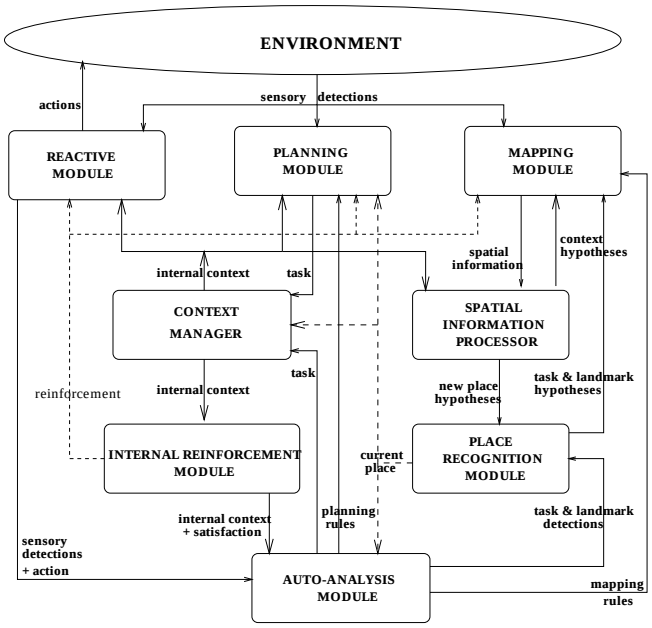

Figure 1: The MonaLysa architecture

The functionalities of the first five modules have been extensively described elsewhere (Donnart and Meyer, 1994; Donnart and Meyer, 1996) and will accordingly be dealt with only briefly. As a whole, they manage a pile of *tasks* that allow the animat to reactively escape from any obstacle it gets trapped into by skirting around it, and to plan a trajectory that will later allow it to avoid the obstacle from a distance. Such an ability relies upon the animat's capacity to analyze its skirting paths and to detect landmarks that will be used for locating itself in the environment.

The reactive module chooses the next move to perform. It uses production rules that take the form :
If <sensory information> and <direction of current goal> Then <action>

The sensory information that these rules take into account is that provided by the animat's proximate sensors.

The planning module decomposes a task into a series of subtasks, that is, into a series of planned trajectories connecting a starting place to an end place. It uses production rules that take the form :
If <sensory information> and <current task> Then <subtask>

The sensory information that these rules take into account is that provided by the animat's proximate sensors, augmented with the animat's coordinates and current orientation.

The context manager contains a pile of the tasks that the animat autonomously generates while it moves in its environment. Such tasks can be either *skirting tasks*, that are posted by the auto-analysis module when the animat detects an obstacle, or *obstacle-avoidance tasks* or subtasks, that are posted by the planning module. At each instant, the task at the top of the pile specifies the *current goal* and the *current task*, i.e., the *internal context* in which reactive and planning rules can be triggered.

The internal-reinforcement module is used to monitor the *satisfaction* of the animat and to adjust the *strengths* of the reactive rules. The satisfaction is an estimation of the success with which a given rule brought the animat closer to, or took it farther from, its current goal. The strengths are used to probabilistically choose which rule to trigger if the condition parts of several rules match the current situation

The role of the auto-analysis module is to analyze the current behavior of the animat in order to alter its current task dynamically and to create new tasks that will enhance its future behavior. It is also responsible for characterizing landmarks in the environment, i.e. places through which it will be useful to travel in the future in order to avoid the obstacles from a distance, or which are used for map building and place recognition.

When an obstacle is detected, as described in (Donnart and Meyer, 1996), the auto-analysis module generates a skirting task that specifies that the animat must cross the line that lies parallel to the direction taken to avoid the obstacle, and that passes through the place where the obstacle has been detected. This task is coded by the pair *<coordinates of the place> < direction vector of the straight line to be crossed>* and is registered at the top of the pile of the context manager. An emergent functionality of the internal dynamics of this module is to enable the animat to skirt around the obstacles it encounters and to extricate itself from dead-ends with arbitrarily complicated shapes.

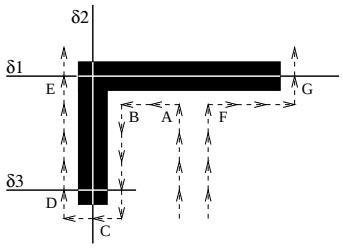

Figure 2: Skirting tasks and skirting trajectories generated by MonaLysa.

For example, two skirting trajectories around a specific obstacle are shown on Figure 2. The left trajectory has been generated through the creation of three skirting tasks at places A, B and C, respectively associated with lines $\delta 1$, $\delta 2$ and $\delta 3$. The task associated with line $\delta 2$ - that will simply be called task $\delta 2$ from now on - has been erased at place C, and tasks $\delta 3$ and $\delta 1$ have been erased respectively at places D and E. The right trajectory has been generated through the creation of task $\delta 1$ at place F and its erasing at place G.

3 Landmark detection and map learning

Another role of the auto-analysis module is to analyze the trajectory followed by the animat in order to detect landmarks in the environment. There are 4 categories of such landmarks : *satisfying* places, *unsatisfying* places, *task-erasing* places and *marker* places.

To detect landmarks belonging to the first two categories, the auto-analysis module monitors the variation of the animat's satisfaction between two successive moves. It also detects *constrained moves*, that is, moves that aren't the most conducive to reaching the goal, and to which the animat resorts because of the presence of an obstacle. Such moves trigger obstacle detection, as described in (Donnart and Meyer, 1996). Places where the satisfaction gradient is positive (resp. negative) and that have been reached through a constrained move are categorized as satisfying (resp. unsatisfying) landmarks. Some such satisfying landmarks are used to define the start and end places of the obstacle avoidance subtasks generated by the planning module, according to a recursive procedure that has also been described in (Donnart and Meyer, 1996). Task-erasing landmarks are places where a task can be erased from the context manager pile. Finally, marker landmarks are places that belong to any of the three preceding categories. They are defined as any first landmark the animat is capable of encountering after the triggering of a skirting task, or as any last landmark the animat is capable of encountering in the context of this skirting task. It turns out that such landmarks are useful for expediting the hierarchical positioning procedure described below.

The " map " that the animat builds while it explores its environment has nothing in common with a classical two-dimensional map. Rather, it is coded as various production rules that are created by the auto-analysis module, that are memorized into the mapping module, and that post their action part to the spatial information processor. Such rules belong to 6 different categories and take the forms :

(1) *If <marker landmark A> and <current task δ > Then <record the pair (A, δ)>*

(2) *If <marker landmark A> and <current task $\delta 2/\delta 1$> Then <record the pair (A, $\delta 2$)>*

(3) *If <link between two landmarks (A, B) detected> and <(X, δ) recorded> Then <support structure [X, Y]/δ >*

(4) *If <unsatisfying landmark A and subtask $\delta 2$ detected> and <(X, $\delta 1$) recorded> Then <support structure [X, Y]/$\delta 1$>*

(5) *If <marker landmark Y> and <(X, δ) recorded> Then <record the structure [X, Y]/δ >*

(6) *If <task-erasing landmark A and new task $\delta 2$ detected> and <current task $\delta 1$ to be erased> Then <record the pair (A, $\delta 2$)>*

where expressions like <task $\delta 2/\delta 1$> and <structure [X, Y]/$\delta 1$> mean that task $\delta 2$ and structure [X, Y] are detected within the context of a specific task $\delta 1$.

Thus, such rules are capable of recording the association between a landmark and a skirting task (type 1), or the association between a landmark and two skirting tasks, which can be hierarchically (type 2) or sequentially (type 6) linked. Such rules are also capable of recording more global *structures* that include two or more landmarks and subtasks (type 5). Finally, they are capable of providing support to some position hypotheses (types 3 and 4), as explained later.

When they involve a skirting task, the corresponding records take into account the coordinates of the place where the task is triggered and the direction vector of the straight line to be crossed. A position and an orientation error are also recorded. When they involve a landmark, the corresponding records take into account the coordinates of the corresponding place and the orientation of the animat in this place. Such records also take into account on-line estimates of the animat's position and orientation errors and the category to which the detected landmark belongs. It must be noted that, although *absolute* coordinates and orientations are recorded into the mapping rules, the matching process between two places or two tasks that will be described later involves *relative* distances or orientations between such places and tasks. Absolute information arbitrarily refer to the animat's initial position. It is taken into account only when matching is ambiguous - for instance when a given place can be matched to two, or more, other places - or when two places are too far apart to be linked within any mapping rule - for instance when they belong to two distinct objects in the environment. It must also be noted that no sensory information is used in the description of the landmarks, even if such information would certainly increase the animat's self-positioning capabilities. Therefore, the self-positioning capacities of MonaLysa do not

depend upon the animat's sensory errors, contrary to what happens in many other realizations.

The process of landmark detection and map learning also entails the management of two lists within the auto-analysis module, a LC_list and a S_list, which are associated with each skirting task triggered. The LC_list records each landmark and each skirting subtask that are encountered or generated within the context of a given skirting task. The S_list records specific portions of the animat's trajectory that are traveled within the context of a given skirting task, that begin and end with a marker landmark, and that can be encapsulated as hierarchical structures likely to be used by mapping rules of types 3, 4 and 5.

The management of these lists and the creation of the mapping rules will be illustrated on the specific example of Figure 3. While the animat is traveling southwards, it encounters an obstacle at place A and triggers 3 skirting tasks successively, characterized by lines $\delta 1$, $\delta 2$ and $\delta 3$. Having moved in turn through places A, B, ... I, it resumes traveling southwards from place I.

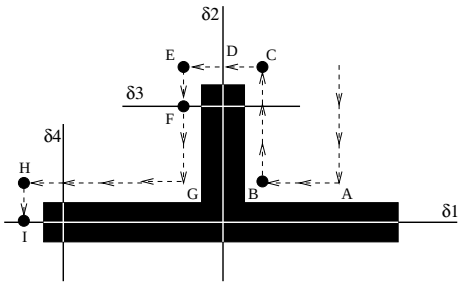

Figure 3: Skirting tasks (δ lines) created and landmarks (filled circles) detected along a trajectory

Along its path, the animat's first encounter with the obstacle occurs at place A. Because the animat cannot proceed straight on, its gradient of satisfaction becomes negative at A. However, this place is not characterized as an unsatisfying place because the move that led to A was not constrained. In place A, the animat chooses to turn right and generates a skirting task characterized by line $\delta 1$. While it moves from A to B, its current goal is to reach the projection of its current position onto line $\delta 1$: the corresponding satisfaction gradient remains constant and no landmark is detected along the path (Figure 4a). However, the corresponding moves are constrained because the presence of the obstacle prevents the animat from turning left, i.e. from reaching its current goal on line $\delta 1$.

In B, the animat encounters a new obstacle, triggers a new task $\delta 2$, and turns to the right. Because its satisfaction gradient with respect to line $\delta 1$ diminishes, and because the preceding move was constrained, place B is detected as both an unsatisfying landmark and a marker landmark within the context of task $\delta 1$. B and $\delta 2$ are added to the LC_list associated with $\delta 1$. Likewise, B is detected as a satisfying landmark and a marker landmark within the context of task $\delta 2$. Thus B is also added to the LC_list associated with $\delta 2$ (Figure 4b). From B to C, the animat's current goal is to cross line $\delta 2$, the corresponding moves are constrained, and the satisfaction gradient remains unchanged.

In C, the animat can turn towards its current goal on line $\delta 2$ and its satisfaction gradient increases : therefore, place C is detected as both a satisfying landmark and a marker landmark associated with $\delta 2$. Place C is then added to the LC_list associated with $\delta 2$ but, because this list begins and ends with two marker landmarks, B and C, these landmarks are encapsulated as a structure [BC] which is added to the S_list associated with $\delta 2$. The corresponding LC_list shrinks into place C (Figure 4c).

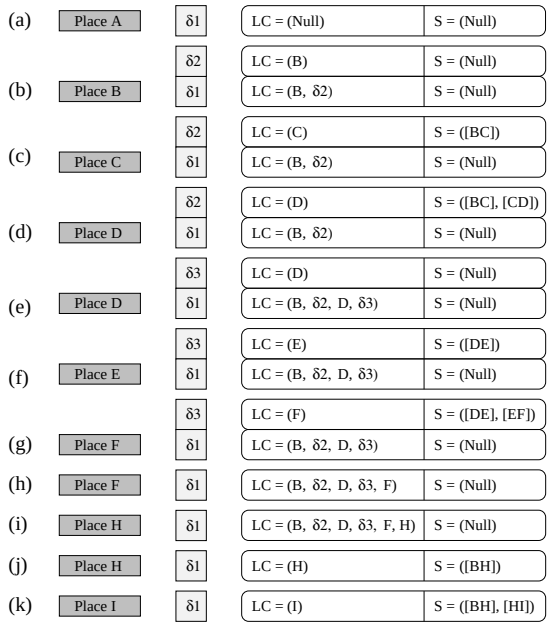

Figure 4: The management of the LC_list and S_list during map learning

When the animat arrives at place D, task $\delta 2$ is erased from the context-manager pile and place D is detected as a task-erasing landmark and a marker landmark that is added to the LC_list associated with $\delta 2$. However, because, this list begins and ends with two marker landmarks, C and D, these landmarks are encapsulated as a structure [CD] which is added to the S_list associated with $\delta 2$. The corresponding LC_list shrinks into place D (Figure 4d). Furthermore, the erasing of task $\delta 2$ allows the creation of 2 type 1 mapping rules, which record the pairs (B, $\delta 2$) and (C, $\delta 2$) and of 2 type 5 mapping rules, which record the structures [B, C]/$\delta 2$ and [C, D]/$\delta 2$. Then, the two lists associated with $\delta 2$ are erased. How-

ever, because the obstacle still prevents the animat from crossing line $\delta 1$, D is also detected as an unsatisfying landmark associated with $\delta 1$ and the skirting task $\delta 3$ is added to $\delta 1$ on the top of the context-manager pile. Finally, within context $\delta 3$, D is also detected as a satisfying landmark and a marker landmark (Figure 4e).

In place E, the animat can turn to its right and move towards its current goal on line $\delta 3$. As the preceeding move was constrained, place E is detected as both a satisfying landmark and a marker landmark associated with $\delta 3$. Place E is added to D, within the corresponding LC_list. This list shrinks to E when structure [D,E] is created and added to the corresponding S_list (Figure 4f).

The same events as those that occurred in place D now occur in place F, which is detected as a task-erasing landmark and a marker landmark in context $\delta 3$, and as a satisfying landmark in context $\delta 1$. In particular, the type 1 and type 5 mapping rules that record pairs (D, $\delta 3$) and (E, $\delta 3$), and structures [C, D]/$\delta 3$ and [D, E]/$\delta 3$ are created (Figure 4g). Then, task $\delta 3$ is erased and the LC_list and S_list of task $\delta 1$ become as shown on Figure 4h.

Place G is not detected as a landmark, because the move that led to it was not constrained. Conversely, place H is detected as both a satisfying and a marker landmark associated with $\delta 1$ and is added to the corresponding LC_list (Figure 4i). Then, the list shrinks to H, while structure [BH] is added to the S_list (Figure 4j).

In place I, structure [HI] is added to the S_list (Figure 4k) and task $\delta 1$ is erased. The type 1 and type 5 mapping rules that record the pairs (B, $\delta 1$) and (H, $\delta 1$), and the structures [B, H]/$\delta 1$ and [H, I]/$\delta 1$ are created. As the navigation between B and H involves several detections of landmarks and subtasks, two type 3 mapping rules and one type 4 are also created, that record the pairs (B, $\delta 2$), (D, $\delta 3$) and (F, H).

From place I, the animat resumes pursuing its initial goal. However, in a purely exploratory mode - i.e., in the absence of any such goal - it might be motivated to explore further the obstacle it is skirting around and, thus, to turn to its left. This would consequently create the task $\delta 4$ shown on Figure 3 and trigger the creation of a type 6 rule, which would record the pair (I, $\delta 4$).

Whatever the case, after the map has been learned, it appears that landmark D, for instance, can be detected as a marker landmark at the beginning of a structure [DE] within context $\delta 3$, or as a task-erasing landmark ending structure [CD] within context $\delta 2$, or as an unsatisfying landmark belonging to structure [BH] within context $\delta 1$. It will be shown in the next paragraph that such different ways of recognizing landmark D substantiate each other and contribute to make the animat's self-positioning easier.

4 Self-positioning

Having learned a map of the environment, the animat must be able to use it to position itself, even if its position and orientation estimates are noisy. In this paragraph, a simplified version of the MonaLysa's self-positioning procedure will be described, assuming that, although no errors were made during a preliminary map-building stage, the animat must now position itself under conditions of noisy position estimates. In the following paragraph, experimental results obtained under more realistic conditions will be presented.

In MonaLysa, the self-positioning procedure relies upon the management of a H_list within the spatial information processor. Such a list is associated with each skirting task triggered and records all the pairs (Ai, δi) on the map that match the animat's *current hypothesis* about its position within the environment.

The procedure also entails the management of a PH_list of *position hypotheses* within the place-recognition module, each such hypothesis being characterized by an *error estimate* and a *confidence estimate*. At every time step, one of these hypotheses is the animat's current hypothesis and designates the place where the animat estimates it is the most likely to be. Nevertheless, the animat also considers that it could be situated in other places, although the corresponding hypotheses are less likely to be true. New hypotheses are posted each time a marker is detected in the context of a skirting task and matches a given landmark on the map. This event occurs when a first marker is detected and triggers type 1, 2 or 6 mapping rules, or when a structure is detected and triggers type 5 mapping rules. When the distance between such an hypothesized position and a place already recorded on the map - that is, a place recorded by mapping rules of types 1, 2, 5, or 6 - doesn't exceed the animat's length, the memorized place replaces the hypothesized position, and the error estimate associated with this position is reset to the memorized error, i.e., to 0 if the map has been learned without error. Between two such reset episodes, the error estimates of each position hypothesis increase with the distance actually traveled by the animat in a manner that will not be described here for lack of space.

As to the confidence estimate associated with each position hypothesis, it increases when the corresponding position is replaced by a landmark already recorded on the map, the importance of this effect being greater when the landmark belongs to a structure than when it doesn't. Moreover, supports provided by individual components of a given structure can modulate this effect, when mapping rules of types 3 and 4 are triggered. The confidence estimate associated with each position hypothesis decreases when no equivalence with a memorized landmark can be found. However, the exact way such confidence estimates are reset will not be described

herein.

Figures 5 and 6 help to describe the self-positioning procedure from a specific example. They assume that the animat, after a preliminary exploratory stage, has detected 4 obstacles in its environment and built the corresponding map, according to the mapping procedure previously described. Such a map involves landmarks B',...N', B",... N" and skirting lines like $\delta 1$', ...$\delta 4$' ,$\delta 1$", ...$\delta 4$" (Figure 5). It also assumes that the animat, currently in a positioning phase, arrives at place A', but erroneously assumes it has reached place A. Thus place A is its current positioning hypothesis - which is supposedly characterized by an error of 250 and a confidence level of 100 - and the top of the context-manager pile contains task $\delta 1$ (Figure 6a).

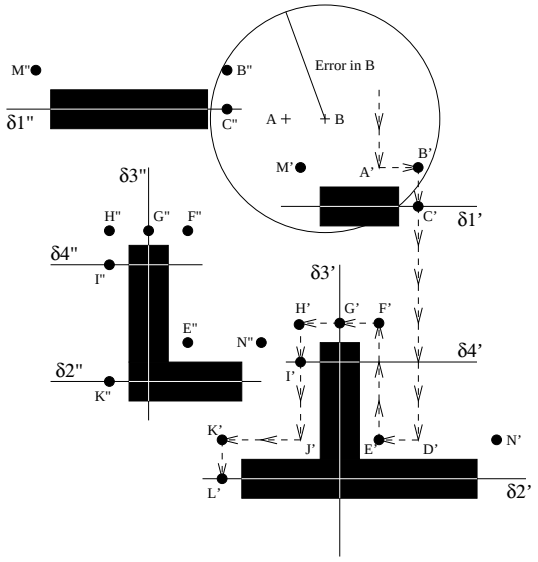

Figure 5: The use of the cognitive map for self-positioning. Filled circles refer to landmarks recorded in the map

In A', the animat turns left and reaches place B', which is characterized as a satisfying landmark B. B is thus added to the LC_list associated with $\delta 1$. Because the coordinates of the animat in places B' and B" fall within the error margin of place B, because the animat's orientation in these places is the same, and because all these places are characterized as landmarks of the same type, places B' and B" are said to *match* place B. Likewise, because the origins of the vectors defining lines $\delta 1$' and $\delta 1$" fall within the error margin associated with the origin of the vector defining $\delta 1$, and because the corresponding orientations are the same, tasks $\delta 1$' and $\delta 1$" match task $\delta 1$. On the contrary, place B doesn't match place M' because, although they are both satisfying landmarks, the animat's orientation in these places differ. Neither does place B match place C", because the latter is a task-erasing landmark and because the animat's orientations in both places differ. Thus, 2 mapping rules of type 1 that have their condition parts matching the current situation are triggered and post pairs (B', $\delta 1$') and (B", $\delta 1$") on the H_list of $\delta 1$. The current positioning hypothesis becomes B, which is characterized by an error of 251, and two new positioning hypotheses, B' and B", are added to the PH_list and are both characterized by an error of 0 and an initial confidence level of 25. The confidence level of B, which is not recognized on the map, decreases to 90 (Figure 6b). In other words, as the animat moves away from a landmark recorded in its map, its position error increases and its confidence in the current positioning hypothesis decreases.

From B' the animat moves to C', where it updates its 3 position hypotheses, and records that, although it has probably moved from B to C [1], it might have moved from B' to C*' or from B" to C*" as well.

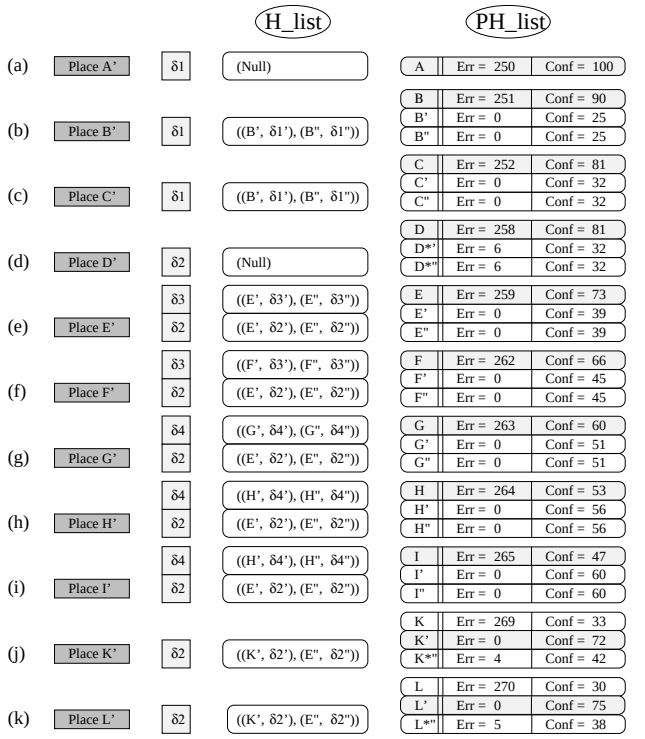

Figure 6: The management of the H_list and PH_list during self-positioning. Shaded components of the PH_list correspond to the animat's current position hypotheses.

At C', the current position C is detected as a task-erasing landmark, and structure [BC] is posted on the S_list of $\delta 1$. Current conditions in C match those of C' and C", allowing the triggering of 2 mapping rules of type 5, which suggest that the place the animat is currently in belongs to structures [B'C'] or [B"C"]. Because place B matches places B' and B", because place C matches

[1] Place C, like several other places that are mentioned in the text, is not shown on Figure 5.

places C' and C", and because the relative displacement the animat made between B and C matches the relative displacement it made from B' to C' and from B" to C", structures [B'C'] and [B"C"] match structure [BC], thus supporting the hypothesis that place C could, in fact, be either place C' or C". These hypotheses are accordingly posted to the PH_list. Because positions of C*' and C"* are respectively no more distant from places C' and C" than the animat's length, the positions of C'* and C*" can be replaced by those of C' and C", and their error estimates can be reset to zero. The confidence estimates of C' and C" thus increase to 32 when the confidence estimate of B decreases to 81 (Figure 6c).

Then the animat travels from C' to D', updating its 3 position hypotheses, and records that, although it has probably moved from C to D, it might have moved from C' to D*' or from C" to D*" as well (Figure 6d). Then it turns to the right and generates skirting task $\delta 2$. At place E', the place E is characterized as an unsatisfying landmark that matches places E' and E" and task $\delta 3$ is created. Type 4 mapping rules post pairs (E', $\delta 2$'), (E", $\delta 2$"), (E', $\delta 3$') and (E", $\delta 3$") respectively on the H_list of $\delta 2$ and $\delta 3$, and the two place hypotheses E' and E" are posted on the PH_list. These hypotheses replace the current hypotheses E*' and E*", and their confidence estimates increases to 39, while the confidence estimate of E decreases to 73 (Figure 6e).

At place F', task-erasing landmark F is matched with F' and F", and structure [EF] is matched with structures [E'F'] and [E"F"]. Thus, the confidence estimates of the hypotheses associated with F' and F" continue to increase, while the confidence estimate of the hypothesis associated with F continues to decrease (Figure 6f).

From F' to G', H' and I', the PH_list maintained by MonaLysa is updated according to Figures 6g, 6h and 6i.

When the animat arrives at K', it supposes itself to be at K, and structure [EK] = ((E, $\delta 3$), (G, $\delta 4$), (I, K)) is posted on its S_list. As such a structure matches [E'K'] but not [E"K"], it turns out that K*' can be replaced by K' and that its error estimate can be reset to 0. Because the confidence in hypothesis K' exceeds a given threshold level (i.e., 50) and because the confidence associated with K and K*" does not, the animat's current hypothesis switches from K to K' (Figure 6j). In other words, the animat has positioned itself accurately in its environment.

Had structure [E"K"] matched the detected structure [EK] in the presence of a high noise level, then the supports posted during navigation from E' to K' by the three components of the detected structure [EK] would nevertheless have been more likely to contribute to a greater increase in the confidence estimate of K' than to that of K". Indeed, components (E, $\delta 3$) and (G, $\delta 4$) of [EK] would match the condition parts of type 3 rules that have been created within the contexts of the exploration of [E'K'] and [E"K"] and would receive support from both categories of rules. On the contrary, component (I, K) would be more likely to match the condition part of a type 4 rule created within the context of the exploration of [E'K'] than within the context of [E"K"].

It should be understood that self-positioning success relies upon the fact that MonaLysa has managed simultaneously several hypotheses about the animat's moves and that some of these hypotheses got support every time a given landmark or structure in the environment was matched with a landmark or structure in the map. Thus, the hypothesis that the animat traveled from E' to K' got support not only from the matching of E with E', of F with F', of G with G', of H with H', of I with I', and of K with K', but also from the matching of [EF] with [E'F'], of [FG] with [F'G'], of [GH] with [G'H'], of [HI] with [H'I'], and finally from the matching of the whole structure [EK] with [E'K']. Had the animat failed to recognize a given landmark on the E'K' trajectory, like landmark I' for instance, the supports from the matchings of I with I', of [HI] with [H'I'] and of (I, K) in [EK] with (I', K') in [E'K'] would have been lost. Nevertheless, the other supports would have been retained, thereby preventing the hypothesis that the animat actually moved from E' to K' from being abandoned. Thus the management of hierarchical structures affords the self-positioning procedure of MonaLysa with interesting robustness features.

5 Experimental results

To demonstrate the efficiency of the mapping and positioning procedures just described in a more realistic context, both procedures have been managed in parallel. The animat has been placed in an initial random position within the environment of Figure 7 and left free to explore it. It has not been provided with any overall goal, but its satisfaction was assumed to be greater when it was moving straight on, as opposed to turning right or left. Its orientation and position estimates were assumed to be correct and the distance traveled after each move was assumed to be equal to the animat's length. Furthermore, a specific procedure, not described herein, allowed it to learn to differentiate between obstacles around which it could skirt, and obstacles that merely delineated the environment's boundaries. Figure 7 also shows parts of the cognitive map thus elaborated after 50000 elementary moves. In particular, although this map doesn't represent any of the skirting lines and structures that have been detected in the environment, it shows the various landmarks and their topological links that the animat has memorized.

Figure 8 shows parts of the cognitive map elaborated when the animat's position estimates were assumed to be noisy and to increase with the distance traveled. Thus, after each elementary move beyond a given place, the error on the corresponding position estimate was sup-

posed to be a random normal variate, with a mean of 0, and a standard deviation proportional to the animat's size. After n such elementary moves, the corresponding standard error was that of a sum of normal variates, i.e., it was proportional to $\sqrt{n} * animat's\_length$. Because such errors were taken into account in both the mapping and positioning processes, the matching of two places was considered successful when the corresponding error margins intersected. Under such conditions, the animat wandered in its environment and, after each move, updated its cognitive map and its position estimate. Figure 8 shows parts of the cognitive map elaborated after 50000 elementary moves.

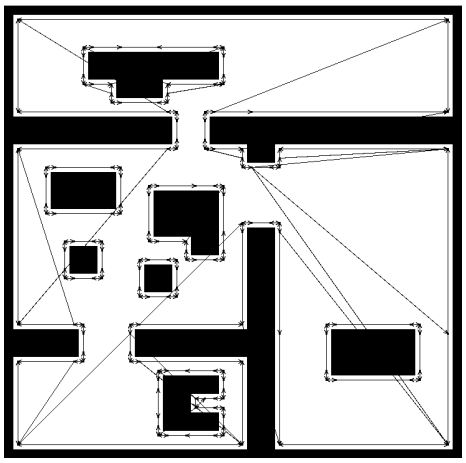

Figure 7: A square environment explored by an animat endowed with the MonaLysa control architecture. The size of the animat is 15 pixels, the size of each side of the environment is 735 pixels. Parts of the cognitive map elaborated by MonaLysa in the absence of noise are shown on the picture, as vectors coding topological links between pairs of landmarks

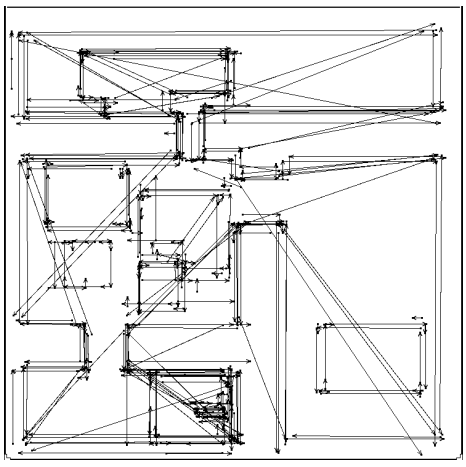

Figure 8: The cognitive map built when the standard deviation of the noise on the position estimate after each elementary move was assumed to be equal to 10% of the animat's length.

Figure 9 shows how the distance between the animat's current position estimate and its actual position varies over time in noisy conditions, when the mapping procedure is used and when it is not. It thus appears that its cognitive map allows the animat to control its positioning error. In particular, this error seldom exceeds twice the animat's length and, when such an event does occur, the error decreases to its basic level in a few moves.

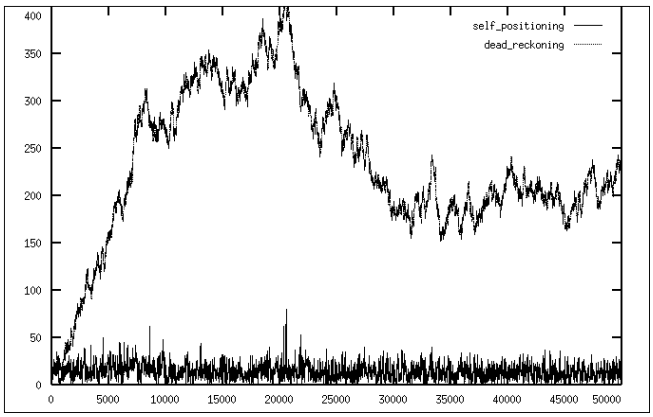

Figure 9: Evolution of the distance between the animat's current position estimate and its actual position, during the construction of the cognitive map of Figure 8. The upper curve correspond to results obtained without using the map, the lower one corresponds to results obtained when the map was used. Distances in pixels (ordinate) versus successive elementary moves (abscissa)

Additional experiments indicate that the control of the positioning error is robust at reasonable noise levels. However, and as might be expected a priori, map learning and self-positioning become impossible beyond a certain noise threshold, roughly equal to 20 % of the animat's length.

6 Discussion

It has been demonstrated here and elsewhere that the MonaLysa control architecture enables an animat equipped with a rudimentary sensory-motor apparatus to explore its environment, to extricate itself from dead-ends with arbitrary complicated shapes, to build a cognitive map of its environment, to accurately estimate its current position, to plan trajectories that avoid obstacles and that lead to a given externally-specified or autonomously-generated goal. Such capabilities qualify MonaLysa as a *cognitive* architecture and the animat as a *motivationally autonomous system* (McFarland and Bösser, 1993; Donnart and Meyer, 1996). They rely upon the use of production rules that take into account, not only the animat's current sensory information, but also their " internal context ", which codes the animat's additional knowledge about the current situation. In the

case of the reactive rules, this knowledge concerns the direction of the current goal. In the case of the planning rules, it concerns the nature of the current task the animat tries to accomplish. In the case of the mapping rules, it concerns the various tasks associated with markers and structures the animat has detected in its environment. Moreover, the fact that this internal context has a hierarchical structure confers several advantages on MonaLysa. Besides expediting the learning of the planning rules as described in (Donnart and Meyer, 1996), it has been shown here that it facilitates the self-positioning procedure and makes it relatively robust with respect to noise.

It has also been shown here that such robustness makes it possible for MonaLysa to detect, memorize and recognize landmarks in the environment, although the corresponding procedures do not rely on the information provided by the animat's proximate sensors. Actually this information is only used by the reactive rules of MonaLysa for the purpose of skirting around obstacles and of escaping from dead-ends, and place recognition depends only upon the proprioceptive position, orientation, and satisfaction estimates. Such a characteristic is in sharp contrast with many other realizations that implement place-recognition capabilities. For instance, several such realizations draw upon biology (Zipser, 1986; Cartwright and Collett, 1987; Muller et al., 1991; Burgess et al., 1994; Bachelder and Waxman, 1994) and implement a neural network in which the firing of some sensory neurons, tuned to the features of some landmarks sensed in a given place, triggers the firing of a specific *place cell* that codes for this place. Likewise, in (Kuipers and Byun, 1991), distinctive places are defined as the local maxima of specific functions that are defined over the various sensor readings of the animat, and, in (Mataric, 1992), landmarks are characterized as features in the world that have physical extensions reliably detectable over time. Few other realizations resort to proprioceptive information for place recognition. A specific example is provided in (Nehmzow and Smithers, 1991), where landmarks are characterized as convex or concave corners, such that, if the time a robot needs to turn towards a wall exceeds a certain threshold time, a convex corner is detected. Conversely, if the time it takes the robot to get away from a detected obstacle exceeds a certain threshold time, a concave corner is detected.

The route-following navigation strategy of MonaLysa could also be contrasted with other realizations which do not fully exploit the information encoded in their cognitive maps. In such realizations, indeed, a specific place is usually recognized because it has been reached by a specific move from another specific place, but no account is taken of the hierarchical structures that MonaLysa manages. As an example, in (Nehmzow and Smithers, 1991), a specific concave corner can be recognized because it has been reached from a convex corner a given distance away, but not because this specific concave corner belonged to a characteristic structure that linked, for instance, three succeeding convex corners to the concave corner in question. It is certainly because such realizations do not take advantage of the robustness that hierachical contexts afford to mapping and positioning procedures that their exploration capacities are much more limited than those of MonaLysa. The Nehmzow and Smithers's robot (Nehmzow and Smithers, 1991) cannot position itself in the environment if it doesn't keep following its boundary walls. Likewise, Mataric's robot (Mataric, 1992) and Kuipers and Byun's animat (Kuipers and Byun, 1991) rely heavily upon their wall-following and corridor-following exploration strategies to avoid getting lost. In contrast, the animat described in this paper is able to navigate randomly from one obstacle to another and to position itself. However, the present work and that of Mataric have in common the fact that place-recognition not only depends upon topological information, but also upon metric information about distances and orientations. Likewise, the present work and that of Kuipers and Byun have in common the fact that place-recognition depends upon topological links that are indexed by the animat's control strategy. With MonaLysa, an actual move between two places can be matched to a possible move on the map provided they occur within the same skirting-task context. In Kuipers and Byun's approach, two moves can be matched if they are actuated by the same *follow-the-midline* or *move-along-object-on-right* control strategies.

Although MonaLysa seems able to cope with less carefully designed environments than those that have been used in other realizations, it is nevertheless true that its current mapping and self-positioning capabilities would be lost if the environment contained non-polygonal obstacles. To allow the management of arbitrarily shaped obstacles, future research will be directed towards the use of more general skirting tasks than mere line crossing.

Future research will also be directed towards the possibility of recognizing whole obstacles and not only single landmarks or structures. This faculty will introduce an additional hierarchical level in the mapping and self-positioning procedures, and thus enhance the animat's cognitive capacities. In the present implementation, although the animat uses relative distances and orientations to self-position itself along the external contours of a given obstacle, it resorts to absolute distances and orientations to position itself when it first encounters a new obstacle. The possibility of characterizing whole objects in the environment will make it possible to evaluate all positions and orientations relatively to each object in the environment, and thus to get rid of any reference to the animat's initial position.

The planning capacities of MonaLysa have already

been demonstrated with a real robot (Donnart and Meyer, 1996). The present work has used simulations to demonstrate its mapping capacities. Future research will aim at combining the planning and mapping capacities of the architecture, first within a simulation framework, then with a robotic application. It will also aim at translating the actual rule-based implementation of MonaLysa into a biologically more realistic neural network architecture.

7 Conclusions

It has been shown here that the MonaLysa control architecture endows an animat with map-building and self-positioning capabilities that are robust with respect to noise. Such capabilities rely upon mapping and positioning procedures that take into account specific landmarks and structures in the environment. This approach is original and exhibits several advantages in comparison with other works that have similar objectives. In particular it can be used in environments that need not be as carefully designed as usual. In the future, it will be extended so that, beyond single landmarks and structures, it can characterize and recognize whole objects. Future work will also target the management of general skirting tasks that will hopefully cope with arbitrarily shaped obstacles. It will also be extended to the implementation of MonaLysa on a Khepera robot and to the demonstration that its mapping and planning capacities do actually work in the real world.

References

Bachelder, I.A. and Waxman, A.M. (1994) "A neural system for qualitative mapping and navigation in visual environments." *In Proceedings of the PerAc Conference: from Perception to Action. IEEE Soc. Press.*

Burgess, N., Recce, M. and O'Keefe, J. (1994) "A model of hippocampal function." *Neural Networks 7(6/7) :1065-1081.*

Cartwright, B.A. and Collett, T.S. (1987) "Landmark learning in bees. Experiments and models." *Journal of Comparative Physiology A 151 :521-543.*

Chatila, R. and Laumond, J. (1985) "Position referencing and consistent world modeling for mobile robots." *In IEEE Proceedings of Int. Conf. on Robotics and Automation. pp. 138-145.*

Donnart, J.Y. and Meyer, J.A. (1994) "A Hierarchical Classifier System Implementing a Motivationally Autonomous Animat." *In Proceedings of the 3rd Int. Conf. on Simulation of Adaptive Behavior. The MIT Press/Bradford Books, pp. 144-153.*

Donnart, J.Y. and Meyer, J.A. (1996) "Learning Reactive and Planning Rules in a Motivationally Autonomous Animat." *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, Vol. 26, No. 3.*

Gallistel, C.R. (1990) "The Organization of Learning." *The MIT Press.*

Kuipers, B.J. (1982) "The ' map in the head ' metaphor." *Environment and Behavior 14(2) :202-220.*

Kuipers, B.J. and Byun, Y.T. (1991) "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations." *Robotics and Autonomous Systems 8 :47-63.*

Latombe, J.C. (1991) "Robot motion planning." *Kluwer Academic Publishers.*

Mataric, M.J. (1992) "Integration of Representation Into Goal-Driven Behavior-Based Robots." *IEEE Transactions on Robotics and Automation 8(3) :304-312.*

McFarland, D. and Bösser, T. (1993) "Intelligent Behavior in Animals and Robots." *The MIT Press/Bradford Books.*

Muller, R.U., Kubie, J.L. and Saypoff, R. (1991) "The hippocampus as a cognitive graph." *Hippocampus 1(3) :243-246.*

Nehmzow, U. and Smithers, T (1991) "Mapbuilding using Self-Organising Networks in 'Really Useful Robots'." *In Proceedings of the 1st Int. Conf. on Simulation of Adaptive Behavior. The MIT Press/Bradford Books, pp. 152-159.*

Nehmzow, U. (1995) "Animal and robot navigation." *Robotics and Autonomous Systems 15 :71-81.*

O'Keefe, J. A. and Nadel, L. (1978) "The Hippocampus as a Cognitive Map." *Oxford University Press*

Penna, M.A. and Wu, J. (1993) "Models for map building and navigation." *IEEE Transactions on Systems, Man, and Cybernetics 23(5) :1276-1301.*

Poucet, B. (1993) "Spatial cognitive maps in animals : New hypotheses on their structure and neural mechanisms." *Psychological Review 100 :163-182.*

Riolo, R. L. (1991) "Lookahead planning and latent learning in a classifier system." *In Proceedings of the 1st Int. Conf. on Simulation of Adaptive Behavior. The MIT Press/Bradford Books, pp. 316-326.*

Roitblat, H. L. (1994) "Mechanism and process in animal behavior: Models of animals, animals as models." *In Proceedings of the 3rd Int. Conf. on Simulation of Adaptive Behavior. The MIT Press/Bradford Books, pp. 144-153.*

Schmajuk, N.A. and Thieme, A.D. (1992) "Purposive behavior and cognitive mapping : A neural network model." *Biological Cybernetics 67 :165-174.*

Scholkopf, B. and Mallot, H.A. (1995) "View-Based Cognitive Mapping and Path Planning." *Adaptive Behavior 3(3) :311-348.*

Thinus-Blanc, C. (1988) "Animal spatial cognition." *In Weiskrantz, L. (Ed.). Thought without language. Clarendon.*

Tolman, E.C. (1948) "Cognitive maps in rats and men." *The Psychological Review 55 :189-208.*

Zipser, D. (1986) "Biologically plausible models of place recognition and goal location." *In Rumelhart, D.E. and McClelland, J.L. (Eds.). Parallel Distributed Processing. The MIT Press/Bradford Books.*