REFERENCES

[1] C. J. Harris, X. Hong, and Q. Gan, *Adaptive Modeling, Estimation and Fusion From Data: A Neurofuzzy Approach*. New York: Springer-Verlag, 2002.

[2] M. Brown and C. J. Harris, *Neurofuzzy Adaptive Modeling and Control*. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[3] L. X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least squares learning," *IEEE Trans. Neural Networks*, vol. 3, pp. 807–814, Sept. 1992.

[4] S. Chen, S. A. Billings, and W. Luo, "Orthogonal least squares methods and their applications to nonlinear system identification," *Int. J. Control*, vol. 50, pp. 1873–1896, 1989.

[5] S. Chen, Y. Wu, and B. L. Luk, "Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 10, pp. 1239–1243, Sept. 1999.

[6] M. J. L. Orr, "Regularization in the selection of radial basis function centers," *Neural Comput.*, vol. 7, no. 3, pp. 954–975, 1993.

[7] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Automat. Contr.*, vol. AC-19, pp. 716–723, Dec. 1974.

[8] A. C. Atkinson and A. N. Donev, *Optimum Experimental Designs*. Oxford, U.K.: Clarendon, 1992.

[9] X. Hong and C. J. Harris, "Nonlinear model structure detection using optimum design and orthogonal least squares," *IEEE Trans. Neural Networks*, vol. 12, pp. 435–439, Mar. 2001.

[10] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function neural networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 302–309, Mar. 1991.

[11] X. Hong and C. J. Harris, "Neurofuzzy design and model construction of nonlinear dynamical processes from data," *IEE Proc D. Control Theory Applications*, vol. 148, no. 6, pp. 530–538, 2001.

[12] ——, "Nonlinear model structure design and construction using orthogonal least squares and D-optimality design," *IEEE Trans. Neural Networks*, vol. 13, pp. 1245–1250, Sept. 2002.

[13] M. Stone, "Cross validatory choice and assessment of statistical predictions," *J. R. Statist. Soc. B.*, vol. 36, pp. 117–147, 1974.

[14] L. Breiman, "Stacked regression," *Mach. Learn.*, vol. 5, pp. 49–64, 1996.

[15] R. H. Myers, *Classical and Modern Regression With Applications*, 2nd ed. Boston, MA: PWS-KENT, 1990.

[16] X. Hong, P. M. Sharkey, and K. Warwick, "Automatic nonlinear predictive model construction algorithm using forward regression and the PRESS statistic," , 2002, submitted for publication.

[17] S. Chen, "Locally regularization assisted orthogonal least squares regression," , 2001, submitted for publication.

[18] D. J. C. MacKay, "Bayesian interpolation," *Neural Comput.*, vol. 4, no. 3, pp. 415–447, 1992.

[19] S. Chen, X. Hong, and C. J. Harris, "Sparse kernel regression modeling using combined locally regularized orthogonal least squares and D-optimality experimental design," , 2002, submitted for publication.

[20] X. Hong, S. Chen, and P. M. Sharkey, "Automatic kernel regression modeling using combined PRESS statistic and regularized orthogonal least squares," , 2002, submitted for publication.

[21] T. Soderström and P. Stoica, *System Identification*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[22] D. W. Marquardt, "Generalized inverse, ridge regression, biased linear estimation and nonlinear estimation," *Technometrics*, vol. 12, no. 3, pp. 591–612, 1970.

[23] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamic systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–27, Mar. 1990.

[24] J. H. Nie and T. H. Lee, "Rule-based modeling: fast construction and optimal manipulation," *IEEE Trans. Syst., Man, Cybern. A*, vol. 26, pp. 728–738, Nov. 1996.

Dynamical Neural Networks for Planning and Low-Level Robot Control

Mathias Quoy, Sorin Moga, and Philippe Gaussier

***Abstract*—We use dynamical neural networks based on the neural field formalism for the control of a mobile robot. The robot navigates in an open environment and is able to plan a path for reaching a particular goal. We will describe how this dynamical approach may be used by a high level system (planning) for controlling a low level behavior (speed of the robot). We give also results about the control of the orientation of a camera and a robot body.**

***Index Terms*—Dynamical systems, neural networks, robot control and planning.**

I. INTRODUCTION

Our research team develops architectures for the control of mobile robots. These robots are able to navigate in an open environment and to find a path toward a particular goal. There are several different approaches for solving this problem ([1], and [2] and references therein]). The recent ones which relate more directly to our work rely on potential functions [3], [4] or neural network (NN) implementations of dynamical systems [5], [6]. See also the more classical NN approach to control given in [7]–[9]. The potential functions (or potential fields) approach is not a new one [10]. The main drawbacks are the existence of local minima and a difficult use in changing environments. It also relies on a Cartesian map of the environment. This map is not always available or accurate enough. Dynamical systems in a NN formalism try to avoid the shortcomings of this approach.

Following Polderman and Willems [11], we will define a *behavior* as the solution of a given dynamical system. From an ethological point of view, a behavior will correspond to a particular trajectory for going from one location to another, for instance. Note that the behavior being the solution of a dynamical system does not mean that the whole trajectory may be computed knowing the initial conditions. The solution may be defined implicitly. Thus the differential equation has to be solved numerically. This definition of a behavior is much used in an animat approach [12]. Indeed, one often defines the behavior as the *transient* dynamics leading from an initial condition to the attractor of the system. It is the case in the potential field case for instance. The nature of the attractor in this case is far less important than the transient dynamics. If the attractor is a fixed point, then the system stays on it and does not evolve anymore. We would like here to extend the notion of behavior to the dynamics *on the attractor*. In this case, a stable behavior can be reached when the system is on an attractor (among others: stable fixed point, limit cycle or even chaotic strange attractors [13], [14]). In this article, we will only consider the case of fixed points with asymptotic stability as defined in [15] for instance. In that case the robot keeps

Manuscript received March 8, 2002; revised November 20, 2002. This work was supported by two French GIS contracts on Cognitive Sciences "comparison of control architectures for the problem of action selection" in collaboration with Animat lab (J.Y. Donnart, A. Guillot, J.A. Meyer), LISC (G. Deffuant) and RFIA (F. Alexandre, H. Frezza), and "Mobile robots as simulation tools for the dynamics of neurobiological models: adaptation and learning for visual navigation tasks" in collaboration with CRNC (G. Schöner) and LAAS (R. Chatila). This paper was recommended by Associate Editor R. A. Hess.

M. Quoy and P. Gaussier are with the Neurocybernetics team, ETIS, Université de Cergy-Pontoise—ENSEA, BP 44, 95014, Cergy-Pontoise, France (e-mail: mathias.quoy@dept-info.u-cergy.fr).

S. Moga is with the Department IASC, ENST Bretagne, 29238 Brest, France.

Digital Object Identifier 10.1109/TSMCA.2003.809224

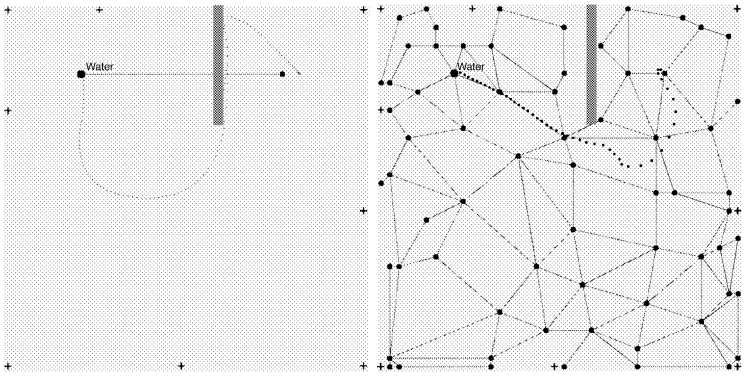

Fig. 1. (a) Direct attraction toward the goal (gradient maximization of learned goal) and reflex behavior of obstacle following for reaching the goal (initial speed toward the upper right. The robot has a mass giving an inertia). The dots indicate the trajectory followed by the robot. (b) "Cognitive map" built by exploration of the same environment. The landmarks are the crosses on the border. Each circle is a subgoal. The links indicate that the two subgoals have been activated in succession. The subgoals and the learned transitions form the goal or cognitive map. The dotted line indicates the robot trajectory.

doing the same actions. The change in behavior is obtained through bifurcations. A local bifurcation leads to a continuous change in the behavior, whereas a global one, as the new attractor may appear anywhere in the state space, leads to a completely different behavior. In this frame, an input is not a change in the initial conditions of the system (like in Hopfield nets for instance [16]), but rather a *change in the parameters of the dynamical system itself*. Hence, a continuous change of inputs provides a continuous change of the dynamical system, and therefore of its attractor(s) (provided there is no global bifurcation occurring). Once the attractor has been reached, the system remains on the attractor. If some perturbation moves it off the attractor, the dynamics forces it back toward it. Even more interesting is that we are able to know (depending on the distance to the attractor) what kind of "drive" moves the system back to the attractor. Hence, we do not only know the attractor, but also how to reach it (read-out mechanism).

A powerful theoretical framework is the neural field developed by Amari [17] and used for control purposes by Schöner (dynamical approach to autonomous robotics [18], [19]). The main interest of the neural field approach is that it is combining the dynamical systems formalism with a neuronal basis so that it may be very easily included in our control architectures using neural networks [20], [21].

We use neural networks for their learning abilities and modular implementation. But we are also involved in developing bio-inspired models which may be useful both for engineers and neurobiologists. Indeed, our system (from which only one part is presented in this article) is a complete architecture addressing the problems of navigation and planning, but also object recognition [22] and low-level imitation [23], [24]. It is our experience that we cannot solve these tasks separately, and then put all parts together. We have to think of the system globally and benefit from the interactions within the system and with the environment. However, the results presented in this article do not depend on the complete implementation of our control architecture. We rather focus on the general dynamical aspects and properties.

In this correspondence, we first present the planning architecture used by a robot for finding a path toward a chosen goal. This architecture is based on a planning map built on-line by the exploration of the environment. Then, we present the notion of activation field. Because the cognitive map may be seen as an activation field, it may be used in return for the speed control of the robot (top-down control). This work was only performed in simulations. Under internal or external constraints, this attractor may evolve in time. In order to address this case, we introduce the neural field formalism and explain the related equation, and apply it to control the orientation of the camera and the body of a real robot. Finally, we draw some conclusions.

II. Planning Map as an Activation Field for Speed Control

A. Planning Map

The setting for the simulations and the robotic experiments are as follows. The robot is put into an unknown environment and has to explore it in order to find "interesting" places (called goals). The places may depend on the task to achieve (going to a reloading station for instance). Once discovered, these places are linked with a particular "motivation". If the motivation goes below a given threshold (for instance the robot needs to resupply), then the robot has to go to the nearest location linked with this motivation. In order to solve this problem, the robot has to 1) localize itself [21], [25] and to 2) plan a path to go back to a particular goal when needed [26] and [27].

1) Location is defined by the recognition of a set of distant landmarks associated with their azimuth. The azimuth is given relative to a compass giving the magnetic north. Each couple (landmark, azimuth) is learned on a neuron map. The set is then learned by a single neuron. The closer the robot to the neuron coding for a position, the higher the response of that neuron. As the robot explores the environment, it learns new locations (subgoals) if the activity of all neurons is below a defined recognition threshold. The higher this threshold is, the more learned places are there (denser coverage of the open space).
2) In order to be able to go back to a learned location, the robot has to keep track of where it was and where it may go from a location. It builds a map of the environment in which the relationships between successively explored places are learned. The temporal proximity being equivalent to a spatial proximity, the system creates a topological representation of its environment (Fig. 1).

We call this last group our "cognitive map" (or planning map) (Fig. 2) [28]–[30]. When an interesting location is found, a motivation is linked with the corresponding neuron on the planning map. The principle of this kind of cognitive maps is not new ("world graph" [31]–[37]). Yet, our algorithm allows to solve action selection problems in a dynamic environment (the sources may disappear and appear again elsewhere) [38]. The map is learned and modified online and allows to manage contradictory goals. We don't provide any external explicit description of the environment (world model), nor learn what to do for each location in the environment [39].

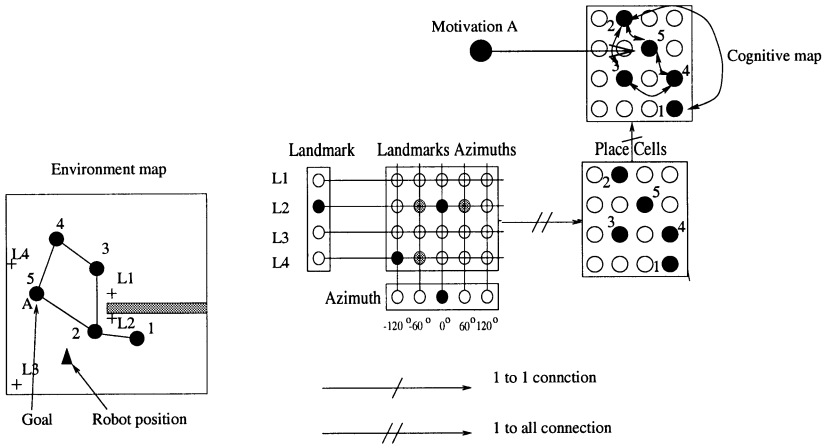

Fig. 2. Global architecture of the planning system. On the left a simplified environment with four landmarks and five learned locations. Location 5 is a goal linked with the satisfaction of motivation A. Each of the five place cells learns a pattern of landmark-azimuth combinations. This corresponds to the recognition level. The planning level stores the "cognitive map." Learning reinforces the connections between place cells coding for neighbor locations in the environment. Note that there is no Cartesian topology on this map (which would directly correspond to the environment), but only the record of the possible transitions between place cells.

When the robot wants to satisfy a particular motivation, the corresponding neuron is activated. This induces the activation of a goal in the planning map which is linked with the satisfaction of this motivation. Then, this activity diffuses on the map beginning from the goals, and activates place cells according to their distance (in number of links) to the goal. First, the robot tries to follow the gradient of neuron activity in this cognitive map to select the next location to reach. The most activated goal or subgoal in the neighborhood of the current robot location is then selected and used to attract the robot in its vicinity. When the robot is close enough to this location, the associated subgoal is inhibited and the robot is attracted by the next subgoal and so on until it reaches the final goal.

Learning the planning map is performed continuously. There is no separation between the learning and test phases. The links between neurons of the graph are reinforced (hebbian associative learning) for neurons associated with successively recognized places. Let W_{ij} be the weight associated with the fact that from the place i it is possible to reach directly place j, its learning rule is the following:

$$\frac{dW_{i,j}}{dt} = -\lambda.W_{i,j} + \left(C + \frac{dR}{dt}\right).(1 - W_{i,j}).\overline{G(i)}.G_{(j)} \qquad (1)$$

where $G(i)$ is the value of neuron i (see [38] for more details on the navigation algorithm). $G(i)$ must be held to a non null value until $G(j)$ (with $i \neq j$) is activated by the recognition of the place cell j. This is performed by a time integration of the $G(i)$ values represented in the equation by $\overline{G(i)}$. $\overline{G(i)}$ decreases in time and can be used as a raw measure of the distance between i and j. λ is a very low positive value. It allows forgetting unused links. $C = 1$ in our simulations. The term dR/dt corresponds to the variation of an external reinforcement signal R (negative or positive) that appears when the robot enters or leaves a "difficult" or "dangerous" area. At the planning level (Figs. 2 and 3), the motivations activate directly the neurons linked with the goal to be achieved (the goal neuron intensity is proportional to the motivation). This activity is then diffused from neuron to neuron through the links.

The goal diffusion algorithm is the following [40]

- i_0 *it is the goal neuron activated because of a particular motivation.* $G(i_0)$ *is its activity.*
- $G(i_0) \leftarrow 1$ and, $G(i) \leftarrow 0 \forall i \neq i_0$
- $WHILE$ *the network activity is not stabilized,* $DO : \forall j, G(j) \leftarrow \max_i (W_{ij}.G(i))$. (2)

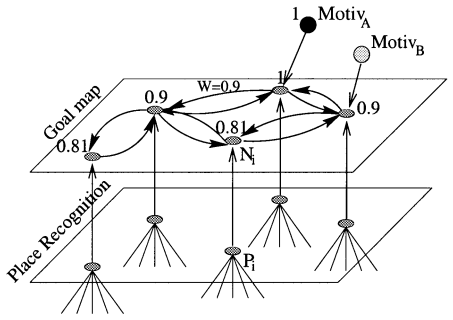

Fig. 3. Motivation diffusion on the planning map. The recognition level allows to identify the situations when the robot arrives in the vicinity of a stored place. These situations are directly linked with the goal level which allows to plan a route from one attractor to the next until the objective. When motivation "A" activates the goal, this activation is propagated through all nodes in the graph (here all weights are equal to 0.9).

W_{ij} has to be strictly inferior to 1 ($0 < W_{ij} < 1$) to ensure the algorithm convergence. On Fig. 3 an example of activity diffusion from motivation "A" is presented. The algorithm allows to find the shortest path in the graph. That path can be found by a gradient "following" from the node associated with the current location to the goal node. The algorithm is proved to always find the shortest path in the graph (it is equivalent to the Bellman-Ford algorithm of graph exploration [40]–[42]).

The planning map is constructed in order to indicate which path should be taken in order to reach the goal. In addition, we will see in next subsection that the map may give us another information: how to control the speed of the robot in order to assure to go to the goal and stop once arrived. Hence, the planning map not only indicates where the goal is, but also gives the possibility to control how to reach it.

B. Activation Field

There are two different dynamics to consider. The first one enables to reach a stable fixed point (activation field presented in this section). The second dynamics (neural field equations in Section III) changes continuously the location of the fixed points depending on the interaction with the neighboring neurons and the input to the system. In this section, we will only deal with the first dynamics.

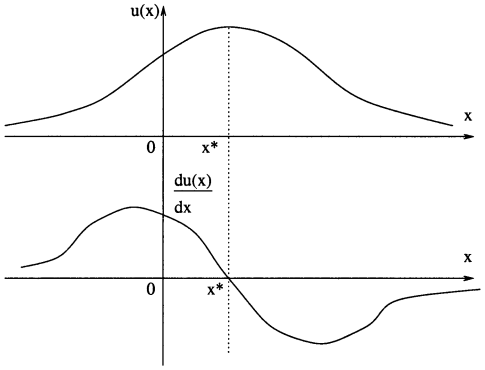

Fig. 4. u versus x (top) and gradient of u versus x (bottom). We have considered here that:$u(x) = e^{-((x-x^*)^2/2\sigma^2)}$.$x^*$ is the stable fixed point of $\dot{x} = \partial u(x(t))/\partial x$.

Let us consider a real variable x. At each point x, we consider a monomodal activation $u(x)$ which is maximal for $x = x^*$. u is the so called activation field. u has to be differentiable at x^*.

As stated before, we are interested in stable behaviors. From the dynamical systems point of view, the stable state is a stable attractor. There may be various kinds of attractors including limit cycles and chaotic attractors [13], [14]. However, we will here only deal with stable fixed points. So our purpose is now to construct a dynamical system where x^* is a stable fixed point (Fig. 4).

x^* is the stable fixed point of the following dynamical system:

$$\dot{x} = \frac{dx(t)}{dt} = \frac{\partial u\,(x(t))}{\partial x}. \tag{3}$$

Hence the value of $(\partial u(x(t)))/(\partial x)$ gives the variation on x for reaching the stable fixed point x^*. Thus, a first-order approximation gives:

$$x(t+\delta t) = x(t) + \delta t.\frac{\partial u\,(x(t))}{\partial x}. \tag{4}$$

Given an initial condition $x(0)$, iteration of equation (4) leads to the stable fixed point x^*. This equation gives how to reach the stable fixed point by relating it to the derivative of the activation. This corresponds to the *"read-out" mechanism*.

The crucial property on u is being continuous and derivable at the maximum location unless there may not be any stable fixed point at all (see for example $G(n)$ on Fig. 6 in the next section). The case where u is multimodal fits in the frame of multiple activation fields (see Section III)

C. Speed Control

The following results were only performed in simulations because for the moment the robot has to stop between two moves in order to take a panoramic image of its environment. This drawback will disappear by implementing an active scanning of the scene.

For speed control, the robot is considered to have a mass M and runs on a floor having a friction F. An internal drive is also present. When the robot is exploring, this drive is random in strength and direction. This enables to randomly explore large areas. When the robot is going toward the nearest goal (or subgoal), the direction of the drive is given by the direction of the goal, and the strength may be either independent from the distance to the final goal (hence constant), or depend on this distance (hence dependent on the gradient). The equations driving the robot movement are now (we took the same form as in [43]).

$$\begin{aligned} v_x(t+dt) &= v_x(t) - \frac{dt.F.v_x(t)^2}{M} + \frac{dt.drive}{M} \\ x(t+dt) &= x(t) + dt v_x(t) \end{aligned} \tag{5}$$

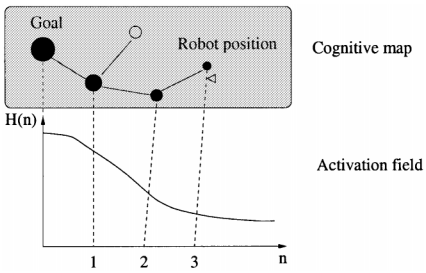

Fig. 5. Planning map and corresponding H versus n. The activated path in the planning map is depicted by black dots. The activation of each node is a function of its distance to the goal. This distance is expressed in a number of subgoals and does not directly reflect the real metric distance.

where the x subscript denotes a projection on the x axis. The same equations hold for the y axis. v is the speed and (x, y) the position of the robot. Note that the robot does not use the (x, y) values in order to "know" where it is in the environment. Its recognition of places follows the place cell recognition (G_i) as described in [38]. There are no nonholonomic constraints such as having a rolling tire without slipping. We only set a threshold speed. Our model allows the robot to glide illustrating the effect of the external drive in this equation for the parameters $F = 5$ and $M = 5$. These parameters are not realistic, but serve to illustrate the control properties of our system. $drive$ is a force (either random or toward the goal). dt is a small integration constant.

The value of each node i in the map is $G(i)$. Let $H(i)$ be the following value:

$$H(i) = \exp - \frac{\left(\frac{\ln G(i)}{\ln(W)}\right)^2}{2.\sigma^2} \tag{6}$$

where σ is fixed to 5. We consider here that all weights W_{ij} are equal (to W). All results shown below may be extended (and were used in simulations) with variable weights [as computed by equation (1)].

In order to stay at the same position (when a goal is reached for instance), the speed must be 0. Hence $\mathrm{drive} = 0$ at the goal location. When the drive is constant, the speed will tend to $\sqrt{\mathrm{drive/F}}$ and the robot cannot stop on the goal. If the drive depends on the distance to the goal, one may choose directly the drive as a function of G or H to make it tend to 0 as the goal becomes closer.

We can extend the notion of distance expressed in number of subgoals by making it continuous. The goal has now an activation field extending over each subgoal (Fig. 5). Practically, in the case when the drive is not constant, the difference between the value of the nearest subgoal and the value of the second nearest subgoal determines the strength of the drive. This value is computed during the diffusion of the motivation from the goal to the subgoals. Using the goal diffusion algorithm (2) presented in the previous section, this value is $G(n) = W^n G(0)$, where W is the weight between the subgoals (supposed to be constant to simplify the explanation), n the distance to the goal[1] and $G(0) = 1$ (for a more general case see Appendix). The discrete value of the distance corresponds to the number of subgoals. The dotted line in Fig. 6 shows the strength of the subgoal $G(n)$ versus the distance to the goal.

The properties of G do not make it a suitable activation field because there is no fixed point on the goal (the derivative of G at the goal location is non 0 (Fig. 6). However, H has the suitable properties. Let the value of the field at subgoal n be $H(n)$. Considering H as an activation field and applying the read-out mechanism defined by equation (4), it is possible to control the position of the robot through modifications of its

[1]Note that we use now n instead of x as defined in the previous section, because x refers now as usual to the projection on the x-axis.

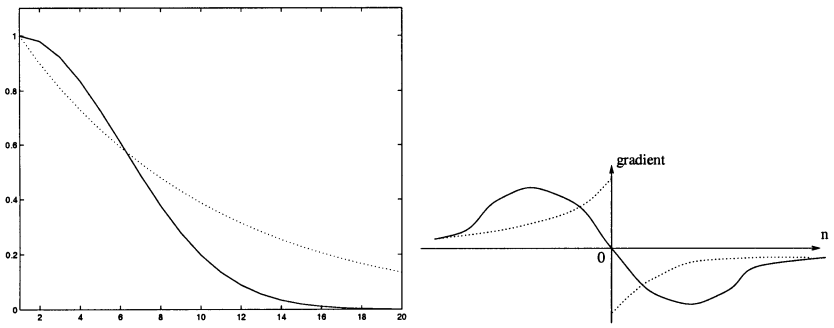

Fig. 6. Left: H (solid line) and G (dotted line) versus n. In the first case, the gradient decreases as the goal is being reached. Whereas in the second case, the gradient increases, leading to a high speed near the goal. Right: Gradient of H (solid line) versus n and gradient of G (dotted line) versus n. In the first case, $n = 0$ is a stable fixed point for the dynamics $\dot{n} = F(n)$. In the second case, this not the case.

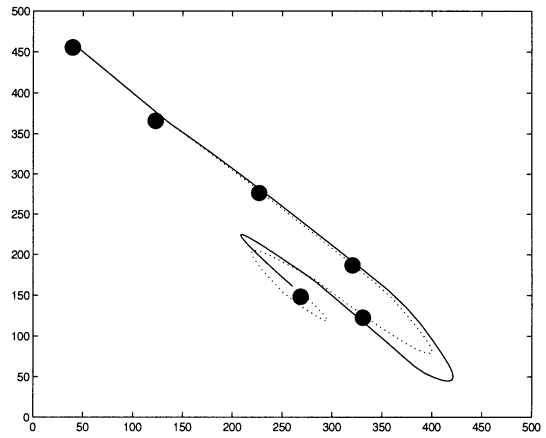

Fig. 7. Trajectory of an robot driven by the gradient computed using $H(n)$ (solid line) and $G(n)$ (dotted line). The dots are the learned subgoals. The planning map links are not shown.

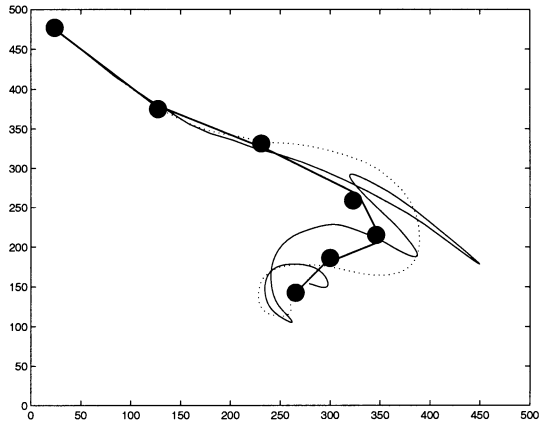

Fig. 8. Trajectory of a robot driven by the gradient computed using $H(n)$ (solid line) and $G(n)$ (dotted line). The robot reaches the goal sooner in the second case, but with a high speed.

speed introduced by the gradient of $H(n)$. In this case the goal position is a stable fixed point of the dynamics $\dot{n} = (\partial H(n))/(\partial n)$ (Fig. 6). See in Appendix for more proofs on the stability of this equation when all W's are equal (case 1), and different (case 2). When $n = 0$, the robot is at the target (and its speed is 0 either). Moreover, as it is a stable fixed point, we are sure that the robot will converge on it. This is not the case when using a gradient computed with $G(n)$. The robot may not converge to the goal at all because the goal is not a fixed point anymore (Fig. 6). As in [44], we have a Gaussian shaped field which is maximal at the target location (here the goal). Adding the dynamics enables to use the gradient of this function for control purposes (here the speed), and makes the target a stable fixed point.

When using G for computing the gradient (dotted line in Fig. 7), the gradient increases as the robot comes closer to the goal. Hence as the robot approaches the goal, the drive toward it increases. This leads to a non null speed when reaching on the goal and unstable reaching behavior. In order to avoid this unstable behavior (or to stop at the target), the speed must decrease as the robot comes closer to the goal by using $H(n)$ for computing the drive. The speed near the goal is now reduced, enabling a smooth approach. This is particularly suitable if the robot has to stop at the target.

Comparing two drive concepts, it appears that use of G enables reaching the target earlier, with a high speed when arriving at it. It is reached faster because the attraction strength of the goal is high, so that the robot does not "glide" as much as in the second case. However, there may be also cases where the goal is not reached at all because the goal is not a fixed point of the dynamics. In the second case, the robot may take a longer time to reach the goal, but arrives there with a very low speed. However, due to the smaller strength of the attraction it may sometimes "circle around" the goal until it is reached (Fig. 8).

Concerning the interpretation of the cognitive map in terms of activation field, it is important to stress that *this process is a top down one*. Indeed, the map has first been built using the place recognition mechanism. Now, in the return phase the use of the map gives information for the low level control of the robot.

Up to now, we consider in the case where there is only one dynamics t. Therefore, we have neglected the fact that the values $G(n)$ [and hence $H(n)$] may evolve in time (in Appendix see case 3). This means, for instance, that the motivation for reaching the goal increases in time. As a consequence, it is not possible to "freeze" the motivation value until the goal is reached (dynamics t). Therefore, the functions G and H also depend on another time scale t', corresponding to the time scale of the evolution of the motivations. If $H(n)$ remains constant during the diffusion algorithm, then the goal is still a fixed point. Practically, $H(n)$ may be modified after each diffusion process. The same kind of remark holds if one considers now that the weights are modified according to the (1) (see case 4 in Appendix). This means that there is a time scale t'' on which these weights are updated. If the weights remain constant during the diffusion process, then the goal is always a fixed point. Again, here, the weights may be updated after each diffusion.

The algorithm may now be defined as follows:

1) initialization of $H(0)$ depending on the motivation level;
2) diffusion of $H(0)$ on the topological map using (2);
3) computation of the drive $(\partial H(n))/(\partial n)$ as $H(n) - H(n-1)$, (4);
4) movement given by (5);

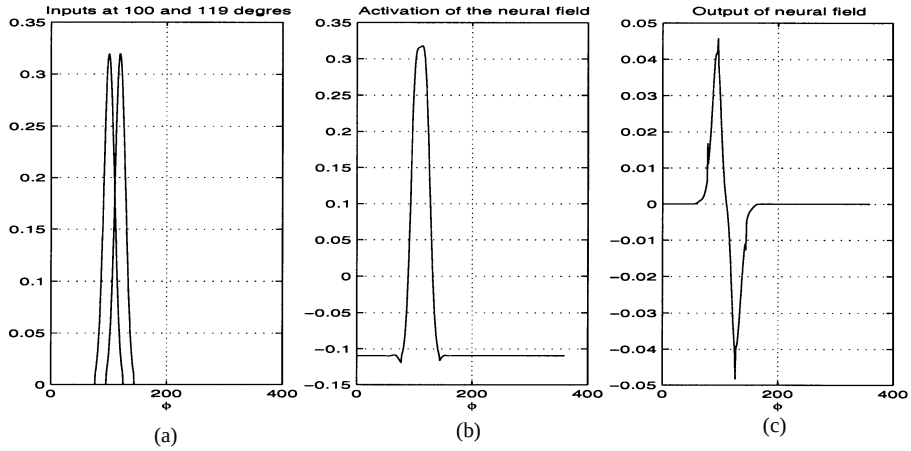

Fig. 9. Cooperation feature of the neural field for 2 inputs. (a) Simulated inputs a 100 and 119°[values of x, or here ϕ (see below)]. (b) Activation of the neural field has a single maximum. (c) Dynamics of the neural field has a single fixed point.

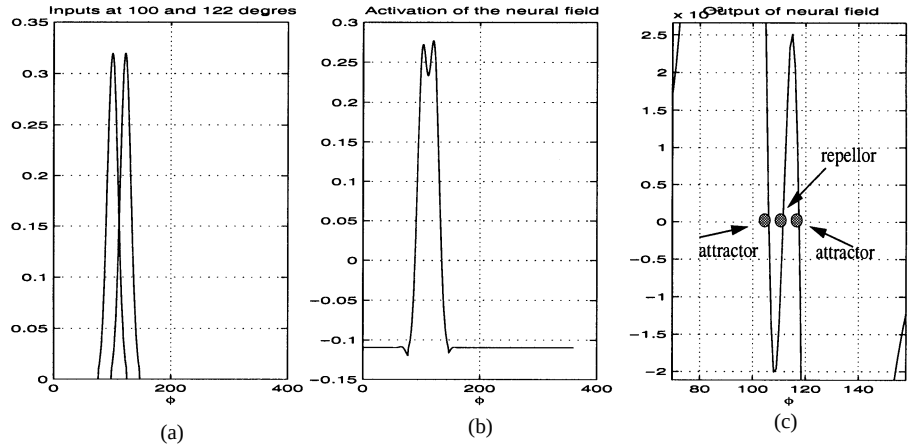

Fig. 10. Competition features of the neural field for two inputs. (a) Simulated inputs at 100 and 122°. (b) Activation of the neural field has two maxima and one minimum. (c) Detail (zoomed) of the output of the derivative of the neural field showing two stable fixed points and one unstable fixed point.

5) modification of the weights (1);
6) go back to first point until the goal is reached.

III. Neural Field for the Control of Camera and Body Orientations

A. Neural Field

Let us now consider the possibility not to have a unique activation on each x. We then have to introduce a way to combine these different activations. Adding theses activations together deeply affects the activation shape and consequently dynamics given by (3). Indeed, the previous maximum may not exist anymore. Therefore, the unique fixed point previously existing may disappear. There may be now one or several fixed points (stable separated by unstable ones).

Until now, we have only considered a fixed activation field. When the activations evolve in time, we define the activation at each x as $u(x,T)$, with T *a priori* not the same time scale as t in equation (3). Since each activation field u changes over time T, the positions of the fixed points also changes. Moreover, the combination of the activations may lead to existence of bifurcations. A fixed point may appear or disappear. This property is crucial since the occurence of bifurcation forces the system to "choose" between one or the other fixed point. And afterwards, once it is in the attraction basin of one point, it remains within it, until it disappears (via another bifurcation).

The neural field equations proposed by Amari [17] have the suitable properties in order to address our problems (combining activation fields and having fixed points evolving in time).

$$\tau \cdot \frac{u(x,T)}{dT} = -u(x,T) + I(x,T) + h + \int_{z \in V_x} w(z) \cdot g\left(u(x-z,T)\right) dz. \tag{7}$$

Without inputs, the homogeneous pattern of the neural field, $u(x,T) = h$, is stable. The inputs to the system, $I(x,T)$, represent the stimuli exciting the different regions of the neural field. τ is the relaxation rate of the system. $w(z)$ is the interaction kernel in the neural field activation. These lateral interactions ("excitatory" and "inhibitory") are modeled by a DOG function. V_x is the lateral interaction interval. $g(u(x,T))$ is the activity of neuron x according to its potential $u(x,T)$. We use a classical ramp function. Taken separately, each input erects an attractor (i.e. a fixed point with asymptotic stability) in the neural field. The neural field equation allows for the cooperation between coherent inputs associated with different goals (spatially separated targets). For closely spaced input information, the dynamics has a single attractor corresponding to the average of the input information (Fig. 9).

For a critical distance between inputs, a bifurcation point appears and the previous attractor becomes a repellor (i.e. an unstable fixed point), and two new attractors emerge (Fig. 10). Depending on the initial state, the robot switches to one of two new fixed points. This mechanism of

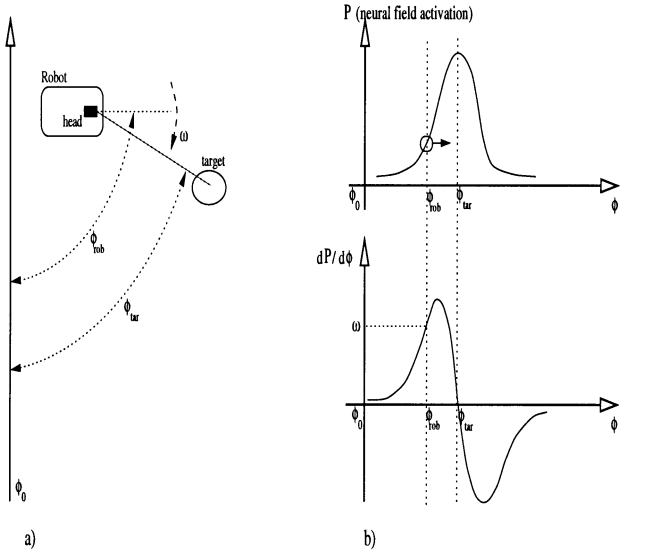

Fig. 11. (a) Robot and the target coordinates are represented in the same reference frame. The reference orientation, ϕ_0, is used to compute $\phi_{\rm rob}$ and $\phi_{\rm tar}$. ϕ_0 is given by a compass for instance. (b) Target position erects an attractor at ϕ_{tar}. As for the speed control in previous section, the read-out mechanism allows to compute the rotation speed ω using the derivative of the neural field activation.

input competition/cooperation has an hysteresis property which avoids oscillations between the two possible behaviors.

Another feature of the neural field is memory. If the parameter h in (7) has a sufficiently negative value, then the neural field operates with a memory effect in which a peak of an attractor is being maintained for a short time interval.

A large positive value of h induces a saturation of all neurons in the neural field. Finally, there may be a "traveling" wave phenomenon, which is not interesting for our purposes.

A dynamical system with two time scales t and T given by equations (3) and (7) has been defined. The coupling between the dynamics is given by the fact that at each time T, the dynamics t enables to reach the new fixed point. In other words this means that the *internal* dynamics t has to be faster than the *external* dynamics T. The dynamics T is external because it changes with the inputs to the system. In contrary, the dynamics t only works on internal variables.

B. Control of the Camera and Body Orientations

It has been proposed to use the properties of the neural field for motor control problems [45]. A major innovation was the "read-out" mechanism proposed by Schöner [18]. As explained before, this mechanism consists of the use of the derivative of the activation field u to compute the motor command.

Following Schöner's works [18] and [19], the orientation of the robot camera, $\phi_{\rm rob}$, relative to a fixed reference is used in our system as a behavioral variable. The state of the system is expressed as a value of this variable. We will use here the notation ϕ instead of x to keep an "angular" notation. If the target orientation is $\phi_{\rm tar}$

1) it erects an attractor in the neural field (Fig. 11) at ϕ_{tar}
2) the robot's camera rotation speed will be $\omega = (\partial u(\phi))/(\partial\phi) = (d\phi)/(dt)$. It sets the equation dynamics of the robot's camera.

This last equations uses the same read-out mechanism as in the previous section, but applied now to the rotation speed of the camera.

We first use this system for tracking a moving object: a Koala robot moves with a constant speed in the visual field of the CCD-camera of a looking robot. The position of the object in the visual field is extracted by computing the optical flow [46] giving $\phi_{\rm tar}$. The results obtained in real time runs are shown in Fig. 12. The object moves for a few iterations, then it stops. The generated activation field is still present after the removal of the input. This illustrates the memory property of the neural field.

Now, if the input moves in the visual field, the camera is able to track it. For instance, let's call "A,""B," and "C," the three successive positions of the target. First, state A is activated. The input in the neural field generates an attractor for orientation ϕ_A (Fig. 13). At time τ, the ϕ_B neuron will be activated (direction of movement B). This activation shifts the attractor to ϕ_B in the neural field. The same process holds for ϕ_C.

Until now, only the camera has been rotated for movement tracking. It is possible to use exactly the same system for rotating the robot body, so that it could face the target. The angle provided by the read-out mechanism is directly used for rotating the camera and turning the robot body. Because of a difference in the inertia of the camera and the robot body, the robot body turns slower than the camera. The camera turns first rapidly to the new direction, then the robot body turns. The top plot in Fig. 13 shows the variation of head and body orientation as a function of time. According to the neural field dynamics, orientation change is continuous. For an external observer, the head orientation anticipates the body orientation.

IV. Discussion

The main limitation of the present implementation of the planning algorithm is the diffusion of an analogical value with a geometrical decreasing law. Indeed, in very big environments (with a great number of subgoals) the gradient may be very small. Hence, first, the drive on the speed would be very small too (leading to a long time before reaching a goal). Second, if there is a small noise on the gradient, it would be now impossible to follow it. Therefore, the maps of different joint environments have to be combined. This means we have to define a planning structure, or plans of plans, in order to address large scale environments. This need is also highlighted by the use of the same neural structure for storing all maps linked with different environments. There will soon be an explosion of the number of neurons needed to code all locations, and moreover a mix up between maps coding for totally different environments. Hence, the map has to be "transfered" in an other structure where it may be memorized.

The experiment on the camera and body control runs on-line without any problem caused by the framework we use. The only problems are linked with the recognition of a moving object or person if it passes too fast in the visual field of the camera. However, once the system is locked on a particular moving object, it stays on it, and follows it. The accuracy depends on the number of neurons used for coding the neural field. We use a 120 neurons array in the case of an angular coding (precision of 3° for a neuron).

One advantage of using dynamical systems is that a bifurcation leads to choose between different attractors. Once an attractor is chosen the system stays locked on it until another bifurcation occurs. So the action selection property is linked with the stabilization of the chosen behavior. Moreover, the read-out mechanism provides a natural feedback command either on the same system (camera and body control), or on a different system (planning controlling speed).

We have shown that dynamical system theory is useful in our robot control framework, and may be easily integrated in a neural network formalism. In this article we have only considered fixed point attractors. Using recurrent neural networks, it is possible to have more complex dynamics such as limit cycles or even chaos [13] and [14]. The same principles hold in this case, in particular the dynamics of the system is always on an attractor. The challenge now is to deal with such complex

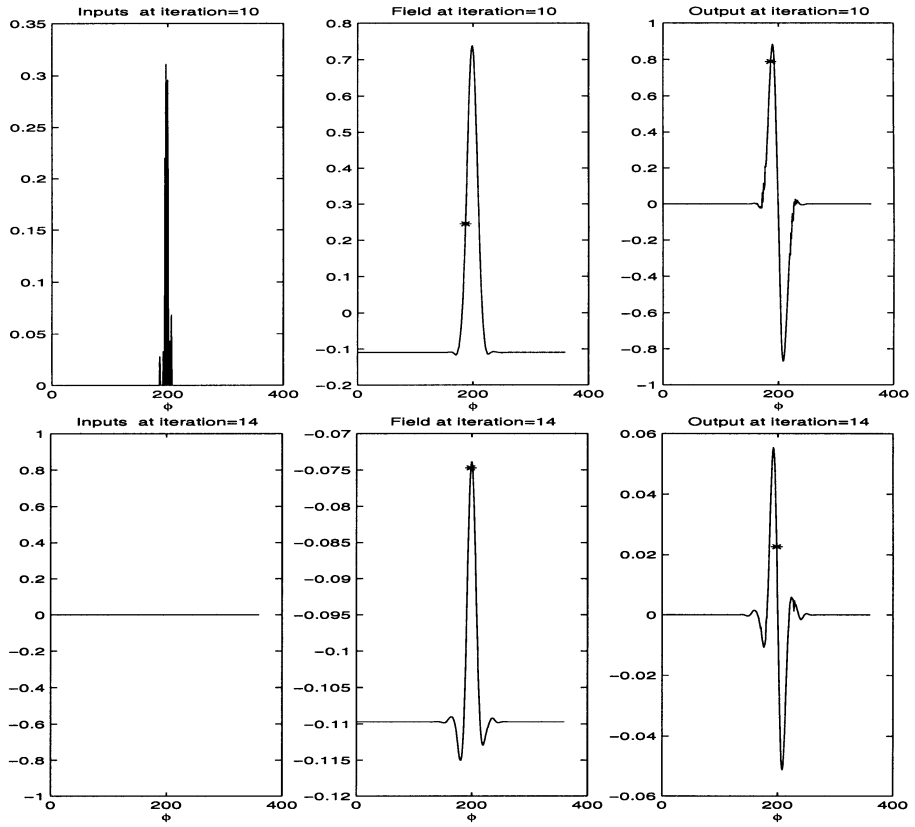

Fig. 12. Memory effect. Top: state of the system at the 10th iteration. Bottom: state of the system at the 14th iteration. From the 11th to the 14th iteration, there is not input. Left: the input to the system. Center: activation field. Right: output of the neural field. "$*$" shows the CCD-camera orientation.

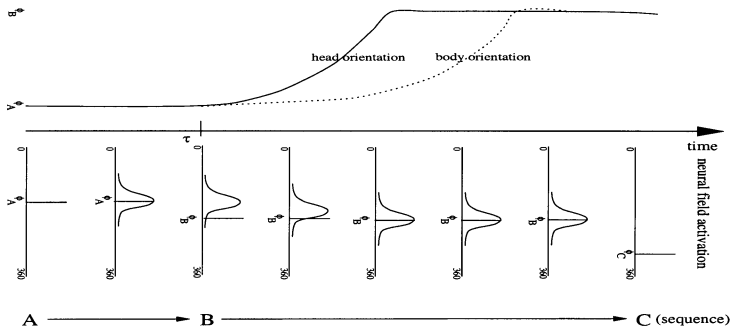

Fig. 13. Top: temporal variation of head and body orientations. Bottom: the neural field activation for an "ABC" moving sequence.

dynamics. Preliminary works on robot control using a chaotic recurrent neural network lead to very encouraging results [47].

APPENDIX

In this appendix, we will give the proofs that the goal ($n = 0$) is the fixed point of equation $\dot{n} = (dn)/(dt) = (\partial H(n))/(\partial n)$. By definition

$$H(n) = \exp - \frac{\left(\frac{\ln G(n)}{\ln(W)}\right)^2}{2.\sigma^2}. \tag{8}$$

We will calculate the derivative $(\partial H(n))/(\partial n)$ and study its sign for the following cases:

1) W is constant;
2) weights have different values W_i;
3) H depends on the motivation on a time scale t';
4) weights are modified on a time scale t'' by (1).

Preliminary 1: The properties of G are as follows:

Property 1:

$$\forall n \; 0 < G(n) \leq 1 \tag{9}$$

Property 2:

$$\forall n \frac{\partial G(n)}{\partial n} \leq 0. \tag{10}$$

Preliminary 2: In the general case, the calculation of the derivative of H with respect to n leads to

$$\frac{\partial H(n)}{\partial n} = \frac{-1}{\sigma^2 G(n) ln^2 W} \cdot \frac{\partial G(n)}{\partial n} ln\,(G(n)).H(n). \tag{11}$$

If n^* is the fixed point, two points to be investigated in all cases are:

1) $\forall n \geq n^* \; (\partial H(n)/\partial n) \leq 0$;
2) $(\partial H(n)/\partial n)|_{n^*} = 0$.

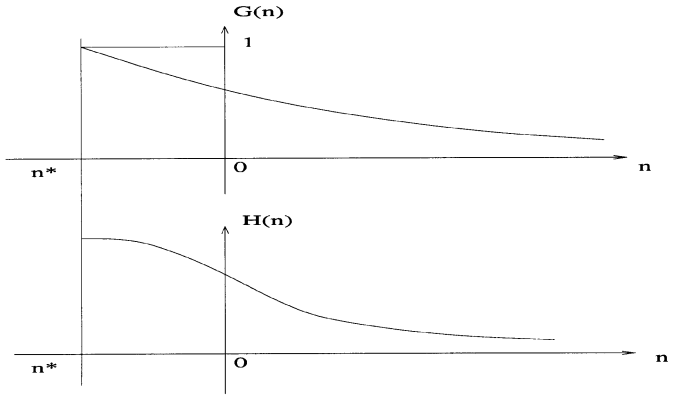

Fig. 14. G and H versus n in the general case. If $G(0) = 1$ then the value of the fixed point n^* is 0. Otherwise, the value of n^* is negative.

First Case: W is Constant: Result 1: If $0 < W < 1$ there exists a fixed point n^*

Proof: In this case, the diffusion algorithm (2) leads to $G(n) = W^n G(0)$. Replacing G in equation (11), one obtains

$$\frac{\partial H(n)}{\partial n} = -\frac{ln\,(G(0)) + nln(W)}{\sigma^2 ln(W)}\, H(n). \tag{12}$$

The derivative $(\partial H(n))/(\partial n)$ is 0 when $ln(G(0)) + nln(W) = 0$. Thus, $n = -(ln(G(0)))/(ln(W))$. Let this value be n^*. Note, that this value is negative because $G(0)$ and W are both between 0 and 1. For instance, if $G(0) = 1$ then $n^* = 0$. And if $G(0) = W$, then $n^* = -1$ (see Fig. 14).

The sign of the derivative is given by the opposite sign of $(ln(G(0)))/(ln(W)) + n$. If n is positive, the derivative is negative. If n is negative, we have $n \geq n^*$, thus $n \geq -(ln(G(0)))/(ln(W))$ and the derivative is negative.

It follows immediately that

1) $\forall n \geq n^*\ (\partial H(n)/\partial n) \leq 0$;
2) $(\partial H(n)/\partial n)|_{n^*} = 0$..

Thus, n^* is a fixed point. ■

Second Case: The Weights Have Different Values W_i: Result 2: Let N be the number of weights. If $\forall i = 0..N$, $0 < W_i < 1$ then there exists a fixed point n^*

Proof: In this case, the discrete diffusion algorithm (2) leads to

$$G(n) = G(0) \prod_{i=0}^{n} W_i$$

Let W_{min}(resp.W_{max}) be the smallest (resp. biggest) of the weights. One may obtain

$$G(0)(W_{min})^n \leq G(n) \leq G(0)(W_{max})^n.$$

It follows that

$$ln\,(G(0)) + nln(W_{min}) \leq lnG(n) \leq ln\,(G(0)) + n\ln(W_{max})$$

W_{min} and W_{max} are positive but less than 1, so that $ln(W_{min})$ and $ln(W_{max})$ are negative. Hence, when n tends to $-\infty$, the first and third terms tend to $+\infty$. Therefore, $lnG(n)$ tends to $+\infty$ when n tends to $-\infty$. Moreover, as $G(0) \leq 1$, $ln(G(0)) \leq 0$. G is also a continuous functions. We can then apply the intermediate value theorem leading to the result that $\exists n \in [-\infty, 0]$ such that $lnG(n) = 0$. We call this value of n, n^*. Using (12), we have

$$\left.\frac{\partial H(n)}{\partial n}\right|_{n^*} = 0.$$

For $n \geq n^*$, $G(n)$ is positive, $(\partial G(n))/(\partial n)$ and $lnG(n)$ are negative, so that $((\partial H(n))/(\partial n)) \leq 0$.

It follows immediately that

1) $\forall n \geq n^*\ (\partial H(n)/\partial n) \leq 0$;
2) $(\partial H(n)/\partial n)|_{n^*} = 0$.

Thus n^* is a fixed point. ■

Practically, as we deal with a discrete product $G(0)\prod_{i=0}^{n} W_i$, it is easy to set $W_{-1} = G(O)$, so that $n = -1$ is the fixed point. Thus, only a coordinate change is needed so that $n = 0$ is a fixed point for the new coordinates.

Third Case: H Depends on the Motivation on a Time Scale t': Result 3: If $0 < W < 1$ there exists a fixed point $n^*(t')$

We have now

$$G(n) = G(0, t') \prod_{i=0}^{n} W_i.$$

If $G(0, t')$ is constant during the diffusion algorithm, then one may use the same result as in the second case. However, now n^* also depends on t', hence, the notation$n^*(t')$.

In practical, we use the same time scale for t and t'.

Fourth Case: The Weights are Modified on a Time Scale t'' by Equation (1): Result 4: If $0 < W < 1$ there exists a fixed point $n^*(t', t'')$

We now have

$$G(n) = G(0, t') \prod_{i=0}^{n} W_i(t'')$$

As in the previous case, if $W_i(tn)$ remains constant during the diffusion algorithm, then $n^*(t', t'')$ is a fixed point. In practical, we use the same time scale for t and t''.

References

[1] W. E. Dixon, D. M. Dawson, F. Zhang, and E. Zergeroglu, "Global exponential tracking control of a mobile robot system via a pe condition," *IEEE Trans. Syst., Man, Cybern. B*, vol. 30, pp. 129–142, Feb. 2000.

[2] G. S. Dordevic, M. Rasic, D. Kostic, and V. Potkonjak, "Representation of robot motion control skills," *IEEE Trans. Syst., Man, Cybern. C*, vol. 30, pp. 219–238, May 2000.

[3] S. S. Ge and Y. J. Cui, "New potential functions for mobile robot path planning," *IEEE Trans. Robot. Automat.*, vol. 16, pp. 615–620, 2000.

[4] A. G. Pipe, "An architecture for learning potential field cognitive maps with an application to mobile robotics," *Adapt. Beh.*, vol. 8, no. 2, pp. 173–204, 2000.

[5] J. Tani and S. Nolfi, "Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor system," in *From Animals to Animats: Simulation of Adaptive Behavior SAB'98*, R. Pfeifer, B. Blumberg, J. A. Meyer, and S. W. Wilson, Eds. Cambridge, MA: MIT Press, 1998, pp. 270–279.

[6] S. X. Yang and M. Meng, "Neural network approaches to dynamic collision-free trajectory generation," *IEEE Trans. Syst., Man, Cybern. B*, vol. 31, pp. 302–318, June 2001.

[7] C. Y. Seong and B. Widrow, "Neural dynamic optimization for control systems part I: Background," *IEEE Trans. Syst., Man, Cybern. B*, vol. 31, pp. 482–489, Aug. 2001.

[8] ——, "Neural dynamic optimization for control systems part II: Theory," *IEEE Trans. Syst., Man, Cybern. B*, vol. 31, pp. 490–501, Aug. 2001.

[9] ——, "Neural dynamic optimization for control systems part iii: applications," *IEEE Trans. Syst., Man, Cybern. B*, vol. 31, pp. 502–513, Aug. 2001.

[10] R. C. Arkin, "Motor schema-based mobile robot navigation," *Int. J. Robot. Res.*, vol. 8, no. 4, pp. 92–112, 1990.

[11] J. W. Polderman and J. C. Willems, *Introduction to Mathematical Systems Theory: A Behavioral Approach*. New York: Springer-Verlag, 1998.

[12] J. A. Meyer and S. W. Wilson, "From animals to animats," in *Proc. First Int. Conf. Simulation Adaptive Behavior*. Cambridge, MA: MIT Press, 1991.

[13] B. Cessac, B. Doyon, M. Quoy, and M. Samuelides, "Mean-field equations, bifurcation map and route to chaos in discrete time neural networks," *Physica D*, vol. 74, pp. 24–44, 1994.

[14] E. Daucé, M. Quoy, B. Cessac, B. Doyon, and M. Samuelides, "Self-organization and pattern-induced reduction of dynamics in recurrent networks," *Neural Networks*, vol. 11, pp. 521–533, 1998.

[15] H. K. Khalil, *Nonlinear Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1996.
[16] J. J. Hopfield, "Neural networks and physical systems with emergent collective behavior," *Proc. Nat. Acad. Sci. USA*, vol. 79, pp. 2554–2558, 1982.
[17] S. Amari, "Dynamics of pattern formation in lateral-inhibition type neural fields," *Biol. Cybern.*, vol. 27, pp. 77–87, 1977.
[18] G. Schöner, M. Dose, and C. Engels, "Dynamics of behavior: theory and applications for autonomous robot architectures," *Robot. Auton. Syst.*, vol. 16, no. 2–4, pp. 213–245, Dec. 1995.
[19] E. Bicho and G. Schöner, "The dynamics approach to autonomous robotics demonstrated on a low-level vehicle platform," *Robot. Auton. Syst.*, vol. 21, pp. 23–35, 1997.
[20] P. Gaussier and S. Zrehen, "A topological map for on-line learning: emergence of obstacle avoidance in a mobile robot," in *From Animals to Animats: SAB'94*. Cambridge, MA: MIT Press, 1994, pp. 282–290.
[21] ——, "PERAC: Aneural architecture to control artificial animals," *Robot. Autonom. Syst.*, vol. 16, no. 2–4, pp. 291–320, 1995.
[22] S. Leprêtre, P. Gaussier, and J. P. Cocquerez, "From navigation to active object recognition," in *SAB 2000*, Paris, France, Sept. 2000.
[23] S. Moga and P. Gaussier, "A neuronal structure of learning by imitation," in *Proc. Eur. Conf. Artificial Life*, vol. 1674, F. Mondada, D. Floreano, and J. D. Nicoud, Eds., Lausanne, France, Sept. 1999, pp. 314–318.
[24] P. Andry, P. Gaussier, S. Moga, J. P. Banquet, and J. Nadel, "The dynamics of imitation processes: from temporal sequence learning to implicit reward communication," *IEEE Trans. Man, Syst., Cybern. A*, vol. 31, pp. 431–442, Sept. 2001.
[25] P. Gaussier, C. Joulain, J. P. Banquet, S. Leprêtre, and A. Revel, "The visual homing problem: an example of robotics/biology cross fertilization," *Robot. Auton. Syst.*, vol. 30, pp. 155–180, 2000.
[26] P. Gaussier, S. Leprêtre, C. Joulain, A. Revel, M. Quoy, and J. P. Banquet, "Animal and robot learning: experiments and models about visual navigation," in *Proc. 7th Eur. Workshop Learning Robots*, Edinburgh, U.K., 1998, pp. 46–55.
[27] P. Gaussier, A. Revel, J. P. Banquet, and V. Babeau, "From view cells and place cells to cognitive map learning: processing stages of the hippocampal system," *Biol. Cybern.*, vol. 86, pp. 15–28, 2002.
[28] E. C. Tolman, "Cognitive maps in rats and men," *Psychol. Rev.*, vol. 55, no. 4, 1948.
[29] C. Thinus-Blanc, *Animal Spatial Navigation*. Singapore: World Scientific, 1996.
[30] A. Revel, P. Gaussier, S. Leprêtre, and J. P. Banquet, "Planification versus sensory-motor conditioning: what are the issues," in *From Animals to Animats: Simulation of Adaptive Behavior*, 1998, p. 423.
[31] M. A. Arbib and I. Lieblich, "Motivational learning of spatial behavior," in *Syst. Neuro.*, J. Metzler, Ed. New York: Academic, 1977, pp. 221–239.
[32] N. A. Schmajuk and A. D. Thieme, "Purposive behavior and cognitive mapping: a neural network model," *Biol. Cybern.*, vol. 67, pp. 165–174, 1992.
[33] I. A. Bachelder and A. M. Waxman, "Mobile robot visual mapping and localization: A view-based neurocomputational architecture that emulates hippocampal place learning," *Neural Networks*, vol. 7, no. 6/7, pp. 1083–1099, 1994.
[34] B. Schölkopf and H. A. Mallot, "View-based cognitive mapping and path-finding," *Adapt. Beh.*, vol. 3, pp. 311–348, 1995.
[35] G. Bugmann, J. G. Taylor, and M. J. Denham, "Route finding by neural nets," in *Neural Networks*, J. G. Taylor, Ed. Henley-on-Thames, U.K.: Alfred Waller Ltd., 1995, pp. 217–230.
[36] O. Trullier, S. I. Wiener, A. Berthoz, and J. A. Meyer, "Biologically based artificial navigation systems: review and prospects," *Progr. Neurobiol.*, vol. 51, pp. 483–544, 1997.
[37] M. O. Franz, B. Schölkopf, H. A. Mallot, and H. H. Büthoff, "Learning view graphs for robot navigation," *Auton. Robots*, vol. 5, pp. 111–125, 1998.
[38] P. Gaussier, S. Leprêtre, M. Quoy, A. Revel, C. Joulain, and J. P. Banquet, *Interdisciplinary Approaches to Robot Learning*. Singapore: World Scientific, 2000, vol. 24, pp. 53–94.
[39] N. Burgess, M. Recce, and J. O'Keefe, "A model hippocampal function," *Neural Networks*, vol. 7, no. 6/7, pp. 1065–1081, 1994.
[40] A. Revel, P. Gaussier, and J. P. Banquet, "Taking inspiration from the hippocampus can help solving robotics problems," in *Proc. IEEE Eur. Symp. Artificial Neural Networks*, Bruges, Belgium, Apr. 1999.
[41] R. E. Bellman, "On a routing problem," *Quart. Appl. Math.*, vol. 16, pp. 87–90, 1958.
[42] M. Quoy, P. Gaussier, S. Leprêtre, A. Revel, C. Joulain, and J. P. Banquet, "A neural model for the visual navigation and planning of a mobile robot," in *Proc. Eur. Conf. Artificial life*, Lausanne, France, Sept. 1999.
[43] N. A. Schmajuk and H. T. Blair, "Place learning and the dynamics of spatial navigation: a neural network approach," *Adapt. Beh.*, vol. 1, pp. 353–385, 1992.
[44] J. Droulez and A. Berthoz, "A neural network model of sensory topic maps with predictive short term memory," *Proc. Nat. Acad. Sci.*, vol. 88, pp. 9653–9657, 1991.
[45] J. A. Scott Kelso, *Dynamic Patterns: The Self-Organization of Brain and Behavior*. Cambridge, MA: MIT Press, 1995.
[46] P. Gaussier, S. Moga, J. P. Banquet, and M. Quoy, "From perception-action loops to imitation processes," *Appl. Artif. Intell.*, vol. 1, no. 7, pp. 701–727, 1998.
[47] E. Daucé, M. Quoy, and B. Doyon, "Resonant spatio-temporal learning in large random neural networks," *Biol. Cybern.*, vol. 87, pp. 185–198, 2002.

Input-to-State Stabilization of Dynamic Neural Networks

Edgar N. Sanchez and Jose P. Perez

***Abstract*—As a continuation of their previous published results, in this paper the authors propose a new methodology, for input-to-state stabilization of a dynamic neural network. This approach is developed on the basis of the recent introduced inverse optimal control technique for nonlinear control. An example illustrates the applicability of the proposed approach.**

***Index Terms*—Dynamic neural networks, Lyapunov analysis, nonlinear systems, stability.**

I. Introduction

Neural networks have became an important methodology to various scientific areas for solving difficult problems and for improving system performance. Among the different proposed neural network schemes, the Hopfield-type neural network [4] remains an important architecture due to successful applications in solving associative memory, pattern recognition, identification and control, and optimization problems as well as its easy VLSI implementation.

Analysis of dynamic neural network stability has attracted a great deal of attention since the late 1980s [9]. When a neural network is employed as an associative memory, the existence of many equilibrium points is a needed feature. However many engineering applications, such as identification and control, involve optimization problems, where it is required to have a well-defined solution for all possible initial conditions. From a mathematical perspective, this means that the neural network should have a unique equilibrium point, which is

Manuscript received October 19, 2001; revised July 2, 2002. This work was supported by CONACYT, Mexico, under Project 32059A and also by the UANL School of Mathematics and Physics. This paper was recommended by Associate Editor H. Takagi.

E. N. Sanchez is with CINVESTAV, Unidad Guadalajara, C.P. 45091, Guadalajara, Mexico (e-mail: sanchez@gdl.cinvestav.mx).

J. P. Perez is with CINVESTAV, Unidad Guadalajara, Guadalajara, Jalisco, C.P. 45091, Mexico, on leave from the School of Mathematics and Physics of Universidad Autonoma de Nuevo Leon (UANL), Monterrey, Mexico.

Digital Object Identifier 10.1109/TSMCA.2003.811509