

A neural framework for adaptive robot control

Mohamed Oubbati · Günther Palm

Received: 22 August 2008 / Accepted: 11 March 2009 / Published online: 31 March 2009
© Springer-Verlag London Limited 2009

Abstract This paper investigates how dynamics in recurrent neural networks can be used to solve some specific mobile robot problems such as *motion control* and *behavior generation*. We have designed an adaptive motion control approach based on a novel recurrent neural network, called Echo state networks. The advantage is that no knowledge about the dynamic model is required, and no synaptic weight changing is needed in presence of time varying parameters in the robot. To generate the robot behavior over time, we adopted a biologically inspired approach called neural fields. Due to its dynamical properties, a neural field produces only one localized peak that indicates the optimum movement direction, which navigates a mobile robot to its goal in an unknown environment without any collisions with static or moving obstacles.

Keywords Recurrent neural networks · Metalearning · Mobile robots · Navigation · Adaptive control

1 Introduction

One important task of mobile robotics is to move a robot *safely* and *precisely* in real environments. The *safety* and *precision* requirements need to address two issues: *Motion Control* and *Behavior Generation*. Motion control issue concerns the low-level control providing the robot the ability to track a desired trajectory. The issue of behavior

generation concerns a higher level control, which offers a mobile robot the ability to make decisions on how to achieve a task under constraints, e.g. moving towards a target while avoiding obstacles.

Many authors have studied motion control of mobile robots in the last decade. At the beginning, the research effort was focused only on the kinematic level, assuming that there is *perfect* velocity tracking [1]. Later on, the research has been conducted to design motion controllers, also including the dynamics of the robot [2, 3]. Taking into account the specific robot dynamics is more realistic, because the assumption, *perfect* velocity tracking does not hold in practice. Moreover, during the motion, the robot parameters may change due to surface friction, additional load, energy supply, among others. Therefore, the control at the dynamic level is at least as important as the control at the kinematic level. In this issue, it is desirable to develop a robust control, which has the following capabilities: (1) ability to successfully handle errors and noise in sensor signals, (2) high-performance velocity tracking, and (3) adaptation ability in presence of time varying parameters in the robot.

Approaches that have been developed for behavior generation issue can be divided into global and local methods. Global methods consider the robot motion planning as an optimization problem, where the goal is to return a continuous collision-free path that satisfies certain optimization criteria. This concept requires that the environment should be completely known with a static configuration. If unexpected changes occur in the environment, the previously found path has to be modified through re-planning which needs an expensive computation cost. Local methods, by contrast, need only local information. It means that the path planning is done while the robot is moving in response to environmental changes. Due

M. Oubbati (✉) · G. Palm
Institute of Neural Information Processing,
University of Ulm, James-Franck-Ring, 89081 Ulm, Germany
e-mail: mohamed.oubbati@uni-ulm.de

G. Palm
e-mail: guenther.palm@uni-ulm.de

to their low-computational costs, local methods are much more suitable for real application where the environmental state changes are continuous. The most popular local method is the potential field approach proposed by Khatib [4]. While this approach is very attractive from a computational point of view since no planning is required, substantial drawbacks have been identified, i.e. local minima (cyclic behavior), no passage between closely spaced obstacles, oscillations in narrow passages, etc. [5].

In this paper, we investigate how recurrent neural networks (RNNs) can offer efficient and practical solutions to provide a mobile robot the capability to perform navigation in an environment that is unknown. Two main tasks are addressed in this work: (1) designing an adaptive motion control system, and (2) finding a robust framework for the behavior generation control. The ability of RNNs to instantiate arbitrary temporal dynamics allows us to design motion control, also including the dynamics part of the robot. Moreover, through *metalearning*, no synaptic weight changing would be needed in presence of parameters' variation in the robot. For behavior generation, the concept of dynamical neural fields [6] is adopted as a framework. A self-stabilized peak of activation is linked to on-line sensory information and decodes optimal behaviors. This approach presents a new framework that unifies both design of behaviors and their coordination.

The rest of this paper is organized as follows. Section 2 describes formally the concept of echo state networks. Section 3 is concerned with the tracking control at the kinematic and dynamic level of mobile robots. The behavior generation system based on the concept of neural fields is described in Sect. 4. In Sect. 5, the whole neuro-control system is designed for navigation. Implementation results are presented in Sect. 6, and finally, discussion and conclusion are drawn in Sect. 7.

2 Echo state networks

Feedforward networks have been successfully used to solve problems that require the computation of a static function, i.e. a function whose output depends only upon the current input, and not on any previous inputs. In the real world, however, one encounters many problems which cannot be solved by learning a static function, because the function being computed changes with each input received. It is clear from the architecture of feedforward neural networks that past inputs have no way of influencing the processing of future inputs. This situation can be rectified by the introduction of feedback (recurrent) connections in the network. RNNs have an internal state, which is essential for many domains where time plays a role, such as telecommunication [7], adaptive identification and control

[8, 9], time series prediction [10], or robotics [11] to name just a few. In principle, RNNs can implement almost arbitrary sequential behavior [12–14], and they have the ability to model multiple non-linear systems with fixed weights (topic covered in Sect. 3). However, RNNs are often regarded as difficult to train. The first difficulty is associated with the derivative calculations that are required for gradient-based training procedures. Due to the dynamical nature of these networks, the derivatives must reflect the network dynamics. Treating these derivatives as static functions of the network weights will result in poor mapping performances, since the weights search will be along inappropriate directions. Several methods have been explored for calculating derivatives with respect to network weights, such as back propagation through time [15] and real-time recurrent learning [12].

In our work, we used a novel RNN called *Echo state network* (ESN) [16]. The basic idea of ESN is that a readout neuron can learn to extract salient information from a high-dimensional transient state of a large RNN. This readout neuron can be trained by supervised or unsupervised learning methods. An ESN (Fig. 1) is formed by a so-called “dynamic reservoir” (DR), which contains a large number of sparsely interconnected neurons with non-trainable weights. It has K inputs, N internal neurons and L output neurons. Activations of input neurons at time step n are $U(n) = (u_1(n), u_2(n), \dots, u_k(n))$, of internal neurons are $X(n) = (x_1(n), \dots, x_N(n))$, and of output neurons are $Y(n) = (y_1(n), \dots, y_L(n))$. Weights for the input connection in a $(N \times K)$ matrix are $W^{in} = (w_{ij}^{in})$, for the internal connection in a $(N \times N)$ matrix are $W = (w_{ij})$, and for the connection to the output neurons in an $L \times (K + N + L)$ matrix are $W^{out} = (w_{ij}^{out})$, and in a $(N \times L)$ matrix $W^{back} = (w_{ij}^{back})$ for the connection from the output to the internal neurons.

The activation of internal and output units is updated according to:

$$X(n+1) = f(W^{in}U(n+1) + WX(n) + W^{back}Y(n+1)) \quad (1)$$

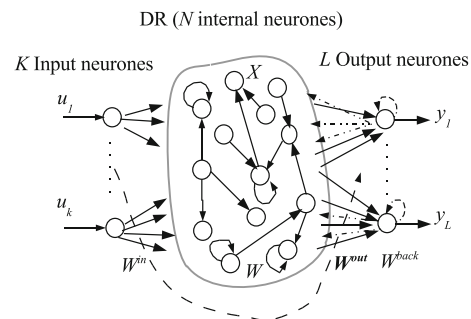


Fig. 1 Basic architecture of ESN. Dotted arrows indicate connections that are possible but not required

where $f = (f_1, \dots, f_N)$ are the internal neurons output sigmoid functions.

The outputs are computed according to:

$$Y(n + 1) = f^{\text{out}}(W^{\text{out}}(U(n + 1), X(n + 1), Y(n))) \quad (2)$$

where $f^{\text{out}} = (f_1^{\text{out}}, \dots, f_L^{\text{out}})$ are the sigmoid functions of the neuron outputs. The term $(U(n + 1), X(n + 1), Y(n))$ is the concatenation of the input, internal, and previous output activation vectors.

2.1 Training ESNs

The ESN approach is based on the concept of *echo states*. It is a property of the network prior to training, related to the weight matrices $(W^{\text{in}}, W, W^{\text{back}})$ and to training data. A network $(W^{\text{in}}, W, W^{\text{back}})$ has echo states with respect to the compact intervals U and D , if the network state $X(n)$ is uniquely determined by the history of the input–output data. Several equivalent characterizations of echo states are collected in [17]. In practice, it is found that when the spectral radius of the internal matrix $|\lambda_{\text{max}}| < 1$, we do have an echo state network. The following procedure seems to give a practical solution to guaranty echo state property:

1. The order of input and output neurons should be stated according to the task at hand.
2. Generate randomly the input weights W^{in} and output backpropagated weights W^{back} .
3. Generate randomly an internal weight matrix W_0 .
4. Normalize W_0 with its spectral radius λ_{max} and put it in $W_1 : W_1 = \frac{1}{|\lambda_{\text{max}}|} W_0$.
5. Scale W_1 with a factor $0 < \alpha < 1$ and put the new internal matrix $W = \alpha W_1$.

It has been always found that using the steps above the untrained network $(W^{\text{in}}, W, W^{\text{back}})$ is an echo state network. Once the echo state property is verified, only weights connections from the internal neurons to the output (W^{out}) are to be adjusted following next steps:

1. Compute the network states by presenting I/O training sequence $(u(n), d(n))$:

$$X(n + 1) = f(W^{\text{in}}U(n + 1) + WX(n) + W^{\text{back}}Y(n + 1)) \quad n = 0, \dots, T \quad (3)$$

2. Collect at each time the state $X(n)$ as a new row into a state collecting matrix M , and collect similarly at each time the sigmoid-inverted teacher output $\tanh^{-1}D(n)$ into a teacher collection matrix C . After these collections, the matrix M has the size of $(T + 1) \times (K + N + L)$, and the matrix C has the size of $(T + 1) \times L$.

3. Adjust the output weights: compute the pseudoinverse of M and put:

$$W^{\text{out}} = (M^{-1}C)^t \quad (4)$$

t indicates transpose operation.

The ESN is now trained. For exploitation, the network can be driven by new input sequences and using (1) and (2).

3 Adaptive motion control with ESNs

We consider a mobile robot having a geometric configuration described by $q = [x, y, \phi]^T$ where (x, y) are its position in the global coordinate system, and ϕ its heading angle. Assuming that (v, w) are linear and angular velocities of the robot, its kinematic model is given by the following equation [3]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos(\phi) & 0 \\ \sin(\phi) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \quad (5)$$

In motion control, the objective is to control the velocity of a robot such that its pose $P = [x, y, \phi]^T$ follows a reference trajectory $P_r = [x_r, y_r, \phi_r]^T$. Various feedback controllers have been proposed to solve this problem (see the survey paper [1] and references cited therein). Most of them neglect the vehicle dynamics and consider only the steering system (5), where the velocities (v, w) are supposed to be the robot inputs. Little has been written about integrating the dynamic part of the robot, and literature on robustness in presence of uncertainties in the dynamical model is rare [18]. This is because exact knowledge about the dynamic model parameters is almost unattainable in practical situations. If we consider that these parameters are time varying, the problem becomes more complicated. To solve this problem, we designed an adaptive neurocontrol system with two levels [19]. In the first level, a recurrent neural network (ESN_K) improves the robustness of a kinematic controller and generates desired linear and angular velocities, necessary to track a reference trajectory. In the second level, another recurrent neural network (ESN_D) converts the desired velocities into a torque control (Fig. 2).

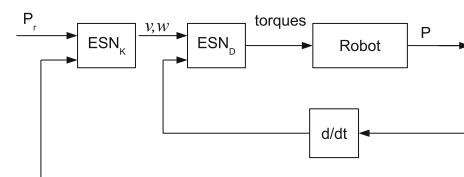


Fig. 2 Motion control using ESNs

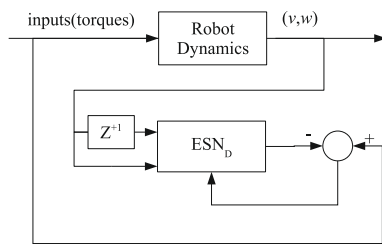


Fig. 3 Training ESN_D as a dynamic controller

3.1 Training of ESN_D

Training procedure involves using actual and future delayed velocities of the robot as inputs, and correspondent torques as teacher signals (Fig. 3). The goal of this training procedure can be summarized as follows: find the weights of the network using the teacher signal, which brings the robot from the actual velocity at time (n) to a future velocity at time $(n + 1)$ [20]. The advantage of this approach is that, no knowledge about the dynamic model is required, since the controller is designed only by learning input/output data collected from the robot. This property is very useful in practical situations, where the exact knowledge about the robot parameters is almost unattainable. Furthermore, no synaptic weight changing is needed in presence of parameters variation. This can be reached through prior *metalearning*.

3.2 Metalearning

It seems that metalearning represents a big puzzle. Each research group see metalearning from a different angle (for a comprehensive survey refer to [21]). In spite of the many research efforts, no clear answer has emerged, but its fundamental research question remains constant: *how can we improve learning algorithms by exploiting past knowledge (learning experience)?* The machine learning community uses meta-learning to build up neural networks, which have the ability to adapt without changing explicit weights. One may directly ask: *how does the network adapt, if its weights are fixed?* This question was first discussed in 1990 by Cotter and Conwell [22]. They prove the Fixed Weight Learning Theorem that describes how fixed-weight neural network can approximate the dynamics of a feedforward network trained by an adaptive weight-learning algorithm. This approximation does not require any weight change, hence the “fixed-weight” label. Later on, Feldkamp et al. [23] have shown that a single fixed weight RNN can perform one-time-step prediction for many distinct time series. In the control domain, it was shown in [24] that a RNN can be trained to act as a stabilizing controller for three unrelated systems and to

handle switch between them. Adaptation with fixed weights in RNNs is possible, since recurrent signal loops can store information by accumulation, and they act like weights in a conventional network [25]. It is the way of selecting training data that determines the difference between (base-) learning and metalearning. In a base-learning, training data are collected from a single functional mapping, whereas in meta-learning, training data are collected from different families of functional mappings. In metalearning, recursive algorithms will suffer from the problem of *recency effect*: the tendency to forget what has been learned in the past. If training data are homogenous, one or more passes through the data can probably produce good results. However, when training data are heterogeneous, the tendency of training will be in a favor of the most recently presented data. To avoid this, an effective solution is to scramble the order of training data presentation, or to run the network through the entire data set, and make an update based on the entire set errors. The later is known as batch update algorithm. Prokhorov et al. [26] propose the multi-stream procedure to deal with the recency effect. It is an extension of Extended Kalman Filter training methods, which combines both scrambling and batch update methods. As a result, this procedure provides the benefits of the batch learning to circumvent the recency effect, while remaining consistent with the Kalman recursion. Recently, we explored the notion that a well-trained ESN needs to change only its internal state to change its behavior policy [27]. It was shown how an ESN identifier was able to recognize and track changing in a non-linear dynamical system with time varying parameters only through its inputs and its own state, without changing any synaptic weight. In our previous work [19, 28], the ESN_D was asked to exhibit a characteristic, normally ascribed to adaptive controllers, whose parameters change in response to an environmental change. *Adaptation* was defined as the ability of the controller ESN_D to recognize changing in the robot parameters only through its inputs and its own state, without changing any synaptic weight. It was first trained for *all* possible operating conditions of the dynamical model, corresponding to “all” possible combinations of the time varying parameters of the robot. After training, the fixed weight ESN_D controller showed a reasonable balance between variety of reference velocity and dynamic properties of the robot.

3.3 Training of ESN_K

A simple approach for training ESN_K as a kinematic controller, is to drive the robot randomly to different positions and collect correspondent I/O data for training. This would provide nice results only when training data cover the whole space of operation. Another approach is to learn a

behavior of a predesigned controller and to improve its performance. In [19], we designed an ESN_K to mimic a kinematic non-linear controller, and to improve its robustness against noise. The non-linear control used is described as following. Let there be prescribed a reference robot (trajectory):

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\phi}_r \end{bmatrix} = \begin{bmatrix} \cos(\phi_r) & 0 \\ \sin(\phi_r) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_r \\ w_r \end{bmatrix} \tag{6}$$

where $x_r, y_r,$ and ϕ_r are the pose of the reference robot, and (v_r, w_r) are its linear and angular velocities. We define e_1, e_2, e_3 as following:

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \phi_r - \phi \end{bmatrix} \tag{7}$$

The inputs (v, w) which make e_1, e_2, e_3 converge to zero are given by [2, 3]:

$$\begin{cases} v_d = v_r \cos e_3 + K_1 e_1 \\ w_d = w_r + v_r K_2 e_2 + K_2 \sin e_3 \end{cases} \tag{8}$$

where K_1, K_2, K_3 are positive constants.

Figure 4 depicts a bloc diagram of the training procedure. First, a random reference trajectories $P_r = [x_r, y_r, \phi_r]$ was generated. The kinematic model of the robot is then controlled by (8) in order to minimize the error between the actual pose of the robot $P = [x, y, \phi]$ and the reference trajectory. At the same time, ESN_K learns the behavior of (8). In addition, to provide the ESN_K robustness against disturbances, a gaussian noise with zero mean was added to training data. After training, the ESN_K showed more robustness to recover perturbation, compared to the feedback controller (see [19] for more details).

4 Behavior generation

Approaches that have been developed for robot navigation range from centralized sense-plan-act (SPA) architectures

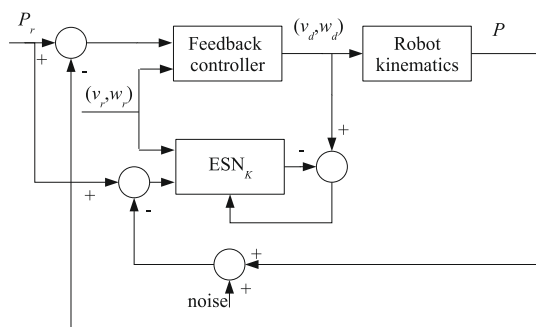


Fig. 4 Training ESN_K as a kinematic controller

to distributed behavior based architectures. SPA needs global explicit representations of the robot with its environment in order to plan ways to solve problems. Behavior-based design, instead, focuses on solutions to sub-problems rather than building a searching machinery for “mostly” unpredictable environments. It tries to find inverse models of certain sub-tasks considered as basic behaviors. Each behavior produces immediate reactions in order to achieve a specific task such as: “moving to a target”, or “avoiding obstacles”. One of the central design challenges of behavior-based systems is to find effective mechanisms for coordination between behaviors. Recently, the theory of dynamical systems has proven to be an elegant and easy paradigm for local navigation [29–34]. It provides a framework to design differential equations for so-called behavior variables that generates the robot’s behaviors over time. A more general form of dynamic systems can be reached through the concept of neural fields. Originally, these fields were proposed by Amari [6] as models of the neurophysiology of cortical processes. They are equivalent to continuous recurrent neural networks, in which, neurons are laterally coupled through an interaction kernel and receive external inputs. This concept has also shown great robustness in mobile robot navigation [35–40].

4.1 Dynamic neural fields

The equation of a one-dimensional dynamic neural field (DNF) is given by

$$\tau \dot{u}(\varphi, t) = -u(\varphi, t) + S(\varphi, t) + h + \int_{-\infty}^{+\infty} w(\varphi, \hat{\varphi}) f(u(\hat{\varphi}, t)) d\hat{\varphi} \tag{9}$$

The parameter $\tau > 0$ defines the time scale of the field, $u(\varphi, t)$ is the field excitation at the position (neuron) φ and at time t . We used the symbol φ for neurons, because each neuron will encode a possible direction for the robot (see next paragraph). The temporal derivative of the excitation is defined by

$$\dot{u}(\varphi, t) = \frac{\partial u(\varphi, t)}{\partial t} \tag{10}$$

The constant h defines the resting activity level of the neurons, and $f(u)$ is the local activation function, usually chosen as a sigmoid function. The stimulus $S(\varphi, t) \in \mathfrak{R}$ represents the input of the field at position φ and at time t . A non-linear interaction between the excitation $u(\varphi, t)$ at the position φ and its neighboring positions $\hat{\varphi}$ is achieved by the convolution of an interaction kernel $w(\varphi, \hat{\varphi})$. That is, the activity change of a neuron φ depends on its actual activity level, the weighted input from other neurons, and the external input (stimulus) at its position. Depending on

the parameter h , and the functions S, f , and w , equation (9) can have different types of solutions:

1. \emptyset -solution, if $u(\varphi) \leq 0, \forall \varphi$.
2. ∞ -solution, if $u(\varphi) > 0, \forall \varphi$.
3. a -solution, if there is localized excitation from a position a_1 to a position a_2 . This solution is also called a *single-peak* or *mono-modal* solution.

In a -solution, weights in local neighborhood are stronger than those for large distances, which has the effect to stabilize a single peak at a position φ when a stimulus $S(\varphi, t)$ is very large compared with the within-field cooperative interaction in that position. Furthermore, due to neurons interaction, the peak remains in its position, even if we remove the stimulus. This interaction has also the effect to “push” the peak towards positions with maximum stimulus level (Fig. 5). More detailed information can be found in [6, 41].

4.2 Neural fields for robot navigation

The key idea of using neural fields in robotics navigation is to provide sensory information as stimulus to the neural field, and the position of the peak will decode the correspondent movement direction of the robot. In the experiments, the neural field has to encode angles from $-\pi$ to $+\pi$ divided into N discrete directions. Therefore, the model of (9) is first discretized as

$$\tau \dot{u}(\varphi, t) = -u(\varphi, t) + S(\varphi, t) + h + \sum_{\hat{\varphi}=1}^N w(\varphi, \hat{\varphi})f(u(\hat{\varphi}, t)) \tag{11}$$

where $h = -1$, the activation function f is chosen as a step-function

$$f(u) = \begin{cases} 1, & u \geq 0 \\ 0 & u < 0 \end{cases} \tag{12}$$

and the interaction kernel as

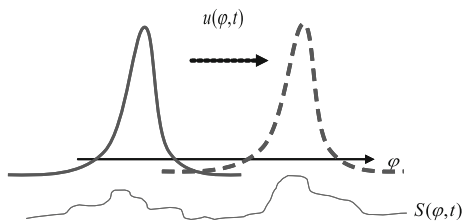


Fig. 5 Neural field in a -solution. In this solution, when an input of a stimulus $S(\varphi, t)$ is very large compared with the within-field cooperative interaction, a single-peak will be stabilized by interaction. It follows then the maximum of the stimulus and remains there, even if the stimulus is removed

$$w(\varphi, \hat{\varphi}) = \begin{cases} 1 & \|\varphi, \hat{\varphi}\| \leq 3 \\ -2.2 & \text{otherwise} \end{cases} \tag{13}$$

where $\|\varphi, \hat{\varphi}\|$ is the minimum distance between neuron φ and neuron $\hat{\varphi}$. The interaction kernel $w(\cdot)$ is chosen as a window function so that excitatory connections dominate for proximate neurons, and inhibitory connections dominate at neurons with greater distances. After the stabilization of the field, the most activated neuron φ_{peak} (peak position) decodes the direction to be executed by the robot:

$$\varphi_{\text{peak}} = \text{argmax}\{u(\varphi)|\varphi \in [1, N]\}. \tag{14}$$

Before selecting an appropriate direction, the neural field needs some necessary information (stimulus) about the robot and the environment states. In our task, the global stimulus is determined according to two stimuli:

4.2.1 Target stimulus

This stimulus is designed excitatory, showing an attraction toward the target direction. It is chosen as

$$S_{\text{target}}(\varphi, t) = c_{\text{max}} - c_a \cdot \|\varphi_{\text{target}}, \varphi\| \tag{15}$$

where c_{max}, c_a are constants positive chosen as 15 and 0.45, respectively. φ_{target} is the field position, equivalent to the main target direction at time t .

4.2.2 Obstacle stimulus

This stimulus shows directions to obstacles $\{S_{\text{OI}}(\varphi, t): l \in [1, N_{\text{Obst}}]\}$, where N_{Obst} is the number of obstacles detected by the robot sensors. It is chosen as gauss function centered at the direction of an obstacle.

$$S_{\text{OI}}(\varphi, t) = C_O e^{-\sigma(\varphi - \varphi_l)^2} \tag{16}$$

where C_O and σ are positive constants. C_O defines the amplitude of the stimulus chosen as 40, and σ_O defines the range of inhibition of an obstacle. In practical situations, this parameter is tuned according to the radius of the robot and the obstacles. φ_l reflects the direction of the obstacle l at time t . However, the stimulus considers only obstacles in distances d_{OI} which are below a threshold d_{Th} . The contribution of all obstacles stimuli determine the final obstacle stimulus:

$$S_{\text{obstacle}}(\varphi, t) = \sum_{l=1}^{N_{\text{Obst}}} g(d_{\text{OI}})S_{\text{OI}}(l, t) \tag{17}$$

where g is a step function:

$$g = \begin{cases} 1, & d_{\text{OI}} < d_{\text{Th}} \\ 0 & d_{\text{OI}} \geq d_{\text{Th}} \end{cases} \tag{18}$$

Using the step function g , only obstacles, which their distances d_{O_i} to the robot are below a threshold d_{Th} , are considered.

4.2.3 Global stimulus

The contributions of all stimuli determine the final state of the field, considering that Obstacles stimulus must be taken inhibitory, since obstacles collision must be avoided. Thus, the global stimulus of the field at time t is determined by

$$S(\varphi, t) = S_{target}(\varphi, t) - S_{obstacle}(\varphi, t) \tag{19}$$

5 Global control system

The global neurocontrol system is decomposed in two stages. A high-level control based on neural fields concept providing desired direction in presence of static and dynamic obstacles, and a low-level control system, which guarantees that the robot follows the desired trajectory (Fig. 6). A more detailed architecture is depicted on Fig. 7. The neural field model is designed in order to produce peak-solutions of the field activation, which encode the appropriate robot direction P_r in response to a change in the environment. The motion control is designed using an outer-loop, where the ESN_K computes the desired velocity necessary to track a reference trajectory P_r , and an inner-loop, depending on the robot dynamics, where the ESN_D converts the velocity control provided by the outer-loop into a torque control.

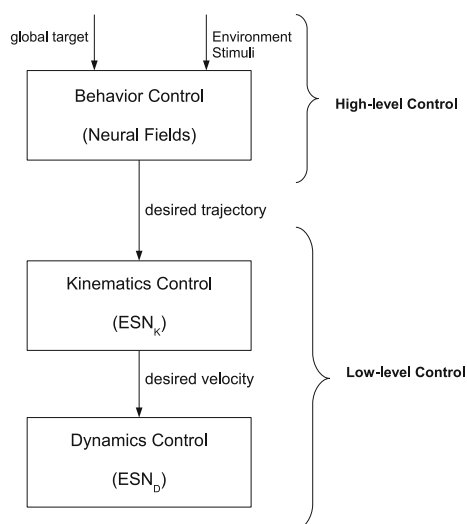


Fig. 6 Recurrent neural networks for robot navigation: a global architecture

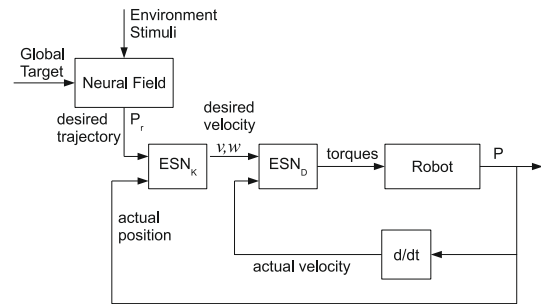


Fig. 7 Recurrent neural networks for robot navigation: a detailed architecture

6 Experimental results

Here, we present some results obtained using an omnidirectional robot (Fig. 8). The robot is equipped with three omni-wheels equally spaced at 120 degrees from one to another, and driven by three 90 W DC motors. A personal computer on board is used to manage different sensors and tasks. For environment sensing, the robot is equipped by an omnidirectional vision system, based on a hyperbolic mirror and a standard IEEE1394 (FireWire) camera. Omnidirectional vision provides our robot a very large field of view, which has some useful properties. For instance, it can facilitate the tracking of robots, the ball, and a set of environmental features used for self-localization. The geometry of the omnidirectional robot and its coordinate definitions are shown in Fig. 9. This model is used to transform linear and angular velocities to wheels' speeds. It is chosen as in [42]:

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} \sin(\theta) & \cos(\theta) & L \\ -\sin(\frac{\pi}{3} - \theta) & -\cos(\frac{\pi}{3} - \theta) & L \\ \sin(\frac{\pi}{3} + \theta) & -\cos(\frac{\pi}{3} + \theta) & L \end{pmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} \tag{20}$$

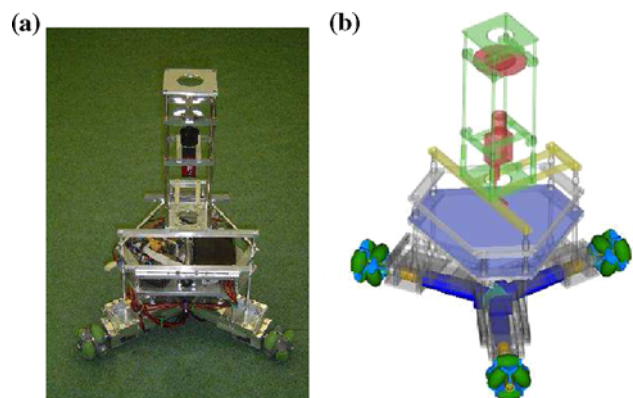


Fig. 8 Omnidirectional robot. **a** Hardware photo. **b** CAD model

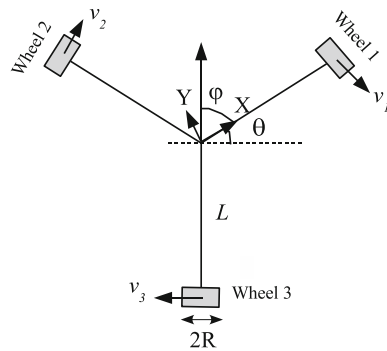


Fig. 9 Kinematic geometry

where w_i is the angular speed of wheel i , and x , y and θ are the pose of the centre of mass (CM) of the robot. L is the distance of the wheels from the CM, and R is the radius of each wheel. In our case: $R = 0.04$ m and $L = 0.2$ m.

6.1 Motion control

Here, the ESN_D controller was asked to make an adaptive dynamic control for the robot, whose global mass is time varying (a switch between 15 and 17.5 kg). This change can result also to variations of the CM position. Upon completion of the training procedure, we expect that the ESN controller will be capable to exhibit characteristics, normally ascribed to adaptive controllers: (1) detect the mass and CM variation, (2) adapt itself to the perceived change and generate the appropriate control signal, and (3) make an acceptable generalization to track reasonably a reference velocity. Naturally, we expect adequate performance only for the two possible global mass (15 and 17.5 kg) of the robot for which the network has been trained. Metalearning procedure can offer the ESN velocity controller to recognize the mass variation only through its inputs, and its own state, without changing any synaptic weight. Training data were prepared by driving the robot with different velocities in different directions. 1,000 I/O sequences were collected from the robot having its initial mass (15 kg) and other 1,000 I/O sequences when the extra mass $m = 2.5$ kg was put on the robot. During training, no multi-stream was needed, since the ESN batch update uses all data at once, and does not suffer from the recency effect. The ESN architecture was chosen as follows. 6 inputs (actual and delayed wheels speeds), 13 internal neurons and 3 outputs (Pulse width modulation-PWMs duty ratio for each motor). No back-connection from the output to the DR, and no connections from the input directly to the output. The input and the internal synaptic connections weights were randomly initialized from a uniform distribution over $[-1, +1]$. The internal weight

matrix W has a sparse connectivity of 20% and scaled such that its maximum eigenvalue $|\lambda_{\max}| \approx 0.3$. After training, the network was implemented in the control system on the host computer on-board, and several experimental tests were performed. We report here one of them (Fig. 10). The desired linear and angular velocities (v_d , w_d) were chosen as follows:

$$\begin{aligned} 1s \leq t \leq 13s : v_d &= 0.23 \text{ m/s}, w_d = 0 & (\varphi = 0) \\ 14s \leq t \leq 18s : v_d &= 0 \text{ m/s}, w_d = 3.5 & (\varphi = 0) \\ 19s \leq t \leq 26s : v_d &= 0.27 \text{ m/s}, w_d = 0 & (\varphi = \frac{2\pi}{3}) \end{aligned}$$

Using the actual (measured) and desired wheel-speeds, the ESN send the appropriate PWMs duty to the motors. During the first 13 s, the ESN was asked to control the robot having a mass of 17.5 kg. In this interval, the robot will move straight ahead (no angular velocity) which requires zero speed for wheel 3. Here, the ESN controller could bring the robot to the desired velocity (0.23 m/s), even the presence of high friction between the carpet and wheel 3 which explains the high frequency in the control signal delivered by the ESN for this wheel. At time $t = 13$ s, the mass of the robot is reduced to 15 kg. Surprisingly, the control is barely affected by this variation. The ESN recognized this change only through its inputs and its own state, and could provide the appropriate control signals in order to bring the robot to the desired velocity. Despite the poor quality of training data, and the performance limitation of the omni-wheels, the ESN controller could achieve acceptable adaptive control results and showed a reasonable balance between the variety of the reference velocity and the robot mass variation. Note that the ESN controller is effective only for the two operation conditions of the robot (15 and 17.5 kg) for which it has been trained, not for arbitrarily chosen robot-mass.

6.2 Behavior control

The concept of neural fields is used to generate the robot behavior over time. In the experiments, the neural field has to encode angles from $-\pi$ to $+\pi$. By means of a codebook, we use N discrete directions. We chose $N = 60$ neurons, which means that each direction N decodes a step of 6° . The neural field is expected to align the robot's heading with the direction of the target, and to bring it away from nearby obstacles. We will illustrate this with an experiment showing target acquisition with collision avoidance for multiple static and moving obstacles (Fig. 11). During the first 20 time steps, both the stimulus (Fig. 11b) as well as the field activation (Fig. 11a) are unimodal, since no obstacles are detected and the stimulus contains only the target entries. In the following time

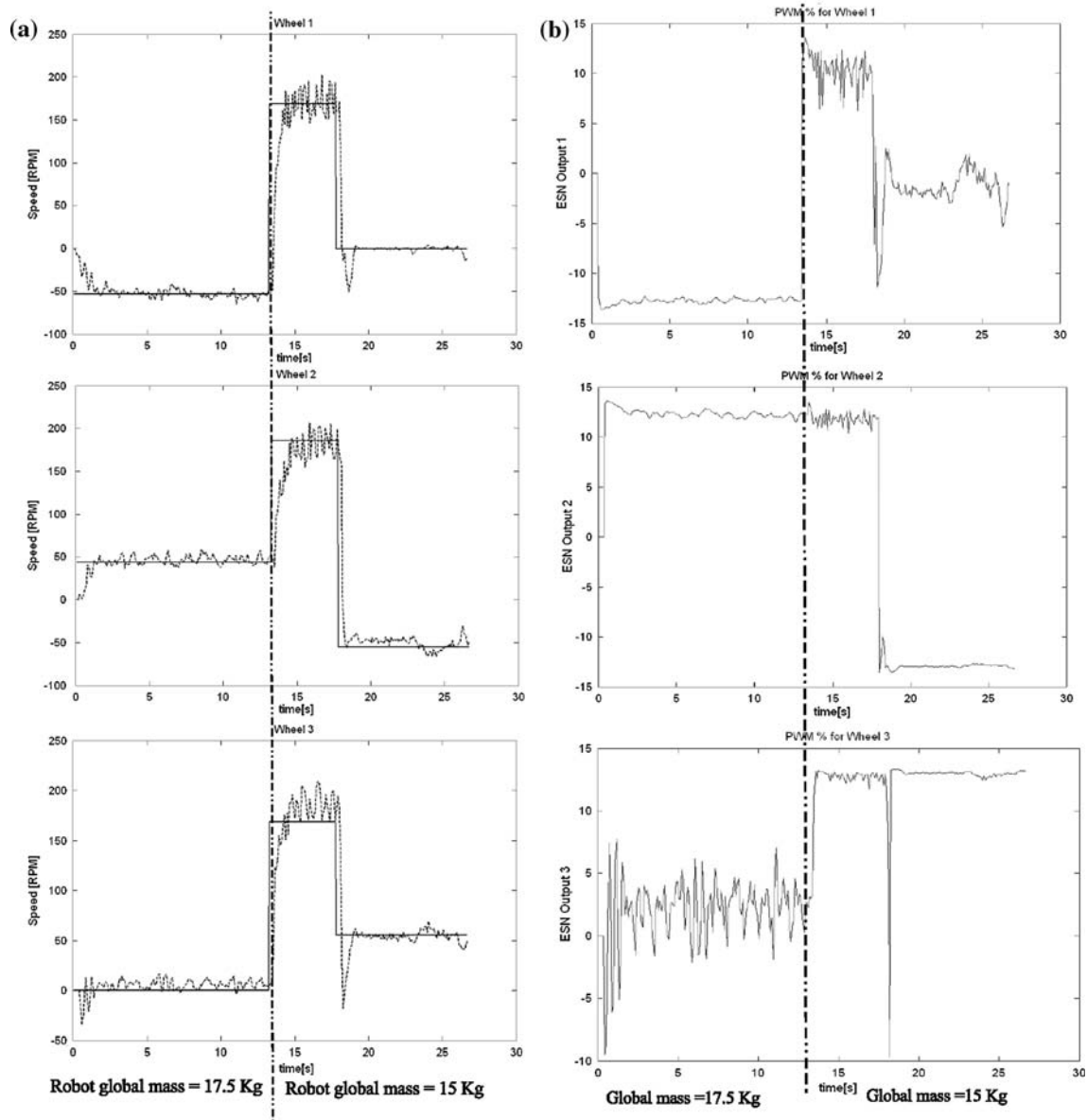


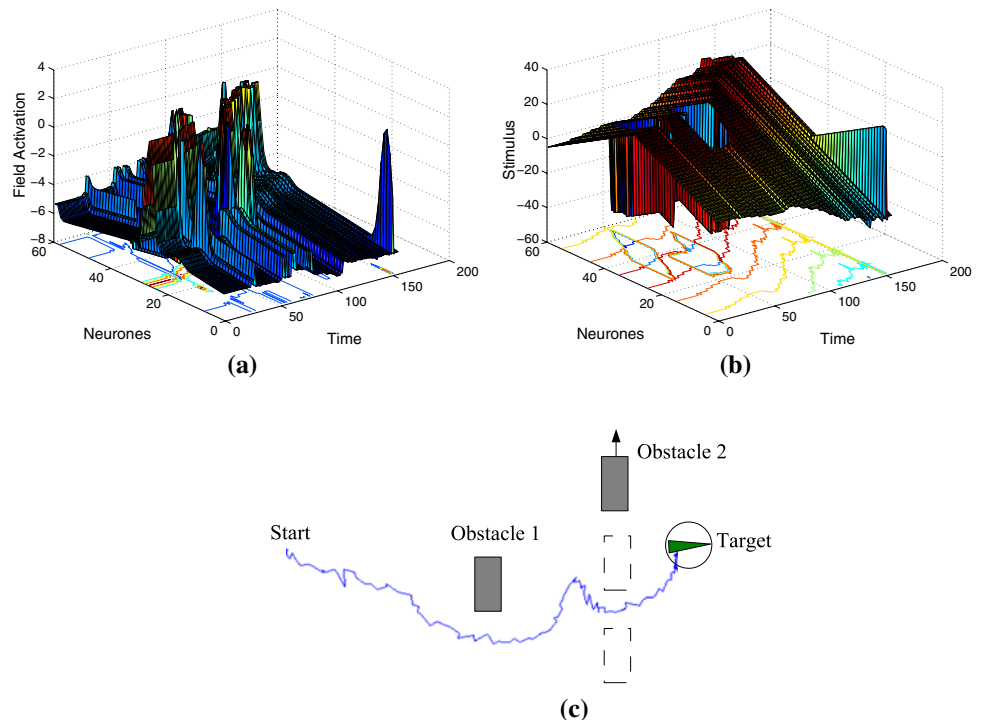
Fig. 10 Dynamic adaptive control with ESN. **a** Desired speeds (*solid*) and actual wheels speeds (*dashed*). **b** ESN control signals

steps, the stimulus contains also obstacle entries. Therefore, it becomes bimodal. By contrast, on the field activation, the peak follows the optimum position resulted from the combination of target-acquisition and obstacle-avoidance behaviors. It now provides an appropriate heading direction in order to avoid obstacle 1. At time step 51, the stimulus (Fig. 11b) contains now entries from the moving obstacle 2. Again, the robot behaves with the same manner to avoid the moving obstacle. After passing obstacle 2, the entries of the target becomes stronger than those of obstacles, and the peak changes its position relative to the direction of the target. The whole path is illustrated on Fig. 11c.

7 Discussion and conclusion

In this paper, we investigated how dynamics in recurrent neural networks can offer solutions for robot navigation. The ability of RNNs to handle arbitrary temporal dynamics, and robustness for noise environment make them an ideal choice for real dynamical systems control. However, this potential is not being exploited because simple and powerful training algorithms were missing. To overcome this problem, we adopted the concept of ESNs. The basic idea is that a large recurrent network can serve as a source of dynamics from which information can be extracted by a readout neuron. The training experiments carried out here

Fig. 11 Behavior generation with neural field. **a** Field activation. **b** Stimulus. **c** 2D Path executed by the robot



demonstrate that small and partially interconnected ESNs could be trained to act as motion controllers for mobile robots. The advantage of ESNs is that no physical model of the robot model is required, and no synaptic weight changing is needed in presence of time-varying parameters. This capability is a natural consequence of prior meta-learning. A further advantage here is that no multi-streaming was needed during training, since ESNs use a batch-learning algorithm and do not suffer from the recency effect. However, during experiments, we had to deal with many practical problems. The major problem in motion control design was training data. It is technically not possible to use random inputs in order to collect training data. A realistic possibility was to drive the robot with different *low* velocities (max 0.5 m/s) in order to avoid slippage of the wheels, and low-frequency inputs to protect the motors. It is clear that using this method, training data will be not rich enough to provide complete information about the robot dynamics. Another problem was the network architecture. Using a relatively large dimension (more than 30 internal neurons), the network lost stability at many times and exhibited sometimes high-frequency oscillations on smooth accelerations. This is due to the high degree of freedom of the closed loop Robot-ESN. With small dimension (say 5–8 internal neurons), oscillations were minimized, but we obtained bad generalization on new data. With 13–18 internal neurons, the ESN could show acceptable results.

For behavior generation, neural fields have been chosen as a framework. They could provide safe trajectories in

presence of static and moving obstacles. Once activated, the field produces a localized peak which decodes a collision-free direction. The possibility to generate only one single peak presents the power of this approach. This property could, for instance, solve some problems observed with methods based on artificial potential fields concept: *oscillations in narrow passages* and *no passage between closely spaced obstacles*. However, neural fields did not solve all problems from which potential field methods suffer. Some experiments (not presented here) show that neural fields did not prevent the robot from being caught in a real local minima, i.e. “trapped” in dead-end situations, and the generated global path is not necessarily the optimal one.

Overall, we have shown that dynamics in RNNs could provide solutions for robot navigation. There are, however, plenty of improvements that could be done in order to achieve better results. Here, only the main points are summarized:

- We are aware of a certain degree of arbitrariness in our choice of the ESN parameters and architecture. Substantial investigation and more experiments on much larger data sets are still needed to ensure that the results we have achieved to date are indeed statistically significant.
- How to guaranty stability? As any RNN, a relatively large dimension ESN lost stability at many times and exhibited sometimes high-frequency oscillations on smooth test signals.

- What is the behavioral capacity of ESNs? In this work, we explored the performance of ESNs to exhibit adaptive behavior with fixed-weights. However, we did not discuss the question whether the obtained fixed-weights ESNs could pass any test in adaptive system. Our network works well only for those situations, for which it has been trained. This may be seen as a limitation relative to explicitly adaptive systems. Does a more efficient training method exist?
- We have provided a control framework based on neural fields theory. Much more efforts must be done to completely integrate movement planning and motor control. One important issue is to integrate the speed control with the temporal evolution of neural activation. A first step towards addressing this issue is to find out what are the optimal neurons activation states before and after performing movements. Further step is to estimate the optimal timing of the movement.

References

1. Kolmanovsky I, McClamroch H (1995) Development in non-holonomic control problems. *IEEE Control Systems* 20–36
2. Fierro R, Lewis F (1998) Control of a nonholonomic mobile robot using neural networks. *IEEE Trans Neural Netw* 9(4):589–600
3. Fukao T, Nakagawa H, Adachi N (2000) Adaptive tracking control of a nonholonomic mobile robot. *IEEE Trans Robot Autom* 16(5):609–615
4. Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. *Int J Robot Res* 5(1):90–98
5. Koren Y, Borenstein J (1991) Potential field methods and their inherent limitations for mobile robot navigation. In: *Proceedings of the IEEE ICRA*, Sacramento, California, pp 1398–1404
6. Amari S (1977) Dynamics of pattern formation in lateral-inhibition type neural fields. *Biol Cybern* 27:77–87
7. Kechriotis G, Zervas E, Manolakos E (1994) Using recurrent neural networks for adaptive communication channel equalization. *IEEE Trans Neural Netw* 5:267–278
8. Chao-Chee K, Lee KY (1995) Diagonal recurrent neural networks for dynamic systems control. *IEEE Trans Neural Netw* 6:144–156
9. Quanmin Z, Guo L (2004) Stable adaptive neurocontrol for nonlinear discrete-time systems. *IEEE Trans Neural Netw* 15:653–662
10. Connor JT, Martin RD, Atlas LE (1994) Recurrent neural networks and robust time series prediction. *IEEE Trans Neural Netw* 5:240–254
11. Harter D, Kozma R (2005) Chaotic neurodynamics for autonomous agents. *IEEE Trans Neural Netw* 16:565–579
12. Williams RJ, Zipser D (1989) A learning algorithm for continually running fully recurrent neural networks. *Neural Comput* 1:270–280
13. Perez-Ortiz JA, Gers FA, Eck D, Schmidhuber J (2003) Kalman filters improve lstm network performance in problems unsolvable by traditional recurrent nets. *Neural Netw* 2:241–250
14. Juergen Schmidhuber (1992) A fixed size storage $O(n^3)$ time complexity learning algorithm for fully recurrent continually running networks. *Neural Comput* 4(2):243–248
15. Werbos PJ (1990) Backpropagation through time: what it does and how to do it. In: *Proceedings of the IEEE*, vol 78, pp 1550–1560
16. Jaeger H (2002) Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach. Technical Report 159, AIS Fraunhofer, St. Augustin, Germany
17. Jaeger H (2001) The 'echo state' approach to analysing and training recurrent neural networks. Technical Report 148, AIS Fraunhofer, St. Augustin, Germany
18. Wenjie D, Huo W, Tso SK, Xu WL (2000) Tracking control of uncertain dynamic nonholonomic system and its application to wheeled mobile robots. *IEEE Trans Robot Autom* 16(6)
19. Oubbati M, Schanz M, Levi P (2005) Kinematic and dynamic adaptive control of a nonholonomic mobile robot using a RNN. In: *Proceedings of the 6th IEEE symposium on computational intelligence in robotics and automation*, pp 27–33
20. Oubbati M, Schanz M, Buchheim T, Levi P (2005) Velocity control of an omnidirectional robocup player with recurrent neural networks. In: *RoboCup*, Osaka, Japan, pp 691–701
21. Vilalta R, Drissi Y (2002) A perspective view and survey of meta-learning. *Artif Intell Rev* 18:77–95
22. Cotter NE, Conwell PR (1990) Fixed-weight networks can learn. In: *International joint conference on neural networks*, vol 3, pp 553–559
23. Feldkamp LA, Puskorius GV, Moore PC (1996) Adaptation from fixed weight dynamic networks. In: *Proceedings IEEE international conference on neural networks*, pp 155–160
24. Feldkamp LA, Puskorius GV, Moore PC (1997) Fixed weight Controller for multiple systems. In: *Proceedings of IEEE international conference on neural networks*, pp 773–778
25. Steven Younger A, Conwell PR, Cotter NE (1999) Fixed weight on line learning. *IEEE Trans Neural Netw* 10(2):272–283
26. Prokhorov D, Puskorius G, Feldkamp L (2001) Dynamical neural networks for control. *IEEE Press*
27. Oubbati M, Schanz M, Levi P (2005) Meta-learning for adaptive identification of non-linear dynamical systems. In: *Proceedings of 20th international symposium on intelligent control*, IEEE, pp 473–478
28. Oubbati M, Schanz M, Levi P (2005) A fixed-weight RNN dynamic controller for multiple mobile robots. In: *Proceedings of 24th IASTED international conference on modelling, identification, and control*, Innsbruck, Austria, pp 277–282
29. Schöner G, Dose M, Engels C (1995) Dynamics of behavior: theory and applications for autonomous robot architectures. *Robot Auton Syst* 16
30. Steinhage A, Schöner G (1997) Self-calibration based on invariant view recognition: dynamic approach to navigation. *Robot Auton Syst* 20:133–156
31. Steinhage A, Schöner G (1997) The dynamic approach to autonomous robot navigation. In: *ISIE97, IEEE international symposium on industrial electronics*
32. Bergener T, Bruckhoff C, Dahm P, Janßen H, Joublin F, Menzner R, Steinhage A, von Seelen W (1999) Complex behavior by means of dynamical systems for an anthropomorphic robot. *Neural Netw* 12(7–8):1087–1099
33. Goldenstein S, Metaxis DM, Large EW (2000) Nonlinear dynamic systems for autonomous agent navigation. In: *Proceedings of the 17th national conference on artificial intelligence*
34. Monteiro S, Bicho E (2002) A dynamical systems approach to behavior-based formation control. In: *Proceedings of the 2002 IEEE ICRA*
35. Dahm P, Bruckhoff C, Joublin F (1998) A neural field approach to robot motion control. In: *Proceedings of the IEEE international conference on systems, man, and cybernetics*, pp 3460–3465
36. Giese MA (2000) Neural field model for the recognition of biological motion. In: *Second international ICSC symposium on neural computation*, Berlin, Germany

37. Edelbrunner H, Handmann U, Igel C, Leefken I, von Seelen W (2001) Application and optimization of neural field dynamics for driver assistance. In: IEEE 4th international conference on intelligent transportation systems, IEEE Press, pp 309–314
38. Erlhagen W, Bicho E (2006) The dynamic neural field approach to cognitive robotics. *J Neural Eng* 3:R36–R54
39. Oubbati M, Schanz M, Levi P (2006) Neural fields for behaviour-based control of mobile robots. In: 8th international IFAC symposium on robot control. Bologna, Italy
40. Oubbati M, Palm G (2007) Neural fields for controlling formation of multiple robots. In: IEEE international symposium on computational intelligence in robotics and automation, pp 90–94
41. Kishimoto K, Amari S (1979) Existence and stability of local excitations in homogeneous neural fields. *J Math Biol* 7:303–318
42. Kalmár-Nagy T, D'Andrea R, Ganguly P (2004) Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle. *Robot Auton Syst* 46:47–64