Natural Dynamics and Neural Networks: Searching for Efficient Preying Dynamics in a Virtual World

Peter András

Department of Psychology, University of Newcastle Newcastle upon Tyne, NE1 7RU, United Kingdom

Eric Postma and Jaap van den Herik

Department of Computer Science/IKAT, Universiteit Maastricht, POB 616, 6200 MD Maastricht, The Netherlands

ABSTRACT

Since the 1970s, the chaotic nature of environmental dynamics of animals is a challenge for computer-science researchers. The intriguing question is how can we develop an effective means for predicting future states? In this paper we assume that animals possess control structures exhibiting chaotic dynamics that can deal with chaotic environmental dynamics. Based on that assumption, we describe a novel model that is characterized by predicting future states through a recursive application of an input-output mapping generated by natural dynamics (the internal chaotic dynamics) and chaotic environmental dynamics. For comparison, we evaluate two neural networks that control the prediction behavior of an agent (for example, an animal) in a dynamical environment. The inspirational source of our simulated environment is the preying dynamics of the frog—fly interactions. First we examine a non-chaotic *direct-prediction network* that predicts future states by mapping the input in a single step to the output. Then we develop and investigate a chaotic *recursive-prediction network* that predicts the future in a recursive manner, using iterations of small-step predictions. Our experiments show that the recursive-prediction network clearly

Reprint requests to: Peter András, Department of Psychology, University of Newcastle Newcastle upon Tyne, NE1 7RU, United Kingdom; e-mail: peter.andras@ncl.ac.uk; postma@cs.unimaas.nl; herik@cs.unimaas.nl

outperforms the direct-prediction network in the sense of the number of accurate predictions. The experimental tasks involved the prediction of future states of a chaotically moving target. The result is also explained in terms of complexity of the underlying function-approximation task. From our observations that recursive-prediction networks are better than direct-prediction networks in dealing with chaotic environments, and from the performance of the recursive-prediction network, we tentatively conclude that our model can serve as a chaotic-deterministic component in the procedure of predicting future states.

KEYWORDS

animat, chaos, control, dynamics, neural network, prediction

1. INTRODUCTION

In nature, adaptation to the complex natural dynamics is a prerequisite for survival. Processes that are characteristic for a natural environment are chaotic and turbulent. Animals employ efficient strategies for the prediction of future events to react adequately (for example, Barnsley, 1993). As a case in point, we see that birds and butterflies cope successfully with unpredictable variations in the amplitude and direction of air streams (Able 1973, Alestam & Hedenström, 1998, Sparks 1999, Spieth et al., 1998). Birds even use wind parameters to determine an optimal flight direction (Able, 1973, Alestam & Hedenström, 1998). Apparently, animals possess special control structures for dealing with environmental dynamics. Modeling such control structures enhances our knowledge of the underlying principles and may lead to novel applications in process control (for example, Balaram, 2000; Battersby, 1997; Weber et al., 1999). In recent years, several models for controlling actions have been proposed (Chown, 1999; Goetz and Walters, 1997; Leow, 1998; Prescott & Redgrave, 1999; Sun & Peterson, 1998, 1999). These methods range from recurrent networks that are applied to behavior modeling (Goetz & Walters 1997) to reinforcement learning that is applied to multi-agent

systems (Sun & Peterson 1998, 1999). A handicap of all these models is that they do not take into full account the chaotic nature of the environment.

This paper presents a novel type of control structure that is inspired by the dynamic reconstruction of chaotic processes (Haykin, 1999). Our approach is to equip the animal (henceforth called 'agent') with internal dynamics that match the external environmental dynamics. To support our model, we analyze briefly the complex dynamics characterizing the flight patterns of the house fly (*Musca domestica*) in the Appendix of this paper. Building upon Freeman's (1987) work, we assume that internal chaotic dynamics are required to cope with the chaotic environment. Below, we begin describing our new type of control structure in the form of a new model mimicking the behavior of the animal agent. Then we compare our new recursive-prediction network with a traditional direct-prediction network. The fundamental difference between the two networks is that recursive-prediction networks function on the basis of chaotic dynamics, whereas direct-prediction networks exhibit fixed-point dynamics. The recursive-prediction networks make long-term predictions by recursive use of small-step intermediate predictions, whereas the direct-prediction networks perform long-term predictions in a single step. The recursive use of small-step predictions enable the recursive-prediction networks to fit the underlying dynamics of the environment. We evaluate the performances of both types of controllers on a prediction task involving a chaotic environment. In addition, we analyze the underlying causes of the results obtained.

Our work has implications for research in situated robotics (Clancey, 1997, Duchon et al., 1998), decision modeling (Bather, 2000), and cognitive-process modeling (Clarck & Grush, 1999). Our approach is a new way of dealing with environmental dynamics through the incorporation of internal matching dynamics. In particular it shows how to create matching dynamics using recursive-prediction networks.

1.1 Related Research and Applications

Our approach is closely related to the approach of system identification applied in the control of non-linear systems (Cichocki & Ubenhauen, 1993; Haykin, 1999; White & Sofge, 1992). Typical system identification methods build explicitly a model of the observed system—for example, an input/output

model that matches the existing functional relation between the inputs and outputs of the system (White & Sofge, 1992). Our method differs from the system-identification approach in that we do not require the explicit identification of the dynamics of the environment (namely, the analog of the controlled non-linear system). Instead, we require only the generation of a matching dynamics. When matching dynamics extract relevant information from the environment, they allow agents to deal with their dynamic environment.

An example of an application domain for modeling and controlling dynamical environments is the control of helicopter rotors. Several papers report on difficulties related to the complex, turbulent airflow dynamics caused by the movements of a rotor (for example, Gulieri & Celi, 1998; Leishman & Bagai, 1998). Our approach is an alternative to explicit dynamical modeling. The generation of matching dynamics of the environment may outperform approaches that are based on explicit dynamical modeling.

1.2 Outline of the Paper

The outline of the paper is as follows. In Sec. 2 we describe the model and the experimental set-up. In Sec. 3 we present the experiments performed and analyses our results. Then in Sec. 4 we examine the structural complexity of the neural networks we used and the causes of the errors that occurred. Finally, Sec. 5 is a summary of our discussion and concludes, amongst others, by stating that internal chaotic dynamics are beneficial in dealing with a chaotic environment.

2. MODEL AND EXPERIMENTAL SET-UP

Our experiment is inspired by the predator-prey behavior of the frog and fly, respectively, and by Beer's (1996) study of an active object-recognition agent. During predating, a frog watches the movements of a fly. The frog captures the fly with its tongue when within reach (Alcock, 1993; 85). The frog typically waits for the approach of the prey in a camouflaging environment with minimal movements until a prey approached near enough to be captured with a rapid tongue movement. By minimizing its movements in an environmental camouflage, the frog largely avoids to be detected by the motion-sensitive visual

systems of flies and other insect preys (Juusola & French, 1997). We assume the erratic flight pattern of the fly to be an expression of the chaotic nature of (a part of) the frog's environment. In the Appendix we provide an analysis of the flight pattern of the house fly as supporting evidence.

2.1 The Model

Our model is composed of two parts: a dynamic environment (the fly) and an agent (the frog). The dynamic environment is a two-dimensional world containing a chaotically moving fly. The agent is a flycatcher that catches its prey by predicting the position of the fly upon reaching the ground level. Fig. 1 illustrates the model.

The flycatcher moves along a line defined as the ground level. The fly moves chaotically in the direction of the ground level. The flycatcher has to predict the future positions of the fly in order to catch the fly when it arrives at the ground level. In our models, the flycatcher predicts the ground-level position of the fly using either a direct-prediction or recursive-prediction neural network[1]. In the following the fly and the flycatcher are discussed in more detail.

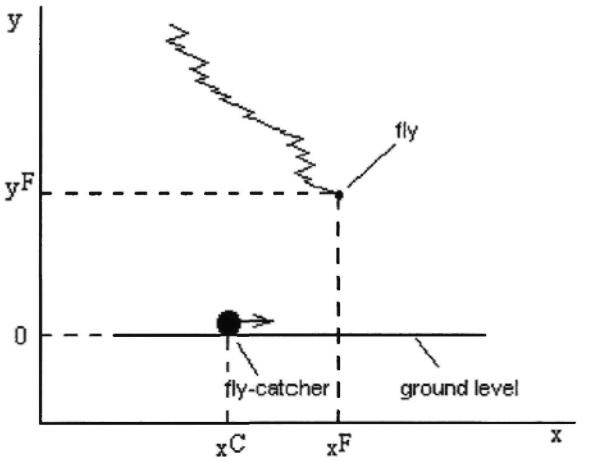

Fig. 1: Illustration of the model containing the fly and the flycatcher. The x^F and y^F are the horizontal and vertical coordinates of the fly's position, and x^C is the horizontal coordinate of the flycatcher's position.

2.2 The Fly

The co-ordinates x^F and y^F represent the position of the fly in a two-dimensional world. Two types of fly movement are defined: random-position (RP) and random-walk (RW) movements[2]. In RP movements, the next horizontal position of the fly (x_{new}^F) is determined by the current position x^F through the logistic map (Hale & Kocak, 1991): $x_{new}^F=\lambda x^F(1-x^F)$. In RW movements, the next horizontal position x_{new}^F of the fly is determined by its current position x^F and its movement ω^F, through the following formula: $x_{new}^F=x^F+\omega_{new}^F$, with $\omega_{new}^F=\lambda(0.25-(\omega^F)^2)-0.5$. The value of the parameter λ is set to a value for which the logistic map exhibits chaotic behavior (Hale & Kocak, 1991). In both the RP and the RW movements, the new vertical position of the fly is defined as $y_{new}^F=y^F-\alpha|x_{new}^F-x^F|$, where y^F is the current vertical position of the fly. The value of α is chosen to have a number of steps that is sufficient for the fly to meet the ground level ($y^F<\theta$, where θ is a small constant). Fig. 2 illustrates both types of fly movements.

To model the random effects of the air stream, the position of the fly is perturbed by a small random component ε, i.e., $x_{effective}^F = x^F + \varepsilon$, with ε having a zero mean normal distribution with variance σ_ε.

2.3 The Flycatcher and Flycatching Strategies

The flycatcher deals with the environmental dynamics by observing the fly and by moving in the appropriate direction. In doing so, it employs a

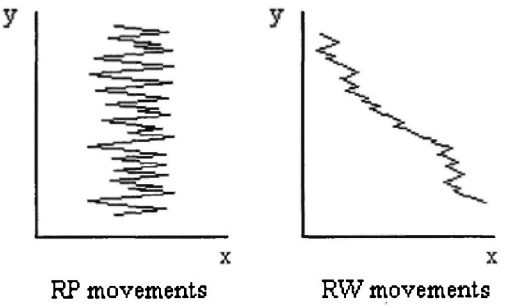

Fig. 2: Illustration of random-position (RP) and random-walk (RW) fly movements.

strategy depending on the type of fly movements (RP and RW). The critical part of our model is the control mechanism of the flycatcher that determines its actions, given the necessary information about the environment. The movements of the flycatcher are determined according to a fly-catching strategy. In both cases the flycatcher uses the available relevant information to predict the landing position of the fly. At each moment of observation it makes a prediction and moves toward the predicted landing position. The difference between the strategies stems from the type of the relevant information (namely, the position of the fly in the RP case and the position and the last move of the fly in the RW case). For RP movements, given the current position of the fly, the expected ground-level position of the fly is predicted, and the flycatcher moves toward the place. For RW movements the flycatcher uses the position and the movement of the fly for predicting its ground-level position and subsequently moves to this place. Below we discuss the control mechanisms in some detail.

2.4 Control Mechanisms

In all cases, RBF-type neural networks were used. Fig. 3 shows an RBF network with five neurons. The networks were trained with standard squared error based training (Haykin, 1994) adapted to the particular control mechanism.

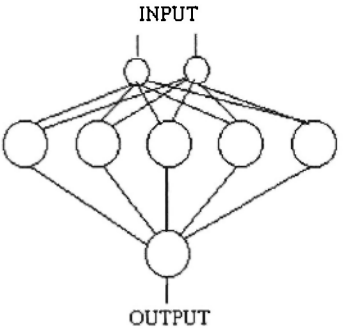

Fig. 3: Illustration of a typical neural network.

For evaluating the ability of both types of networks (direct-prediction and recursive-prediction networks) to deal with environmental dynamics, we defined two control mechanisms. The first control mechanism applies the trained network in a feedforward or direct manner. The second control mechanism applies the trained network recursively. For explanatory reasons, we describe the latter network in iterative terminology.

The direct-prediction network is trained to predict the ground-level position of the fly (x_{ground}^F) given the fly' current position (x^F, y^F) or the fly's current position and movement (x^F, y^F, ω^F) for RP and RW movements, respectively. Using the same inputs, the recursive-prediction network is trained to predict the new horizontal position of the fly (x_{new}^F), instead of the final ground-level position. Both networks are made up of artificial neurons with a Gaussian activation function defined as

$$f(t) = e^{-\frac{||t-c||^2}{2r^2}} \tag{1}$$

with c the center and r the width of the Gaussian function. Using this activation function the direct-prediction network generates a prediction of the form:

for RP movements: $$\hat{x}_{ground,FF}^F = \sum_{k=1}^{n} a_k e^{-\frac{||(x^F, y^F) - c_k||^2}{2r_k^2}} \tag{2}$$

for RW movements: $$\hat{x}_{ground,FF}^F = x^F + \sum_{k=1}^{n} a_k e^{-\frac{||(\omega^F, y^F) - c_k||^2}{2r_k^2}} \tag{3}$$

The c_k-s are two-dimensional vectors, representing the centers of the Gaussian functions.

Fig. 4 shows in graphic form the functioning of the direct-prediction neural network.

For RP movements, the recursive-prediction network generates predictions according to:

$$\hat{x}_{p+1} = \sum_{k=1}^{n} b_k e^{-\frac{(\hat{x}_p - d_k)^2}{2s_k^2}}, \qquad \hat{y}_{p+1} = \hat{y}_p - \alpha |\hat{x}_{p+1} - \hat{x}_p| \tag{4}$$

in which p represents the p-th application of the network to generate a new intermediary output, and d_k is a real number representing the center of the Gaussian function, and

$$\hat{x}_1 = x^F, \qquad \hat{y}_1 = y^F \tag{5}$$

Equation (4) is applied iteratively, until the predicted value of $\hat{y}_p$ is at the ground level, that is,

$$\hat{y}_p < \theta \tag{6}$$

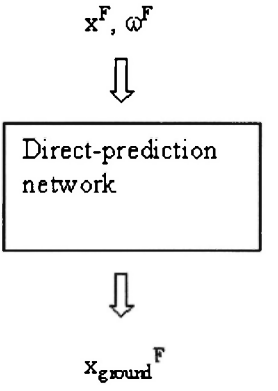

Fig. 4: The prediction mechanism of direct-prediction networks.

Then the prediction of the ground-level horizontal position is

$$\hat{x}^F_{ground,R} = \hat{x}_p \tag{7}$$

Similarly, for RW movements the predictions are generated according to

$$\begin{aligned} \hat{\omega}_{p+1} &= \sum_{k=1}^{n} b_k e^{-\frac{(\hat{\omega}_p - d_k)^2}{2s_k^2}}, \\ \hat{x}_{p+1} &= \hat{x}_p + \hat{\omega}_{p+1}, \\ \hat{y}_{p+1} &= \hat{y}_p - \alpha |\hat{\omega}_{p+1}| \end{aligned} \tag{8}$$

where d_k is the real-valued centre, and

$$\begin{aligned} \hat{\omega}_1 &= \omega^F, \\ \hat{x}_1 &= x^F, \\ \hat{y}_1 &= y^F \end{aligned} \tag{9}$$

The application is iterative and terminates when

$$\hat{y}_p < \theta \tag{10}$$

yielding the predicted ground-level position

$$\hat{x}^F_{ground,R} = \hat{x}_p \tag{11}$$

Figure 5 illustrates the functioning of the recursive-prediction neural network. The flycatcher makes a prediction of the fly's ground-level position at each time step. After making the prediction, the flycatcher moves in the direction of the predicted position, or if it is possible, it moves to the predicted position (namely, if its maximum speed is sufficiently high to move to that place in one time step).

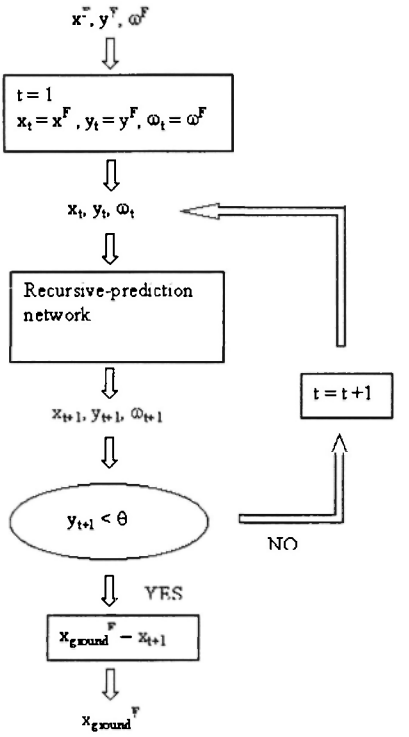

Fig. 5: The prediction mechanism of recursive-prediction networks.

3. EXPERIMENTS AND ANALYSIS

In this section we report our experimental observations in the simulated world of fly and flycatcher. Furthermore, we analyze and interpret these observations. We start by describing the experimental conditions.

3.1 Experimental Conditions

The experiments evaluate the performances of the agents under four experimental conditions. The two control mechanisms (direct-prediction and recursive-prediction networks) are combined with the two types of fly movements (RP and RW movements), making four experimental settings possible. These were assessed for a range of maximum-speed values of the flycatcher.

In all settings flies start from the initial vertical position $y_{initial}^F = 40$. Moreover, in all experiments, the fly-movement parameters discussed in Section 2.2 were set to $\lambda = 3.891$ and $\alpha = 2.5$[3]. The first experiment ignored random effects by setting $\varepsilon = \sigma_\varepsilon = 0$. The initial horizontal position and movement of the flies were set to random values within the interval [0,1]. We defined separate ranges of admissible speed values v for RP and RW movements, i.e., $0.015 < v < 0.05$, and $0.06 < v < 0.2$, respectively (the speed value is defined as maximum distance that the flycatcher can make between two consecutive observations of the fly).

Denoting the position of the flycatcher by x^C, a fly is caught if both $|x^F - x^C| < \rho$ and $y^F < \theta$. Considering the rules defining the RP and RW movements and the initial distance of the flies from the ground level, we can determine the range of the x_{ground}^F values in both cases. We get that in our case for RW movements $x_{ground}^F \in [0,8]$, and for RP movements $x_{ground}^F \in [0,1]$. Taking into account the range of values for x_{ground}^F the ρ value is larger for RW movements than for RP movements, namely, $\rho=0.2$ and $\rho=0.02$, respectively. The fly arrives at the ground level (and in potential reach of the flycatcher) if $y^F < \theta$, with $\theta = 0.5$.

In all settings, the networks contained five neurons, later on established as optimal (see Sec. 4). The networks were first trained and then tested. Each simulation run comprises a complete trajectory of a fly from its starting point to the ground level. At the start of each run, the flycatcher is set to its initial position $x_{initial}^C = 0.5$. We used 1000 runs training and 200 for testing the networks. We measured the percentage of caught flies during the testing phase. Performances are based on averages over twenty replications for each maximum-speed value.

3.2 Experimental Observations

The graphs shown in Fig. 6 (RP movements) and Fig. 7 (RW movements) present the average performances of both networks as a function of (maximum) speed. For both RP and RW movements the recursive-prediction network outperforms the direct-prediction network over the entire range of speed values. The performance of the recursive-prediction network

increases with the speed, whereas the performance of the direct-prediction network is hardly affected by the speed of the flycatcher. The variance of the performances in all cases is much smaller than the average value (less than 20% of the average).

3.3 Analyses and Discussion

We analyze two aspects of all observations. First, we analyze the nature of the prediction processes realized by the two kinds of neural networks. Second, we compare the current observations with those obtained by ever-simpler prediction techniques.

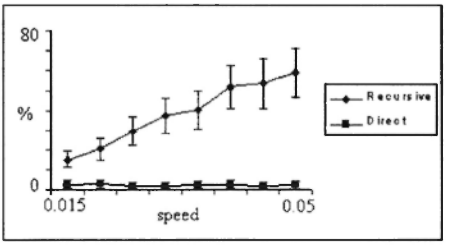

Fig. 6: The average performance of the flycatchers for RP movements. Error bars show 99% confidence intervals.

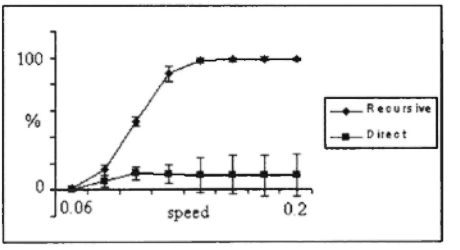

Fig. 7: The average performance of the flycatchers for RW movements. Error bars show 99% confidence intervals.

The nature of the predictions: The results can be interpreted in terms of differences in complexity of the approximation task for both networks. The direct-prediction network tries to approximate the 30^{th} to 40^{th} iteration of the logistic function governing the fly's flight pattern. This function is very complex. It is a polynomial within a degree of more than a million that transforms the [0,1] interval onto itself. As an illustration Fig. 8 shows the 8^{th} iteration that has 255 minima and maxima.

Generally, the p-th iterate of the logistic function has 2^p-1 extreme points within the [0,1] interval for properly chosen λ values, i.e., $\lambda \in (0,4)$.

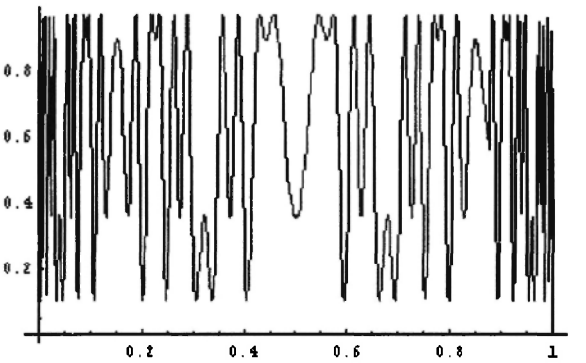

Fig. 8: The graph of the 8^{th} iteration of the logistic function.

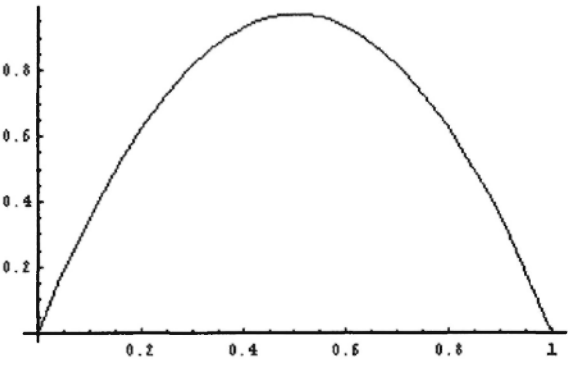

Fig. 9: The graph of the logistic function.

Consequently, a small direct-prediction network may not be able to approximate such a complex function. In contrast, the recursive-prediction network appropriately approximates the simple logistic function, which has a single peak only (see Fig. 9).

One may argue, therefore, that the performance on the fly-catching task is directly related to the complexity of the associated approximation task. The recursive-prediction network predicts the final position by recursive application of the approximating function. For our choice of λ the recursive application of the approximated logistic map behaves chaotically. Consequently, small approximation errors may be amplified yielding a large prediction error.

Apparently, in our experiment the simpler approximation outweighs the amplification effects of the recursive calculations. In other words, the recursive-prediction network splits the prediction task into a separate approximation and prediction task, which turns out to be beneficial for the prediction performance.

Comparison with simple techniques. Hale and Kocak (1991) note that the iterative application of a logistic function with chaotic dynamics yields a distribution of values (namely, the distribution of the values x_t, where $x_0 \in [0,1]$, and $x_{t+1}=\lambda x_t(1-x_t)$) approximately equivalent with the $\beta(1/2,1/2)$ distribution, independent of the initial value. For our study this implies that the conditional expectation of the ground-level position is independent of the initial condition and is equal to the mean value of a random variable with the $\beta(1/2,1/2)$ distribution. Consequently, taking into account the horizontal catching condition ($|x_{ground}^F - x^C| < \rho$), we can calculate the performance of a statistical prediction technique. Applying this technique yields a prediction performance of 8.03% and 21.98% for RP and RW movements, respectively. For high maximum-speed values, the recursive-prediction network outperforms the statistical prediction in all settings. The direct-prediction network performs worse than the statistical prediction for RP movements (<5%). For RW movements, the direct-prediction performance is approximately equal to the statistical prediction performance.

Another way to deal with the prediction task is to make one-step predictions of the position of the fly and then move toward that position. In this case it is possible to predict the new position of the fly very accurately. The problem, however, is that the fly moves much faster than the flycatcher.

The delayed correct response of the agent yields a bad performance. For instance, if the last movement of the fly is large, the flycatcher is not able to move to the appropriate position. As the maximum speed of the agent increases, the performance of the flycatcher increases steadily. Nevertheless, the performance of the flycatcher with one-step predictions and low speed is worse than the performance with final-position prediction. As the speed increases, the one-step prediction flycatchers outperform those with direct-prediction networks, whereas their performance falls short when compared to the performance of the recursive-prediction flycatchers.

We limited our comparisons to simple techniques. We note that there might be other control techniques that might lead to better performance than these simple methods; for example, extended-Kalman filter control (Haykin, 1999). We believe that to cope effectively with complex environmental dynamics, it is essential that the control technique allows the construction of some kind of internal matching dynamics, as is the case in our recursive-prediction networks.

3.4 The Effect of Noise

A potentially disturbing aspect of recursive processes is their sensitivity to initial conditions (Schuster, 1988). As a result, small initial errors may be amplified to large errors. To study the effect of noise on the prediction performances, we added noise to the fly's position. The effect of noise is modeled as:

$$x'^{F}_{new} = x^{F}_{new} + \varepsilon \tag{12}$$

The noise has an additive effect in the short run but gets a highly worthwhile and multiplicative effect in the long run:

$$x^{F}(t+2) = \lambda^2 (x^{F}(t)(1-x^{F}(t)) + \varepsilon(t))(1-\lambda x^{F}(t)(1-x^{F}(t)) - \varepsilon(t)) + \varepsilon(t+1) \tag{13}$$

Figures 10 and 11 show the average performances, in the presence of noise ($\sigma_\varepsilon = 0.2$), of both networks as a function of (maximum) speed, for RP and RW movements, respectively.

We see that although the performance of the recursive-prediction network is altered by the presence of noise, the relative performances of both networks are not changed. The reason for this is that the recursive-prediction networks perform approximation and prediction separately (see Sec. 3.3).

The performance of the direct-prediction networks is hardly affected by noise. This result agrees with the result of our analysis that the direct-prediction networks provide basically a pure statistical estimation of the mean expected arrival position of the fly. The zero-mean noise does not have an effect on the mean arrival position of the fly.

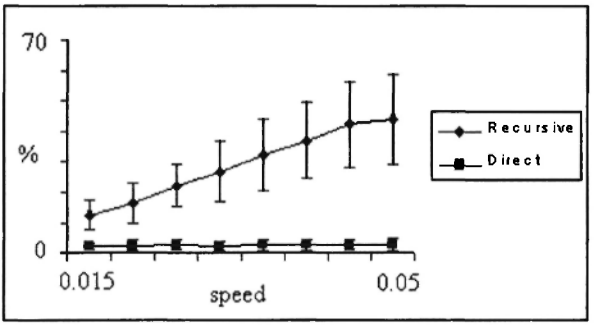

Fig. 10: The average performance of the flycatchers for RP movements of noisy flies. Error bars show 99% confidence intervals.

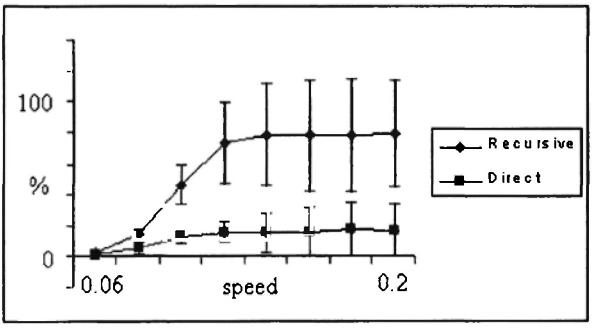

Fig. 11: The average performance of the flycatchers for RW movements. Error bars show 99% confidence intervals.

4. THEORETICAL ANALYSIS

This section provides an analysis of two aspects of our experimental results. First we analyze how the complexity of the neural networks affects the performance of the flycatchers. Second we analyze the performance of the recursive-prediction network in terms of its errors.

4.1 Complexity of Neural Networks

In this subsection we intend to determine the minimal complexity of a neural network fit for our task (fly catching) performed in a chaotic environment. To measure the effect of the complexity of the networks on the fly-catching performance, we systematically vary the number of neurons that represent the basis functions. Each network was tested with 1, 3, 5, 7, and 10 neurons for both types of fly movement. For each network and type of movement we performed five experiments and calculated the average and the variance of the performances. Figures 12 and 13 illustrate the performance of neural networks comprising three (instead of five) neurons. By comparing the graphs of these figures to those of Fig. 6 and 7 (which show the performances for five-neuron networks), the detrimental effect of reducing the network complexity is evident. For both the RP and RW movements, there is an apparent drop in performance.

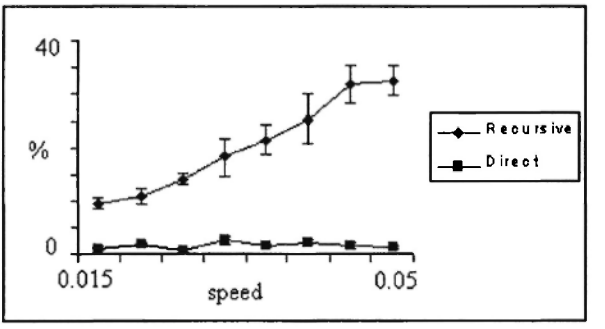

Fig. 12: The average performance of the flycatchers with 3 neurons for RP movements. Error bars show 95% confidence intervals.

Figures 14 and 15 show performances as a function of complexity (number of hidden neurons) for a fixed maximum-speed level, for RP and RW movements respectively. Both graphs illustrate the beneficial effect of increasing the complexity of the recursive-prediction network. With more neurons, the network makes a better approximation of the underlying function. Hence, a better performance is obtained with more complex recursive-prediction networks. For the direct-prediction networks, a positive effect of network complexity could not be established for the range of neurons tested. Apparently, even with 10 neurons, the underlying function is too complex to be approximated[4].

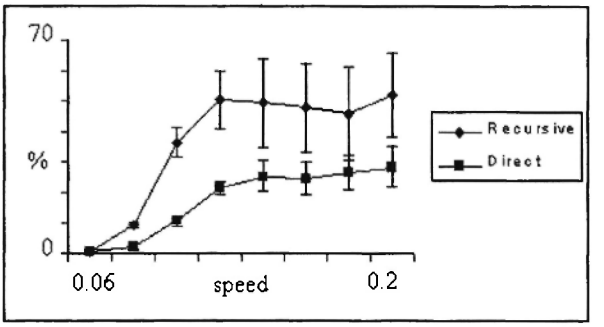

Fig. 13: The average performance of the flycatchers with 3 neurons for RW movements. Error bars show 95% confidence intervals.

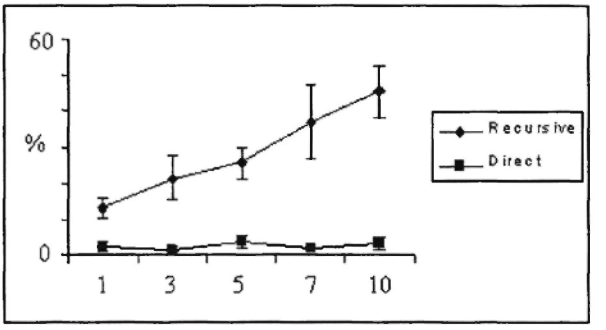

Fig. 14: The performance of flycatchers with varying number of neurons for flies with RP movements, flycatcher speed = 0.025

The graphs shown in Fig. 16 illustrate the dependency of network performance as a function of the speed (horizontal axes) and the number of neurons (vertical axes). Each graph is partitioned into several shaded areas representing levels of performance. Fig. 16A reveals the performance of flycatchers with recursive-prediction neural networks (RP movements) improved by both the flycatcher's speed and the network's complexity. A similar result is observed for RW movements (see Fig. 16B). We note that the top performance is achieved starting from networks with five neurons. This is observable in Fig. 15, as well. The performances for the direct-prediction network (Figs. 16C and 16D for RP and RW movements, respectively) do not reveal such a clear dependency on flycatcher speed and network complexity, although there is a positive effect of increased speed on performance for RW movements.

Taken together the results indicate that the question of finding minimal complexity network structure with high performance has sense only in the case of recursive-prediction networks with RW movements. In this case, the results show that we can achieve the maximal performance with networks with only five neurons, and that the performance does not significantly increase by adding more neurons to the network.

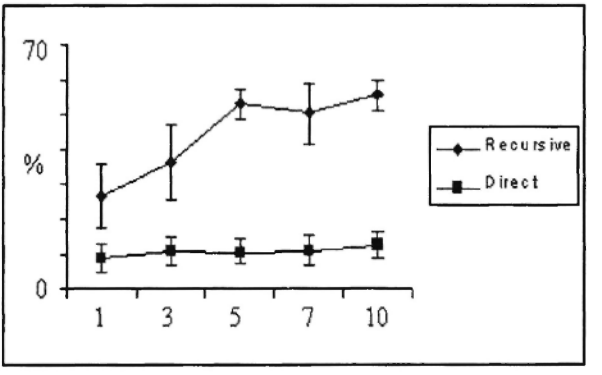

Fig. 15: The performance of flycatchers with varying number of neurons for flies with RW movements, flycatcher speed = 0.1.

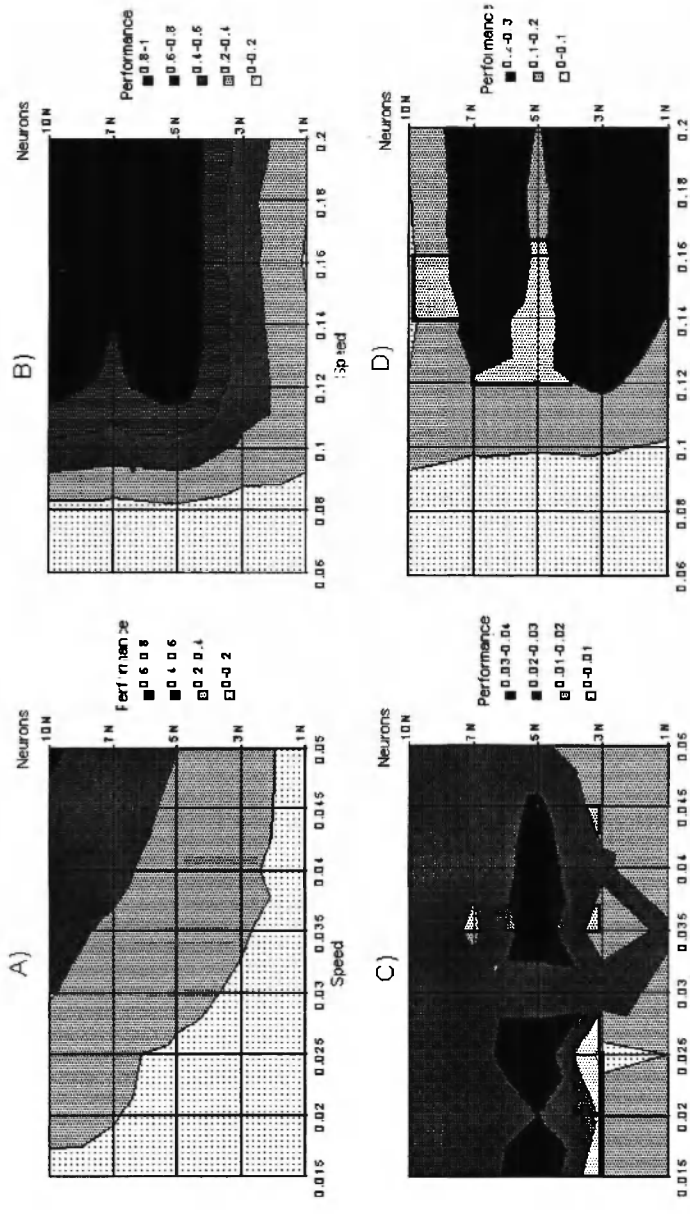

Fig. 16: A, B: Performance of recursive-prediction flycatchers as a function of flycatcher speed and network complexity (number of neurons) for RP (A) and RW (B) movements. C, D: Performance of direct-prediction flycatchers for RP (C) and RW (D) movements. Shaded areas represent similar performances (see legend).

4.2 Error Analysis

In this subsection we analyze the error of recursive prediction networks when approximating a logistic function.

Our analysis employs the following notation.

f(x) the logistic function

f*(x) the approximation of f(x)

$g_n(x) = f^n(x)$ for $n>1$ the iterated logistic function

$g_n^*(x)$ the approximation of g(x)

e_1 the approximation error of f* with respect to f

e_n the approximation error of g_n^* with respect to g_n

The following relations hold.

$$f^*(g_n^*(x)) = f(g_n(x)+e_n)+e_1$$
$$= \lambda g_n(x) - \lambda (g_n(x))^2 - 2\lambda e_n g_n(x) + \lambda e_n + e_1 - \lambda e_n^2$$

Taking into account that $g_n(x) \in [0,1]$ for all $x \in [0,1]$, we get

$$|f^*(g_n^*(x)) - f(g_n(x))| < 3\lambda e_n + \lambda e_n^2 + e_1$$

So, we can write

$$e_{n+1} < 3\lambda e_n + \lambda e_n^2 + e_1$$

There exist numbers e'_n such that $e_n < e'_n$, $e'_1 = e_1$, and we have that

$$e'_{n+1} = 3\lambda e'_n + \lambda e'^2_n + e'_1$$

It is apparent that e'_n grows to infinity with n. Nevertheless, taking into consideration the construction of the functions represented by the neural networks (finite combinations of localized Gaussians), it is evident that the absolute values of recurrent applications of these functions are bound by some value $M > 0$ (practically $M \approx 1$). Consequently, M bounds the values of e_n too.

Considering e'_n values well below M and imposing a more strict limit on the values of e_n, we aim at determining a range of error values e_1 for which e'_n remains significantly below 1 for n = 10 to 15. Numerical simulations show that

to have $e'_{15}<<1$ (ranging from 0.01 to 0.1), the initial error should be $e_1<10^{-15}$. Consequently, if the initial approximation of the logistic function is sufficiently close (namely, $e_1=\max|f^*(x)-f(x)|<10^{-15}$), the 10^{th} to 15^{th} iterate of the approximating function is a good approximation of the iterated logistic function. Of course, e'_n is a coarse approximation of e_n, so less precise approximations of the logistic function may be sufficient. Taking into consideration that neural networks can make arbitrarily good approximations of a continuous function (Park & Sandberg, 1991), we conclude that recursive-prediction networks approximate the iterates of the original approximated function well.

5. CONCLUSIONS AND CONSEQUENCES

In the introduction we stated that dealing with chaotic natural dynamics requires an internal dynamics that matches the environmental dynamics. Thereafter, we presented an experimental set-up that is simple and realistic, contains the relevant natural dynamics and is analyzable. We studied the behaviour of simulated flycatcher animats in our simple model world, in which natural chaos was represented by a chaotically flying fly. We analyzed whether direct-prediction or recursive-prediction neural-network architectures fit best to our simulated dynamic natural environment. Our results show that recursive-prediction networks outperform direct-prediction networks even in the case of stochastic noise. The power of the recursive-prediction networks stems from the knowledge that these networks separate the prediction and the approximation processes, while the latter are not separated in the case of direct-prediction networks. In this way, the recursive-prediction networks build up accurate predictions by using small-step intermediate approximations.

In situated robotics (Clancey 1997), the embedding of an agent in a realistic environment is taken as a starting point for artificial intelligence. Commonly, the animats used in the situated approach incorporate direct-prediction neural networks for the sake of simplicity and ease of analysis (Srinivasan & Venkatesh, 1998). Although our recursive-prediction networks did not appear as evolutionary results of the considered environment, the results presented here show that they offer a better architectural setting to evolve successful solutions in an environment having complex nonlinear

dynamics. Our results suggest that for building anticipating robots, recurrent neural networks are more suitable than non-recurrent types of neural networks. Nevertheless, their application to effective situated robotics requires our results to be extended to recurrent neural networks operating and interacting in real time (cf. Beer, 1996; Freeman, 1987).

Our work also has implications for decision modeling (Bather, 2000) and cognitive process modeling (Clarck & Grush, 1999). Recent proposals suggest that these models should take account of the environmental factors, and particularly environmental dynamics (for example, Kuniyoshi & Berthouze, 1998). In our view, this approach can be done successfully by incorporating generic dynamical components into the models that can develop internal dynamics matching the external dynamics of the environment. In accordance with our results, an effective way to do this is to use recursive-prediction models, namely, the recursive-prediction networks analyzed here. We suggest that decisional and cognitive process models incorporating matching internal dynamics should be validated using real data (for example, data from animals performing decisional tasks) to determine the optimal application and practical limits of the methodology.

We conclude by stating that our approach opens new perspectives on dealing with dynamic environments by agents. Our future work aims at investigating the viability of our approach in more realistic cases. One of our intended ways to do this is to analyze the preying behavior of real animals (for example, real frogs) and to compare that with the results provided by our simulated animats.

APPENDIX

Analysis of the flight pattern of the house fly

The flight of the house fly (*Musca domestica*) is analyzed using a video camera. Flies were captured in a glass box and filmed using a Sony camera (DCR-TRV 110) firmly mounted on a tripod. Using a capture card (FAST AVMaster version 2.2) we digitized four sequences of 60 to 180 seconds of recorded flights. The digitized video sequences were pre-processed by a digital

video-processing program developed by the first two authors. The pre-processing yielded eight sequences of three-dimensional fly movements projected onto a two-dimensional plane. Each sequence comprised 15 to 34 snapshots of fly positions. Typical examples of horizontal and vertical position sequences are shown in Fig. 17 (x positions) and Fig. 18 (y positions), respectively. The position is expressed in pixels and the time in video-capture frames. To analyze the flight pattern, we started by calculating the so-called *difference series* of the flight data, namely, $u'_t = u'_{t+1} - u_t$, where u_t denotes

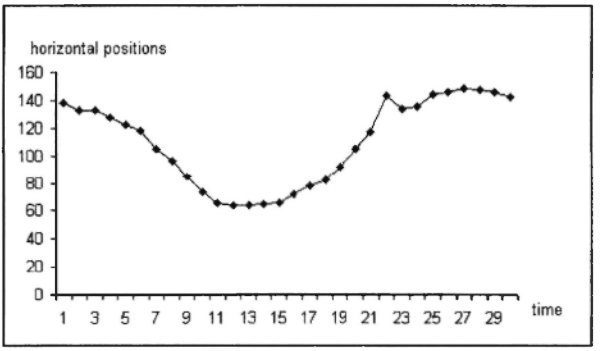

Fig. 17: The horizontal positions of the fly as a function of time (the time is measured in video-capture frames).

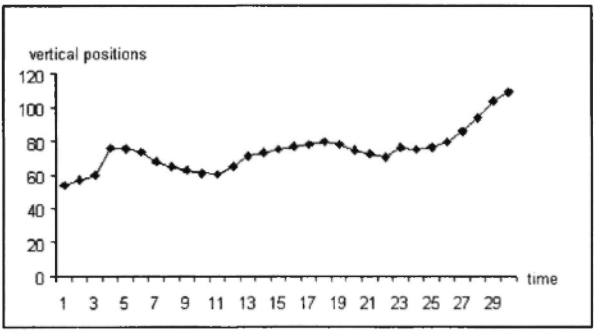

Fig. 18: The vertical positions of the fly as a function of time (the time is measured in video-capture frames).

the position at time t in the original series. Subsequently, we analyzed the difference series by calculating the correlation dimension (Basar, 1990; Granger & Teräsvirta, 1996; Peters 1996) to determine if it contains a chaotic component. If the data series is random than we should see that the correlation dimension increases as we increase the embedding dimension.

If the data series has a dominating linear relationship among consecutive data values, the correlation dimension should saturate[5] at an integer number, that is, the same as the dimensionality of the linear relationship. If the correlation dimension saturates at a non-integer value, it is likely that the data series contain a complex nonlinear relationship between consecutive members of the series[6].

We performed these calculations separately for horizontal and vertical difference series. Furthermore, the effective correlation dimension was determined by calculating the correlation dimension for a range of embedding dimensions. We found that the correlation dimension saturates around 3.4 for the X-values and around 5.4 for the Y-values (see Figs. 19 and 20).

Supported by these results, we suggest that the fly flight data contain a chaotic component. Consequently, the erratic flight pattern of the fly is an example of the chaotic nature of the frog's environment

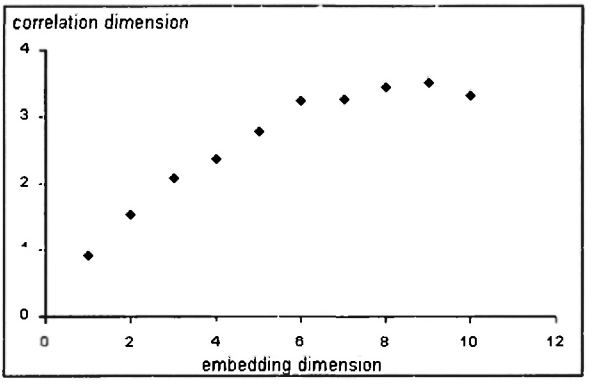

Fig. 19: Correlation dimension of the horizontal position difference series.

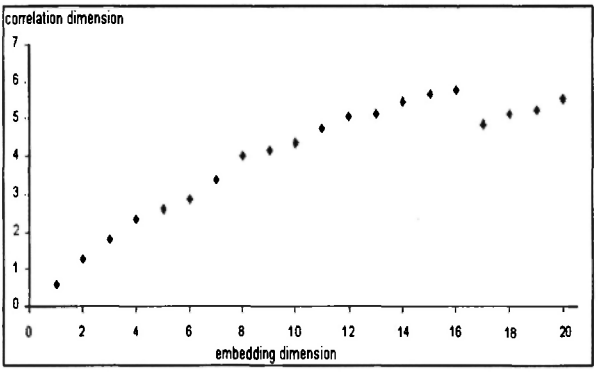

Fig. 20: Correlation dimension of the vertical position difference series.

REFERENCES

Able, K.P. 1973. The role of weather variables and flight direction in determining the magnitude of nocturnal bird migration, *Ecology*, **54**, 1031–1041.

Alcock, J. 1993. *Animal behavior: An evolutionary approach*, Sunderland, Massachusetts, USA: Sinauer Associates.

Alestam, T. and Hedenström, A. 1998. The development of bird migration theory, *Journal of Avian Biology*, **29**, 343–369.

Balaram J. 2000. Kinematic state estimation for a Mars rover, *Robotica*, **18**, 251–262.

Barnsley, M.F. 1993. *Fractals everywhere*, Chestnut Hill: AP Professional.

Bather, J. 2000. *Decision theory : An introduction to dynamic programming and sequential decisions*, New York, NY, USA: Wiley.

Battersby, S. 1997. Mars rover meets rock with complex past, *Nature*, **388**, 215.

Basar, E., editor. 1990. *Chaos in brain function*. Berlin, Germany: Springer-Verlag.

Beer, R.D. 1996. Toward the evolution of dynamical neural networks for minimally cognitive behavior. in: *From animals to animats 4*, edited by Maes, P., Mataric, M., Meyer, J.-A., Pollack, J. and Wilson, S, Cambridge, Massachusetts, USA: MIT Press, 421–429.

Chown, E. 1999. Making predictions in an uncertain world: Environmental structure and cognitive maps, *Adaptive Behavior,* **7**, 17–34.

Cichocki, A. & Ubenhauen, R. 1993. *Neural Networks for Optimization and Signal Processing*, New York: John Wiley & Son.

Clancey, W. 1997. *Situated cognition*, Cambridge, UK: Cambridge University Press.

Clarck, A. and Grush, R. 1999. Towards a cognitive robotics, *Adaptive Behavior,* **7**, 5–16.

Duchon, A.P., Warren, W.H. and Kaelbling, L.P. 1998. Ecological robotics, *Adaptive Behavior,* **6**, 473–508.

Freeman, W.J. 1987. Simulation of chaotic EEG patterns with a dynamic model of the olfactory system, *Biological Cybernetics,* **56**, 139–150.

Goetz, P. and Walters, D. 1997. The dynamics of recurrent behavior networks, *Adaptive Behavior,* **6**, 247–284.

Granger, C.W.J. and Teräsvirta, T. 1996. *Modelling Nonlinear Economic Relationships*, Oxford, UK: Oxford University Press.

Gulieri, G. and Celi, R. 1998. Some aspects of helicopter flight dynamics in steady turns, *Journal of Guidance, Control and Dynamics,* **21**, 383–390.

Hale, J.K. and Kocak, H. 1991. *Dynamics and Bifurcations*. New York: Springer-Verlag.

Haykin, S. 1999. *Neural Networks* (second edition), Upper Saddle River, New Jersey, USA: Prentice Hall.

Juusola, M. and French, A.S. 1997. Visual acuity for moving objects in first- and second-order neurons of the fly compound eye, *Journal of Neurophysiology*, 77, 1487–1495.

Kuniyoshi, Y. and Berthouze, L. 1998. Neural learning of embodied interaction dynamics, *Neural Networks* **11**, 1259–1276.

Leishman, J.G. and Bagai, A. 1998. Challenges in understanding the vortex dynamics of helicopter rotor wakes, *AIAA Journal,* **36**, 1130–1140.

Leow, W.K. 1998. Computational studies of exploration by smell, *Adaptive Behavior,* **6**, 411–434.

Park, J. and Sandberg, I.W. 1991. Universal approximation using radial-basis function, *Neural Computation*, **3**, 246–257.

Peters, E.E. 1996. *Chaos and order in the capital markets*, New York, NY, USA: John Wiley and Sons.

Prescott, T.J. and Redgrave, P. 1999. Layered control architectures in robots and vertebrates, *Adaptive Behavior,* 7, 99–128.

Schuster, H.G. 1988. *Deterministic chaos* (2nd edition), Berlin, Germany: Springer-Verlag.

Sparks, T.H. 1999. Phenology and the changing pattern of bird migration in Britain, *International Journal of Biometeorology,* **42**, 134–138.

Spieth, H.R., Cordes, R.-G. and Dorka, M. 1998. Flight direction in the migratory butterfly *Pieris brassicae*: Results from semi-natural experiments, *Ethology,* **104**, 339–352.

Srinivasan, M.V. and Venkatesh, S., editors. 1997. *From living eyes to seeing machines*, Oxford, UK: Oxford University Press.

Sun, R. and Peterson, T. 1998. Autonomous learning of sequential tasks: Experiments and analyzes, *IEEE Transactions on Neural networks,* **9**, 1217–1234.

Sun, R. and Peterson, T. 1999. Multi-agent reinforcement learning: weighting and partitioning, *Neural Networks,* **12**, 727–753.

Weber, K., Venkatesh, S. and Srinivasan, M.V. 1999. Insect-inspired robot homing, *Adaptive Behavior,* 7, 65–98.

White, D.A. and Sofge, D.A., editors. 1992. *Handbook of intelligent control. Neural, fuzzy and adaptive approaches*, New York, NY, USA: Van Nostrand Reinhold.

Notes:

[1]For the sake of simplicity we kept the vertical movement of the fly as a simple function of its horizontal movement. In a more complex case the fly's vertical position could vary similarly as its horizontal position in our simulation, and the flycatcher could be required to make correspondingly more complex predictions.

[2]We note that the use of the word 'random' does not mean that these movements are truly random, but that their distribution is similar to that of a random series.

[3]The λ determines the chaotic nature of the flight patterns of the simulated fly. If λ is close to 4 and $\lambda<4$ the flight pattern is likely to be chaotic. The $\lambda=3.891$ is a particular choice which gives chaotic flight patterns. For other values of λ the flight pattern might have a periodic or fixed point dynamics. For details on the dependence of the chaotic nature of the logistic function on the λ parameter see Hale and Kocak (1991). The α determines how many iterations are before the fly hits the ground level. The bigger the α, the less iterations are calculated, and the easier are the predictions.

[4]Looking at Fig. 8, we see how complicated is the underlying function after only 8 iterations. This suggests that we need a very large number of neurons in the network to have a good direct approximation of such iterated functions. Furthermore, the number of necessary neurons depends on the number of iterations. Too many neurons may lead to over fitting, if the number of practical iterations turns to be

smaller than the ideal one. Too few neurons may lead to low performance if the number of practical iterations is higher than the ideal one. As the number of practical iterations may vary, it is hard to find a good direct-prediction network for this task, even if we do not limit the number of the neurons of the network.

[5]The correlation dimension is said to saturate if it does not change with increasing embedding dimension. The value of the correlation dimension at the saturation is defined as the effective correlation dimension.

[6]We note that this test is not an absolute test, and it is possible to construct counterexamples for the method (Basar 1990).