The dynamic wave expansion neural network model for robot motion planning in time-varying environments

Dmitry V. Lebedev, Jochen J. Steil, Helge J. Ritter

October 11, 2004

We introduce a new type of neural network - the dynamic *wave expansion neural network* (DWENN) - for path generation in a dynamic environment for both mobile robots and robotic manipulators. Our model is parameter-free, computationally efficient, and its complexity does not explicitly depend on the dimensionality of the configuration space. We give a review of existing neural networks for trajectory generation in a time-varying domain, which are compared to the presented model. We demonstrate several representative simulative comparisons as well as the results of long-run comparisons in a number of randomly-generated scenes, which reveal that the proposed model yields dominantly shorter paths, especially in highly-dynamic environments.

List of symbols

$C \subset \Re^N$, robot's configuration space;
N, the number of robot's degrees-of-freedom;
s_i, the neighbourhood of the i-th neuron in the network field;
x_i, activity level of neuron i;
I_i, the external input of neuron i;
$g(.)$, the transfer function of a neuron;
δ_{ij}, the Kronecker symbol;

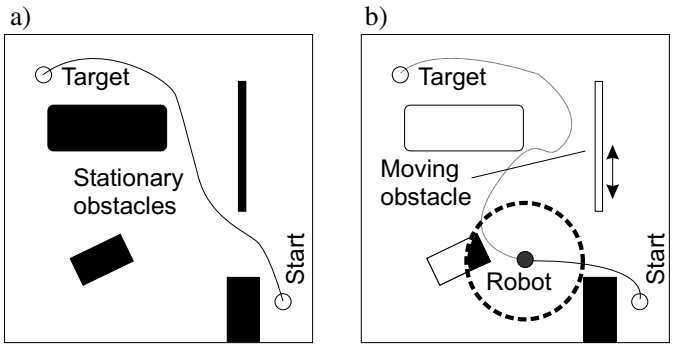

Figure 1: The path planning problem at two extremes: a) The simple problem: to find a path from the start to the target in a stationary environment with a given description of obstacles. b) The complex problem: the environment is initially unknown and the information about (potentially moving) obstacles is acquired during the motion. To get a reasonable path in a time-varying environment and to escape possible local minima effects, this information has to be efficiently integrated into the path planning process.

1 Introduction

One of the major challenges in the development of intelligent robotic systems is endowing them with an ability to plan motions and to navigate autonomously. This ability becomes critical particularly for robots which operate in dynamic environments, where unpredictable and sudden changes may occur. Whenever the robot's sensory system detects a dynamic change, its planning system has to adapt the path accordingly. Prominent examples are real world environments that involve interaction with people, like museums, shops, or households. Usually, it is required that the path of a robot is *safe* (i.e. collision-free), *optimal* or close to optimal, and *natural*, i.e., in a complex situation the robot does not get lost and goes far away from its destination.

Different types (and complexity levels) of the path planning problem can be distinguished (Figure 1). The simplest problem is, given the exact description of the environment, to find a continuous path from a starting location to a target location. There exists a number of global approaches, such as decomposition, road-map, and retraction methods (Latombe, 1991; Hwang & Ahuja, 1992; Henrich, 1997), randomised approaches (Kavraki, Svestka, Latombe, & Overmars, 1996; Barraquand et al., 1997; Song, Thomas, & Amato, 2003), genetic algorithms (Paredis & Westra, 1997; Mazer, Ahuactzin, & Bessiere, 1998; Eldershaw & Cameron, 2000), as well as several local approaches, e.g., potential field methods (Khatib, 1986; Barraquand & Latombe, 1991) to solve this problem. Usually, global approaches require a preprocessing stage, during which a graph structure containing the information about the connectivity of the robot's free space is formed, before the path search can be performed. Local methods need some heuristics, as, e.g., the estimation of local gradients in a potential field to provide an effective path search.

If the environment is *dynamic* (i.e., if obstacles and/or the target are moving), then two cases are possible. If trajectories of obstacles are *known* in advance and the robot dynamics is not considered (like for free-flying objects), the problem is reduced to the stationary case by adding the time axis to the planning space (moving obstacles become stationary in the new space) (Latombe, 1991). There exist also a number of methods, which account for the constraints on the robot dynamics during the planning (see, e.g., (Fraichard & Laugier, 1993; Fiorini & Shiller, 1998; Hsu, Kindel, Latombe, & Rock, 2002)). For the most complex case, when obstacle placements/trajectories are *unknown* in advance, there

exist much fewer approaches. Obstacles in that case are detected locally during the robot movement and are dynamically incorporated into the path generation process, what often makes global approaches with replanning computationally demanding. In (Zelinsky, 1992), for instance, the whole path is replanned from scratch each time the robot bumps into an obstacle. Stentz (1995) and Koenig and Likhachev (2002) proposed graph-search algorithms which utilise the information from previous searches to accelerate the replanning. The algorithms in (Lumelsky & Stepanov, 1986) guarantee to find a path to the target (if one exists) in an unknown stationary environment based on the local "tactile" input. Other approaches (Miura, Uozumi, & Shirai, 1999; Yu & Su, 2001; Bennewitz, Burgard, & Thrun, 2003) try to predict and to approximate the movement of obstacles in the workspace, what reduces the problem to the previous case. Several neural network models for path generation in a non-stationary environment have been proposed, which are surveyed in Section 2 and evaluated in simulations in Section 4. Generally, the local nature of these methods allows to integrate the information about changes in the environment into the path generation process in an efficient way, such that real-time planning is possible in many situations.

In this paper, we present a novel type of neural network – the dynamic wave expansion neural network (DWENN) – which is capable of generating dynamic distance potentials for real-time path planning in a time-varying environment. This model can be applied to all aforementioned types of the path planning problem. The underlying idea of the DWENN algorithm is to organise wave propagation in a way similar to waves in water spreading, for instance, around a dropped stone. The neurons of the network are arranged in a regularly discretised lattice. In our model a scalar potential field is formed by repetitively generated waves of neural activity, which originate from the target location. Each subsequent wave "brings" an updated distance information from the target, and increases the potential of lattice nodes in such a way that farther (from the target) neurons accumulate larger activity values. If at some instance of time a location is not reached by the actual wave front, it is regarded as untraversable for the robot.

To prevent local minima problems, in our model the propagation of inhibitory waves (waves of zero activity) is triggered in particular situations to temporarily interrupt the planning process, and thus to avoid undesired path oscillations. The robot then waits several time steps until a new activity wave reaches its position from an appropriate direction, and then continues to move. Thus, no replanning from scratch is needed, since the potential field adapts to changes in the environment dynamically and rapidly. The DWENN's update rules are computationally very efficient, and its state equations are *parameter-free*. Preliminary versions of the model have been reported in (Lebedev, Steil, & Ritter, 2002, 2003b, 2003a). In (Lebedev et al., 2003b), we have shown that DWENN can be viewed (with minor simplifications) as a dynamic version of the distance transform algorithm (Zelinsky, 1992), used for path planning in stationary environments.

The paper is organised as follows. Section 2 provides a taxonomy and review of existing neural network approaches for path planning with particular attention to neural network models for trajectory generation in a time-varying domain. In Section 3 we describe the proposed DWENN model and analyse its dynamics. Comparative simulation studies and a complexity analysis are presented in Section 4, and, finally, conclusions are discussed in Section 5.

2 Review of neural network models for path planning

Table 1 summarises most of the existing neural network models for path planning. They are ordered along two main axes: (i) with respect to the environment type (stationary

Table 1: Neural network models for path planning.

Authors	Network type	Dynamics	Repres.	Type
(Ageev & Istratov, 1998)	multi-layer, feed-forward	gradient minimiz. method	algebraic, explicit	stationary
(Lee & Kardaras, 1997)	multilayer network	simulated annealing	algebraic, explicit	stationary
(Meng & Picton, 1992)	single hidden layer back-propagation	learns collision penalties	algebraic, explicit	stationary
(Vleugels et al., 1997)	Kohonen-type, with 2 classes of nodes	road-map construction	algebraic, explicit	stationary
(Xia & Wang, 2000)	discrete-time, recurrent	linear optimization	algebraic, explicit	stationary
(Dracopoulos, 1998)	multilayer perceptron	learns to predict movement direction	grid-based	stationary
(Kassim & Vijaya Kumar, 1995)	two layers, locally connected	wave fronts propagation	grid-based	stationary
(Kindermann et al., 1996)	three layers, with local reccurent connections	diffusion-like activity propagation	grid-based	stationary
(Lemmon, 1991)	single layer, locally connected, oscillatory dynamics	produces oscillatory behavior	grid-based	stationary
(Siemiatkowska & Dubrawski, 1998)	cellular, two layers	diffusion-like activity propagation	grid-based	stationary
(Bugmann et al., 1995), (Tarassenko et al., 1991)	one or two layers, locally connected	diffusion-like activity propagation	grid-based	stationary, dynamic
(Chen et al., 2003)	Hopfield-type, topologically organised, locally connected	diffusion-like activity propagation	grid-based	stationary, dynamic
(Glasius et al., 1995)	Hopfield-type, topologically organised, locally connected	diffusion-like activity propagation	grid-based	stationary, dynamic
(Yang & Meng, 2000)	single layer, topologically organised, locally connected	diffusion-like activity propagation	grid-based	stationary, dynamic

vs. dynamic), and (ii) with respect to the environment representation (algebraic vs. grid-based). Since DWENN is a model for fast planning in a non-stationary domain, we review for further reference and comparison the models in the second part of Table 1.

These models generate scalar potentials over a distributed representation of the configuration space of the robot. Such potentials are an efficient alternative to analytically-described potential fields (Khatib, 1986; Rimon & Koditschek, 1992; Li & Bui, 1998; Wang & Chirikjian, 2000) because of their easy implementation and high performance. An attempt to draw analogies between potential fields and neural networks was made in (Liu & Khatib, 2002).

In all models surveyed below, the path planning process is performed in the robot's *configuration space* C, a regularly discretised hypercube in $\Re^N$, where N is the number of degrees-of-freedom of the robot (Lozano-Perez, 1983). Each discrete position in C is associated with a formal neuron. Each neuron i is connected to its neighbours within a certain radius, which comprise its neighbourhood s_i. The neural networks have a locally connected, highly parallel architecture, such that they can be realized in a distributed fashion as parallel processes (see, e.g., (Shu & Buxton, 1995)). All neurons together comprise the *network field*.

The existing models have in common that their dynamics perform an averaging of the potentials of their local neighbouring neurons. In all these models the initial activity potential associated with the target location is distributed through the network field in such a way that activity of all neighbours of a neuron is used to compute its potential. Neurons associated with obstacle locations receive typically a negative value on their external inputs, and have, therefore, smaller potential values. Each path step in that case is done in the direction of the neighbour with the maximal activity value, what leads the robot to the target. As we show in Section 3, our newly introduced model follows a different idea of *selective* activity propagation, and in many situations there is no need to query all neighbours of a neuron to calculate its activity level, what results in a more efficient update rule.

2.1 The resistive grid model

A resistive grid is represented as a regular lattice, in which nodes are associated with neurons. Each neuron is connected to its $2N$ closest neighbours, where N is the dimensionality of the state space. The evolution of the i-th node is given in discrete time by

$$x_i(t+1) = I_i + \frac{1}{2N} \sum_{j \in s_i} x_j(t) \ ,$$

where I_i is the external current applied to the node.

The resistive grid model is based on a numerical approximation of a solution $\varphi(x, y)$ of the Laplace equation: $\partial^2\varphi/\partial x^2 + \partial^2\varphi/\partial y^2 = 0$ (in $2D$). By expansion of φ into a Taylor series with a small discretisation step for each variable, it can be shown that in the first-order approximation the potential value of a grid node is equal to the average sum of the potential values of its immediate neighbours (Connolly, Burns, & Weiss, 1990; Karnik, Dasgupta, & Eswaran, 2002; Alvarez, Alvarez, & Gonzalez, 2003).

In (Bugmann et al., 1995), a cellular automaton is used to calculate resistive grid potentials and the influence of Dirichlet and von Neumann boundary conditions on route generation is analysed.

2.2 The model by Glasius et al.

Glasius et al. (1995); Glasius, Komoda, and Gielen (1996) proposed a discrete-time Hopfield-type neural network, whose dynamics is described by

$$x_i(t+1) = g\left(\sum_{j \in s_i} w_{ij} x_j(t) + I_i\right), \qquad w_{ij} = \begin{cases} e^{-\gamma |i-j|^2}, & \text{if } |i-j| \le r \\ 0, & \text{if } |i-j| > r \end{cases},$$

where w_{ij} are symmetric connections weights, $|i-j|$ is the Euclidian distance between corresponding grid nodes, $\gamma, r > 0$ are scalars, and g is the transfer function. The external input I_i encodes the information about the target and obstacle positions on the network field and is given by

$$I_i = \begin{cases} v, & \text{i - target} \\ -v, & \text{i - obstacle} \\ 0, & \text{else} \end{cases}, \quad v \ge 1.$$

Since any monotonically-increasing function can be used as the transfer function g (Glasius et al., 1995), a piecewise linear transfer function was chosen:

$$g(x) = \begin{cases} 0, & x \le 0 \\ \beta x, & x \in [0,1] \\ 1, & x \ge 1 \end{cases}, \quad \text{and} \quad \beta > 0.$$

2.3 The model by Yang et al.

Yang and Meng (2000, 2001, 2003) proposed a continuous-time dynamics, which is derived from the Grossberg's shunting model (Grossberg, 1988). The dynamics of neuron i is given by

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)\left([I_i]^+ + \sum_{j \in s_i} w_{ij}[x_j]^+\right) - (D + x_i)[I_i]^-,$$

where the parameters A, B and D are the passive decay rate, the upper and lower bounds of the neural activity, accordingly, and the neural activity $x_i \in [-D, B]$. The function $[a]^+$ is defined as $[a]^+ = max\{a, 0\}$, whereas the function $[a]^-$ is given by $[a]^- = max\{-a, 0\}$. The symmetric connection weights are defined by

$$w_{ij} = \begin{cases} \mu/|i-j|, & \text{if } 0 < |i-j| \leq r \\ 0, & \text{if } |i-j| > r \end{cases},$$

where $|i-j|$ is the Euclidean distance between corresponding lattice nodes, μ and r are positive constants. Note, that some additional efforts for tuning and selection of proper network parameters are required for this model.

2.4 The model by Chen et al.

While in the model by Glasius et al. (1995) the decay rate is fixed and equal to one, Chen et al. (2003) introduced recently a similar model, where the decay rate $A > 0$ may be chosen arbitrarily:

$$\frac{dx_i}{dt} = -Ax_i + D_i \cdot m \cdot \sum_{j \in s_i} w_{ij} x_j + I_i.$$

Here m is a positive gain coefficient, $D_i = 0$ for the obstacles, and $D_i = 1$, otherwise. The external input I_i is positive only for the target neuron, and is zero, otherwise. The connection weights $w_{ij} = 1$ if network neighbourhoods are 4-connected, while $w_{ij} = (8m/A)^{\sqrt{2}-1} < 1$ if the latter are 8-connected (therefore, it is required that $8m < A$).

3 The dynamic wave expansion neural network (DWENN)

In the DWENN's network field waves of neural activity are spread around the neuron associated with the target location. At each instance of time a new wave emanates from that point and carries the information about the distance to the target, i.e., it propagates in such a way, that neurons associated with farther locations accumulate larger activity values. The update rule employs one addition and some binary checks only, it is parameter-free, and involves only integer-valued computation. Hence, the activity propagation is computationally efficient and makes real-time planning possible.

At each time step, a neuron inherits (and increments by two) the activity from a neighbour, which is (i) closer to the target neuron, (ii) is not an obstacle neuron, and which belongs to (iii) the active (i.e., this neighbour has a positive activity value) and (iv) actual (i.e., this neighbour has changed its activity level at the previous time step) wave front. If for a neighbour all the conditions (i)-(iv) are satisfied, then the corresponding connection weight becomes equal to one, while the connection weights from all other neighbours become (or stay) zero.

The activities of all neurons constitute a scalar potential field, in which the minimal positive value is always at the target location (the zero state is the prohibited state in the potential field). The robot is globally "attracted" by the target, and starts to move as soon as the first wave front reaches its initial position. It can move only to the neighbouring location, from which the activity has been inherited. This ensures that robot path steps are safe, and the path tends to be L_1-optimal.

3.1 The network dynamics

Using the definitions from Section 2, each neuron i is connected to its set $s_i = \{i_1, ..., i_n\}$ of neighbours, for which we assume an arbitrary and (possibly) fixed enumeration (Figure 2a).

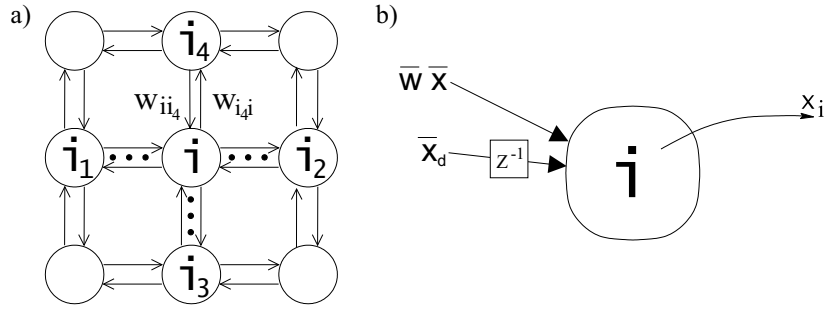

Figure 2: The network architecture: a) Network neighbourhood in $2D$. Neurons are locally connected and arranged on a regular lattice. Neurons in local neighbourhoods are arbitrarily enumerated. b) The model of the neuron. Each neuron is characterised by its activity level x_i, which depends on its own state and on the state of its neighbours at the current and the previous time step.

Figure 2b shows the neuron model.

The DWENN model can be viewed as a discrete-time dynamical system. The activity of neuron i at the moment of time $t+1$ will depend on the current activities of its neighbours (the vector $\vec{x} = (x_{i_1}(t), .., x_{i_n}(t))$) as well as on the activities of its neighbours and its own activity at the previous time step (the vector $\vec{x}_d = (x_i(t-1), x_{i_1}(t-1), .., x_{i_n}(t-1))$). Furthermore, neuron i is active, if $x_i > 0$, and inactive, otherwise.

Initially, the activity values of all neurons and all connection weights are zero. Let $i_*(t)$ define the index of the target neuron at time t. Three classes of neurons are distinguished with dynamics given by:
(i) for the target $(i = i_*(t))$:

$$x_i(t+1) = 1, \tag{1}$$

(ii) for its direct neighbours $(i \in s_{i_*(t)})$:

$$x_i(t+1) = \begin{cases} x_i(t) + 1, & \text{if } i_*(t+1) = i_*(t) \\ 2, & \text{otherwise} \end{cases}, \tag{2}$$

(iii) for all other neurons

$$x_i(t+1) = \sum_{j \in s_i} w_{ij}(t) \cdot (x_j(t) + 2). \tag{3}$$

At each time step, before updating the activity level of neuron i, the corresponding connection weights are determined in accordance with:

$$w_{ij}(t+1) = \begin{cases} \delta_{jk}, & \text{if } k \in s_i \text{ is the } \textit{first} \text{ neuron,} \\ & \text{for which (a)-(d) below hold ,} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where δ_{jk} is the Kronecker symbol, and the corresponding conditions for neuron k are:
(a) k is not an obstacle,
(b) $x_k(t) > 0$, i.e., it is part of the activity wave front and carries *some* information about the changes in the workspace,
(c) $x_k(t) \neq x_k(t-1)$, i.e., it carries *new* information,
(d) if $(x_i(t) + x_i(t-1)) > 0$, then $x_k(t) < x_i(t)$ must hold, i.e., its location is closer to the target than the robot's current position.

Note, that the neighbours of neuron i are considered with respect to some preassigned ordering.

Equations (1) assures that the neuron at the (potentially moving) target always has the minimal activity value in the neural field. The weights update rule (4) enforces that for each neuron at most one connection weight $w_{ij^*} = 1$, where j^* is selected in accordance with the rules (a)-(d), and $w_{ij} = 0$ for all others $j \in s_i$.

The rules (a)-(d) determine the *selective* flow of neural activity because the weight $w_{ik}(t+1)$ is equal to one, if and only if the neuron k has changed its state at the previous evolution step (rule c)), if at time t it is active (rule b)), and is not associated with an obstacle (rule a)), and if its activity level is less than that of neuron i, i.e., if neuron k is closer to the target neuron with respect to the actual global distribution of the potentials (rule d)).

The following properties can immediately be obtained from DWENN's dynamics equations (1, 4):

Property 1. If the first wave front reaches neuron x_i at the time step t_k, then this neuron becomes active with value $x_i(t_k) = (2t_k - 1)$;

Property 2. If there exists a positive weight w_{ij} for neuron i, then this weight indicates the gradient direction in the potential field: $w_{ij} > 0 \Rightarrow x_j < x_i$;

Property 3. The activity level of neuron i is bounded by the number n of network iterations: $x_i(t) \leq n$;

Property 4. If an active neuron i has become inactive, then it will stay inactive at the following time step. Indeed, if $x_i(t-1) > 0$ and $x_i(t) = 0$, then $x_k(t) < x_i(t)$ is always false, therefore, the condition (d) in (4) is also false, and $\forall\, j \in s_i : w_{ij} = 0$, and, hence, $x_i(t+1) = 0$.

Further, if the target is stationary:

Property 5. If an active neuron remains active, then its activity level is increased by one at each time step: $x_i(t) > 0 \Rightarrow x_i(t+k) = x_i(t) + k$;

Property 6. If neuron i became active at time t_i and neuron j at time $t_j > t_i$, then $x_i(t) < x_j(t)$ for all $t \geq t_j > t_i$.

When the robot is at the configuration represented by neuron i, its next path step is done in the direction indicated by the only non-zero weight, i.e. to the configuration represented by neuron j if $w_{ij} > 0$ (or j is the target neuron). From Property 2, each path step of the robot is done along the gradient descent direction in the potential field. Consequently, the resulting path tends to be optimal in a L_1 metrics.

The preassigned and fixed enumeration of the neighbours results in a tendency to generate paths consisting of straight lines. Therefore, it is reasonable to query first the same neighbour from which the activity at the previous time step has been inherited, since the probability that a new wave will arrive along this direction is high.

3.2 Obstacle avoidance and global adaptation in dynamic environments

In dynamic environments, a fast reconfiguration of the potential field is possible because active neurons can temporarily become inactive to signal dynamic changes (Property 4). This key property is encoded in Condition (d) in (4). An inactive neuron, therefore, may initiate the propagation of an *inhibitory wave*. This is illustrated in Figure 3a, where the initial placement of two obstacles is shown.

The neuron, which is behind the gate (the position of this neuron is denoted by the black border), receives new activity only through the gate and all its other neighbours have a larger activity (Figure 3b). Thus, when the gate is closed, this neuron becomes inactive (Property 4). This triggers the generation of an inhibitory wave, which sequentially

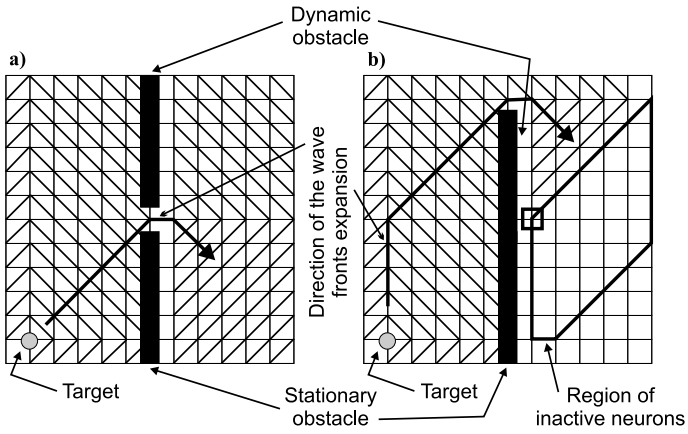

Figure 3: Global adaptivity of the potential field to environmental changes by means of inhibitory waves. a) Propagation of wave fronts around the target in a $2D$ workspace. b) Rapid adaptation to the new placement of the dynamic obstacle. The neuron shown by a black border becomes inactive and inhibits (inactivates) other neurons. Inactive neurons become active after a new wave front has reached them again.

inactivates neurons behind the gate until they receive new activity carried by a wave front passing through the new gate position (Figure 3b).

The same inhibitory mechanism provides a natural behaviour for avoidance of dynamic obstacles appearing on the robot's path. This is illustrated in Figure 4 for a dynamic obstacle emerging at time t_k at m path steps before the robot[1]. The neuron i between the robot and the obstacle, which is next to the obstacle, becomes inactive at time t_k+1, since all of its neighbours (except the obstacle neuron) have a larger activity, and, therefore, Condition (d) in (4) is false for them. This is similar to the "gate" situation illustrated above.

As follows from Property 4, the neuron i stays inactive also at the next time step t_k+2, and (at least) one of its neighbours becomes also inactive (analogous as discussed above for the neuron i). Therefore, two time steps after the obstacle has emerged, at least a pair of two inactive neurons appears. It moves away from the target along the direction of the wave front expansions. After $m/2$ steps the robot position coincides with one of these inactive neurons, the robot stops and waits at least one more time step, and at time $t_k + m/2 + 2$ continues the navigation, if a new wave front has reached its position (Figure 4b). The direction of robot's next path step depends on the enumeration of the neurons in the local neighbourhood. The final path of the robot, which corresponds to the neighbours' enumeration as in Figure 4a, is shown in Figure 4c.

This inhibitory mechanism distinguishes our model from other models, for instance, from the classical resistive grid model, which requires a large number of iterations to escape from local maxima and to converge to a solution. During these iterations oscillatory movements of the robot centred at the point of a local extremum are observed, which lead to unnatural "hither and thither" paths with far from optimal lengths.

4 Comparative simulation and complexity analysis

Below we present comparative simulations of the models discussed in Section 2 together with DWENN in three different dynamic environments: (i) the results for the "closing gate"

[1]here m is even, the case of an odd m is similar.

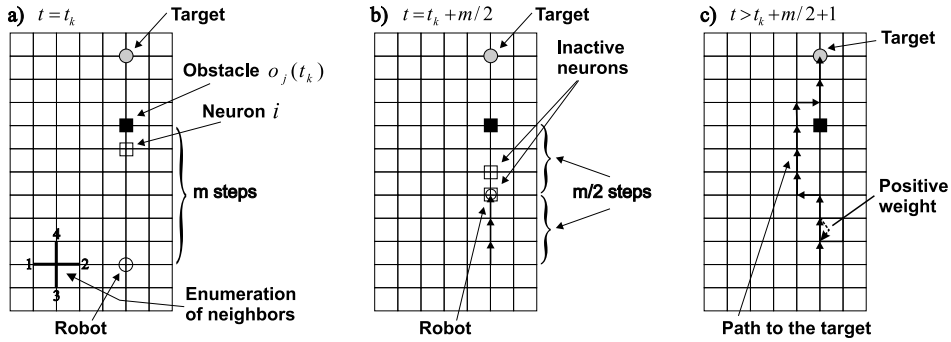

Figure 4: Avoidance of dynamic obstacles by travelling inhibitory waves. a) An obstacle appears on the way of the robot. b) This causes the appearance of a chain of two neurons, which moves along the direction of the wave front expansions. c) The resulting path avoiding the obstacle depends on the enumeration of neurons in local neighbourhoods (the shown path corresponds to the enumeration as in a)).

Table 2: Model parameters chosen for the shown simulations (the first part of the table), and for the statistical evaluations (the second part of the table).

Glasius et al.	Yang et al.	Chen et al.
$r = 1$, $\gamma = 0.9$, $\beta = 0.43$	$r = 1$, $A = 40$, $B = D = 1$, $\mu = 8$, $E = 15$	$A = 100$, $m = 17$
$\beta = 0.437$, 0.43 for the "closing gate" and the "freezing up obstacles" scenario, respectively.		$A = 93.2$

scenario similar to the one discussed in the previous section are shown in Figure 5; (ii) Figure 6 displays the results for the scene with forward and backward moving obstacles, which finally freeze up at their initial position, and, (iii) the results for moving target pursuit are shown in Figure 7.

In all simulations the network consists of 3600 (60×60) neurons representing a 2D workspace. The borders of the workspace are treated as obstacles, SP and TP denote the starting and target positions, respectively, and arrows indicate the movement directions of obstacles. On the $3D$ plots the values along the z-axis are scaled by the $log()$ function (for all models, except DWENN). The models were simulated with the parameters listed in Table 2. Lacking general rules, we tried to optimise the performance of these models by trial and error with heuristically chosen parameters.

The statistical evaluations were based on 500 runs per model and scenario each. For the "closing gate" scenario (Figure 5), the robot's starting position and starting times of the moving obstacle were selected randomly. The maximum number of iterations was limited by 1000. In the second "freezing obstacles" scenario (Figure 6) the x positions of the obstacles and their stopping time were selected randomly and a maximum of 1500 iterations was allowed. In this case two experiments were conducted: in the first series the obstacles stop at their current positions. In the second series, despite their current positions, the obstacles were placed at the last visited wall (i.e., either to the top, or to the bottom of the workspace). For both scenarios, the average number of path steps, the average number of iterations required to reach the target, the corresponding standard

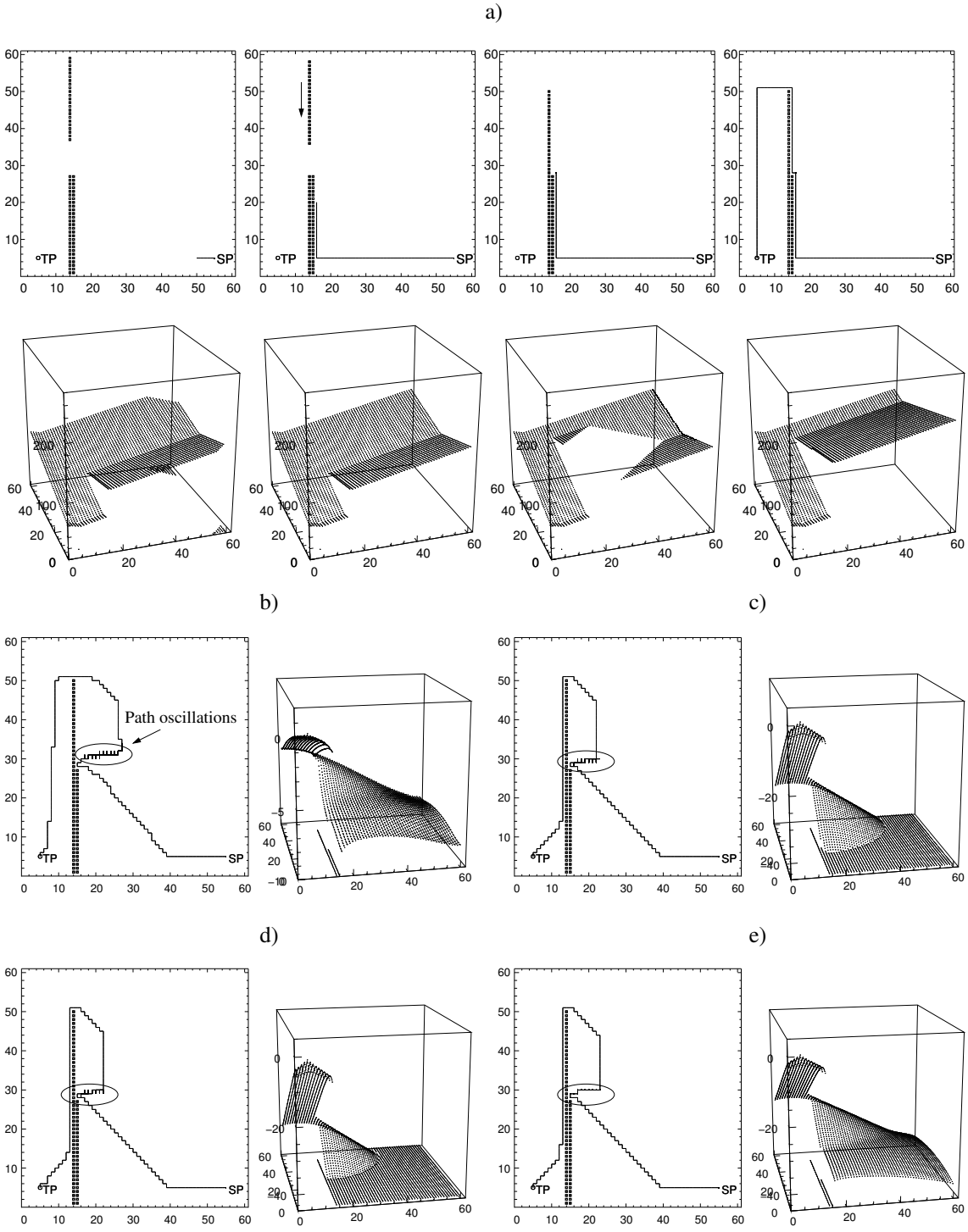

Figure 5: Scene 1: the "closing gate" scenario. The moving obstacle closes the "gate" before the robot has come through it. a) Three intermediate planning steps, the resulting path, and corresponding activity landscapes for the DWENN model (142 path steps, 283 network iterations). b) The final path and the activity landscape for the resistive grid model (464 path steps, 560 iterations). c) For the model by Glasius et al. (274 path steps, 380 iterations). d) For the model by Yang et al. (158 path steps, 743 iterations). e) For the model by Chen et al. (270 path steps, 412 iterations). While models b)-e) lead to significant detours, the DWENN model a) can be seen to produce a very parsimonious path that is of minimal L_1-length in this case.

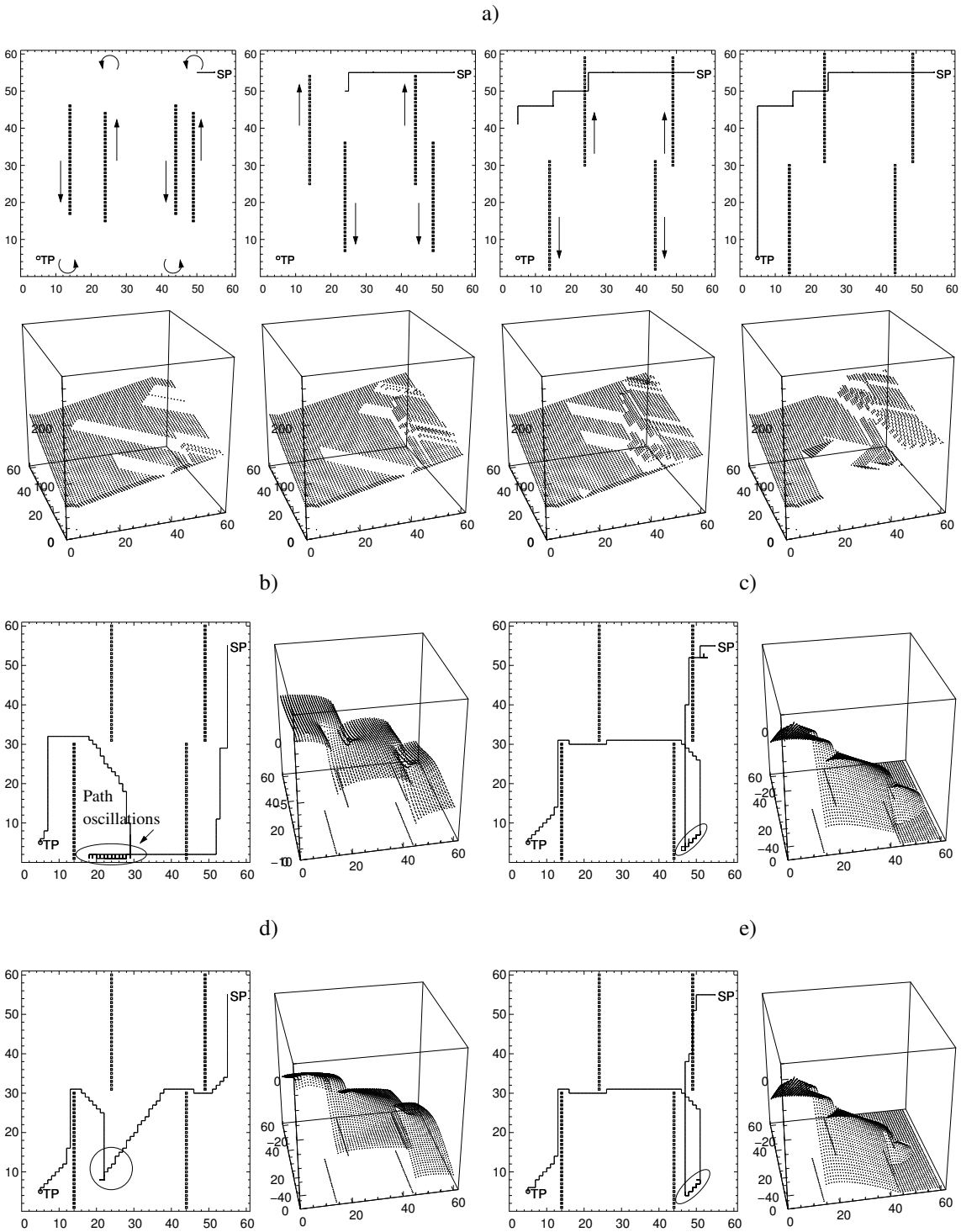

Figure 6: Scene 2: "freezing up obstacles" scenario. The dynamic obstacles start at the positions as shown on the fourth $2D$ plot, move from the bottom to the top and back, and freeze up at their initial positions. a) Three intermediate planning steps, the resulting path, and corresponding activity landscapes for the DWENN model (100 path steps, 216 network iterations). b) The final path and the activity landscape for the resistive grid model (1152 path steps, 1252 iterations). c) For the model by Glasius et al. (250 path steps, 368 iterations). d) For the model by Yang et al. (150 path steps, 652 iterations). e) For the model by Chen et al. (232 path steps, 368 iterations).

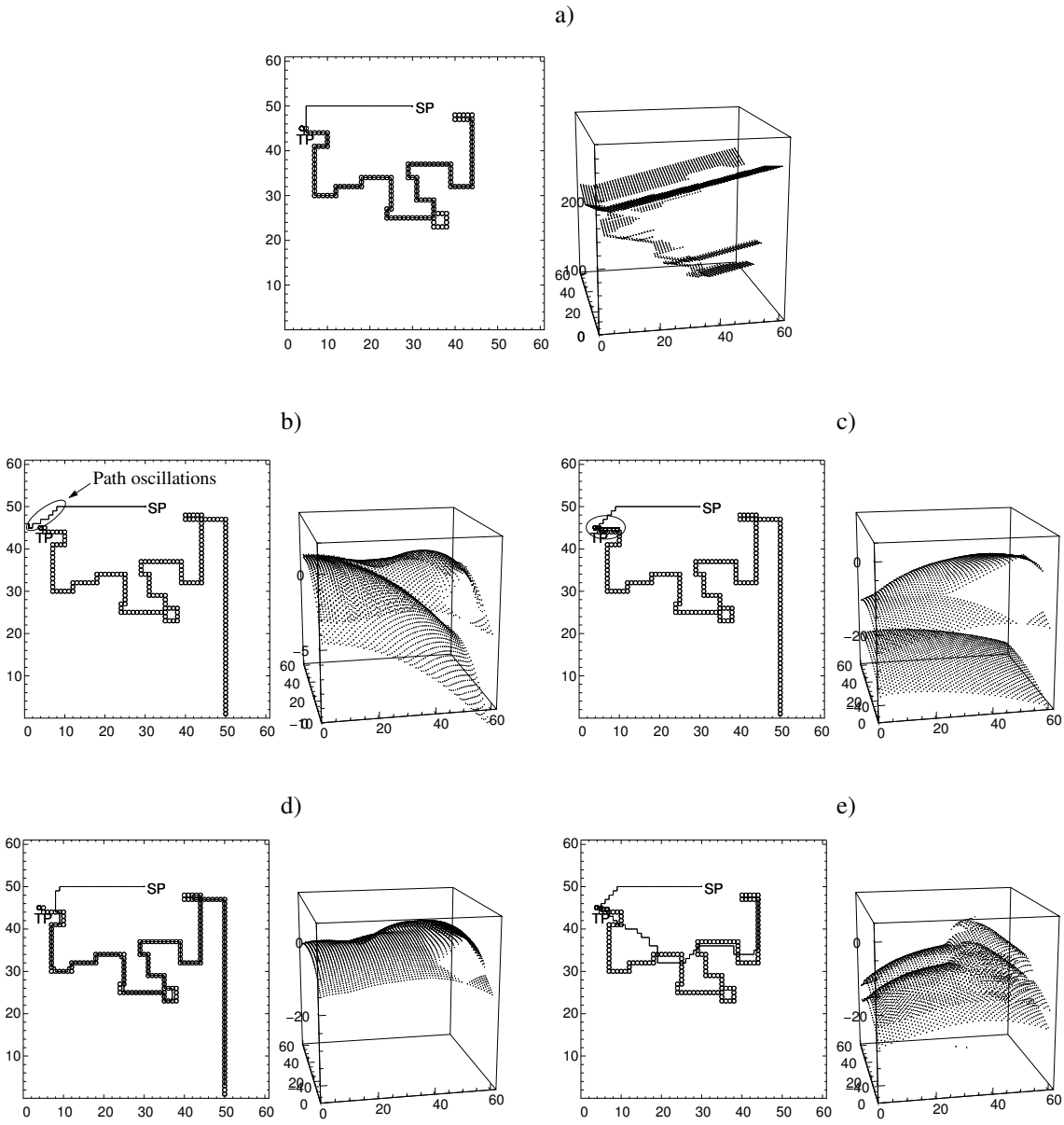

Figure 7: Scene 3: pursuit of the moving target. The target starts to move after several path steps of the robot. a) The path of the robot and the resulting activity landscape for the DWENN model: the target is captured. b) The resistive grid model: failed. The robot got trapped by a local maximum (several local maxima appear during the network evolution; this can be seen in the corresponding $3D$ plot). c) The model by Glasius et al.: failed. The robot got trapped by a local maximum, and the network dynamics is too slow to continue the pursuit; the target left the border of the workspace. d) The model by Yang et al.: failed. This model demonstrated a reliable pursuit, but it is not fast enough to capture the target. e) The model by Chen et al.: the target is captured.

Table 3: Statistical comparison for the "closing gate" scenario (an example is demonstrated in Figure 5). The table shows for each model the average number of path steps (the average path length) and its standard deviation as well as the average number of iterations and its standard deviation in 500 runs. For each run, the starting time of the moving obstacle and the initial robot position are randomly selected. Each network is given the maximum of 1000 iterations.

	DWENN	resist. grid	Glasius et al.	Yang et al.	Chen et al.
av. # of steps	106.63	287.67	188.6	121.51	199.32
std. deviation	25.49	162.79	86.29	35.56	94.45
av. # of iter.	212.95	370.93	274.64	505.51	288.83
std. deviation	50.76	169.16	94.85	241.45	104.67

deviations, and the number of failures are summarised in Table 3 and Table 4, respectively.

4.1 Discussion of results

In all experiments, the DWENN model clearly outperforms all other models: in the "closing gate" scenario with respect to the path length as well as to the number of network iterations. In the "freezing obstacle" scenario, the results vary in two series of runs (the latter differ with respect to the "freezing" positions of the obstacles). In the first series, all models generate paths of approximately the same length, but the model by Yang et al. requires much more iterations. The results for the second and more complex series reveal that the final placement of obstacles does not influence the DWENN efficiency, while all other models need significantly more iterations to converge to a solution and the path length at least doubles in comparison with DWENN. Obviously, the dynamics of these models is not fast enough to provide rapid adaptation to the sudden stopping of the obstacles (hence some models failed to find a solution in many trials).

The following remarks summarise further observations with respect to the typical behaviour of the existing models. As revealed by the simulations, the resistive grid model has the slowest dynamics. Usually, several local maxima appear when the environment is changing rapidly. As a result, a large number of iterations is needed to escape from a local maximum and to converge to a solution.

The model by Glasius et al. in many situations is more efficient than the resistive grid model. However, in complex dynamic scenes the network dynamics is not fast enough to adapt to the changes effectively. Some efforts are also needed to find an appropriate set of the model parameters.

The model by Yang et al. generates mainly paths of a good quality, which are usually shorter, than those of the resistive grid and the Glasius's et al. models. But typically, this model requires a larger number of iteration to converge to a solution. As noticed also in (Chen et al., 2003), the network dynamics and, therefore, the quality of a generated path, significantly depend on the choice of parameters. The most important model parameters are A (the passive decay rate) and μ, which defines "how much" activity is transferred between a neuron and its neighbours. Incorrect choice of these parameters may lead (a) to a quick saturation of the neural activity (as soon as the activity landscape is in its steady state, no further planning is possible, since all neurons have the same activity level, and the robot stops at its current position), and (b) to the case, when the neural activity decays and becomes too small to reach the position of the neuron, corresponding to the robot's starting point. In both cases path planning fails. An illustration of how the parameters in the model by Yang et al. influence the "quality" of trajectory generation

Table 4: Statistical comparison for the "freezing up obstacles" scenario (an example is demonstrated in Figure 6). The table shows for each model the average number of path steps (the average path length) and its standard deviation as well as the average number of iterations and its standard deviation in 500 runs per series. For each run, the stopping time of the moving obstacles and their initial x positions are randomly selected. Each network is given the maximum of 1500 iterations. In Series 1 the obstacles stopped at their current positions. In Series 2 their were placed to the last visited wall (either to the top, or to the bottom of the workspace).

Series 1	DWENN	resist. grid	Glasius et al.	Yang et al.	Chen et al.
av. # of steps	101.88	119.98	105.35	103.4	105.36
std. deviation	8.25	101.64	20.67	8.46	23.0
av. # of iter.	221.48	220.79	235.12	437.93	236.5
std. deviation	19.5	101.58	20.41	44.8	22.86
# of failures	0	5	0	0	0
Series 2					
av. # of steps	105.53	812.63	221.99	133.2	234.84
std. deviation	23.24	249.63	22.36	23.22	23.75
av. # of iter.	227.74	913.08	352.53	571.41	367.19
std. deviation	40.19	249.47	19.74	96.67	20.43
# of failures	0	98	16	0	3

is shown in Figure 8. As demonstrated in Figure 5, when $A = 40$ and $\mu = 8$, the path length is 158 and the number of iterations is 743. For $\mu = 10$, the path length and the necessary number of iterations is 166 and 975, correspondingly (this path is illustrated in Figure 8a). For $A = 10$ and $\mu = 2$, the path length is 158 steps again, but in this case the model requires much more, namely 2507 iterations to converge to a solution (this path is depicted in Figure 8b). For $A = 2$ and $\mu = 10$, the neural activity saturates quickly, such that the robot is not able to move further and to reach the target (the position, where the robot stopped, is shown in Figure 8c; the activity landscape is shown for the moment, when the neuron, corresponding to the robot's current position becomes saturated).

The paths, generated by the model by Chen et al., are of an acceptable quality, and are similar to those produced by the model by Glasius et al. (what indicates also the similarity of these models).

It is worth to notice, that there are no explicit rules of selecting the model parameters for the models by Glasius et al., by Yang et al., and by Chen et al.. Even if some parameter set appears to be suitable for a certain task, it is not known, whether a generated path is optimal and what parameters are needed to get a good path. In contrast, the DWENN model is *parameter-free* and its performance exclusively depends on the task.

4.2 Complexity issue

Besides the performance in terms of path length and network iterations, two major issues are important in robotic tasks: (i) the computational load needed to carry out the path planning, and (ii) possible applicability of the method for planning in higher dimensional configuration spaces and corresponding memory requirements.

Table 5 displays for each model the number of additions and multiplications needed for updating the activity level of a single neuron. Note that in all possible cases the DWENN needs at most a single addition to increment the activity inherited from a neighbour, re-

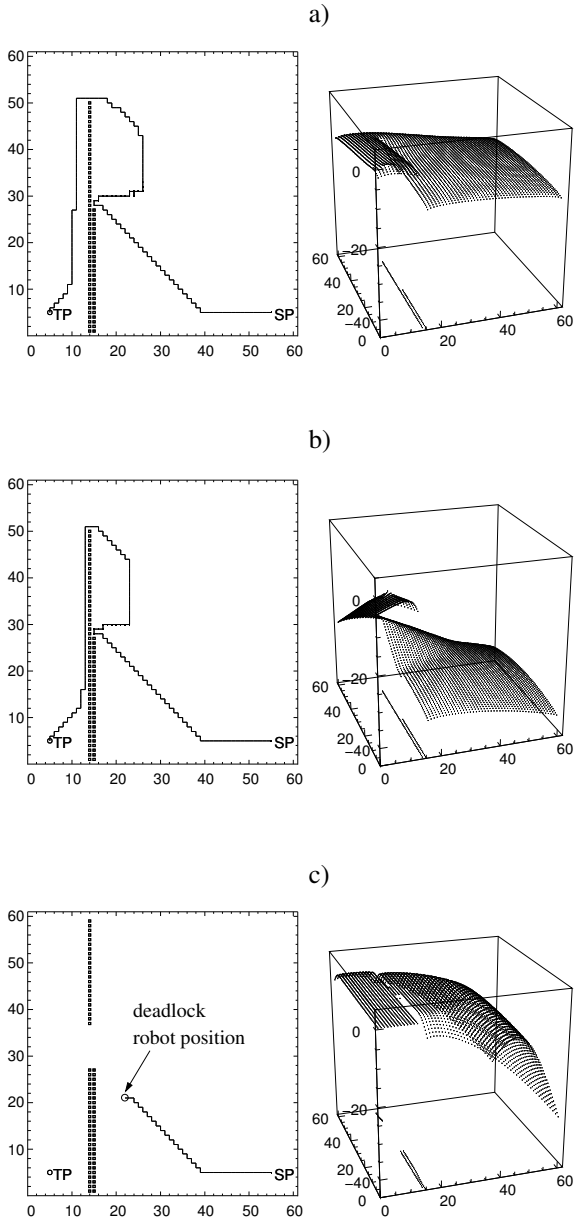

Figure 8: Influence of the parameters in the model by Yang et al. on path planning. a) The resulting path and the activity landscape for $A = 40$ and $\mu = 10$. The path length and the necessary number of iterations is 166 and 975, correspondingly (note, that for $\mu = 8$ the path length is 158 and the number of iterations is 743, Figure 5). b) The resulting path and the activity landscape for $A = 10$ and $\mu = 2$. The path length is 158 steps again, but the number of iterations is 2507. c) The resulting path and the activity landscape for $A = 2$ and $\mu = 10$. The neural activity saturates when the robot is in the shown position. No further movements are possible since the "saturated" neurons have the same activity level.

Table 5: The number of additions and multiplications per state update of a neuron (N is the dimensionality of the configuration space).

DWENN	resist. grid	Glasius et al.	Yang et al.	Chen et al.
1	$(2N+1)$	$(2N+3)$	$(2N+5)$	$(2N+6)$

gardless of the number of neighbours involved and, thus, *independent* of the dimensionality of the configuration space. However, one needs to query the neighbour(s) and evaluate a number of binary attributes in Conditions (a)-(d) in (4) (at most $2N \times 5$ such binary operations are needed in the worst case). In practice, however, the number of these evaluations is dramatically reduced, if the neighbour is queried first, which has a positive weight, i.e., which carried the last activity wave to the updating neuron at the previous time step.

The DWENN model is extremely efficient with respect to memory requirements as well, because only integer-valued computation is involved. Assume that the neural field has k^N neurons, then a simple field of $2k^N$ short integers of 16 bit is sufficient to store the potential values together with all additionally needed information to evaluate Conditions (a)-(d) in (4), which can be encoded in two bits. This and an additional array of $k^N \times 2N$ bits to store the index of the positive weight for each neuron is sufficient to implement the whole DWENN dynamics.

5 Conclusions

We proposed a new type of neural network – the dynamic wave expansion neural network (DWENN) – which is capable of calculating dynamic distance potentials, useful for route generation in time-varying environments. By means of local interactions between neurons, waves of neural activity propagate in the network field and the corresponding activity levels of neurons are updated by summation of the transferred activity. In particular situations (when the activity propagation is interrupted by obstacle movements), DWENN neurons may initiate the propagation of inhibitory waves, which prevent the appearance of local minima, and, hence, undesirable situations, when the robot "gets lost".

For the DWENN model, no *a priori* knowledge of the environment is needed, as well as no learning process is required to perform path generation. Since the network is only locally connected, the computational complexity grows linearly in the number of neurons in the network field. Therefore, the generation of grid potentials is an extremely fast process, which makes real-time planning possible. The DWENN dynamics is *parameter-free*, therefore, no efforts are needed to tune the model. The DWENN paths are safe and tend to be optimal in a L_1 metrics even in complex, highly dynamic situations.

The DWENN algorithm may be characterised as an *active*, *exploratory* method, in which the activity propagation is *selective*. It is sufficient that only one neuron with required properties exists in a local neighbourhood to provide further activity spreading. A sequential principle of adaptation of connection weights was used also in (Kassim & Vijaya Kumar, 1995, 1997b, 1997a, 1999), where a model of neural network for path planning in a stationary domain was proposed. In contrast to DWENN, other models are *passive*, since the initial quantity of neural activity is distributed through the network field in such a way, that all neurons in a local neighbourhood participate in the process of activity propagation (such methods are called sometimes *relaxation* methods).

Since the activity propagation is selective, DWENN's complexity (i.e., the number of operations per state update of a neuron, see Table 5), unlike all other models, does not explicitly depend on the dimensionality of the configuration space, and depends only on

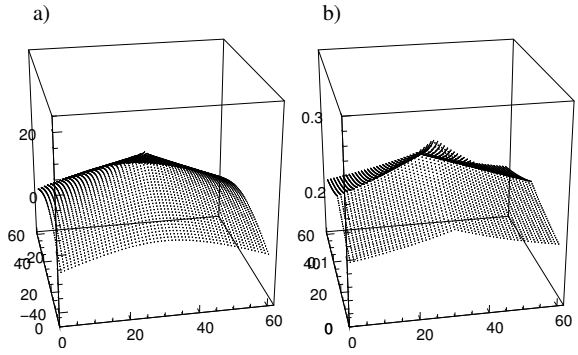

Figure 9: a) Scalar potential field generated by the model by Glasius et al. (the target is in the centre of the workspace). b) Scalar potential field with $z_i = 10/x_i$, where x_i are the potential values generated by the DWENN model. This potential field can be considered as a coarse, but qualitatively similar approximation of the first one.

the "complexity" of the dynamic scene. Therefore, in the case of a stationary environment, for instance, only *one* addition operation and *five* binary checks per neighbour are required to calculate the activity level of a neuron in a configuration space of an *arbitrary* dimensionality.

The DWENN algorithm triggers in particular situation the propagation of inhibitory waves in the network field, what provides an *active*, rapid, and effective reaction and adaptation to sudden environmental changes, as well as local-minima-free trajectory generation in the most cases.

If one takes $z_i = g(x_i)$ for $x_i > 0$ (x_i are computed according to (3), $g(x) = k/x, k \in R^+$), then a potential field generated by the DWENN model can be considered as a coarse, but qualitatively similar approximation of a potential field obtained by the considered relaxation models (Figure 9).

The DWENN dynamics is fast enough to adapt to rapid changes in the environment. It requires only a small number of iterations to stabilise in the presence of environmental changes and to lead the robot to its destination. In the simulations DWENN demonstrated a robust planning and typically outperformed other models by generating shorter and more "efficient" paths. The model can deal effectively with all kinds of dynamic changes in the environment, and, hence, it may be applied to an efficient path planning of mobile robots in (i) environments with a highly dynamic nature, as well as in (ii) multi-agent environments (when robots are considered as moving obstacles for each other). Since the computational efficiency of the model does not explicitely depend on the dimensionality of the configuration space, it can potentially be applied to the trajectory generation of multi-joint robotic manipulators with many degrees-of-freedom, even if corresponding higher-dimensional configuration spaces are changing in time. Further, a distributed planning for multi-body articulated systems, as e.g., a robotic hand with several fingers, or multiple robot arms sharing a common workspace is another prospective application of the DWENN model.

Acknowledgement

The authors would like to thank R. Haschke for his comments on a draft version of the manuscript.

References

Ageev, D. A., & Istratov, A. Y. (1998). Neural network implementation for the optimal path problem. *Journal of Computer and System Sciences International*, *37*, 118–125.

Alvarez, D., Alvarez, J. C., & Gonzalez, R. C. (2003). Online motion planning using Laplace potential fields. In *Proceedings of IEEE International Conference on Robotics and Automation* (pp. 3347–3352). Taipei, Taiwan: IEEE.

Barraquand, J., Kavraki, L., Latombe, J. C., Li, T.-Y., Motwani, R., & Raghavan, P. (1997). A random sampling scheme for path planning. *International Journal of Robotics Research*, *16*, 759–774.

Barraquand, J., & Latombe, J. C. (1991). Robot motion planning: a distributed representation approach. *International Journal of Robotics Research*, *10*, 628–649.

Bennewitz, M., Burgard, W., & Thrun, S. (2003). Adapting navigation strategies using motions patterns of people. In *Proceedings of IEEE International Conference on Robotics and Automation* (pp. 2000–2005). Taipei, Taiwan: IEEE.

Bugmann, G., Taylor, J. G., & Denham, M. (1995). Route finding by neural nets. In J. G. Taylor (Ed.), *Neural networks* (pp. 217–230). Alfred Waller Ltd.

Chen, W., Fan, C., & Xi, Y. (2003). On-line safe path planning in unknown environments. In *Proceedings of IEEE International Conference on Robotics and Automation* (pp. 4191–4196). Taipei, Taiwan: IEEE.

Connolly, C., Burns, J. B., & Weiss, R. (1990). Path planning using Laplace's equation. In *Proceedings of IEEE International Conference on Robotics and Automation* (pp. 2102–2106). Cincinnati, OH: IEEE.

Dracopoulos, D. C. (1998). Neural robot path planning: the maze problem. *Neural Computing & Applications*, *7*, 115–120.

Eldershaw, C., & Cameron, S. (2000). Using genetic algorithms to solve the motion planning problem. *Journal of Universal Computer Science*, *6*, 422–432.

Fiorini, P., & Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, *17*, 760–772.

Fraichard, T., & Laugier, C. (1993). Path-velocity decomposition revisited and applied to dynamic trajectory planning. In *Proceedings of IEEE International Conference on Robotics and Automation* (pp. 40–45). Atlanta, GA: IEEE.

Glasius, R., Komoda, A., & Gielen, S. C. A. M. (1995). Neural network dynamics for path planning and obstacle avoidance. *Neural Networks*, *8*, 125–133.

Glasius, R., Komoda, A., & Gielen, S. C. A. M. (1996). A biologically inspired neural net for trajectory formation and obstacle avoidance. *Biological Cybernetics*, *74*, 511–520.

Grossberg, S. (1988). Nonlinear neural networks: principles, mechanisms, and architectures. *Neural Networks*, *1*, 17–61.

Henrich, D. (1997). Fast motion planning by parallel processing - a review. *Journal of Intelligent and Robotic Systems*, *20*, 45–69.

Hsu, D., Kindel, R., Latombe, J. C., & Rock, S. (2002). Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, *21*, 233–255.

Hwang, Y. K., & Ahuja, N. (1992). Gross motion planning - a survey. *ACM Computing Surveys*, *24*, 219–291.

Karnik, M., Dasgupta, B., & Eswaran, V. (2002). A comparative study of Dirichlet and Neumann conditions for path planning through harmonic functions. In P. M. A. Sloot, C. J. K. Tan, J. J. Dongarra, & A. G. Hoekstra (Eds.), *Lecture notes in computer science 2330* (pp. 442–451). Springer-Verlag Berlin Heidelberg.

Kassim, A. A., & Vijaya Kumar, B. V. K. (1995). Potential fields and neural networks. In

M. A. Arbib (Ed.), *The handbook of brain theory and neural networks* (pp. 749–753). Cambridge, MA: The MIT Press.

Kassim, A. A., & Vijaya Kumar, B. V. K. (1997a). Path planning for autonomous robots using neural networks. *Journal of Intelligent Systems*, *7*, 33–56.

Kassim, A. A., & Vijaya Kumar, B. V. K. (1997b). The wave expansion neural network. *Neurocomputing*, *16*, 237–258.

Kassim, A. A., & Vijaya Kumar, B. V. K. (1999). Path planners based on the wave expansion neural network. *Robotics and Autonomous Systems*, *26*, 1–22.

Kavraki, L. E., Svestka, P., Latombe, J. C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional space. *IEEE Transactions on Robotics and Automation*, *12*, 566–580.

Khatib, O. (1986). Real time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, *5*, 90–99.

Kindermann, T., Cruse, H., & Dautenhahn, K. (1996). A fast, three-layer neural network for path finding. *Network: Computation in Neural Systems*, *7*, 423–436.

Koenig, S., & Likhachev, M. (2002). Incremental A*. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing systems 14.* Cambridge, MA: MIT Press.

Latombe, J. C. (1991). *Robot motion planning.* Boston, MA: Kluwer Academic Publishers.

Lebedev, D. V., Steil, J. J., & Ritter, H. (2002). A new wave neural network dynamics for planning safe paths of autonomous objects in a dynamically changing world. In *Proceedings of WSEAS International Conference on Neural Networks and Applications* (pp. 4171–4176).

Lebedev, D. V., Steil, J. J., & Ritter, H. (2003a). Real-time path planning in dynamic environments: a comparison of three neural network models. In *Proceeding of IEEE International Conference on Systems, Man, and Cybernetics* (pp. 3408–3413).

Lebedev, D. V., Steil, J. J., & Ritter, H. (2003b). A neural network model that calculates dynamic distance transform for path planning and exploration in a changing environment. In *Proceedings of IEEE International Conference on Robotics and Automation* (pp. 4209–4214). Taipei, Taiwan: IEEE.

Lee, S., & Kardaras, G. (1997). Collision-free path planning with neural networks. In *Proceedings of IEEE International Conference on Robotics and Automation* (pp. 3565–3570). Albuquerque, NM: IEEE.

Lemmon, M. (1991). 2-degree-of-freedom robot path planning using cooperative neural fields. *Neural Computation*, *3*, 350–362.

Li, Z. X., & Bui, T. D. (1998). Robot path planning using fluid model. *Journal of Intelligent and Robotic Systems*, *21*, 29–50.

Liu, J., & Khatib, O. (2002). Practical connection between potential fields and neural networks. In M. A. Arbib (Ed.), *The handbook of brain theory and neural networks (second edition).* Cambridge, MA: The MIT Press.

Lozano-Perez, T. (1983). Spatial planning: a configuration space approach. *IEEE Transactions on Computers*, C-32:108–120.

Lumelsky, V. J., & Stepanov, A. A. (1986). Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE Transactions on Automatic Control*, *31*, 1058–1063.

Mazer, E., Ahuactzin, J. M., & Bessiere, P. (1998). The Ariadne's clew algorithm. *Journal of Artificial Intelligence Research*, *9*, 295–316.

Meng, H., & Picton, P. D. (1992). A neural network for collision-free path planning. In I. Aleksander & J. Taylor (Eds.), *Artificial neural networks 2* (p. 591-594). North-Holland.

Miura, J., Uozumi, H., & Shirai, Y. (1999). Mobile robot motion planning considering

the motion uncertainty of moving obstacles. In *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics* (pp. 692–697).

Paredis, J., & Westra, R. (1997). Coevolutionary computation for path planning. In H.-J. Zimmermann (Ed.), *Proceedings of 5th Europian Congress on Intelligent Techniques and Soft Computing* (pp. 394–398). Aachen: Verlag Mainz.

Rimon, E., & Koditschek, D. (1992). Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, *8*, 501–518.

Shu, C., & Buxton, H. (1995). Parallel path planning on the distributed array processor. *Parallel Computing*, *21*, 1749–1767.

Siemiatkowska, B., & Dubrawski, A. (1998). Cellular neural networks for navigation of a mobile robot. In L. Polkowski & A. Skowron (Eds.), *Lecture notes on artificial intelligence 1424* (pp. 147–154). Heidelberg: Springer-Verlag.

Song, G., Thomas, S., & Amato, N. M. (2003). A general framework for PRM motion planning. In *Proceedings of IEEE International Conference on Robotics and Automation* (pp. 4445–4450). Taipei, Taiwan: IEEE.

Stentz, A. (1995). The focussed D* algorithm for real-time replanning. In *Proceedings of Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1652–1659). Montreal, CA: Morgan Kaufmann.

Tarassenko, L., Brownlow, M., Marshall, G., Tombs, J., & Murray, A. F. (1991). Real-time autonomous robot navigation using VLSI neural networks. In R. Lippmann, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems 3* (pp. 422–428). San Fransisco, CA: Morgan Kaufmann.

Vleugels, J., Kok, J. N., & Overmars, M. (1997). Motion planning with complete knowledge using a colored SOM. *International Journal of Neural Systems*, *8*, 613–628.

Wang, Y., & Chirikjian, G. S. (2000). A new potential field method for robot path planning. In *Proceedings of IEEE International Conference on Robotics and Automation* (pp. 977–982). San Francisco, CA: IEEE.

Xia, Y., & Wang, J. (2000). A discrete-time reccurent neural network for shortest-path routing. *IEEE Transactions on Automatic Control*, *45*, 2129-2134.

Yang, S. X., & Meng, M. (2000). An efficient neural network approach to dynamic robot motion planning. *Neural Networks*, *13*, 143–148.

Yang, S. X., & Meng, M. (2001). Neural network approaches to dynamic collision-free trajectory generation. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics*, *31*, 302–318.

Yang, S. X., & Meng, M. (2003). Real-time collision-free motion planning of a mobile robot using a neural dynamics-based approach. *IEEE Transactions on Neural Networks*, *14*, 1541–1552.

Yu, H., & Su, T. (2001). A destination driven navigator with dynamic obstacle motion prediction. In *Proceedings of IEEE International Conference on Robotics and Automation* (pp. 2692–2697). Seoul, Korea: IEEE.

Zelinsky, A. (1992). A mobile robot navigation exploration algorithm. *IEEE Transactions on Robotics and Automation*, *8*, 707–717.