Neural network dynamics for path planning and obstacle avoidance [1]

R. Glasius A. Komoda S. Gielen

Department of Medical Physics and Biophysics,
University of Nijmegen, Geert Grooteplein Noord 21,
6525 EZ Nijmegen, The Netherlands,

Abstract

A model of a topologically organized neural network of a Hopfield type with nonlinear analog neurons is shown to be very effective for path planning and obstacle avoidance. This deterministic system can rapidly provide a proper path, from any arbitrary start position to any target position, avoiding both static and moving obstacles of arbitrary shape. The model assumes that an (external) input activates a target neuron, corresponding to the target position, and specifies obstacles in the topologically ordered neural map. The path follows from the neural network dynamics and the neural activity gradient in the topologically ordered map. The analytical results are supported by computer simulations to illustrate the performance of the network.

[1]This work has been accepted by Neural Networks (March 1994).

1 Introduction

Human motor control reveals a versatility of function and economy of space, that is yet beyond the reach of robots. One of the important themes of research in the field of robotics is the design of a motion planning system. Such a system should guide a robot or a robot manipulator from an initial position to a target position, avoiding obstacles which are located between these positions. If we take into account that the obstacles as well as the target can move and that the obstacles can have any shape, it becomes clear that this problem is not a trivial one. Especially planning a path for a robot in an environment which is unknown and changing, is a difficult problem.

In the past, several authors [2, 7, 16, 18, 20, 31] have worked on the path planning problem. Most work so far deals with static environments and used global methods, which can generally be viewed as a search process for a path in a graph. However, global methods will limit the real-time capabilities of robots in a cluttered environment, especially when new information about changes in the environment is becoming available continuously, because of the time needed to perform the planning task. Later, the idea of using an artificial potential field around each obstacle was proposed combined with an attractive potential around the target [17, 19]. Unfortunately, the potential functions proposed suffered from several problems, one being that of undesired local minima. Later, several papers have attempted to provide solutions to this problem (e.g. [1, 3, 6, 24, 30]). None of these papers mentioned above offers an exact potential-function-based algorithm that is guaranteed to work. Recently, a new algorithm was proposed which addressed the path-finding problem using an iterative approach under the constraints of minimum time or minimum energy [27]. However, this method will be very time-consuming, especially for many, complex objects in the environment.

In this paper we present a path planning system which can control robot motion in a static as well as in a dynamic environment with a guaranteed solution, which is a shortest path in terms of the metric of the representation, if one exists. We will assume that information about the position and shape of obstacles in the environment is not known beforehand and that it appears during the path planning, for example, through real-time sensing. This may be rather artificial for path planning in a static environment, but it is an inevitable situation for path planning in an environment in which the position of target and obstacles are changing continuously. This makes our approach distinct from other approaches [8, 20, 26], which presume an algebraic representation of a robot environment and which call for one-time off-line computation. In our approach we have only studied path-planning, and not trajectory formation such as in [29]. In the latter approach one has to deal with forward and backward models of the system under control before trajectories can be generated from an initial to a final point. For complex paths, such as in a cluttered or changing environment, the total path has to be segmented in order to create a sequence of trajectories.

Our path planning system is based on a Hopfield-type of network with continuous neurons [13]. Analog neurons provide the possibility to make such a system a fast computing device, which can process the massive amount of sensory data necessary to move in a changing environment. In early attempts to understand biological computation, neurons were modeled as two-state threshold devices only. However, parallel computation with discrete neurons faces stability problems not found in sequential dynamics. However, sequential dynamics - when neuron states are updated one at a time - implemented in software on a conventional computer is simply too slow for any large network application.

These stability problems associated with parallel dynamics can be eliminated by using continuous neurons [21]. They also provide the possibility for hardware implementation

The paper is organized as follows: In section 2 we define the model and show its basic properties. We define two types of dynamics: parallel discrete- and continuous-time dynamics. We show for both types of dynamics that the network, after receiving an external stimulus, evolves towards a specific state of neural activity which corresponds to a minimum of a Liapunov function. The path is given by the neural activity gradient. This path has the shortest length among all paths connecting initial and goal position. Computer simulations illustrating the performance of the network are presented in section 3. Section 4 summarizes the conclusions.

2 The model

2.1 Work space and configuration space

Consider a robot manipulator $\mathcal{R}$, equipped with sensors and actuators, is operating in a subset of the real world. This robot should have the capability of planning its own motion by generating a path specified by an initial configuration, a final configuration and by the position of obstacles in the workspace. We assume that the non-redundant robot manipulator has d degrees of freedom which may correspond to the d joints of the robot arm. In this paper we will only deal with the kinematics of the movement-trajectory of the manipulator. We will assume that the d degrees of freedom span a d-dimensional configuration space $\mathcal{C}$. Each point in $\mathcal{C}$ defines a unique configuration of the non-redundant robot-manipulator in workspace.

The obstacles $\mathcal{O}_\alpha$, $\alpha = 1 \ldots m$ in workspace , are represented in the configuration space as the subset of those points in configuration space, which cannot be reached by the manipulator. This includes also configurations which are not allowed since it would require that one of the links touches or passes through an object. Therefore, the "forbidden" regions in configuration space are larger than the regions that correspond to the positions of obstacles. In mathematical terms the obstacle regions are defined in $\mathcal{C}$ by the points where the objects and the links of the robot manipulator share at least one point:

$$\hat{\mathcal{O}}_\alpha = \{\mathbf{q} \in \mathcal{C} \ \mid \ \mathcal{R}(\mathbf{q}) \cap \mathcal{O}_\alpha \neq \emptyset\}, \tag{1}$$

where $\mathcal{R}(\mathbf{q})$ is an area occupied by $\mathcal{R}$ at configuration $\mathbf{q}$ in the workspace.

Feasible paths between an initial configuration $\mathbf{q}_{\text{init}}$ and a target configuration $\mathbf{q}_{\text{targ}}$ are continuous maps from a closed real interval into the free configuration $\mathcal{C}_{\text{free}}$ space which remains after removing the points $\hat{\mathcal{O}}_\alpha$ from $\mathcal{C}$

$$\mathcal{C}_{\text{free}} = \mathcal{C} \setminus \bigcup_{\alpha=1}^{m} \hat{\mathcal{O}}_\alpha \tag{2}$$

2.2 Neuronal space and phase space

Our path planning system consists of a large collection of identical processing units called neurons. These neurons are arranged in a d-dimensional cubic lattice (CL). The number of neurons in the network is N. They are connected only to their z nearest neighbors by excitatory and symmetric connections T_{ij}. Generalization to other types of lattices can be obtained very easily. We assume that the lattice represents a topologically ordered map, which can be obtained by a Kohonen-type of learning [9, 22, 25]. This map gives a discrete topologically ordered representation of the robot configuration space $\mathcal{C}$ [12, 23, 28] with nodes homogeneously distributed over the configuration space. We will call it the neuronal space $\mathcal{N}$. To each neuron in $\mathcal{N}$ corresponds a certain subset of $\mathcal{C}$ called its receptive field. In this discrete representation of the configuration space $\mathcal{C}$ the regions $\bigcup_{\alpha=1}^{m}\hat{\mathcal{O}}_\alpha$ are represented by a subset of nodes of the CL. We will call the nodes, which correspond to $\bigcup_{\alpha=1}^{m}\hat{\mathcal{O}}_\alpha$, occupied nodes. Similarly the initial configuration $\mathbf{q}_{\text{init}}$ and the target configuration $\mathbf{q}_{targ}$ are represented by one node each. The external input signals, which provide information about the position of obstacles in workspace, are supposed to clamp the activity of all neurons in the occupied nodes to the value "zero" corresponding to the minimal activity of the neuron. The activity of the neuron corresponding to the target configuration node is clamped to the value "one". The initial configuration of the robot may correspond to any node which is not occupied.

The activities of the neurons are characterized by real-valued variables $\sigma_i \in [0,1]$ and describe the state of the system. The set $\mathcal{S}$ of all possible states σ of the network is called the phase space of the system. In our model $\mathcal{S} = [0,1]^N$, where $[0,1]^N$ stands for a hypercube in $\mathbb{R}^N$.

2.3 Dynamics

The state variables σ_i, $i = 1 \ldots N$, of the neurons can change due to inputs from other neurons in the network and due to external sensory input. The total input u_i to the neuron i is a weighted sum of activities from other neurons and of an external sensory input I:

$$u_i(t) = \sum_j^N T_{ij}\sigma_j(t) + I_i \tag{3}$$

where the strength of the synaptic connection from node j to node i is represented by T_{ij}. In our model the connections T_{ij} are excitatory ($T_{ij} > 0$), symmetric ($T_{ij} = T_{ji}$) and short range:

$$T_{ij} = \begin{cases} 1 & \text{if } \rho(i,j) < r \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Here r is a positive number and $\rho(i,j)$ is the Euclidean distance between neuron i and j in $\mathcal{N}$.

The evolution of the configuration of this network can be defined by the discrete-time or continuous-time dynamics. In the case of a time-discrete evolution the states of neurons are updated by

$$\sigma_i(t+1) = g\left(\sum_j^N T_{ij}\,\sigma_j(t) + I_i\right) \tag{5}$$

where $g(x)$ is a sigmoid function.

The dynamics (5) may be parallel, sequential or may be a random , asynchronous sequence. In the continuous-time updating, all neurons continuously and simultaneously change their states. The change of activity of the neurons is then given by the set of nonlinear differential equations [11]

$$\frac{d\sigma_i(t)}{dt} = g\left(\sum_j^N T_{ij}\,\sigma_j(t) + I_i(t)\right) - \sigma_i(t) \tag{6}$$

Both types of dynamics lead to the same equilibrium state. The function

$$L(\sigma) = -\frac{1}{2}\sum_{i,j} T_{ij}\,\sigma_i\,\sigma_j - \sum_i I_i\sigma_i + \sum_i G(\sigma_i) \tag{7}$$

with $G(\sigma_i) = \int_0^{\sigma_i} g^{-1}(x)\,dx$, is a global Liapunov function for (5) and (6). The vector $\boldsymbol{\sigma}$ refers to the state of the neural network with components σ_i. The existence of a Liapunov function guarantees that the dynamics of the network always converges to a fixed point. However, in the discrete-time parallel dynamics L is a Liapunov function if the stability condition

$$\beta < |1/\lambda_{\min}| \tag{8}$$

is satisfied. Here β is the steepest slope of the function $g(x)$ and $\lambda_{\min}$ is the most negative eigenvalue of the matrix $\mathbf{T}$. If all eigenvalues are positive then L is a Liapunov function for any β. This can be understood if we calculate $\Delta L \equiv L(\sigma(t+1)) - L(\sigma(t))$:

$$\begin{aligned}\Delta L = -\frac{1}{2}\sum_{i,j} T_{ij}\left[\Delta\sigma_i\Delta\sigma_j + 2\sigma_j(t)\,\Delta\sigma_i\right] - \sum_i I_i\Delta\sigma_i \\ + \sum_i \left[G(\sigma_i(t+1)) - G(\sigma_i(t))\right]\end{aligned} \tag{9}$$

where $\Delta\sigma_i \equiv \sigma_i(t+1) - \sigma_i(t)$. By Taylor's theorem we obtain

$$\begin{aligned}G(\sigma_i(t+1)) - G(\sigma_i(t)) &= G'(\sigma_i(t+1))\,\Delta\sigma_i - \frac{1}{2}G''(\xi)\,(\Delta\sigma_i)^2 \\ &\leq G'(\sigma_i(t+1))\,\Delta\sigma_i - \frac{1}{2}(\Delta\sigma_i)^2 \min_{x\in[0,1]} G''(x)\end{aligned} \tag{10}$$

where $\xi \in [\sigma_i(t), \sigma_i(t+1)]$. Inserting equation (10) in equation (9) and using the definition of $G(x)$ we obtain

$$\Delta L \leq -\frac{1}{2}\sum_{i,j}[T_{ij} + \delta_{ij}/\beta]\Delta\sigma_i\Delta\sigma_j \tag{11}$$

where δ_{ij} is the Kronecker delta-function. Here $1/\beta = \min_x G''(x)$ is the minimum curvature of the function $G(x)$ in the closed interval $[0,1]$. The right-hand side of the inequality (11) is negative if the matrix $T_{ij} + \delta_{ij}/\beta$ is positive definite. A sufficient condition for matrix $\mathbf{T} + \beta^{-\mathbf{1}}\mathbf{I}$ to be positive definite is given by equation (8).

In the case of the continuous-time dynamics, for each finite value of β the dynamics (6) converges to a fixed point which is a (local or global) minimum of the Liapunov function. By differentiation of (7) and using (6) we obtain

$$\frac{dL(\sigma(t))}{dt} = \sum_i^N \frac{\partial L(\sigma(t))}{\partial\sigma_i}\frac{d\sigma_i(t)}{dt} = -\sum_i^N \left(g^{-1}(\sigma_i) - u_i\right)(\sigma_i - g(u_i)) \leq 0 \tag{12}$$

since $g^{-1}(\sigma) - u$ and $\sigma - g(u)$ have the same sign. Calculating the time derivative of (3) we can also obtain a representation of (6) in terms of the input variables u_i:

$$\frac{du_i(t)}{dt} = \sum_j^N T_{ij}\,g(u_j(t)) + I_i - u_i(t) \tag{13}$$

The set of equations (13) was first proposed by Hopfield [14]. There is a subtle difference though, between (13) and (6). Hopfield assumed that the activity of neurons changes at the same time as when the neurons receive input:

$$\sigma_i(t) = g\left(u_i(t)\right) \tag{14}$$

In the case of dynamics (3) and (6) this assumption is only valid when the system has reached a stationary state. The dynamical system defined by (13) together with relation (14) has the same Liapunov function as (3) and (6) [14]. By direct computation,

$$\frac{dL\left(\sigma\left(t\right)\right)}{dt} = \sum_i^N \frac{\partial L\left(\sigma\left(t\right)\right)}{\partial \sigma_i}\frac{d\sigma_i\left(t\right)}{dt} = -\sum_i^N \left(\frac{du_i\left(t\right)}{dt}\right)^2 g'\left(u_i\right) \leq 0 \tag{15}$$

Here we have used the symmetry of T_{ij} and the monotonic increase of the function g. The set of equations (13) and the Liapunov function (7) is a particular case of the more general dynamical system, the stability of which was studied by Cohen and Grossberg [5].

2.4 Feasible paths

To construct a feasible path we can use both discrete- and continuous-time dynamics given by (5), (6) or (13). In our simulations we have used discrete-time dynamics (5) for reasons of simplicity. At the initial time, $t = 0$, the activity of all neurons is set to "zero". Under influence of an external input, which clamps the neuron at the target node to the value "one" and which clamps all neurons in the occupied nodes to "zero", the state of the system begins to change according to the dynamics of the network described by equation (5). The evolution of the network can be seen in the phase space $\mathcal{S}$ as a motion of a point along the curve on which the Liapunov function decreases. This downhill motion ends when the network reaches an equilibrium state which is a local or global minimum of L. The network cannot display an oscillating behavior. This is guaranteed by condition (8). The final equilibrium states are solutions of the fixed point equations

$$\sigma_i^* = g\left(\sum_j^N T_{ij}\sigma_j^* + I_i\right) \tag{16}$$

for $i = 1, \ldots, N$. This final equilibrium state is unique when the Liapunov function is strictly convex. If the second order partial derivatives of L exist then L is strictly convex if and only if the Hessian matrix $L_{ij} \equiv \partial^2 L/\partial\sigma_i\partial\sigma_j$ is positive definite. In our case

$$L_{ij} = -T_{ij} + \delta_{ij}\frac{1}{g'\left(g^{-1}(\sigma_i)\right)} < -T_{ij} + \delta_{ij}/\beta \tag{17}$$

When all eigenvalue of the matrix T_{ij} are negative then L is strictly convex function. If some of them are positive then from sufficient condition for strict convexity is given by

$$\beta < 1/\lambda_{\text{max}} \tag{18}$$

In the case of parallel discrete-time dynamics we have to satisfy simultaneously the stability condition (8). When we choose

$$\beta < 1/\lambda, \qquad \lambda = \max\{|\lambda_{\text{min}}|, |\lambda_{\text{max}}|\}, \tag{19}$$

then the Liapunov function (equation 7) is strictly convex on $\mathcal{S}$ and at the same time condition (8) is satisfied. The strict convexity implies that a local minimum of the function (7) is a global one and is unique. In this case the set of equations (16) has only one solution. This equilibrium state depends only on the position of the obstacles and on the position of the target.

Consider first the situation with static obstacles and with a static target. If the neuron in the initial configuration node j_{init} is active, then a path connecting initial position and target-position is created in the following way. A new configuration for the robot manipulator is given by the position of the neighboring neuron with the largest activity, which serves then as the starting position for a new change. Every time when a new configuration is reached a new decision is made concerning the next configuration. This procedure is repeated until the neuron corresponding to the actual position of the robot becomes the target neuron. The created path $\mathcal{P}$ passes through a sequence of points $j_{\text{init}} = j_0, j_1, \ldots, j_{\text{targ}} = j_m$. The length of the path $\mathcal{P}$ passing through these sequence of points is given by

$$D(\mathcal{P}) = \sum_{i=0}^{m-1} \rho(j_i, j_{i+1}), \tag{20}$$

In our case $\rho(j_i, j_{i+1})$ is equal to the lattice constant. Since the path leads from one node of the lattice to the neighboring node with the largest activity and ends in the node with activity "one", the number of steps and the length of the path on the grid is minimal. All other paths which pass through a sequence of points in which the activity does not increase monotonically or does not increase in the optimal way (i.e. largest increment), consist of larger number of steps and have a length larger than $D(\mathcal{P})$ since all steps have the same length. Due to the discretization of the configuration space, the path $\mathcal{P}$ has to stay on the nodes of the grid and is only an approximation of a smooth curve in $\mathcal{C}$. The degree of accuracy is only limited by the grain size of the grid.

3 Computer simulations

In this section we will illustrate some of the analytical results of the algorithm outlined above. The range of possible applications includes problems classically handled by the path planning community such as: controlling a robot arm, car manoeuvring [8] , or the piano movers problem. We have chosen four examples: an autonomous point-robot trying to find the center of a labyrinth; a two-link robot manipulator moving in a cluttered environment; a point-robot avoiding a moving obstacle in order to reach the target and a point-robot following and trying to catch up with a moving target.

In our demonstration we used a two-dimensional lattice ($d = 2$) with $N = 2500$ neurons ordered in a 50x50 neural map. So, all four examples are two-dimensional path planning problems. Note, however, that the model can easily be extended to higher dimensions and may have other types of homogeneous lattices. According to equation (4) each neuron is connected bi-directionally to the neurons which are lying within a distance r in the neural space around it. By choosing $r = 1.5$ each neuron has eight neighbors ($z = 8$). Each simulation starts with the activity of the neuron closest to the target position clamped to value "one". The activities of all neurons which correspond to obstacle positions are clamped to the value "zero".

At each time-step, neurons update their activations in parallel, according to equation (5). The changes in neuron activations stop when the network reaches an equilibrium state.

The new actual neuron is the neuron with the highest activity, chosen from the last actual neuron and its eight neighbors. If two or more of the neighboring neurons have the highest activation then one of them is chosen randomly.

Any monotonically increasing function can be used in our algorithm as the transfer function. In the simulations we chose both $g(x) = \tanh(\beta x)$ and $g(x) = \beta x$.
To chose a value of β which satisfies the condition (19), we have to estimate the largest and the smallest eigenvalue of the matrix $\mathbf{T}$. Since each neuron is connected to eight neighbors, the number of nonzero elements in each row or column is at most equal to eight. The nonzero elements are all equal to one. From Frobenius theorem (see [4]) it follows then that all eigenvalues $\lambda \in [-8, 8]$. In the absence of obstacles and when we impose periodicity of the neural map, all eigenvalues are positive and the largest eigenvalue is equal to eight. The matrix $\frac{1}{8}\mathbf{T}$ in this case is a double stochastic matrix. So we have to choose $\beta \in [0, 1.25\rangle$ to guarantee the stability condition and uniqueness of the equilibrium state.

For both, the linear and tanh(.) input/output behaviour of the neurons, we chose $\beta = 0.1$. The paths generated by these functions are the same. The computational time however, is far more shorter in the linear case.

An optimal path can be found even before the network settles into an equilibrium state. In our simulations the configuration of the robot manipulator starts to change at the moment when one of the neighboring neurons of the actual neuron is activated by the target neuron. This is essential when the trajectory must be planned in a changing environment. To understand that planning can start before an stable state is reached, we consider the activity in the neural map at equilibrium. The shape of the activity in the map is a hill with the top located at the target neuron and activity zero at the obstacle neurons. In this activation pattern lines of iso-activation can be defined. The structure and shape of these lines depend only on the location of the target and obstacles. Assume that target and obstacles are fixed. At time zero only the target neuron has a non-zero activity. Initially, due to the dynamics of the neurons, all neurons located on the first iso-activation line receive input from the target neuron and are activated. In the following moment they raise their activity and send information to the neurons on the second iso-activation line. This will go on, and the activity of neurons on successive iso-activation lines is raised from zero until the activation landscape is stable. During this dynamics the shape of the iso-activation lines is constant and the amplitude of activity decreases with increasing number of the iso-activation line. Hence, the maximal gradient on every position has a constant direction from the moment when the activation front reaches that location.

3.1 The labyrinth

In the first simulation we test the performance of the network in the complex environment of a labyrinth. Each configuration of a point-robot moving in a 2-dimensional workspace is described by the two position coordinates x and y. In the case of the point-robot the configuration space is isomorphic to the 2-dimensional work space. Hence the topological neural map represents a discretization of the workspace. Each neuron in it corresponds to a small unit area in the workspace. Neighboring neurons correspond to neighboring

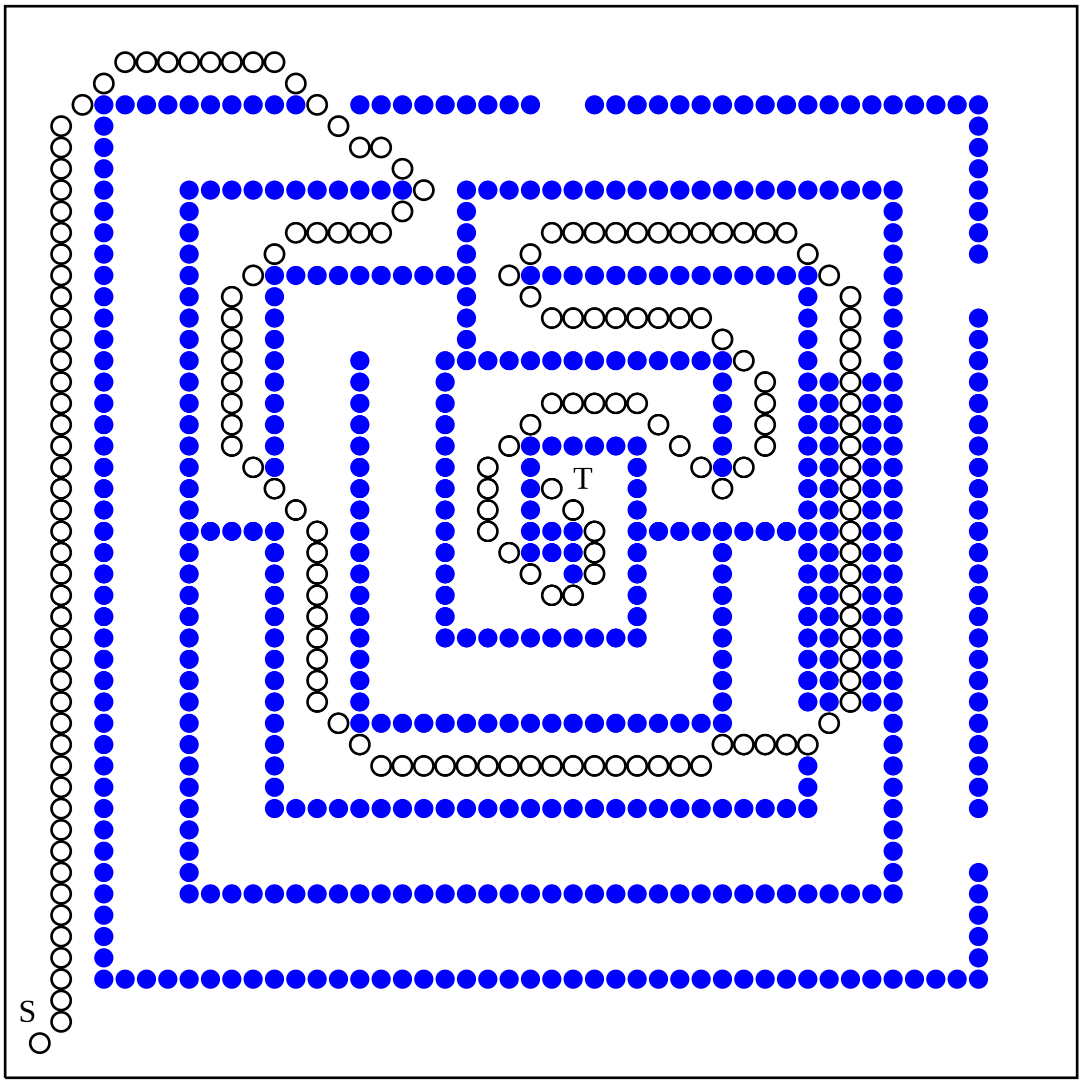

Figure 1 The labyrinth. Black dots represent neurons corresponding to walls of the labyrinth. The sequence of open circles gives the path generated by the network. These dots correspond also to the actual neuron at the successive time-steps. The initial position is at point S, the target-position at point T.

areas. By correspondence, neurons in the neural map represent the target-position, the initial actual position and the obstacles. We simulated several situations. In each case, if a continuous path between target and initial position existed, the neural net found it. In the case that more than one solution was possible, the one with the shortest path-length was chosen. In figure 1 we show the neural map with a set of obstacle neurons which corresponds to labyrinth walls. The initial actual position is chosen outside the labyrinth at point S and the target position is chosen in the center of the labyrinth, at point T. The obstacle neurons are represented by black dots and the neurons corresponding to the actual position in successive time-steps by open circles. Note that several paths may lead to the target position, but that the path generated by the network has the shortest length.

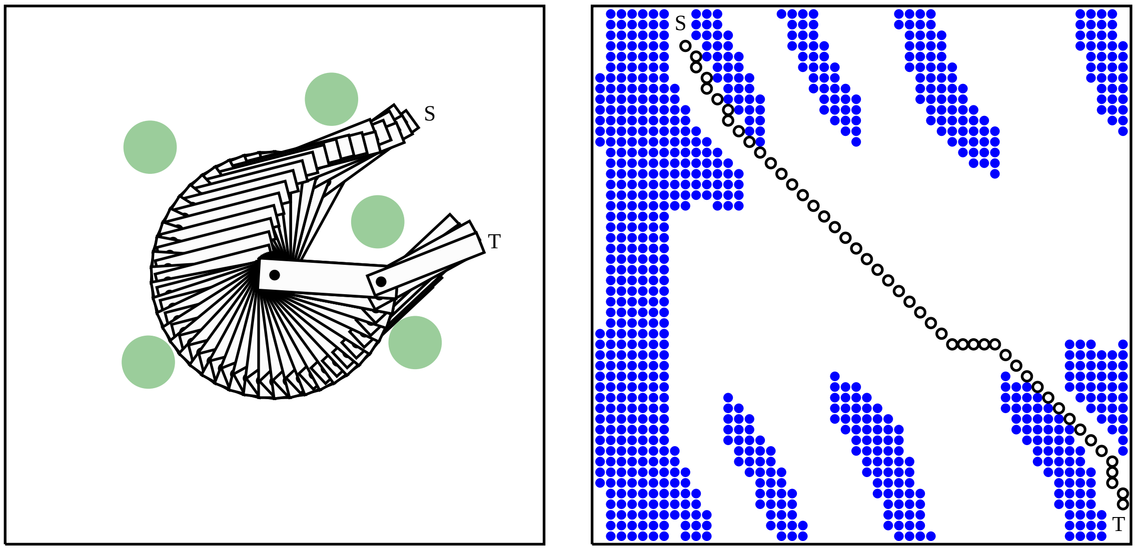

Figure 2 Two link robot manipulator. A) Representation of the workspace. Gray circles represent obstacles. The robot manipulator is drawn in successive positions. B) The configuration space. Black dots represent the neurons corresponding to the obstacles. Open circles show the actual neuron at successive time-steps.

3.2 Planar robot

In this example we will consider a two-joint robot arm moving in a 2-dimensional workspace in the presence of obstacles. In contrast with the point-robot all objects have real physical dimensions. The configuration of the robot arm is described by two joint angles θ_1 and θ_2. We assume no constraints on the joints. In this case, we impose periodic boundary conditions: the neurons on the border of the map are connected to the neurons on the opposite border. The 2-dimensional map is topologically equivalent to a torus and the joint angles can vary from 0 to 2π periodically. Both joints can move simultaneously or move only one at a time. The optimal path generated by the system is given by the minimum joint motion of the robot arm in the workspace.

In workspace we can choose an initial configuration of the two-link manipulator, a target-configuration and the obstacle-positions. The shape of the obstacles in the configuration space depends on the exact shape of the robot arm.

In figure 2A the workspace with the two-link manipulator is presented. The gray circles are obstacles. The manipulator has been drawn at each time-step, illustrating the movement. In figure 2B we show the corresponding neural space which is a discretization of the configuration space. The black dots represent the neurons which correspond to the forbidden configurations of the manipulator. The open circles, drawn in figure 2B, show the successive actual neurons which correspond to the manipulator-configurations at every time-step.

3.3 Changing environment

3.3.1 Moving Target

Consider a situation in which the target is moving and the robot's task is to follow and to try to catch it. In this case the sensory information has to be processed continuously in order to update the network for changes in the environment. The activity of neurons

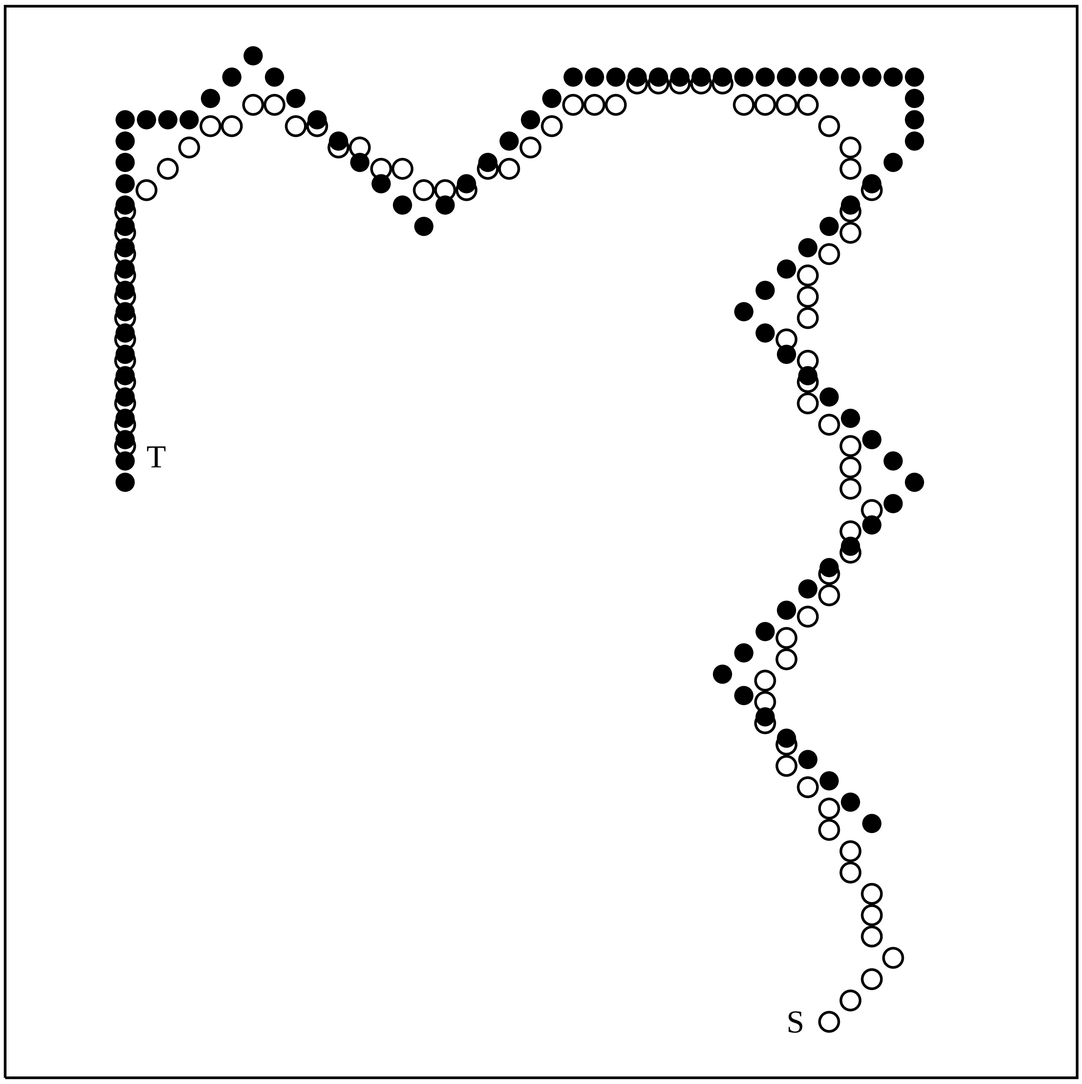

Figure 3 The pursuit. Black dots represent the target-position in successive time-steps. Open circles show the actual position at successive time-steps.

will change all the time in the direction of the equilibrium configuration imposed by the instantaneous position of the obstacles and the target. The path, as in the static case, is determined by the neural activity gradient. The difference with the static environment is that the network can not settle in an equilibrium state since the neuronal representation of target and obstacles will change all the time. This will influence the direction of the gradient which will change not only from node to node but also change with time for a given node. This gradient contains the relation between neural activities and the direction of movement. A collision free path is generated only when the speed of the moving obstacles is smaller than the rate of change of neural activities.

As an example of how the network generates a path in this case, we displaced the target by one step, after each iteration. Generally, the path made by the network is always shorter than the trajectory of the target (see figure 3). This gives the robot the opportunity to catch up with the target, even if both have the same speed.

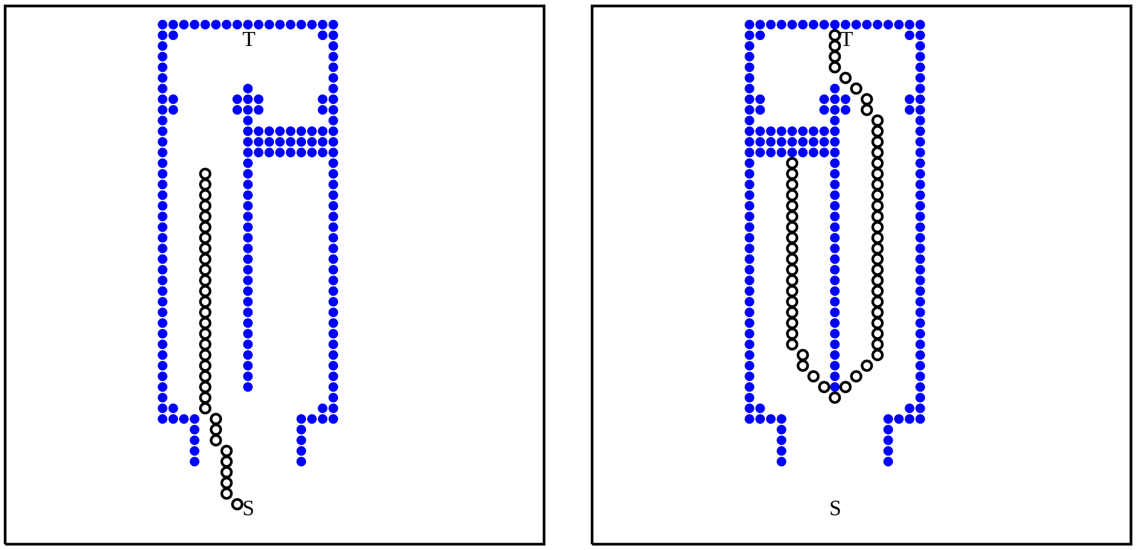

Figure 4 Moving Obstacle. A) Black dots represent neurons corresponding to the labyrinth walls and the obstacle. Open circles show the actual position of the point robot at successive time-steps from the start until the moment when the obstacle is moved. B) When the open pathway is blocked, the robot changes its direction towards the new open "gate".

3.3.2 Moving obstacles

In addition to avoiding static obstacles, path planning must also include collision avoidance with moving objects. To illustrate this we consider the case in which the point-robot can reach the target in two ways. At the beginning of the simulation one path is blocked. During the simulation, after the robot went along a substantial part of the path (see figure 4A), we move the obstacle to obstruct this path and free the other. This causes a rapid change of neural activities and the direction of the neural activity-gradient. As a result the robot reverses and reaches the target via the other path (see figure 4B).

4 Conclusions

We have proposed a model for path planning and obstacle avoidance based on an analog neural network. The network is a large collection of locally and symmetrically connected elementary processors. The evolution of the network is given by parallel discrete- or continuous-time dynamics. When the network receives an external input, the neurons in the network start to change their activity towards a specific value which corresponds to a minimum of the Liapunov function. The direction of the motor response, the path, is determined by the neural activity gradient.

Our neural network algorithm to compute optimal feasible paths is an activity wave propagation, analogous to the Huygens principle. A distance transform method [15] is an example of this kind of algorithm used in robotics for path planning. However, our network can more easily respond to sudden changes in the environment, like moving obstacles and target, and with the possibility for hardware implementation the neural network is more powerful than the distance transform.

Our method resembles Dijkstra's algorithm of finding the lengths of the shortest paths from a given vertex (target neuron) to all the remaining vertices of the graph. However the time taken by Dijkstra's algorithm on a graph with N vertices is $O(N^2)$ and in our method is $O(N)$.

Four points are worth noting about the functional properties of the network. First, pilot computer experiments demonstrated that the system could remain functional despite of local damages in the network. The quality of the path will be somewhat degraded, but it will still be a feasible path. Secondly, we want to stress the network's insensibility to the choice of the transfer function. The system is able to perform correctly, even if different transfer functions are used for the neural map. Thirdly, continuous neurons eliminate oscillations related to parallel dynamics of discrete neurons [21]. Fourth, the parallel architecture and dynamics of the model give it the large speed needed for real-time processing of sensory information because all neurons simultaneously and continuously change their analog states. Since the basic idea of the network can be expressed by electrical circuits, modern technologies should provide the possibilities to implement the network with a large number of processing elements in hardware. Graf et al [10] ware able to make custom chips with $N = 256$ fully connected units, using $2N^2 \approx 130,000$ resistors. Since in our network all connections are positive and local the number of resistors required will be of the same order as the number of neurons in the network.

Acknowledgments

This work was partly supported by the Dutch Foundation for Neural Networks and by the Dutch Foundation for Scientific Research (NWO).

References

[1] J. Barraquand, B. Langlois, and J.C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22:224–241, 1992.

[2] J. Barraquand and J.C. Latombe. Robot motion planning with many degrees of freedom and dynamic constraints. In *Proceedings of the 5th International Symposium on Robotics Research (Tokyo).*, pages 74–83, 1989.

[3] J. Barraquand and J.C. Latombe. A monte-carlo algorithm for path-planning with many degrees of freedom. In *Proceedings of the IEEE International Conference on Robotics and Automation (Cincinnati, OH).*, pages 1712–1717, 1990.

[4] E. Bodewig. North-Holland, Amsterdam, 1959.

[5] M. A. Cohen and S. Grossberg. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:815, 1983.

[6] C.I. Connolly, J.B. Burns, and R. Weiss. Path planning using Laplace's equation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2102–2106, 1991.

[7] J.L. Crowley. Navigation for an intelligent mobile robot. *IEEE Journal of Robotics and Automation*, RA-1:31, 1985.

[8] L. Dorst, I. Mandhyan, and K. Trovato. The geometrical representation of path planning problems. *Robotics and Autonomous Systems*, 7:181, 1991.

[9] B. Fritzke. Unsupervised clustering with growing cell structures. In *Proceedings of the International Joint Conference on Neural Networks (Seatle/WA).*, volume II, pages 531–536, Piscataway, NJ, 1991. IEEE.

[10] H. P. Graf, L. D. Jackel, R. E. Howard, B. Straughn, J. S. Denker, W. Hubbard, D. M. Tennant, and D. Schwartz. VLSI implementation of a neural network memory with several hundreds of neurons. In J. S. Denker, editor, *Neural Networks for Computing (Snowbird 1986).*, page 182, New York, 1986. American Institute of Physics.

[11] S. Grossberg. Nonlinear neural networks: principles, mechanisms, and architectures. *Neural Networks*, 1:17, 1988.

[12] J. Heikonen, P. Koikkalainen, and E. Oja. From situations to actions: motion behavior learning by self-organization. In *Proceedings of the International Conference on Artificial Neural Networks (Amsterdam/The Netherlands)*, pages 262–267, Amsterdam; North-Holland, 1993.

[13] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79:2554, 1982.

[14] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the United States of America*, 81:3088, 1984.

[15] R. A. Jarris. Collission-free trajectory planning using distance transforms. *Mech Eng Trans of the IE Aust.*, ME10:187, 1985.

[16] K. Kant and S.W. Zucker. Towards efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research*, 5:72–89, 1986.

[17] O. Kathib. Real-time obstacle avoindance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5:90–98, 1986.

[18] B.H. Krogh. A generalized potential field approach to obstacle avoidance control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, page 948, 1984.

[19] B.H. Krogh and C.E. Thorpe. Integrated path planning and dynamic steering control for autonomous vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1664–1669, 1986.

[20] T. Lozano-Perez. Spacial planning: a configuration space approach. *IEEE Transactions on Computers*, C-32:108–120, 1983.

[21] C. M. Marcus, F. R. Waugh, and R. M. Westervelt. Associative memory in an analog iterated-map neural network. *Physical Review A*, 41:3355, 1990.

[22] T. M. Martinetz. Competitive hebbian learning rule forms perfectly topology preserving maps. In *Proceedings of the International Conference on Artificial Neural Networks (Amsterdam/The Netherlands)*, pages 427–434, Amsterdam; North-Holland, 1993.

[23] P. Morasso, V. Sanguineti, and T. Tsuji. Neural architecture for robot planning. In *Proceedings of the International Conference on Artificial Neural Networks (Amsterdam/The Netherlands)*, pages 256–261, Amsterdam; North-Holland, 1993.

[24] W.S. Newman and N. Hogan. High speed robot control and obstacle avoidance using dynamic potential function. In *Proceedings of the IEEE International Conference on Robotics and Automation (Raleigh, NC)*, pages 14–24, 1987.

[25] H. J. Ritter, T. M. Martinetz, and K. J. Schulten. Topology-conservingmaps for learning visuo-motor-coordination. *Neural Networks*, 2:159–168, 1989.

[26] J. T. Schwartz and M. Sharir. On the "piano" movers problem. II General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, pages 298–351, 1983.

[27] C. Seshadri and A. Ghosh. Optimum path planning for robot manipulators amid static and dynamic obstacles. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:576–584, 1993.

[28] J. M. Vleugels, J. N. Kok, and M. H. Overmars. A self-organizing neural network for robot motion planning. In *Proceedings of the International Conference on Artificial Neural Networks (Amsterdam/The Netherlands)*, pages 256–261, Amsterdam; North-Holland, 1993.

[29] Y. Wada and M. Kawato. A neural network model for arm trajectory formation using forward and inverse dynamics models. *Neural Networks*, 6:919–932, 1993.

[30] C.W. Warren. Global path planning using artificial potential fields. In *Proceedings of the IEEE International Conference on Robotics and Automation (Scottsdale, AZ)*, pages 316–321, 1989.

[31] C.K. Yap. Hillsdale, N.J., Erlbaum, 1986.