PLANNING AND LEARNING GOAL-DIRECTED SEQUENCES OF ROBOT ARM MOVEMENTS

Kaspar Althöfer[*] and Guido Bugmann[†]

[*]Department of Electronic and Electrical Engineering, King's College London,
Strand, London WC2R 2LS, United Kingdom
k.althoefer@bay.cc.kcl.ac.uk, tel (+44) 171 836 54 54 (ext 3656)
[†]Neurodynamics Research Group, School of Computing, University of Plymouth,
Plymouth PL4 8AA, United Kingdom.
gbugmann@soc.plym.ac.uk, tel (+44) 1752 23 25 66

Abstract: *This paper describes two new types of neural networks. Firstly, a neural implementation of a resistive grid for path planning. Its advantages and limitations are discussed using the example of a 2-joint robot arm. Two of the limitations are the jerkiness of the movements and the inaccurate end-position. Secondly, as a solution to these two problems, a new type of sequence learning neural network is proposed. This 2-layer network can learn in one pass the sequence of movements defined by the resistive-grid. It uses RBF nodes with receptive fields centered on a sequence of starting positions in the configuration space of the arm, and with weights to the output layer being used to point to the next position in configuration space. The output layer uses a new type of node with output activity being equal to the activity-weighted average of the input weights. This network generates a smooth trajectory of the arm and an accurate end position.*

1. INTRODUCTION

Object grasping is likely to be one of the tasks of future mobile autonomous robots. When obstacles have to be avoided by the arm, simple ballistic trajectories are not possible and the robot is faced with a complex planning problem. One of the best planning methods is the resistive grid method, also called Laplacian path planning [1]. It has been introduced to solve the problem of local minima shown by previous methods. A neural network implementation of this method has been proposed recently [2]. In section 2, its properties are discussed in the case of the control of a 2-joint robot arm. In section 3, a new neural network for sequence storage and replay is described. It allows a smoother and precise control of the robot arm. In section 4, a solution to the problem of an excessively large number of node required by high dimensional state spaces is outlined.

2. PLANNING WITH A NEURAL RESISTIVE-GRID

To use a resistive grid for planning, the state space must be divided into a set of small N-dimensional cubes. Each cube corresponds to a node in the resistive grid. Each node is connected to its $2N$ neighbors. Planning can be performed by setting the potential of the node corresponding to the goal state to a high value, e.g. 1. The nodes corresponding to obstacles, or forbidden states, are set to a potential zero. Electric current flows from the goal, through the grid and towards the obstacles. At any point in the grid, the direction of the current flow indicates the shortest path to the goal.

[*] The research carried out by the first author was supported by the Dr. Jost Henkel-Stiftung of Henkel KGaA, Düsseldorf, Germany.

The resistive-grid method differs from commonly called potential field methods. In these methods, the goal is the source of an omnidirectional attractive field and obstacles produce repulsive fields. The local field is obtained by a vector sum of the fields and its direction determines the direction of the next action. This method is plagued by local minima problems. A typical example is a path planning problem where the goal is placed behind a U-shaped obstacle [2]. When a mobile robot is inside the U, it is equally attracted by the goal behind the lower bar of the U and repelled by the 3 bars of the U, and cannot move (see also references in [1]).

In contrast, in a resistive grid, the direction of the local potential gradient reflects the direction of the electric current flowing along a path from the goal to the nearest obstacles. See ref. [2] for an illustration of the maps of current lines. If there is current flowing locally, then there exists a path to the goal. Therefore, the resistive-grid method is very robust. It fails to provide a direction only in particular situations where any action is equally good. This is then not a failure of the method but a property of the problem.

The principle of path planning by the resistive grid-method was initially formulated over a continuous conductive medium and involved the solution of the Laplace equation with suitable boundary conditions [1]. Modeling obstacles as current sinks, as done in this paper, corresponds to the Dirichlet boundary condition. Modeling obstacles as insulating regions corresponds to the Neumann boundary condition. The two boundary conditions are compared in [6,2]. In a neural implementation, the Dirichlet condition bears a smaller computational cost.

In a neural resistive grid [2], one neuron is assigned to each node. The neuron sets its output y_j to the average value of the outputs of its neighbors y_i:

$$y_j = f\left(\sum_i W_{ij} y_i\right) + I_j \qquad (1)$$

where

$$f(x) = 0 \ \ for\, x < 0; \quad f(x) = x \ \ for\, x \in [0,1]; \quad f(x) = 1 \ \ for\, x > 1.$$

After all neurons in the grid have been updated a large number of times, their outputs approximate the potential distribution in a physical resistive grid. It is shown in [2] that, in the limit of a very large number of nodes in the grid, the updating rule (1) is equivalent to the Poisson equation:

$$\nabla^2 y = -bI. \qquad (2)$$

Where $b=2N/\Delta r^2$ and Δr is the horizontal and vertical distance between nodes. The use of a linear saturating transfer function $f(x)$ in (1) allows external control to be exerted on the resistive grid by the input I_j from the map[1] layer in Figure 1. Each neuron in that layer has only one output which is connected to the neuron in the resistive grid with corresponding location. By setting the output of the goal node in the map[1] to 1 (white node in the Figure), it forces the goal node in the neural resistive grid

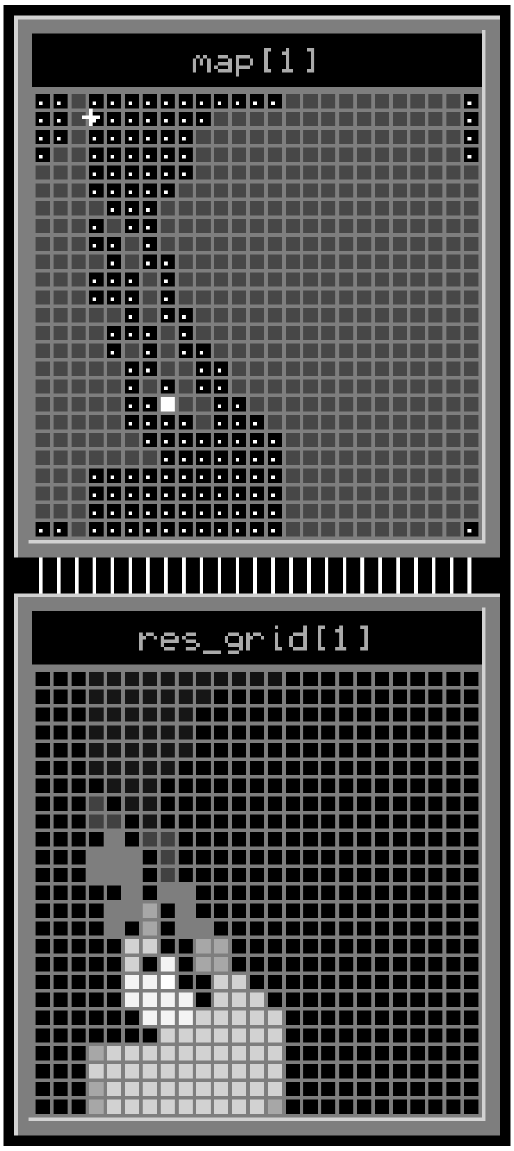

Figure 1. Neural resistive grid (bottom) and its control layer map[1] (top). The initial arm configuration is indicated by a cross in map[1]. The goal is indicated by the white node.

to have an output 1. Similarly, setting the activity of obstacles nodes in map[1] to -1 (gray nodes in the Figure) will force their counterparts in the neural resistive grid to have outputs zero. This flexible architecture allows to deal with any configuration of obstacles and goal(s).

The 2-dimensional resistive grid in Figure 1 represents the configuration space of a 2-joint robot-arm which is shown in Figure 2. A point in the configuration space represents a configuration of the robot arm. A path in the configuration space represents a sequence of arm configurations. The arm in this example has only two mobile joints, so its configuration space is two-dimensional. In Figure 1, the x-axis is the shoulder joint angle and the y-axis is the elbow joint angle. Those joint angles which correspond to collisions with walls and obstacles are represented by forbidden positions in the grid. Forbidden positions can be calculated using data from visual exploration of the workspace [3]. The goal node corresponds to a configuration where the grip is in contact with the filled circle in Figure 2.

A planned sequence of movements is obtained by: i) reading out the direction of the potential gradient at the neuron corresponding to the current state, ii) selecting the next state, iii) executing the corresponding movement. The resulting trajectory in configuration space is shown in Figure 3, and the resulting arm trajectory in the workspace is shown in Figure 4.

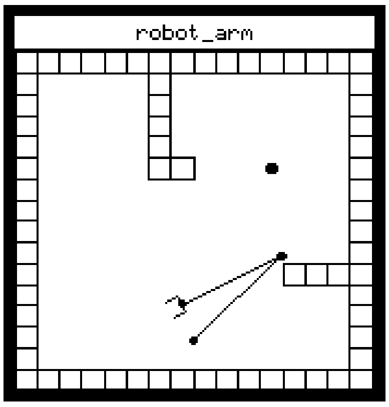

Figure 2. Robot arm in its initial position in the workspace.

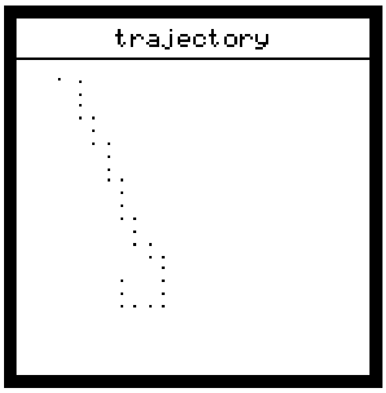

Figure 3. Trajectory in configuration space. The x-axis is the shoulder joint angle. The y-axis is the elbow joint angle.

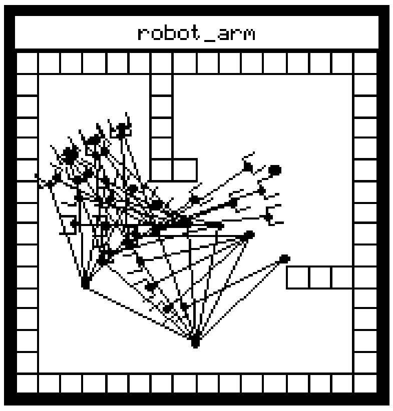

Figure 4. Sequence of arm positions corresponding to the trajectory in Figure 3.

The main advantage of the resistive-grid method is that it guarantees finding a solution to the planning problem, if there is one. However, it has also the limitations of grid-based methods: a limited resolution. In Figure 1, a 25 x 25 grid is used to represent all possible combinations of the two joint angles. As the joints can rotate by a full 360 degree, there is a 14.4 degree angular step change from one node to the next. This has two consequences, i) the planned movement is very saccadic (Figure 3 and 4), ii) the goal is not reached (Figure 4) because its exact location does not necessarily correspond to the centre of a node in the grid. It is indeed possible to increase the resolution of the grid but at the cost of a higher computational load. Another solution is to use the sequence-learning network presented in the next section, which interpolates smoothly between the discrete sequence steps produced by the resistive grid.

The neural implementation of the resistive grid is very flexible due to the possibility of incorporating knowledge in the form of a temporary activation of nodes in the control layer (map[1] in Figure 1).

Mobile robot navigation has been considered as an application of resistive grids [2]. However, as soon as long distances and a high-resolution are required, grids become intractably large in size. The control of a robot arm with a bounded configuration space is probably a better application for a resistive grid.

3. NEURAL NETWORK FOR SEQUENCE LEARNING

The sequence learning/replaying network uses current values α_1, α_2 of joint angles as input and produces the next set of values α'_1, α'_2 in the sequence as output.

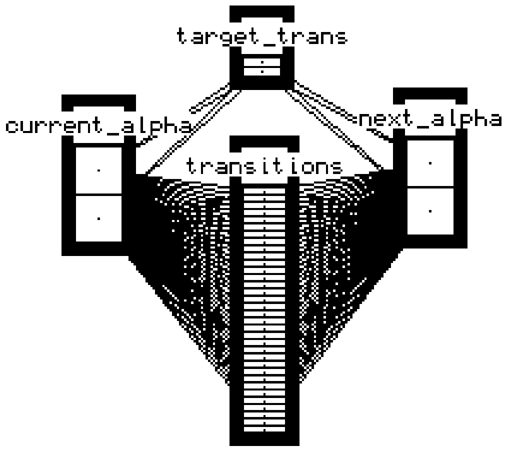

Figure 5. Neural network for sequence learning and replay.

The network has two active layers, as depicted in Figure 5. The hidden layer, named "transitions", is composed of N_h neurons with Radial Basis Functions (RBF) [4]. In our network, these "neurons" have two joint angles α_1 and α_2 as input. They respond maximally when these angles are equal to preset values α^j_{10} and α^j_{20} (the centre of their "receptive field" in the configuration space). The width of the tuning of their response y_j (or size of the receptive field) is determined by σ_j :

$$y_j = \exp[-\frac{1}{2\sigma_j^2}\sum_i(\alpha_i^j - \alpha_{i0}^j)^2]. \qquad (3)$$

These neurons project with weights W_{ij} to the output layer ("next_alpha" in the Figure 5). The output layer comprises two nodes with outputs α'_1 and α'_2. These outputs are calculated as follows:

$$\alpha'_i = \frac{\sum_j W_{ij} y_j}{\sum_j y_j} \qquad (4)$$

This operation corresponds to the weighted average over the input weights, where the input activities are used as weights for the weighted average. It is possible to approximate such a function with biological neurons, assuming feedforward inhibition. The function (4) allows the encoding of the values of the next angles in the weights from hidden to output layer.

The network is trained during the read-out of a sequence of movements as determined by the resistive grid. At each transition from one node n_t to the next, n_{t+1}, a new RBF node is recruited, its receptive field is centred on the starting point in configuration space corresponding to the node n_t, and its *weights* to the output layer are set to the values of the joint angles corresponding to node n_{t+1}.

During replay, the output of this network alone is used to control the arm movement (The resistive grid is not used). As soon as one configuration has been reached, the network provides the next values of joint angles. As RBFs have some overlap, a small number of nodes in the hidden layer are simultaneously active and may point to different future configurations. However, the function (4) of the output nodes allows the most *active* of the inputs to have the largest *weight* in the decision.

To solve the problem of inaccurate end position, two extra-nodes "target_trans" are used (Figure 5). These nodes have receptive fields centred on the two possible final configuration of the arm, and have their output weights pointing to these exact final configurations. (There are two combinations of joint angles allowing the wrist to touch the filled circle, but one turns out to be forbidden, because the arm would have to go through the short wall protruding from the right in Figure 2). Each of these "target_trans" nodes is linked by 5 connections to the output nodes. The number 5 is arbitrary. It must just be is large enough to give a dominant weight to the angles indicated by the "target_trans" nodes. Thereby, when the arm approaches one of the selected final configuration, it becomes strongly attracted to the exact final position.

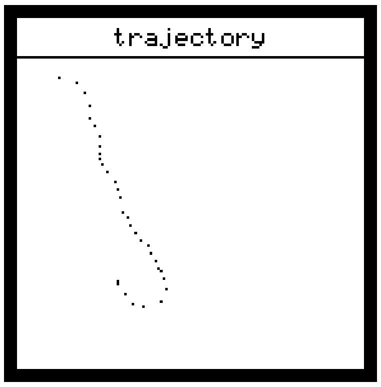

Figure 6. Trajectory in configuration space produced by the trained sequence learning network.

Figure 6 shows the trajectory in configuration space resulting from a movement controlled by the sequence learning/replaying network. Figure 7 shows the corresponding sequence of positions in workspace. One may note the much smoother trajectories and the correct final position of the arm.

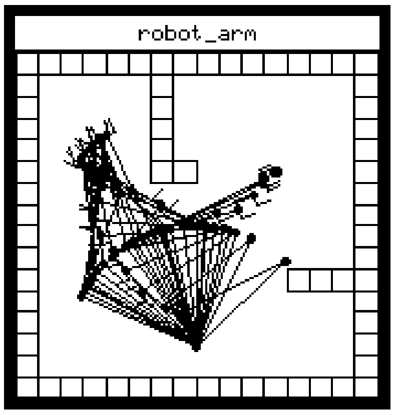

Figure 7. Sequence of arm positions corresponding to the trajectory in Figure 6.

The sequence is not replayed with the same number of steps as the sequence used for the training (Figure 3). In practice, we calculate the next position of the arm using current inputs. Then we set the arm to its new position, which modifies the inputs to the RBF nodes, and calculate the new position. The speed at which the complete movement is performed depends on the size of the receptive fields of the RBF nodes. If these are too large, there is little change in the response of the RBF nodes after a movement and, conversely, little change in the calculated next position. If the receptive fields are too small, the smoothness of the trajectory is lost. For the results shown here we used $\sigma = 2\pi/25$.

4. CONCLUDING REMARKS

We have briefly described here one form of a neural resistive grid. A more complete discussion of this approach to path planning can be found in reference [2]. From an application point of view, an arm with two joints is probably not of great interest. Most industrial arms have 6 degrees of freedom, and biological arms many more. Theoretically, a resistive grid can be designed with any dimension. This is especially easy in a simulated neural implementation (We have been using the simulation package CORTEX-PRO). For instance, a 4-dimensional resistive grid has recently been applied to the pole-balancing problem [5]. Due to a very coarse discretisation of the state space, this resistive grid required only 162 nodes.

If we use 25 discrete values for each degree of freedom, as in Figure 1, we would end up with 244 million nodes for a 6-dimensional problem.

This dimensionality problem can be avoided if planning does not require simultaneous access to all dimensions at the same time. For instance, for the arm shown in Figure 2, it would be reasonable to plan a trajectory by assuming a closed hand during most of the trajectory and fine hand control only towards the end of the trajectory. Such a scenario can theoretically be implemented with a sequence learning/replaying network, where hand configurations are modified as the arm trajectory progresses.

5. REFERENCES

[1] Connolly C.I., Burns J.B. and Weiss R. (1990) "Path planning using Laplace's equation", Proc. IEEE Int. Conf. on Robotics and Automation, 2102-2106.

[2]. Bugmann, G., Taylor, J.G. and Denham M. (1994) "Route finding by neural nets" in Taylor J.G (ed) "Neural Networks", Alfred Waller Ltd, Henley-on-Thames, UK, p. 217-230.

[3] Althöfer, K., Fraser, D.A., Bugmann, G. and Plumbley, M.D. (1995) "Asymetric-B-Splines for the fast Calculation of C Space Patterns of Robot Arms", Proc. of the International Conference on Neural Networks (ICANN'95), Oct. 9-13, Paris.

[4] Moody, J. and Darken, Ch. (1989) "Fast learning in networks of locally-tuned processing units.", Neural Computation, 1, 281-294.

[5] Bapi, R., D'Cruz, B. and Bugmann, G. (1995) "Neuro-Resistive Grid Approach to Pole-Balancing Problem", Proc. of the International Conference on Neural Networks (ICANN'95), Oct. 9-13, Paris.

[6] Tarassenko L. and Blake A. (1991) "Analogue computation of collision-free paths", Proc. IEEE Int. Conf. on Robotics and Automation, 540-545.