THE CONFIGURATION SPACE TRANSFORMATION FOR ARTICULATED MANIPULATORS: A NOVEL APPROACH BASED ON RBF-NETWORKS

K Althöfer, D A Fraser, G Bugmann[†], J Turán[‡]

King's College, University of London, UK - [†]University of Plymouth, UK - [‡]Technical University of Košice, Slovakia

ABSTRACT

This paper proposes a neural-network-based method for the computation of the configuration space for robotic manipulators. The configuration space can be obtained by repeatedly computing configuration space patterns for elementary obstacle primitives. For any manipulator, these patterns depend only on the distance between the base of the manipulator and the obstacle primitive. An RBF-network is trained to recognise the distance of an obstacle primitive and to respond with the associated pattern. The interpolating features of radial basis functions are exploited to achieve a good approximation for untrained patterns. The main principles of the method are explained for a two-link manipulator. Training results are reported.

1 INTRODUCTION

Many path planning strategies and obstacle avoidance techniques for robotic manipulators presented in literature (including those based on neural techniques) [1, 2, 3, 4, 5, 7, 8] make use of the configuration space (C-Space) representation of the robot. To utilise those strategies in a practical application, a transformation from workspace into C-Space becomes necessary. This paper introduces a novel neural network method based on radial basis functions to accomplish this task.

To cope with the real-time constraints which appear during on-line path planning, the C-Space has to be computed rapidly. One way of achieving this is to store the configuration space representations (C-Space patterns) of primitive elements [2, 3, 4, 5]. The occurrence of a primitive in the workspace triggers the retrieval of the corresponding pattern. Further, the C-Space of a complex workspace obstacle can be composed by unifying the C-Space of those primitives which make up the complex object in the workspace.

Newman et al. [3, 4] suggest the use of a bitmap to store the configuration space as well as conventional techniques to compress the data. Our paper expands on their research by suggesting a neural network architecture which is trained on C-Space patterns of workspace primitives. In contrast to conventional approaches our technique interpolates for those patterns which are not part of the stored data set. The access to the stored patterns can be accomplished without a time-consuming decompression step.

Some path planning strategies exploit properties of the robot′s workspace; however, as it has been reported [2, 3, 4], the advantage of using the C-Space for path planning is that every point in C-Space is mathematically well defined, hence, there are no problems with redundancy, multiple solutions, undefined states and singularities.

In the following section we give a description of the C-Space computation for an articulated manipulator. We explain the structure of the used network and the training method. Finally, we report results.
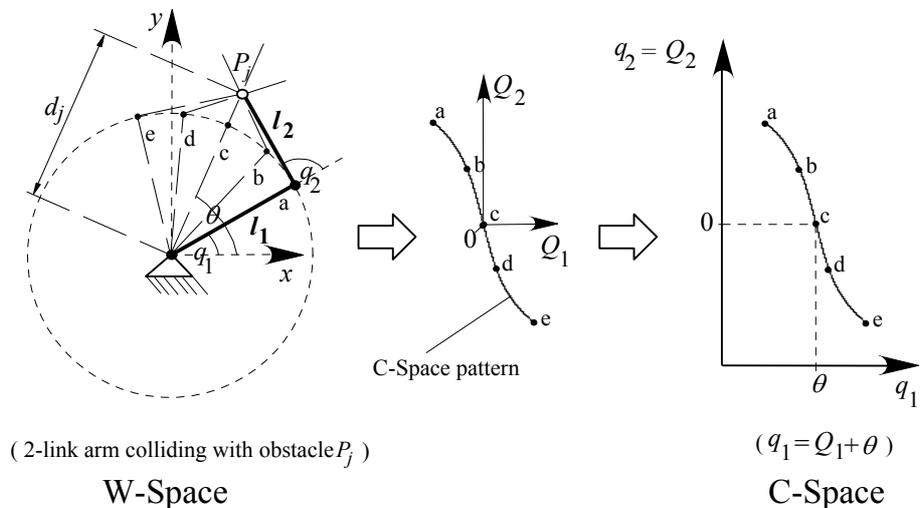
2 DEFINITIONS

(2-link arm colliding with obstacle P_j)

W-Space

($q_1 = Q_1 + \theta$)

C-Space

Figure 1: Depiction of a two-link revolute manipulator (ratio of links: $l_2/l_1 = 0.75$) in its W-Space and C-Space. P_j is an obstacle point in distance d_j. Collisions with P_j are transformed into a C-Space pattern. Postures "a" to "e" in W-Space transform into separate configurations in C-Space shown on the right hand side of this figure.

The basic kinematic performance of any physical manipulator in relation to obstacles located in its range can be investigated by analysing the motion of the manipulator's skeleton. The skeleton of a manipulator, as understood in this paper, is a simplified robot arm with links which are as long as their physical counterpart, but which are shrunk to infinitesimal width. Equivalently, the joints of the skeleton arm are reduced to a point of infinitesimal expansion. Although the following explanations focus on two-link manipulators, the proposed method can be applied to manipulators with n links [7].

This paper describes the application of the proposed technique to manipulators with revolute joints $g_i \in [1,2]$. Furthermore, experimental simulations carried out show that this technique can also be applied to certain types of prismatic arms as well as arms which have a combination of revolute and prismatic joints [7].

Hence, we consider an articulated manipulator A which is composed of two rigid "stick-like" links l_1 and l_2. The first link l_1 is connected via joint g_1 to the base of the manipulator. The other end of link l_i is connected to the succeeding link l_2 via joint g_2. The free end of link l_2 is denoted as endeffector.

The workspace, or *W-Space*, is the description of the area which can be reached by any part of the manipulator. The W-Space can be divided into two subsets. The first subset is the obstacle W-Space and describes the space which is occupied by obstacles. The second subset which is the complement of the first is denoted as free space. It describes the area which is free of obstacles.

A commonly chosen configuration space or *C-Space* of a revolute manipulator is the joint space. This is the space which is described by the parameters $(q_1, q_2) \in [0,2\pi \times [0,2\pi)$. Those represent the angular displacements of the joints g_1 and g_2. The dimension of the C-Space is equal to the degree of freedom of the manipulator. A C-Space representation is the representation of a constraint imposed on any part of the manipulator due to an obstacle in W-Space [6].

3 THE C-SPACE FOR A 2-LINK ARM

The C-Space caused by the collision between an obstacle point in W-Space and any part of a planar two-link manipulator is either a 2-dimensional area or, as we consider here the links of the manipulator to be of zero width, a line (see Fig. 1 and Fig. 2). The investigation of the C-Space of point obstacles is of special interest because a point can be interpreted as the smallest unit of a discretised W-Space. It has been shown that the C-Space of a complex obstacle is the union of C-Space transforms of all the point obstacles which constitute the complex obstacle [3, 4]. Equivalently, the C-Space of the entire obstacle W-Space can be computed by unifying all the C-Space representations of the individual obstacles. The complement of the obstacle W-Space is the free space in which the manipulator can move [3, 4].
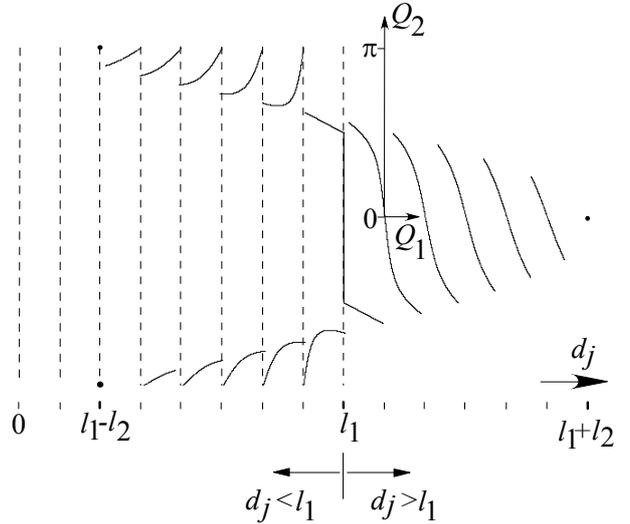

Figure 2: This figure shows C-Space patterns for a two-link manipulator like shown in Fig. 1. Dashed lines depict collisions between obstacle point and link l_1. Solid lines are patterns representing a collision caused by link l_2. The l_2-patterns shrink to points at $d_j = l_1 - l_2$ and $d_i = l_1 + l_2$, respectively. Note, the d_j-axis indicates the distance between the base of the manipulator and obstacle points P_j.

The collision of the stick-like manipulator with an obstacle point P_j results in a line in C-Space (see Fig. 1 and Fig. 2). For any manipulator, the shape of this C-Space representation depends only on the distance between base and obstacle point. Those C-Space representations we call *C-Space patterns*. A C-Space pattern is described by the joint coordinates Q_1 and Q_2. The C-Space pattern is invariant to a rotation of q_1. The centre of any pattern lies at $Q_1 = 0$, $Q_2 = 0$.

Fig. 1 shows how different postures of a manipulator are transformed into a C-Space pattern. To determine its location in configuration space the pattern is to be shifted by θ along axis q_1 (see Fig. 1). Fig. 2 shows the C-Space patterns for obstacle points at different distances d_i. Hence, if we train a network with C-Space patterns for obstacle points at discretised distances d_j we can retrieve the C-Space of any obstacle point in the range of the arm.
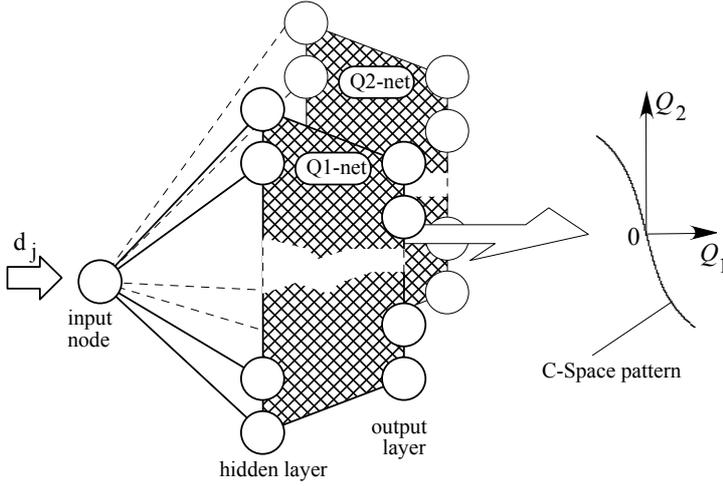
Figure 3: The network structure which is composed of the Q1- and the Q2-net. The two nets compute separately the angles Q_1 and Q_2.

4 NETWORK IMPLEMENTATION

The network structure used for the task presented in the previous sections is based on radial basis functions [10, 11]. The structure is composed of two neural nets (Q1-net and Q2-net) which have a scalar input (see Fig. 3). The two nets produce at their multiple-outputs a set of angular parameters Q_1, Q_2. This set represents the two-dimensional C-Space pattern.

The Q1-net and the Q2-net consist each of three layers. The input layer is a single node which buffers the incoming signals. This node is shared by both nets, thus, each node of the hidden layers are connected with the common input node. The nodes of the hidden layer perform a radial basis function. The nodes of the third layer form a weighted sum of the data received from the hidden layer. Both nets are fully interconnected.

The output of each node in the output layer is as follows:

$$Q_{k,i} = \sum_{j=1}^{m} w_{k,ij} \cdot B(||d_j - c_{k,j}||) \qquad (1)$$

with

$Q_{k,i}$ representing the angles Q_k, $k \in [1, 2]$ of the C-Space patterns at discretised points $i \in [1, p]$. (p represents the resolution with which each pattern is sampled.)

d_j representing the scalar network input,

$c_{k,j}$ representing the weights (or centres) of the nodes in the hidden layer,

$w_{k,ij}$ representing the weights between hidden and output layer.

Most commonly, the Gaussian function is used as radial basis function in the hidden layer [9, 10], but experiments carried out showed that the Gaussian function was not suitable for the task described here. A second order function of the family of dilated B-splines functions [12] seems to produce more promising results:

$$B(x) = \begin{cases} -|\sigma^{-1} \cdot x| + 1 & \text{if } |x| \le \frac{1}{\sigma^{-1}} \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

with σ^{-1} representing the slope.

Many training tasks described in literature are confronted with the problem of a training set with a high number of examples [9, 10, 11]. Instead of assigning each training pattern with one RBF node, different techniques have been developed to reduce the number of nodes by making them sensitive to more than just one pattern and to estimate appropriate width parameters [9, 10, 11]. However, due to the structure of the input data, the training of the networks described here is achieved easily. If m is chosen to be the resolution of the input space, then the number of hidden nodes per net can be set to m as well (see Eq. 1). Hence, for training, we simply set each centre $c_{k,j}$, $i \in [1, m]$ to a value $d_j \in [0, l_1 + l_2]$. In this experiment, the range $[0, l_1 + l_2]$ is divided into equidistant values of d_j. In other words, the nodes in the hidden layer are sensitive to obstacle points in equidistant distances from the base of the manipulator. Obviously, the training of the hidden layer is done in one iteration. The parameter σ is set in such a manner that B(.) falls to zero at neighbouring centre points, thus, the nets perform a linear interpolation for those input values which lie between two centres of hidden layer nodes.

The training of the output layer is independent of the hidden layer training. The used training algorithm is the least-means-square algorithm [9, 10]. The training data are the C-Space patterns described in section 3. The data is stored in a matrix where each column represents a C-Space pattern corresponding to a distance value d_j. Each training pattern is discretised in equidistant values $Q_{k,i}$, $k \in [1, 2]$, $j \in [1, p]$ over the range $[0, Q_{k,p}]$. $Q_{k,p}$ denotes the angular displacement where the endeffector is "touching" the obstacle point. Hence, for each net we have a (p x m)-matrix where each column represents a C-Space pattern corresponding to a distance value d_j and each element in a column represents a single posture of the manipulator.

The training of the output nodes turned out to be very fast. After only ten iterations the summed error falls below 0.0001. Fig. 6 shows the network error which is
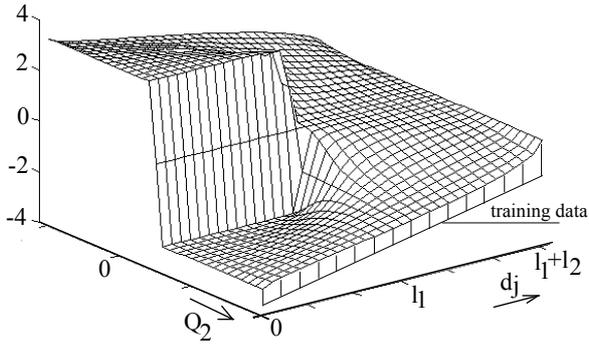
Figure 4: Graphical depiction of the $(p \times m)$-matrix representing the calculated Q2-pattern data. Every second pattern along the d_j-axis was used for the training of the Q2-net.
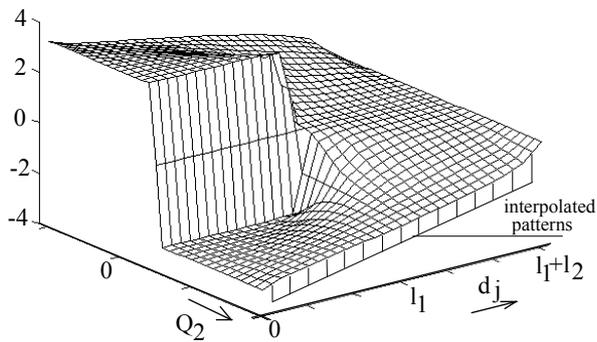

Figure 5: Graphical depiction of the $(p \times m)$-matrix showing the response of the Q2-net. The patterns denoted "interpolated patterns" are produced by the net interpolating between the neighbouring values which had been used for training.
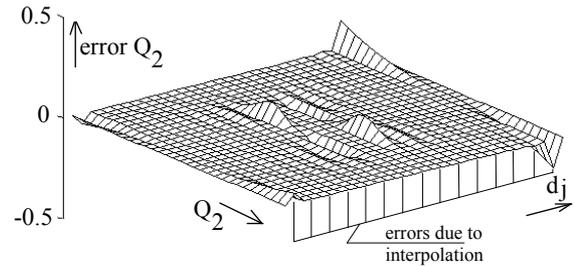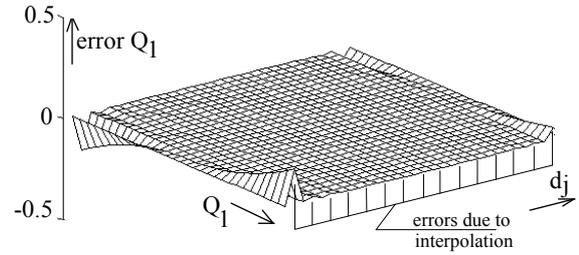

Figure 6: The error of the Q1-net (top) and the Q2-net (bottom) after ten iterations. As expected the reconstruction of patterns which had been part of the training set is done with high accuracy. For the interpolated ones a slight deviation of the error occurs especially at the edges of the data set and in the case of the Q2-net in the centre.

This paper describes a novel technique to compute the configuration space for a manipulator based on a neural network architecture. It shows promising results in this early stage of research. The training was very fast and, hence, it might be possible to have an adaptive C-Space pattern computation to incorporate changes in the structure of the manipulator. Urgent improvement will be to enhance the interpolating features of the network.

Further investigations will focus on the expansion of this approach to other obstacle primitives (circles, lines) for robotic manipulators with two and more degrees of freedom. Additionally, the applicability of the approach to a physical manipulator will be studied. Comparative studies between our method and other techniques will be carried out under real-time conditions.

the difference between the calculated values and the response of the Q1-net and the Q2-net, respectively. To illustrate the functionality of the nets a data set was presented containing d_j-values which are members of the training set as well as intermediate values. Fig. 4 and Fig. 5 show the calculated data used for the Q2-net. As expected the response to the already known values matches the calculated ones accurately. The response to intermediate values shows a slight deviation especially for values near zero in the case of the Q2-net as well as at the edges of both error presentations. The latter might be attributed to transient effects at the beginning and end of the data set.

5 CONCLUSIONS

ACKNOWLEDGEMENTS

REFERENCES

[1] Khatib O, 1985, "Real-time obstacle avoidance for manipulators and mobile robots", <u>Proceedings of the 1985 IEEE International Conference on Robotics and Automation</u>, pp. 500-505

[2] Lozano-Perez T, 1987, "A Simple Motion-Planning Algorithm for General Robot Manipulators", <u>IEEE Journal of Robotics and Automation</u>, Vol. RA-3, No. 3, pp. 224-238

[3] Newman W S, Branicky M S, 1991, "Real-Time Configuration Space Transforms for Obstacle Avoidance", <u>The International Journal of Robotics Research</u>, Vol. 10, No. 6

[4] Branicky M S, Newman W S, 1990, "Rapid Computation of Configuration Space Obstacles", <u>Proceedings of the 1990 IEEE International Conference on Robotics and Automation</u>, pp. 304-310,

[5] Meyer W, Benedict P, 1988, "Path Planning and the Geometry of Joint Space Obstacles", <u>Proc. IEEE International Conference on Robotics and Automation</u>, PA, pp. 304-310

[6] Latombe J C, 1990, "Robot Motion Planning", Boston, MA: Kluwer Academic, USA

[7] Althöfer K, 1994, "On the C- Space Transformation for Planar Manipulators with n Links", <u>Internal Report</u>, Oct. 1994

[8] Bugmann G, Taylor J G, Denham M J, 1995, "Route finding by neural nets", in <u>Neural Networks</u>, Taylor J G (ed.), Alfred Waller Ltd., Henley-on-Thames, pp. 217-230, UK

[9] Hush D R, Horne B G, 1993, "Progress in Supervised Networks - What's New Since Lippmann?", <u>IEEE Signal Processing Magazine</u>, Jan. 1993

[10] Musavi M T, Ahmed W, Chan K H, Faris K B, Hummels D M, 1992, "On the Training of Radial Basis Function Classifiers", <u>Neural Networks</u>, Pergamon Press Ltd., Vol. 5, pp. 595-603

[11] Hlavackova K, Neruda R, 1993, "Radial Basis Function Networks", <u>Neural Network World</u>, Journal 1/93

[12] Brown M, Harris C, 1994, "Neurofuzzy Adaptive Modelling and Control", Prentice Hall International (UK) Limited, UK