Department of
AUTOMATIC
CONTROL
and SYSTEMS
ENGINEERING
UNIVERSITY of SHEFFIELD

A Neural Network Based Collision Detection Engine for Multi-Arm Robotic Systems

A.S. RANA and A.M.S. ZALZALA

Robotics Research Group
Department of Automatic Control and Systems Engineering
The University of Sheffield
Mappin Street, Sheffield S1 3JD, United Kingdom

Research Report #615
February 1996

Tel : +44 (0)114 2825250
Fax : +44 (0)114 2731729
EMail : rrg@sheffield.ac.uk
RRG Robotics Research Group

A Neural Network Based Collision Detection Engine for Multi-arm Robotic Systems

A. S. Rana and A. M. S. Zalzala

Department of Automatic Control & Systems Engineering, University of Sheffield, Sheffield S1 3JD, UK.

SUMMARY
A neural network is proposed for collision detection among multiple robotic arms sharing a common workspace. The structure of the neural network is a hybrid between Guassian Radial Basis Function (RBF) neural networks and Multi-layer perceptron back-propagation (BP) neural networks. This network is used to generate potential fields in the configuration space of the robotic arms. A path planning algorithm based on heuristics is presented. It is shown that this algorithm works better than the conventional potential field methods which carry out the planning in the operational space of the robots. To show the effectiveness of the algorithm, simulation results are presented for a single 2-DOF roboic arm in presence of a static obstacle, and then for two planar manipulator sharing a common workspace. The algorithm is then extended to the case of two 3-DOF arms moving in 3-D space.

KEYWORDS: Multi-arm robots, Neural Network, Collision Free Motion planning.

1. INTRODUCTION

The problem of motion planning of robotic manipulators is different from that of mobile robots, in which the motion for a single rigid body has to be planned. Robotic manipulators form open chains of connected links. The motion planning of these links consists of motion planning of multiple rigid bodies constrained by arm kinematics. The use of multi-arm robotic systems can enhance the working capacity and utilization of the robots, but their motion planning becomes even more complex, since one arm may act as a moving obstacle in the path of another arm.

The potential field approach[4] is perhaps the simplest approach to motion planning of robots in a dynamically changing environment. It constructs an artificial force field under which the robot has to move so that it is repelled by the obstacles while being attracted towards the goal position. This force field is termed as the potential field. Timing analysis of the path planning algorithms in the operational space of the robotic manipulators using potential field approach indicates* that the generation of potential field forms a bottle-neck in the algorithm. It is indicated that more than 80% of time is used in calculating the potential field and in performing the associated calculations. An alternative to this could be planning the path of the manipulator in configuration space.

Duffy and Slawek[2] presented a hardware based collision detection engine for multi-arm robotic system. Lee and Park[5] proposed a connectionist approach to the path planning problem of robots by proposing a multi-layer perceptron neural network for collision detection, and Meng and Picton[7] extended this work by using back-propagation learning technique to teach the neural network the location of the obstacles. Tseng and Wu[8] proposed neural networks for collision detection of multi-arm robots. But they used the neural networks to solve the forward kinematic model of the robots, and the collision itself was determined seperately. Moreover, they stopped short of proposing an actual algorithm for the path planning of the robots.

Motion planning of robotic manipulators in configuration space has the advantage that the problem is reduced to that of planning the path of a point object[6], and hence complexities arising from considering the motion of an articulated chain of links are taken into account. The mapping of obstacles in the path of the manipulators into configuration space is quite time-consuming. In operational space, the obstacles can be approximated by convenient geometric shapes, such as lines and rectangles, and an analytical function found due to potential field generated by charges distributed over these geometric shapes. But in case of path planning in configuration space, formulation of an analytical function to express the potential field is quite difficult, since the obstacles do not form regular shapes. Numerical methods have been suggested[1] for the construction of potential fields in configuration space, and hill climbing techniques have been used to find a collision free path for the robot by following the gradient of these potential fields. Nonetheless, these methods deal with path planning in presence of static obstacles only, where the field has to be calculated only once. They are not applicable to moving obstacles, in which the field would change with the motion of the obstacles. In case of dynamically changing environment, these methods become computationally quite intensive, since the numerical potential field would have to be calculated repeatedly to take into account the changes in potential field due to the movement of the obstacles.

The main emphasis of this paper is the research done towards the development of a nerual-network based collision detection engine for multi-arm robots. An associated motion planning algorithm based on heuristics is also presented.

This paper is organised as follows. Section 2 specifies the problem for 2-DOF planar robots. Section 3 gives a brief overview of two neural network paradigms. Section 4 briefly discusses the generation of obstacles in configuration space.

* unpublished work

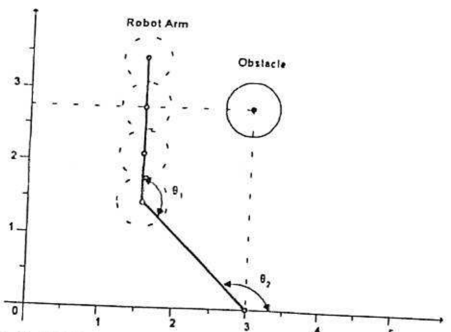


Figure 1. A 2-DOF planar robotic manipulator with base at (3,0) and each link of length 2 units. The circular obstacle is located at (3,2.5).

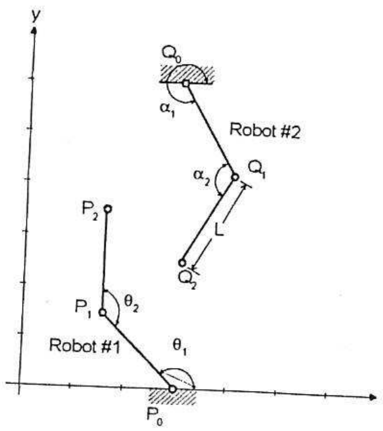


Figure 2. Two robotic manipulators sharing a common workspace.

Section 5 gives the application of neural networks to the generation of potential fields for the collision free motion planning of the robots. Section 6 gives an associated heuristic based path planning algorithm for 2-DOF planar robotic manipulators. Section 7 discusses an extension of this algorithm to the case of two 3-DOF manipulators moving in three dimensional workspace. Finally, section 8 discusses the conclusions.

2. PROBLEM SPECIFICATION

A 2-DOF planar arm is considered first with a stationary circular obstacle in its operational space, as shown in Figure 1. The upper link on the robotic arm collides with the obstacle only. The collision is determined first by approximating the link with touching circles, and then by determining the distance between the centres of each circle and the centre of the static obstacle. If this distance is less than the sum of the radii of the two circles, collision is assumed to occur.

For two robotic arms sharing a common workspace as shown in Figure 2, the links on both the arms are approximated by touching circles in the same way and collision between them is detemined by considering the distance between the centres of the circles.

3. NEURAL NETWORKS

Neural networks are massively parallel systems in which processing elements are linked together by connecting weights. Two frequently used paradigms in neural networks are multi-layered perceptron (MLP) back-propagation (BP) networks and Guassian Radial Basis Function (RBF) networks.

3.1 Multi-layer perceptron BP (Back Propagation) Networks

These networks have one or more hidden layers of neurons in addition to input and output layers. They demonstrate very powerful capabilities to learn non-linear mappings. These networks are called back-propagation networks because during the training (or learning) phase of the network, the weights are adjusted by propagating error at the output of the network backwards through the hidden layers towards the input layer.

Learning consists of feedforward and back-propagation phases. During the feedforward calculations, the output of each neuron in layer *j* recieving inputs from outputs of neurons in layer *i* in the network is calculated as:

$$\text{net-input}_j = \mathrm{i}_j = \sum_i \mathrm{w}_{ji}\mathrm{o}_i + \theta_j \tag{1}$$

$$\text{output}_j = \mathrm{o}_j = f(\mathrm{i}_j) = \frac{1}{1+\exp(-\mathrm{i}_j)} \tag{2}$$

The error at the output is calculated by comparing it to the desired output as:

$$\delta_j = f'(\mathrm{i}_j)(\mathrm{d}_j - \mathrm{o}_j); \tag{3}$$

where d_j is the desired output, and the error in the subsequent layers is calculated as:

$$\delta_j = f'(\mathrm{i}_i)\sum_j \mathrm{w}_{ji}\delta_i \tag{4}$$

In case above mentioned sigmoidal activation function $f(.)$ is used, then:

$$f'(\mathrm{i}_j) = f(\mathrm{i}_j)(1 - f(\mathrm{i}_j)) = \mathrm{o}_j(1-\mathrm{o}_j) \tag{5}$$

If for the output neuron, a linear activation function is used, i.e.

$$o = \mathrm{i} \tag{6}$$

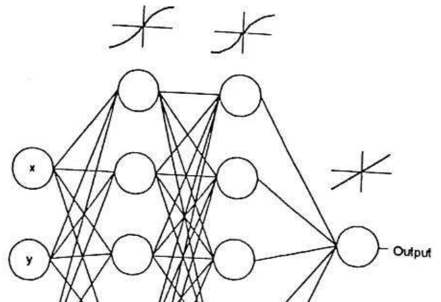


Figure 3. Back propagation Multi-layered Perceptrons Neural Network

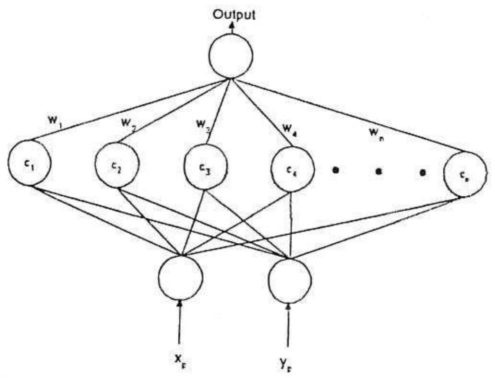


Figure 4. Radial Basis Function Neural Network

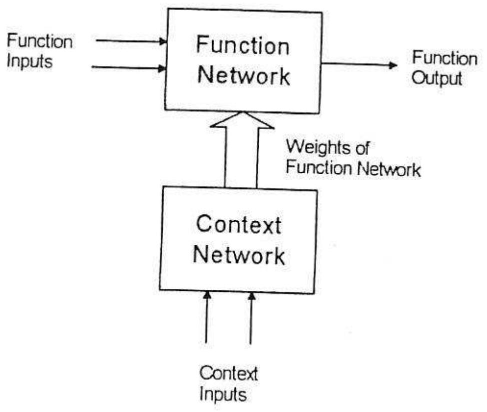


Figure 5. Block Diagram of Context Sensitive Network

then

$$f' = 1 \tag{7}$$

The weights are updated as:

$$w_{ji}(\text{new}) = w_{ji}(\text{old}) + \eta\delta_j o_i \tag{8}$$

where η is a constant called the learning rate of the network.

Once the network has learned through all possible values on inputs and outputs of the mapping, the network calculates the output simply by feedforward calculations.

3.2 Guassian RBF (Radial Basis Function) Networks

Gaussian Radial Basis Function Networks are a powerful method for learning complex input/output mappings. These networks are not very good at learning high frequency parts in the mapping, but provide a very good approximation of the mapping.

Radial Basis Function Networks are locally tuned neurons to learn a mapping. They have a single layer of hidden units which do not just evaluate the weighted sum of inputs but encode the inputs by computing how close they are to the centres of receptive fields. Each hidden unit has an activation function of the form:

$$g(\|\vec{x} - \vec{c}_i\|) \tag{9}$$

Output layer is linear, so:

$$y = \sum_i w_i g(\|\vec{x} - \vec{c}_i\|) \tag{10}$$

g(.) is a guassian activation function given by:

$$g(y) = \frac{1}{1 + \exp(-y)} \tag{11}$$

where

$$y = \|\vec{x} - \vec{c}_i\| \tag{12}$$

The weights can be learned by using Hebb's activation rule, i.e. the change in the weight associated with a neuron is proportional to the error at the output and the activation of the neuron.

3.3 Context Sensitive Networks

Sometimes it is difficult to train large networks. An approach has been suggested[o] in which the input to the network is divided into two groups. One group of variables determines the basic mathematical function/mapping while the other group determines context/setting in which the function is determined. The output of context network determines the weights of function network.

4. GENERATION OF CONFIGURATION SPACE OBSTACLES

Several methods are available for the computation of the configuration space obstacles[3]. The simplest of these is the point evaluation method, in which the robot is placed in all possible configurations and it is determined whether it collides with the obstacles in those configurations or not. If it does, then that configuration belongs to the C-space obstacle, otherwise it constitutes free space. Hence a function f can be formed such that

$$f = \begin{cases} 1 & \text{If the robot collides with the obstacle in operational space in that configuration} \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

5. NEURAL NETWORKS FOR C-SPACE OBSTACLES

Computation of artificial potential field intensity due to obstacles forms a bottleneck in computation time in the path planning algorithm. Since neural networks distribute the computation on parallel precessing elements (neurons), mapping of potential fields on neural networks would decrease the overall computational time of the algorithm. The function expressed in (13) can be taught to a neural network to obtain a potential field for the robot in configuration space. Since the neural network has an inherent property of interpolating in between the values of inputs that it is taught, the output of the NN will vary continously in between 1 and 0 giving a continuous potential field for the robot in configuration space. The application of NN parallelizes the computation of the field at the same time. The only drawback is that the output will only give the intensity of the potential field, and does not provide any information about the gradient of the field which the robot could follow in order to avoid collision with the obstacle.

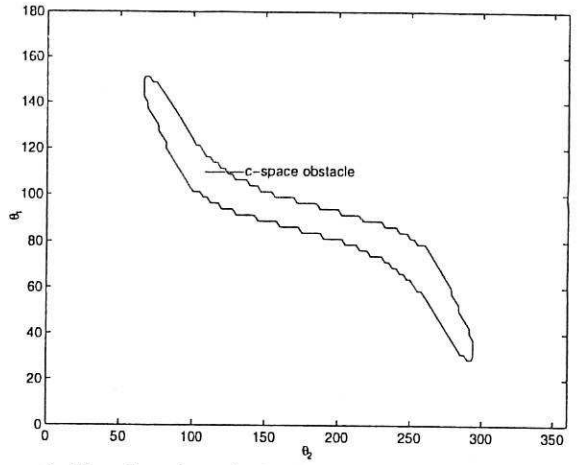


Figure 6. Circular obstacle in the operational space of the 2-DOF robot mapped into the c-space.

5.1 Mapping of Potential Field in C-space of a 2-DOF Planar Manipulator in the Presence of a Static Obstacle

Both MLP and RBF networks were used to map the potential field for a single 2-DOF robotic manipulator in the presence of a static circular obstacle shown in Figure 1. Back propagation rule was used for the training of MLP networks and Hebb's activation rule was used for the training of RBF network. A total of eight neurons were used in each of the hidden layer of the MLP network. For RBF network, the centres of the network were distributed evenly over the configuration space of the manipulator. Both the networks were trained over a grid distributed evenly over the entire range of the c-space variables. The results were quite satisfactory, even though some spurious minima were shown at some places where the network did not learn very well. Figure 6 shows the circular obstacle mapped in the c-space of the manipulator. The results are shown in Figure 7 to Figure 10. It can be seen that the MLP networks show better results at learning the field. For an equivalent learning, the RBF networks require a much larger number of units (neurons), but the actual learning time for the RBF network is much smaller than that for BP networks.

5.2 Mapping of Potential Field for Two 2-DOF Planar Manipulators Sharing a Common Workspace

For two 2-DOF manipulators sharing a common workspace, the problem becomes more complex. In this case, one manipulator acts as an obstacle in the path of the other manipulator. In c-space, this obstacles changes not only in position, but size as well, disappearing at certain configurations altogether.

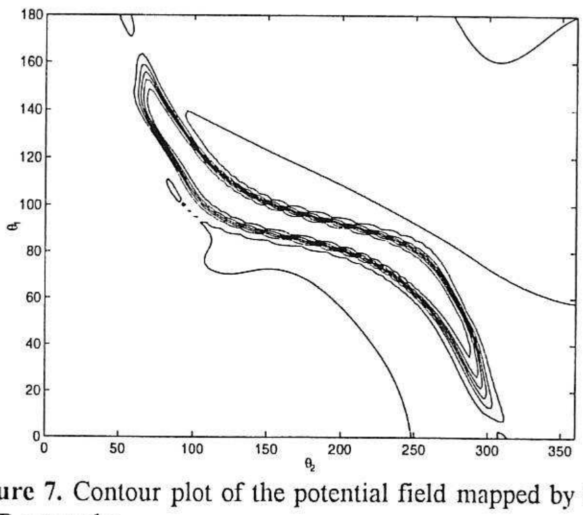


Figure 7. Contour plot of the potential field mapped by the MLP networks.

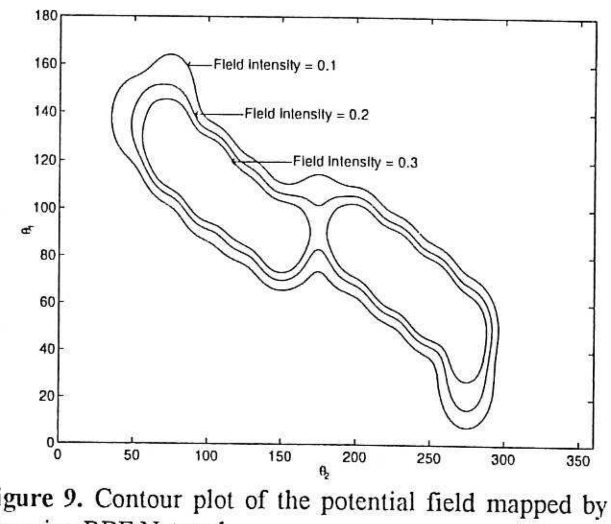


Figure 9. Contour plot of the potential field mapped by a Guassian RBF Network.

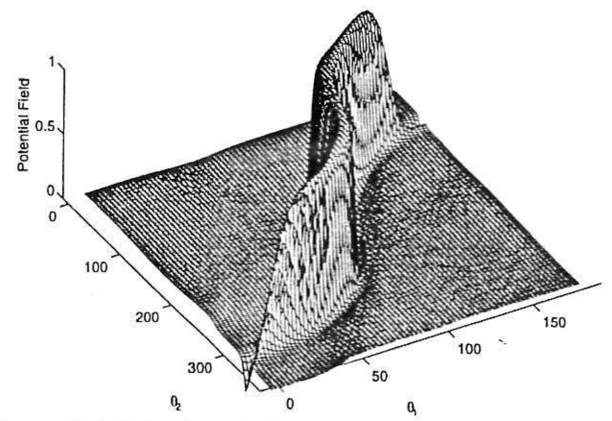


Figure 8. Mesh plot of the potential field mapped by the MLP network.

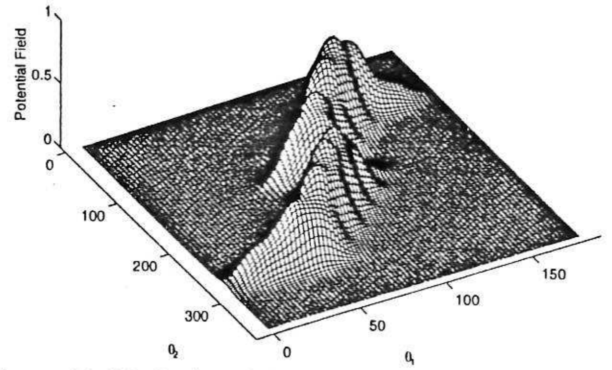


Figure 10. Mesh plot of the potential field mapped by the Guassian RBF network.

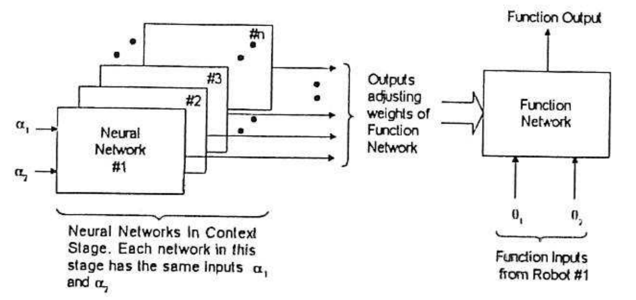

#n
#3
#2
α_1
α_2
Neural Network #1
Outputs adjusting weights of Function Network
Function Output
Function Network
θ_1
θ_2
Neural Networks In Context Stage. Each network in this stage has the same inputs α_1 and α_2
Function Inputs from Robot #1

Figure 11. Block Diagram of Context Sensitive Network for generation of Potential Field in Operational Space of Robotic Arms.

5.2.1 MLP Network

The simplest option for two robotic manipulators is to use a single MLP BP network. This was tried out for two 2-DOF planar robots. A BP network with four inputs (α_1, α_2, θ_1 and θ_2) with different sizes was tried, but it did not show good results. The reason for this is that the four dimensional space becomes quite large for the network. The network has to be taught at the entire range of the joint angles. The range of these angles is 0 to 180° for θ_1, 0 to 360° for θ_2, 180° to 360° for α_1 and 0° to 360° for α_2. If a step angle of 5° is chosed for each joint angle and the network is taught over a grid in the space of input variables, then the number of data points to be taught to the network becomes $36\times72\times36\times72=6.718\times10^6$, which is quite large. The neural network was taught at 10° steps, but it failed to show good results after considerable number of training epochs.

5.2.2 Context Sensitive Networks

Since a single MLP network did not show any good results, context sensitive results were used instead. Two different structures were tried. In the first structure, MLP networks were used in both the funciton and the context stages of the network. In the second structure, MLP networks were used in the context stage and a Guassian RBF network was used in the function stage of the network. The joint angles for robot #1 were used as input to the function network, and joint angoles of robot #2 were used as inputs to the context network.

The use of context sensitive networks give better results than a single MPL network even for a fewer number of training steps. Moreover, the context sensitive network allows the training problem to be devided into independent sub-training problems, which can be carried out on independent machines. The training procedure is carried out in the following three steps:

1. Teach the function network the c-space obstacles at fixed context inputs and store the weights of the function network for each case.
2. From the stored sets of weights for different values of context inputs, generate a training set for each network in the context stage by determining how each weight of the funciton network would vary against the variation in context inputs.
3. Teach each network in context stage the generated data set in step 2.

MLP Networks used in Both Stages of the Network:

In the first instance, MLP networks were tried in both the function stage and the context stage of the network. Even though on the whole it showed good results at the configurations for which it was trained for, it did not interpolate at all in between these configurations. The reason for this is that back propagation networks do not have a unique solution of connection weights for a particular mapping. For this reason, it is not necessary for the connection weights in MLP networks in function network stage to show smooth transition if the context is varied smoothly. This can be seen from Figure 12, which shows how a sample weight in the MLP function network varies if the context inputs are varied and the function network is trained independently from random inital weights.

MLP Networks in the Context Stage and Guassian RBF Networks in the Function Stage of the Network:

RBF networks used in function network stage however showed very good results even for configurations in between the configurations for which the context networks were trained.

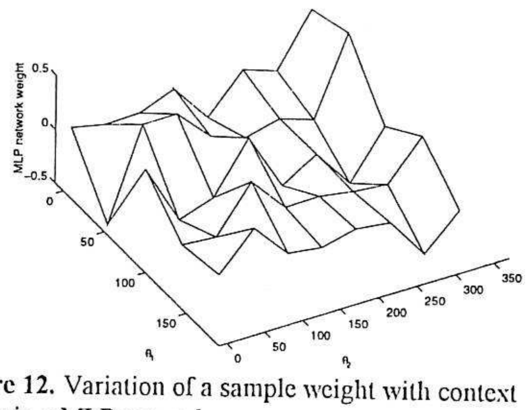


Figure 12. Variation of a sample weight with context inputs in a MLP network.

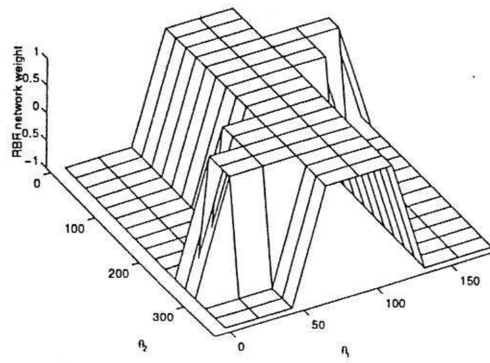


Figure 13. Variation of a sample weight with context inputs in a Guassian RBF network.

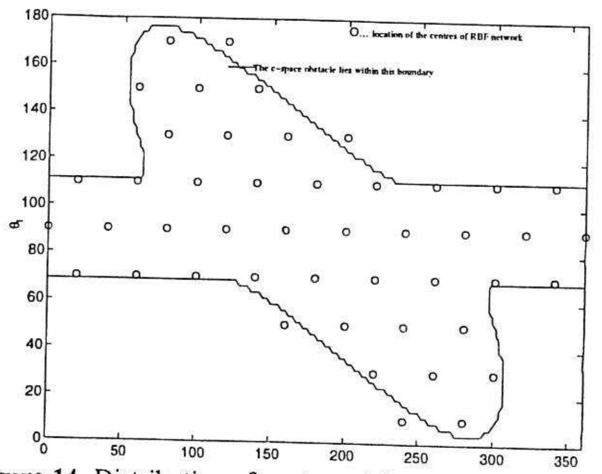


Figure 14. Distribution of centres of the RBF network for the function stage of context sensitive network.

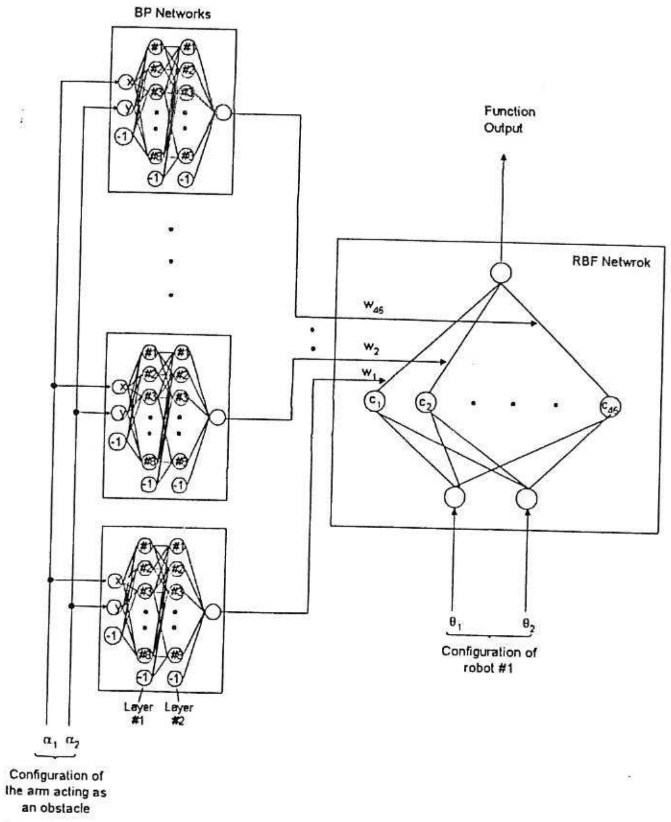


Figure 15. Structure of the hybrid context sensitive network for mapping the potential field for two 2-DOF robotic planar manipulators.

In RBF networks, each neuron has a local receptive field and hence they have a unique solution to the connection weights for a particular set of inputs to the network if the centres of the network are kept fixed. Connection weights in RBF networks will therefore show a uniform variation with variation in context, and hence the network on the whole will show good interpolation in between the configurations for which it is trained. Figure 13 shows the variation of a sample weight in the RBF network against the context inputs.

The centres of RBF network were not spread over the whole c-space, but a grid was formed only on all those regions where the c-obstacle appeared for different configurations of the robotic arm #2. A total of 46 neurons were used in function stage and 46 independent MLP networks were used as context networks. Each context network had two hidden layers with 8 neurons each. Figure 14 shows the distribution of these centres in configuration space. The structure of the neural network is shown in Figure 15.

The following learning rule was used for RBF networks:
If desired output =1, then

$$w_i = \begin{cases} 1 & \text{if } \|C_i - P_m\| \le \frac{D}{2} \\ 0 & \text{otherwise} \end{cases} \qquad (12)$$

where

$w_i =$	connection weight for ith neuron.
$C_i =$	Centre of ith neuron,
$P_m =$	test input, which is varied over the whole of c-space
$D =$	minimum distance between centres.

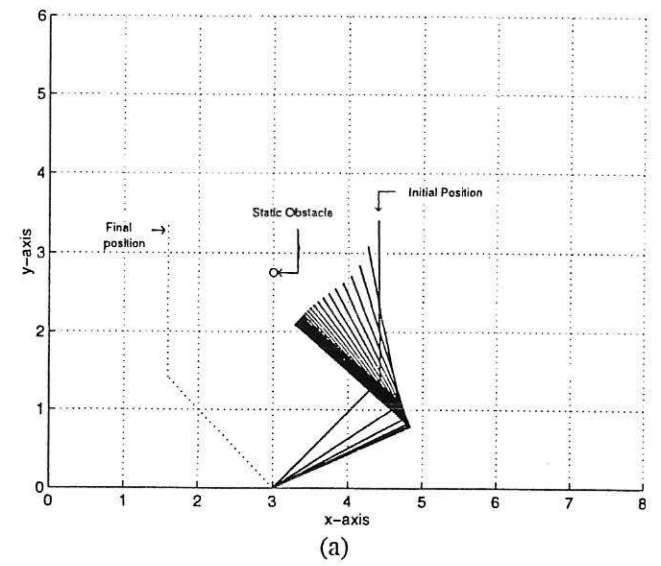
Training step of 1° for the function network's inputs θ_1 and θ_2 (robot #1), and a step of 22.5° was chosen for context networks' inputs α_1 and α_2 (robot #2).

6. A HEURISTICS BASED PATH PLANNING ALGORITHM

Working in c-space reduces the path planning problem into planning the motion of a point robot between given initial and final configurations while avoiding collisions with c-space obstacle. Most of the algorithms based on potential field approach use the gradient of the field. If the robot follows the negative gradient of the field, it can avoid collision with the obstacles, while at the same time, moving towards the goal position, which lies at the global minimum of the potential field. However, if the obstacle is concave, or if the motion of multiple connected rigid bodies is being planned simultaneously, then there is a chance that the robot might get stuck in the local minima formed in the potential field. In configuration space of the robot, the obstacles form irregular shapes, which would result in local minima being formed in the potential field. Another problem with working in c-space using neural networks is that the output of the network gives only the intensity of the potential field, and does not provide any information about the gradient of the field. So a heuristic based algorithm was used to carry out the planning in c-space of the robot. The following heuristics were used

- If the field intensity is lower than a certain value *mid_field*, move towards the goal configuration in a straight line.
- If the field intensity at any point becomes equal to or larger than *mid_field*, follow the contour of the field until the rovot goes around the obstacle, and then follow the first heuristic.
- If the value of the potential field intensity becomes larger than or equal to a certain value *hi_field*, follow the negative gradient of the field (by using a pseudo-gradient descent method) until the field becomes less than or equal to *lo_field*.

It may be noted that *hi_field*≥*mid_field*≥*lo_field*. For the given examples, these values have been chosen as 0.1, 0.2 and 0.3 for *lo_field*, *mid_field* and *hi_field*, respectively.



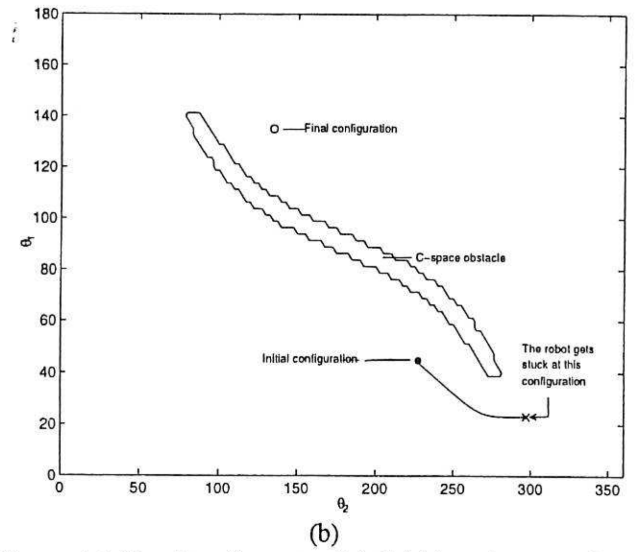


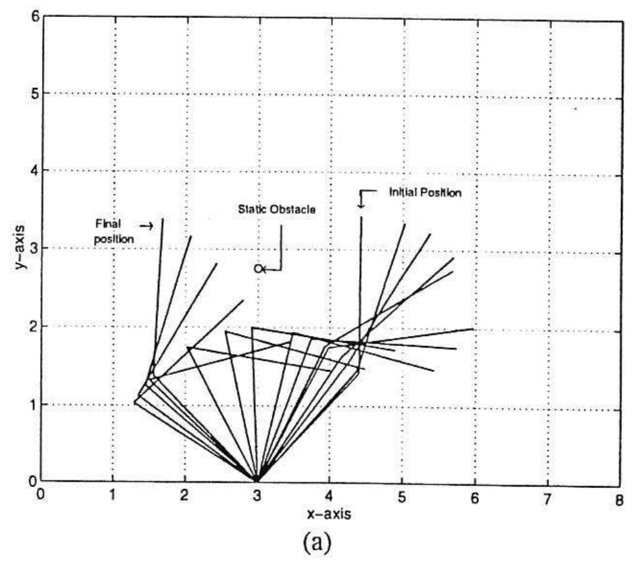
Figure 16. Results of a potential field based conventional path planning algorithm for a single 2-DOF planar manipulator in (a) operational space and (b) configuration space.

6.1 Path Planning for a Single 2-DOF Planar Robotic Manipulator in the Presence of a Static Obstacle

A path planning algorithm based on potential field approach suffers from the drawback that the robot may get stuck in the local minima of the potential field. For the sake of comparison, a potential field based algorithm was developed in which the static obstacle was approximated by a point charge and the links of the 2-DOF planar manipulator by uniformly distributed line charges, thus establishing a repelling potential between the obstacle and the manipulator. At the same time, an attractive potential is set up between the robot and its final configuration, and the robot has to move under the forces resulting from the potential field.

As an example, the a path was planned between the initial and final configurations of (45°,225°) and (135°,135°), respectively. Figure 16 shows the results of the potential field based path planning algorithm. Even though the path can be planned easily if the obstacle is further away from the base of the robot, in this particular case it can be seen that the robot was unable to get to the final configuration and got stuck in the local minimum. For the same initial and final configuration, the heuristics-based algorithm was able to find the path. Figure 17 shows the results this path planning algorithm. It can be clearly seen that the planned path takes the robot around the obstacle in c-space to its final configuration.

At this point, it may be pointed out that if the final configuration happens to lie at a point where the intensity of the potenial field is higher than the value at which the robot starts to follow the contour, then the planner fails to take the robot to the final configuration. Additional heuristics may be needed to avoid this situation. Moreover, at the moment the algorithm does not incorporate the reporting of the fact that it has failed to plan the path, and the path planning stops a certain time after the instant at which the robot starts off from the initial configuration.



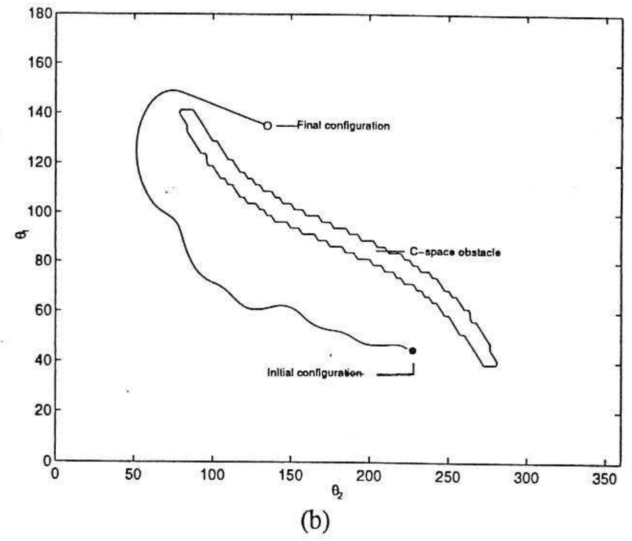


Figure 17. Results of the heuristics-based path planning algorithm for a single 2-DOF planar manipulator in (a) operational space and (b) configuration space.

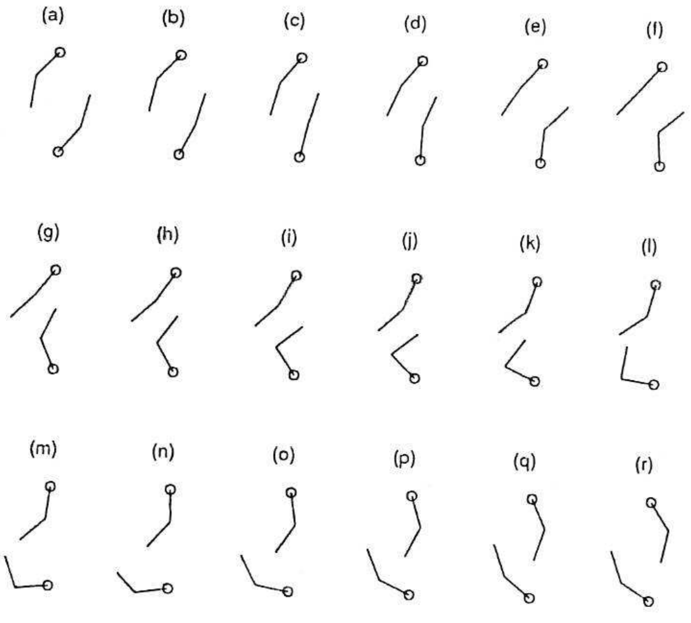


Figure 18. (a) to (r) show the results of the path planning algorithm for two robotic manipulators at different timing instants.

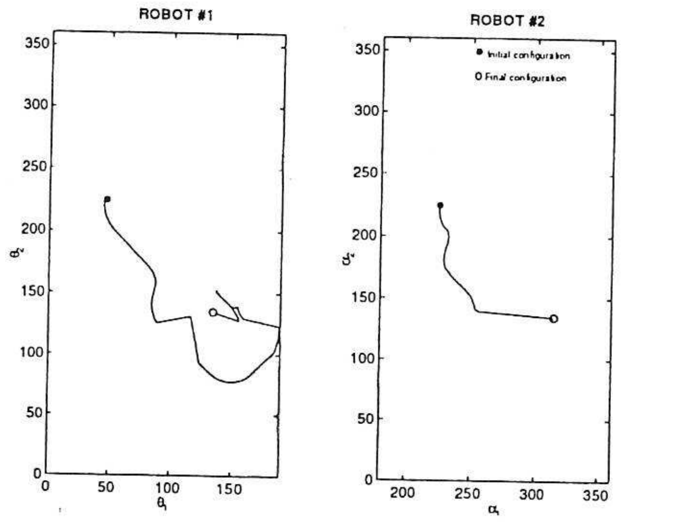


Figure 19. The paths of the two robots in configuration space.

6.2 Path Planning for Two 2-DOF Planar Robotic Manipulators

Figure 18 shows the collision free motion for two 2-DOF planar robotic manipulators planned by using the heuristics based algorithm. Figure 19 shows the paths in configuration space. It may be pointed out here that different speeds for the motion were chosen for robot #1 and robot #2. Robot #2 was infact moving slower than robot #1. The reason for this is that if the same speed was chosen in this particular example, the robots get stuck due to the deadlock occuring from the robots switching between the different heuristics. This happens because of their motion being symmetric. The path was planned for robot #1 from an initial configuration of (45°,225 °) to the final configuration of (135°,135°) and for robot #2 from an initial configuration (225°,225°) to the final configuration (315°,135°).

7. EXTENSION TO TWO 3-DOF ROBOTS MOVING IN THREE DIMENSIONAL SPACE

The path planning algorithm in Section 6 is extended to the case of motion planning of two 3-DOF robots moving in three dimensional space.

7.1 Problem Formulation

The acutal problem is similar to that given in Section 2. The two 3-DOF robots are to work with their work envelops overlapping. For the sake of simplicity, it is assumed that only the third links of both robots may collide with each other. Figure 20 (a) shows the two robots when value of all the joint angles is zero. Figure 20 (b) shows the portion of c-space for both robots where c-obstacles are likely to lie. Figure 20 (c) and (d) show the work envelops of the robots from top and side respectively. Shaded area indicates where the work envelops overlap.

7.2 Generation of C-Obstacles

Point evaluation technique is again used to determine c-obstacles. The robots are modelled by touching spheres. The position of the centres of the spheres approximating the robots is calculated by using forward kinematic model.

7.3 Simulation Results

The algorithm was adopted for two 3-DOF robots moving in three dimensional workspace. The path was planned between initial configurations of (15°,-15°,-30°) and (-150°,-15°,30°), and the final configurations of (150°,-15°,30°) and (-150°,-15 °,-30°) for robot #1 and robot #2, respectively.

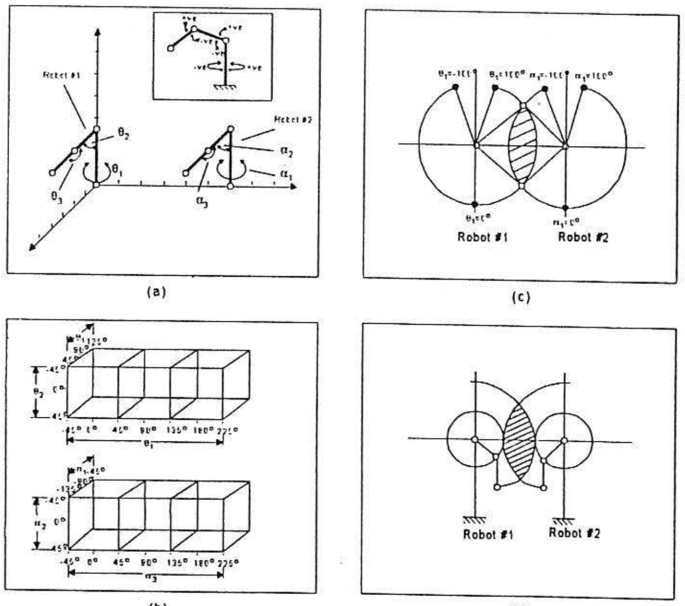


Figure 20. (a) Robots when all joint angles are zero (inset: direction in which angles change positively or negatively). (b) part of configuration space where c-obstacles are likely to lie. (c) top view of work envelops of the robots and (d) side view of work envelops of the two robots.

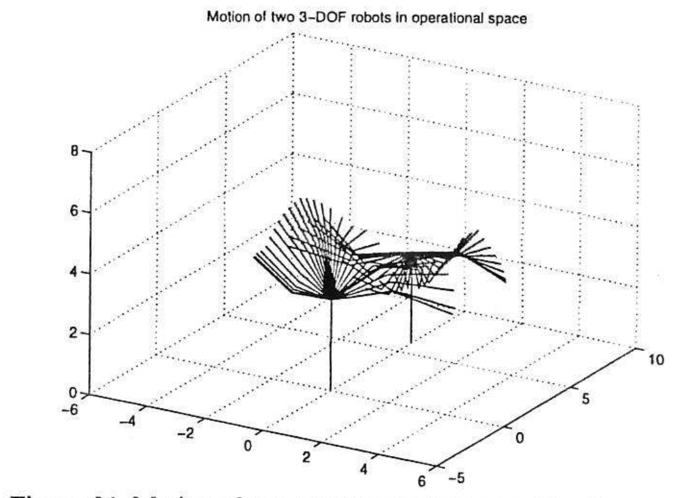


Figure 21. Motion of two 3-DOF robots in operational space.

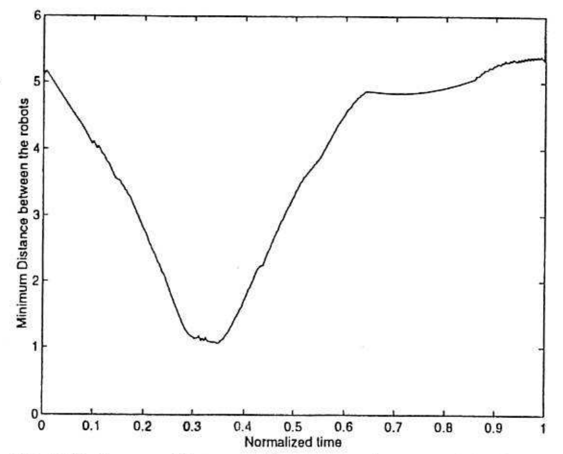


Figure 22. Minimum distance between the two 3-DOF robots moving in three dimensional space against normalized time.

Figure 21 shows the motion of the two robots in operational space. Figure 22 gives the minimum distance between the two robots against normalized time.

8. CONCLUSIONS

A collision detection engine for two robotic manipulators sharing a common workspace is presented. The structure of teh neural network is a cotext sensitive hybrid Guassian Radial Basis Function (RBF) network/ Multi-layered perceptron (MLP) back propagation network. The network is trained off-line to learn the collision detection among the robots. Once trained, it servers two purposes: (i) It parallelizes the computations involved in mapping the configuration space obstacles and (ii) it provides a continuous transition at the output of the neural network between the free space and space occupied by c-space obstacles, thus forming a continuous potential field in c-space. A path planning algorithm based on heuristics ispresented, which uses the output from the neural network to plan the paths in configuration space.

At the moment, the proposed algorithm has two shortcomings. One is that if the final configuration to which the robot has to go to lies withing the rigion where the potential field intensity is higher than the value where the robot starts following the contour of the field (*mid_field*), then the planning algorithm fails to find the path. This shortcoming can be overcome by introducing a new heuristic whereby the robot moves towards the goal configuration irrespective of the intensity of potential field when it reaches within a certain distance of the goal configuraiton, but incorporation of this heuristic would not guarantee that the planned path is collision free or not. The second shortcoming of the algorithm is that sometimes the two robots get stuck due to deadlock because of switching between the heuristics by using crisp values of potential field intensity (viz. *hi_field*, *mid_field* and *lo_field*). This suggests that the use of fuzzy controller for path planning might give better results. The developement of this controller will be considered in future.

REFERENCES

1. J. Barraquand, Langlois and J. C. Latombe, "Numerical Potential Field Function Techniques for Robot Path Planning", *Report No. STAN-CS-89-1285, Dept. of Computer Science, Stanford University*, Oct. 1989.
2. N. Duffy and P. Slavek, "Collision Avoidance for a Multi-arm Robotic Cell" *IEE Colloquium on 'Multi-arm Robotics'* (Digest No. 147 p.2/1-4) June 1992.
3. Y. K. Hwang and N. Ahuja, "Gross Motion Planning- A Survey" *ACM Computing Surveys*, **24**(3), 219-291, Sept. 1992.
4. Y. K. Hwang and N. Ahuja, "A Potential Field Approach to Path Planning", *IEEE Trans.Robotics and Automation*, **8**(1), 23-31, Feb. 1992.
5. Lee, S.; Park, J. "Neural Computation for Collision-free Path Planning" *J. Intelligent Manufacturing*, **2**, 315-326, 1991.
6. T. Lozano-Perez, "Spatial Planning: A Configuration Space Approach" *IEEE Trans. on Computing*, **C-32**(2), 1983.
7. H. Meng and P. C. Picton, "A Neural Network for Collision-free Path Planning" *Proc. 1992 Int. Conf. Artificial Neural Networks (ICANN-92)*, Brighton, U.K., 4-7 Sept. 1992.
8. C. X. Tseng and C. C. Wu, "Collision Detection for Multiple Robot Manipulators by using Orthogonal Neural Networks", *J. Robotic Systems*, **12**(17), 479-490, 1995.
9. D. T. Yeung and G. A. Bekey, "Using a Context Sensitive Learning Network for Robotic Arm Control" *Proc. IEEE Int. conf. on Robotics and Automation*, 1989.