International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 1, Issue 7, ISSN No. 2455-2143, Pages 107-113
Published Online May - June 2016 in IJEAST (http://www.ijeast.com)

A STUDY OF MOVEMENT OF POINT ROBOT THROUGH NEURAL NETWORK

Lokesh Kumar Shrivastav
Computer Science & Engineering,
AMITY UNIVERSITY
Noida, Uttar Pradesh – 201303, India

I. INTRODUCTION

An artificial neural network (ANN), usually called neural network (NN), is a computational model that is inspired by the structure and functional aspects of biological neural networks. A neural network is set of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. It is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. Now a day, it is non-linear statistical data modeling tools. It is used to model complex relationships between inputs and outputs or to find patterns in data.

II. PROBLEM DESCRIPTION

During the last decades numerous and extensive investigations have been made using soft computing techniques towards the problem of navigation in diverse environments by mobile robots. Our project will be concerned with point robot navigation in an unknown cluttered environment based on neural networks .Our task will be to provide a steering command letting a point robot avoid a collision with obstacles.

Obstacle avoidance is a common problem in the field of mobile robotics. In our program, the task will be to reach a specified “goal” location, given bearing to the goal, some measure of distance to the goal. The “obstacles” will be pseudo-randomly-generated polygons of a given expected radius. This particular problem is important to nearly every aspect of mobile robotics, as a robot cannot accomplish very many tasks in an open environment without navigating and avoiding obstacles.

III. BRIEF LITERATURE SURVEY

Neural network simulations are a recent development. Although, this field was established before the advent of computers, but survived several in eras. Several important advances have been boosted by the use of inexpensive computer emulations. Following the initial period of enthusiasm, the field survived a period of frustration and disrepute. During this period when funding and professional support was minimal, important development were made by few researchers. These pioneers were able to development convincing technology, which surpassed the limitations identified by Minsky and Papert. Minsky and Papert, published a book in which they summed up a general feeling of frustration among researchers, and was thus accepted by most without further deep analysis. Now a day, the neural network field enjoys a resurgence of interest and a corresponding increase in funding.

The history of neural networks can be divided into several periods:

1. **First Attempts**: There were some initial and fundamental simulations using formal logic. McCulloch and Pitts developed models of neural networks were based on their understanding of neurology. These models were depends on several assumptions about how neurons works. Their networks were based on simple neurons which were considered to be binary devices with fixed thresholds. The output of their model were simple logic functions such as "a or b" and "a and b". Another attempt made by using computer simulations. Two groups (Farley and Clark, 1954; Rochester, Holland, Haibit and Duda, 1956). The first group (IBM researchers) maintained closed contact with neuroscientists at McGill University. Whenever their models did not work, they consulted the neuroscientists. This interaction established a multidisciplinary dimension which continues till now.

2. **Promising & Emerging Technology**: Due to neuroscience influential in the development of neural networks, psychologists and engineers also started to contribute in the progress of neural network simulations. Rosenblatt stirred considerable interest and activity in the field, he designed and developed the Perceptron. The Perceptron had three layers with the middle layer known as the association layer. This system could learn to associate a given input to a random output. Another model was the ADALINE (ADAptive LInear Element) which was developed in 1960 by Widrow and Hoff (of Stanford University). The ADALINE was an analogue electronic device implemented

from simple components. The method of learning was different to that of the Perceptron, it employed the Least-Mean-Squares (LMS) learning rule.

3. **Period of Frustration & Disrepute:** In 1969 Minsky and Papert wrote a book in which they described the limitations of single layer Perceptrons to multilayered systems. In the book they wrote: "our intuitive judgment that the extension (to multilayer systems) is sterile". The significant result of their book was to eliminate funding for research with neural network simulations. The conclusions supported the disenchantment of researchers in this field. As a result, considerable prejudice against the field was activated.

4. **Innovation**: Although public interest and available funding were minimal, several researchers continued working to develop computational methods for problems such as pattern recognition. During this period several paradigms were generated which modern work continues to enhance. Grossberg's (Steve Grossberg and Gail Carpenter in 1988) influence started a school of thought which explores resonating algorithms. They formed the ART (Adaptive Resonance Theory) networks based on biologically plausible models. Anderson and Kohonen formed associative techniques independent of each other. Klopf (A. Henry Klopf) in 1972, explores a basis for learning in artificial neurons based on a biological principle for neuronal learning called heterostasis. Werbos (Paul Werbos 1974) formed and used the back-propagation learning method, however several years passed before this approach was popularized. Back-propagation nets are possibly the most well-known and widely applied of the neural networks today. The back-propagation net is a Perceptron with multiple layers, a different threshold function in the artificial neuron, and a more robust and capable learning rule.
Amari (A. Shun-Ichi 1967) was involved in theoretical developments: he published a paper which established a mathematical theory for a learning basis (error-correction method) dealing with adaptive patern classification. Fukushima (F. Kunihiko) developed a step wise trained multilayered neural network for interpretation of handwritten characters. The original network was published in 1975 which was known as Cognitron.

5. **Re-Emergence**: The development during the late 1970s and early 1980s was important to the re-emergence on interest in the neural network field. Several factors affect this movement. As, comprehensive books and conferences provided a forum for people in diverse fields with specialized technical languages, and the response to conferences and publications was quite positive. The news media took up on the increased activity and tutorials helped disseminate the technology. New academic programs appeared and courses were introduced at most major Universities (in US and Europe). Attentions were now focused on funding levels throughout Europe, Japan and the US and as this funding becomes available, several new commercial with applications in industry and financial institutions are emerging.

6. **Today**: Essential progress has been made in the field of neural networks-enough to attract a great deal of attention and fund further research. Achievement and advancement beyond current commercial applications appears to be possible, and research is advancing the field on many fronts. Neurally based chips are emerging and applied to solve complex problems of development. Clearly, it is a period of transition for neural network technology.

- In their paper, Qiang Yao, Daryl Beetner, Donald C. Wunsch II and Bjorn Osterloh, demonstrated a RAM based neural network for a mobile robot controlled by a simple microprocessor system. Training a multi-layer neural network requires presentation of training data, which often results in very long learning time. They demonstrated that RAM based neural networks are a suitable choice for embedded applications with few computational resources.

- In the paper, Ten-Min Lee, Ulrich Nehmzow and Roger Hubbold have shown experimental results with nomad 200 mobile robot, acquiring sensor model of a specific environment and using this model to predict the robot-environment interaction. Data obtained by operating the real robot in real target environment is used to train a set of 16 artificial neural networks, which can later be used to model robot-environment interaction and predict the behavior of real robot in offline simulation.

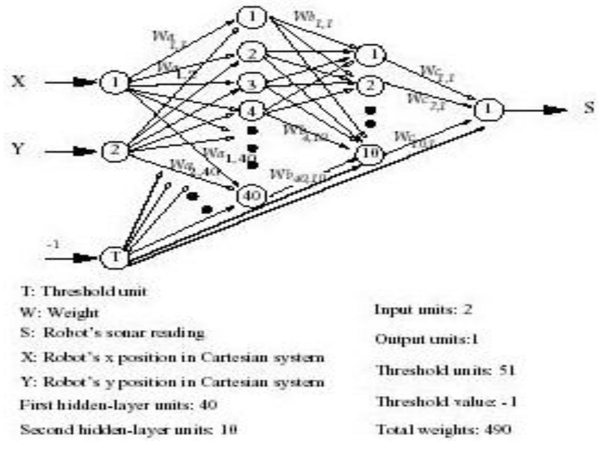


Figure 2.1 Ten-Min Lee, Ulrich Nehmzow and Roger Hubbold's Model

The purpose of their experiments was to model the 16 sonar sensors by means of real data obtained from the Nomad interacting with a target environment. To achieve this, 16 multi-layer perceptrons were used: one to model each sonar sensor. In neural network theory, it is well known that a two-hidden-layer structure can learn arbitrary curves. The total number of hidden units they tried ranges from 10 to 200. As a compromise between learning time and accuracy, each network has 50 hidden units in two layers, two input units and one output unit (see figure 2). The two input units encode the robot's current position in Cartesian space as obtained from the robot's odometry system. The single output unit encodes the range reading obtained from that sonar sensor at that particular position.

- In September 2003, at the IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems, Oleh Adamiv, Vasyl Koval, Iryna Turchenko [10] have presented paper describing the experimental results of neural networks application for mobile robot control on predetermined trajectory of the road. Investigations showed robust mobile robot movement on different parts of the road. The comparative error of deviation of robot movement from predetermined road is 4% on average.

- In their paper, Nicolás Navarro, Cesar Muñoz, Wolfgang Freund, Tomás Arredondo [9] described the use of soft computing techniques for acquiring adaptive behaviors to be used in mobile robot exploration. They have used Action-based Environment Modeling (AEM) based navigation within unknown environments and unsupervised adaptive learning for obtaining of the dynamic behaviors.

The experiments were made using the YAKS Khepera simulator. The simulator has a map where the robot moves; the simulator provides the readings for the sensors according to the current map (room). It also handles the information of zones visited, not visited and the various obstacles in the room. The rooms are square with length and width of 2750 mm in size. The rooms are differentiated by the amount and type of obstacles they present to the robot. The rooms were divided into 2500 zones (each 55 mm by 55 mm). The robot generates an internal map in which the zones are marked with various values: obstacles are indicated with a value of -1, those not visited by the robot are marked with 0 and the visited ones with 1. The robot executes 1000 steps in each simulation, not every step produces forward motion as some only rotate the robot.In order to reduce the search space of behaviors, we use a limited number of actions for the robot to execute in each step.

IV. APPROACH TOWARDS SOLUTIONS

Our approach to the solution of the problem is based on generating a map of the unknown cluttered environment. For this purpose, we will perceive the environment as a grid comprising of a 2D array of cells. Associated with each cell is a value indicating whether the cell is free or has been occupied by an obstacle. The techniques will be used for the approach described, are as follows:

1. Autonomous Map Building:
2. Cell Decomposition Path Planning

Software Used

We will use Visual Basic to implement **Feed forward Back propagation Neural Network** describing below.

V. FEEDFORWARD BACKPROPAGATION NETWORK

The feedforward backpropagation network is a very famous model in neural networks. It doesn't have feedback connections as shown in the figure below, but errors are back propagated during training. Errors in the output determine measures of hidden layer, which are used as a basis for adjustment of connection weights between the input and hidden layers. Mutually adjusting the two sets of weights between the pairs of layers and recalculating the outputs is an iterative process that is carried on until the errors fall below a tolerance level. Learning rate parameters scale the adjustments to weights.

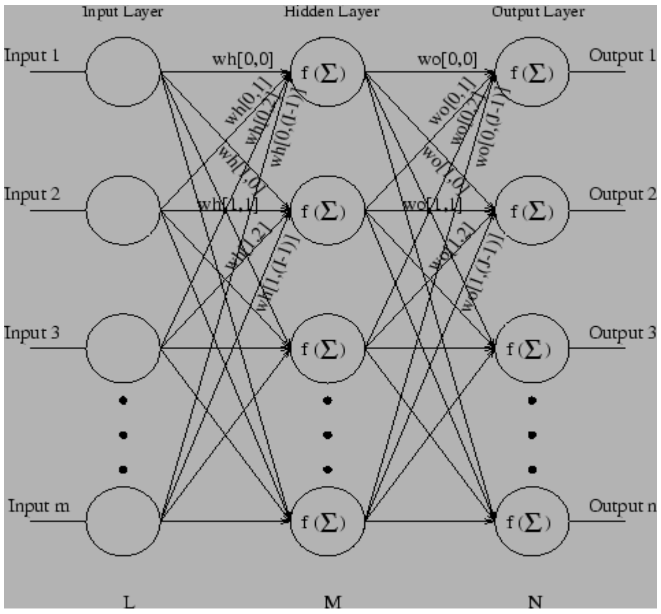


Figure : Backpropagation Network

VI. MAPPING

The feedforward backpropagation network maps the input vectors to output vectors. Pairs of input and output vectors are chosen to train the network first. Once training is finished, the weights are set and the network can be used to find outputs for new inputs. The dimension of the input vector determines the number of neurons in the input layer, and the number of neurons in the output layer is determined by the dimension of the outputs.

VII. TRAINING OF THE BACKPROPAGATION NETWORK

The feed forward back propagation network undergoes supervised training, with a finite number of pattern pairs consisting of an input pattern and a desired or target output pattern. An input pattern is presented at the input layer. The neurons here pass the pattern activations to the next layer neurons, which are in a hidden layer. The outputs of the hidden layer neurons are obtained by using perhaps a bias, and also a threshold function with the activations determined by the weights and the inputs. These hidden layer outputs become inputs to the output neurons, which process the inputs using an optional bias and a threshold function. The final output of the network is determined by the activations from the output layer.
The computed pattern and the input pattern are compared, a function of this error for each component of the pattern is determined, and adjustment to weights of connections between the hidden layer and the output layer is computed. A similar computation, still based on the error in the output, is made for the connection weights between the input and hidden layers. The procedure is repeated with each pattern pair assigned for training the network. Each pass through all the training patterns is called a cycle or an epoch. The process is then repeated as many cycles as needed until the error is within a prescribed tolerance. There can be more than one learning rate parameter used in training in a feed forward back propagation network.

VIII. PROBLEM FORMULATION

Even before the advent of affordable mobile robots, the field of path-planning was heavily studied because of its applications in the area of industrial manipulator robotics. Interestingly, the path planning problem for a manipulator with, for instance, six degrees of freedom is far more complex than that of a differential-drive robot operating in a flat environment. Therefore, although we can take inspiration from the techniques invented for manipulation, the path-planning algorithms used by mobile robots tend to be simpler approximations owing to the greatly reduced degrees of freedom.

Path-planning overview: The robot's environment representation can range from a continuous geometric description to a decomposition-based geometric map or even a topological map. The first step of any path-planning system is to transform this possibly continuous environmental model into a discrete map suitable for the chosen path-planning algorithm. Path planners differ as to how they effect this discrete decomposition. We can identify three general strategies for decomposition:

1. **Road map**: identify a set of routes within the free space.
2. **Cell decomposition**: discriminate between free and occupied cells.
3. **Potential field**: impose a mathematical function over the space.

The following sections present common instantiations of the road map and cell decomposition path-planning techniques, noting in each case whether completeness is sacrificed by the particular representation.

Cell decomposition path planning

The idea behind cell decomposition is to discriminate between geometric areas, or cells, that are free and areas that are occupied by objects. The basic cell decomposition path-planning algorithm can be summarized as follows :

• Divide into simple, connected regions called "cells".
• Determine which opens cells are adjacent and construct a "connectivity graph".
• Find the cells in which the initial and goal configurations lie and search for a path in the connectivity graph to join the initial and goal cell.
• From the sequence of cells found with an appropriate searching algorithm, compute a path within each cell, for example, passing through the midpoints of the cell boundaries or by a sequence of wall-following motions and movements along straight lines.

An important aspect of cell decomposition methods is the placement of the boundaries between cells. If the boundaries are placed as a function of the structure of the environment, such that the decomposition is lossless, then the method is termed *exact cell decomposition*. If the decomposition results in an approximation of the actual map, the system is termed approximate *cell decomposition*.
Here, we briefly summarize these two cell decomposition techniques providing greater detail about their advantages and disadvantages relative to path planning.

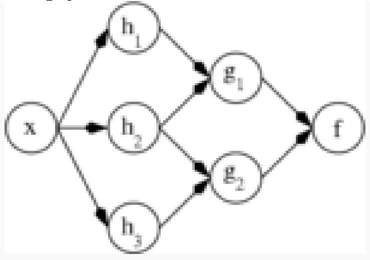
IX. METHODOLOGY

Neural network models in artificial intelligence are usually referred to as artificial neural networks (ANNs); these are essentially simple mathematical models defining a function $f: X \rightarrow Y$ or a distribution over X or both X and Y, but sometimes models are also intimately associated with a particular learning algorithm or learning rule. A common use of the phrase ANN model really means the definition of a *class*of such functions (where members of the class are obtained by varying parameters, connection weights, or specifics of the architecture such as the number of neurons or their connectivity).

The word *network* in the term of 'artificial neural network' defines to the inter–connections between the neurons in the different layers of each system. An example system has three layers. The first layer has input neurons, which send data via synapses to the second layer of neurons, and then via more synapses to the third layer of output neurons. More complicated systems will have more layers of neurons with some having increased layers of input neurons and output neurons. The synapses store parameters called "weights" that explore the data in the calculations.
An ANN can define by three types of parameters:

1. The interconnection pattern between different layers of neurons
2. The learning process for modifying the weights of the interconnections.
3. The activation function that transforms a neuron's weighted input to its output activation.

A neuron's network function $f(x)$ is defined as a composition of other functions $g_i(x)$, which can further be defined as a composition of other functions. This can be represented as a network structure, with arrows depicting the dependencies between variables. A widely used type of composition is the *nonlinear weighted sum*, where $f(x)=K\left(\sum_i w_i g_i(x)\right)$, where K (commonly referred to as the activation function [1]) is some predefined function, such as the hyperbolic tangent. It will be convenient for the following to refer to a collection of functions g_i as simply a vector $g=(g_1,g_2,\ldots,g_n)$.



ANN dependency graph

This figure explores such a decomposition of f, with dependencies between variables indicated by arrows. These can be explained in two ways.
The first view is the functional view: the input x is transformed into a 3-dimensional vector h, which is then transformed into a 2-dimensional vector g, which is finally transformed into f. This view is most fundamental encountered in the context of optimization.

The second view is the probabilistic view: the random variable $F=f(G)$ depends upon the random variable $G=g(H)$, which depends upon $H=h(X)$, which depends upon the random variable X. This view is most fundamentally encountered in the context of graphical models.
The two views are largely equivalent. In either case, for this specific network architecture, the components of individual layers are independent of each other (e.g., the components of g are independent of each other given their input h). This naturally enables a degree of parallelism in the development.

Two separate depictions of the recurrent ANN dependency graph
Networks such as the first one are commonly called feedforward, because their graph is a directed acyclic graph. Networks with cycles are fundamentally called recurrent. Such networks are commonly depicted in the manner shown at the top of the figure, where f is shown as being dependent upon itself. However, an implied temporal dependence is not shown.

What has attracted the most interest in neural networks is the probability of *learning*. Given a specific *task* to solve, and a *class* of functions F, learning means using a set of *observations* to find $f^*\in F$ which solves the task in some*optimal* sense. It isdefining a cost function $C:F\rightarrow\mathbb{R}$ such that, for the optimal solution f^*, $C(f^*)\leq C(f)\ \forall f\in F$ (i.e., no solution has a cost less than the cost of the optimal solution).

The cost function C is an necessary concept in learning, as it is a measure of how far away a particular solution is from an optimal solution to the problem to be solved. Learning algorithms search through the solution space to find a function that has the smallest possible cost.

A applications where the solution is dependent on some data, the cost must necessarily be a *function of the observations*, otherwise we would not be modeling anything related to the data. It can frequently define as a statistic to which only approximations can be made. As a simple example, consider the problem of finding the model f, which minimizes $C=E\left[(f(x)-y)^2\right]$, for data pairs (x,y) drawn from some distribution $\mathcal{D}$. In implementation situations we would only have N samples from $\mathcal{D}$ and thus, for the above example, we would only minimize $\hat{C}=\frac{1}{N}\sum_{i=1}^{N}(f(x_i)-y_i)^2$. Thus, the cost is minimized over a sample of the data rather than the entire data set.
When $N\rightarrow\infty$ some form of online machine learning must be used, where the cost is partially minimized as each new example is seen. While online machine learning is often used when $\mathcal{D}$ is fixed, it is most useful in the case where the distribution changes slowly over time. In neural network

methods, some form of online machine learning is frequently used for finite datasets.

X. EXPECTED OUTCOMES

Figure below depicts exact cell decomposition, whereby the boundary of cells is based on geometric criticality. The resulting cells are each either completely free or completely occupied, and therefore path planning in the network is complete, like the road map based methods above. The basic abstraction behind such decomposition is that the particular position of the robot within each cell of free space does not matter; what matters is rather the robot's ability to traverse from each free cell to adjacent free cells.
The key disadvantage of exact cell decomposition is that the number of cells and, therefore, overall path planning computational efficiency depends upon the density and complexity of objects in the environment, just as with road map based systems. The key advantage is a result of this same correlation. In environments that are extremely sparse, the number of cells will be small, even if the geometric size of the environment is very large. Thus the representation will be efficient in the case of large, sparse environments.

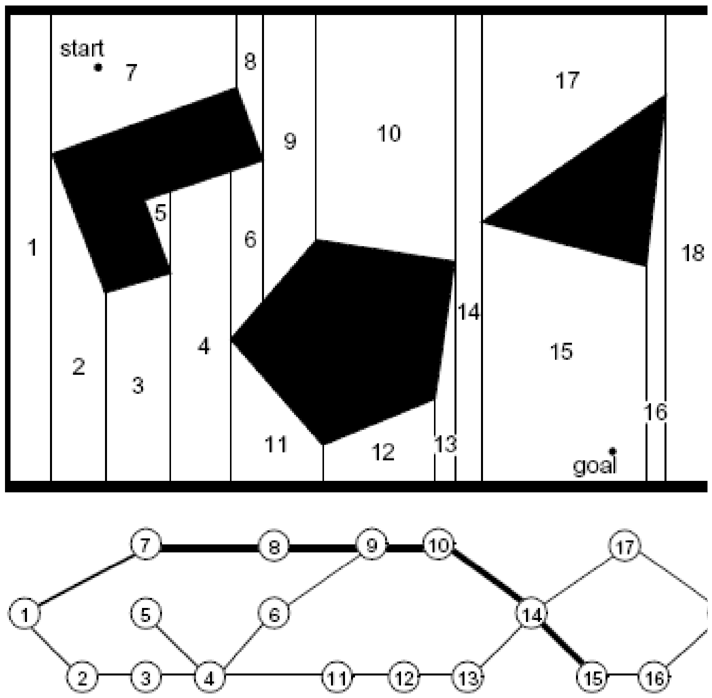
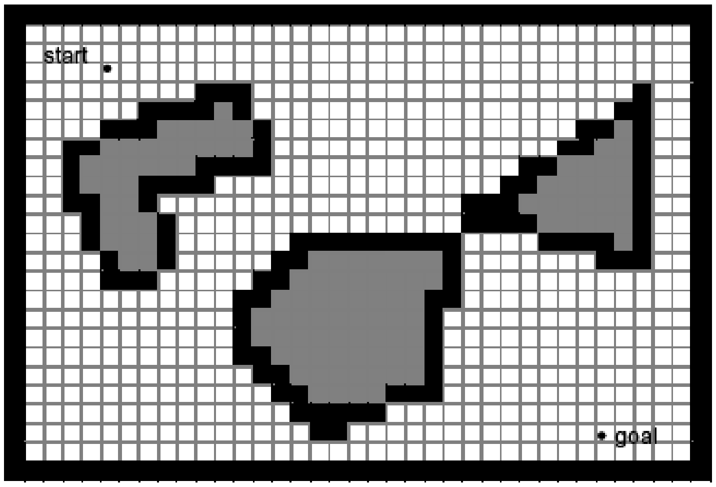


Figure 1.7
Example of exact cell decomposition

Practically speaking, due to complexities in implementation, the exact cell decomposition technique is used relatively rarely in mobile robot applications, although it remains a solid choice when a lossless representation is highly desirable, for instance to preserve completeness fully.



Approximate cell decomposition. By contrast, approximate cell decomposition is one of the most popular techniques for mobile robot path planning. This is partly due to the popularity of grid-based environmental representations. These grid-based representations are themselves fixed grid-size decompositions and so they are identical to an approximate cell decomposition of the environment.
The most popular form of this, shown in figure below, is the fixed-size cell decomposition.

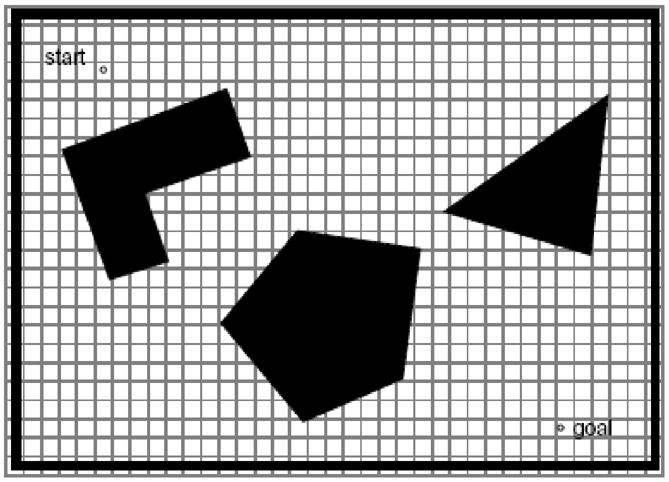


Figure 1.8

Fixed decomposition of the same space (narrow passage disappears).
The cell size is not dependent on the particular objects in an environment at all, and so narrow passageways can be lost due to the inexact nature of the tessellation. Practically speaking, this is rarely a problem owing to the very small cell size used (e.g., 5 cm on each side). The great benefit of fixed size cell decomposition is the low computational complexity of path planning.

Given that the entire array can be in memory, each cell is only visited once when looking for the shortest discrete path from the initial position to the goal position. So, the search is linear in the number of cells only. Thus complexity does not depend on the sparseness and density of the environment, nor on the complexity of the objects' shapes in the environment. The fundamental cost of the fixed decomposition approach is memory. For a large environment, even when sparse, this grid must be represented in its entirety. Practically, due to the falling cost of computer memory, this disadvantage has been mitigated in recent years. The Cye robot is an example of a commercially available robot that performs all its path planning on a 2D 2 cm fixed-cell decomposition of the environment using a sophisticated grassfire algorithm that avoids known obstacles and prefers known routes.

XI. CONCLUSION

The computing world has a lot to gain from neural networks. Their ability to learn by example makes them very flexible and powerful. Furthermore there is no need to devise an algorithm in order to perform a specific task; i.e. there is no need to understand the internal mechanisms of that task. They are also very well suited for real time systems because of their fast response and computational times which are due to their parallel architecture.

Neural networks also contribute to other areas of research such as neurology and psychology. They are regularly used to model parts of living organisms and to investigate the internal mechanisms of the brain.

Perhaps the most exciting aspect of neural networks is the possibility that someday 'conscious' networks might be produced. There is a number of scientists arguing that consciousness is a 'mechanical' property and that 'conscious' neural networks are a realistic possibility.

Finally, I would like to state that even though neural networks have a huge potential we will only get the best of them when they are integrated with computing, AI, fuzzy logic and related subjects.

XII. REFERENCES

[1]Alan C. Schultz, John J. Gefenstette, and William Adams,"Robo-Shepherd: Learning Complex Robotic Behaviors",Naval Research Laboratory, Washington, D.C. 20375-5337, U.S.A.

[2]C D'Souza, K Sivayoganathan , D Al-Dabass, V Balendran and J Keat , " Machine vision for robotic assembly: issues and experiments". Proc. 13th National Conference on Manufacturing Research, Glasgow, 9-11 Sept, 1997, pp 114-118. ISBN 1 9012 48119

[3]C D'Souza, K Sivayoganathan, D Al-Dabass*, V. Balendran ," Simulation of Object Recognition by a Robot", Manufacturing Automation Research Group, Dept. of Mechanical and Manufacturing Engg., *Department of Computing,Nottingham Trent University, Nottingham NG1 4BU.

[4]Danica Janglová, "Neural Networks in Mobile Robot Motion", International Journal of Advanced Robotic Systems, Volume 1, Number 1, March 2004, pp.15-22

[5]Daniel Hein and Manfred Hild and Ralf Berger, "Evolution of BipedWalking Using Neural Oscillators and Physical Simulation", Humboldt University Berlin, Department of Computer Science, {dhein|hild|berger}@informatik.hu-berlin.de

[6]Michael Schuresko,"The Use of Neural Networks and Reinforcement Learning to Train Obstacle-Avoidance Behavior in a Simulated Robot", Machine Learning (cmps 242), Winter 2005.