# MOBILE ROBOT GUIDANCE USING CELLULAR NEURAL NETWORKS AND FUZZY LOGIC

Marco Balsi†, Alessandro Maraschini†, Giada Apicella†
Sonia Luengo‡, Jordi Solsona‡, Xavier Vilasís-Cardona‡

†Dipartimento di Ingegneria Elettronica, Università La Sapienza,
via Eudossiana 18, 00184 Roma, Italy - balsi@uniroma1.it
‡Enginyeria i Arquitectura La Salle, U. Ramon Llull, Departament d'Electrònica,
Pg. Bonanova 8, 08022 Barcelona, Spain - xvilasis@salleURL.edu

## Abstract

We show how a Cellular Neural Networks based image processing system together with a Fuzzy Logic controller are capable of providing the necessary signal processing to guide an autonomous mobile robot in a maze drawn on the floor. In this way, a non-trivial navigation task is obtained by very simple hardware, making real autonomous operation feasible. An autonomous line-following robot has been first simulated and is currently being implemented by simulating the CNN with a DSP, while the fuzzy algorithms run on a 386-microprocessor-based microcontroller.

**Keywords:** Cellular Neural Networks, Robot Vision, Robot Navigation, Fuzzy Logic.

## 1 Introduction

Navigation of a mobile autonomous robot in an unknown (or not completely known) environment can be achieved by use of visual information and feedback. When the environment is structured, the task required from the vision system is recognition of visual clues, and evaluation of their location relative to the robot position and orientation. Such information is then used to take suitable action by a control system.

Line following is one of the reference problems connected with navigation in a structured environment. It is a relatively simple, yet non-trivial problem that is relevant to real-life situations, such as road navigation, or motion through a maze (e.g. industrial environments). Line (or equivalent marking) following has been considered elsewhere. Solutions proposed in the literature may be classified into two main categories: simple solutions with very limited capability, or solutions for difficult, realistic problems obtained by use of rather complicated hardware.

An example of simple system is the ARGO partially autonomous vehicle [1]. It is a normal passenger car fitted with an automatic steering mechanism, controlled by a 486PC. The PC processes the images taken by two B/W cameras with the aim of finding the right line of the road, and keeping it in the appropriate position in the field of view. The processing is very simple, yet effective for the purpose.

The more complex systems are generally based on a computer which is either on board (for large vehicles, e.g. [2, 3]), or remotely connected (e.g. [4]). In the quoted systems, images are processed by an additional unit, due to the necessity of high computing power. Of course, these vehicles can generally tackle fairly complex problems, such as unstructured road navigation, target following, obstacle avoidance, besides line following.

Cellular Neural Networks (CNN [5]) have been widely applied to sophisticated nonlinear real-time image processing tasks, so that they are natural candidates for silicon retina applications [6]. Their most appealing feature is the possibility of fully parallel analogic hardware implementation, which may include on-board image sensing capability [7], and the possibility of executing multiple consecutive and/or conditional operations without transferring images in and out of the network (CNN

Universal Machine [8]). Such networks are the engine of our image processing system. CNNs have been applied to robot navigation tasks before, in particular to stereo vision [9, 10], and optical flow computation [11, 12]. A simple line-following problem was tackled by a hardware-implemented CNN by Szolgay et al. [13]. In this paper, we consider a simple, yet significant, robot navigation task as testbed. Our objective is to drive an autonomous robot equipped with a camera to follow a path in a maze of black straight lines on white background, crossing or joined at right angles. No other sensor information shall be used but the visual information given by the camera. We have put an emphasis on using the simplest and least expensive hardware possible; real autonomy of the robot also called for compact and power-thrifty solutions. We decided to start by implementing a fully functional benchmark autonomous robot capable of navigating in a quite simple environment in order to thoroughly verify the validity of our approach. The results obtained will then be taken as a basis for addressing more complex navigation tasks.

## 2 Background: Cellular Neural Networks

Cellular Neural Networks are arrays of continuous-time dynamical artificial neurons (cells), that are only locally interconnected. This is the essential feature allowing for the VLSI implementation of large networks, fitted in chips with on board sensing, distributed memory and the capability of executing complex sequences of operations by global control (also called 'analogue' programmes). Among the CNNs subclasses we resort to a Discrete-Time CNN model (DTCNN [14]) because, besides being also realisable as a fully parallel VLSI chip, is also more apt to our simulated implementation.

DTCNN operation is described by the following system of iterative equations:

$$x_{ij}(n+1) = \mathrm{sgn}\Big( \sum_{kl \in N(ij)} A_{k-i,l-j} x_{ij}(n) + \sum_{kl \in N(ij)} B_{k-i,l-j} u_{ij}(n) + I \Big) \quad (1)$$

where $x_{ij}$ is the state of the cell (neuron) in position $ij$, that corresponds to the image pixel in the same position; $u_{ij}$ is the input to the same cell, representing the luminosity of the corresponding image pixel, suitably normalised. Matrix $A$ represents interaction between cells, which is local (since summations are taken over the set $N$ of indices of cells that are nearest neighbours of the one considered) and space-invariant (as implied by the fact that weights depend on the difference between cell indices, rather than their absolute values). Matrix $B$ represents forward connections issuing from a neighbourhood of inputs, and $I$ is a threshold. The operation of the network is fully defined by the so-called cloning template $\{A, B, I\}$. Under suitable conditions, a time-invariant input $u$ leads to a steady state $x(\infty)$ that depends in general on initial state values $x(0)$ and input $u$. Images to be processed will be fed to the network as initial state and/or input, and the result taken as steady state value, which realistically means a state value after some (order of 10 to 100 according to the task) time steps.

Many cloning templates have been designed for the most diverse tasks [15], so that some of the operations we need in this context were immediately obtained from the existing library.

## 3 Visual control system

The robot guidance problem is split into two processes, namely, the image processing to extract the mathematical features of the lines and the navigation of the robot to follow them. From the first, we expect to obtain the relative position of the mobile robot with respect to the line being followed, in order to correct possible deviations owing to misalignment of the wheels or mechanical or electrical fluctuations. This correction is to be performed by the navigation module. From the image processing we also need to extract information on the forthcoming crossings or bends, so as to decide which direction is to be taken and what is the correct moment to start turning. The mathematical information needed to control the robot is the angle between the robot and the path $A_{th}$ and the distance from the centre of the steering wheel to the forward axis $X_c$. The result should be the steering angle of the front wheel $A_{st}$.

## 4 Image processing

The image processing relies on the following stages: acquisition, preprocessing (cleaning and contrast enhancement), line recognition, extraction of line parameters (angle, position). Acquisition of course

| | $A$ | $B$ | $I$ |
|---|---|---|---|
| small object killer | $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ | 0 | 0 |
| left edge | 2 | $\begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ | -1 |
| vertically-tuned filter | 2 | $\begin{pmatrix} -1 & 0.5 & 1 & 0.5 & -1 \\ -1 & 1 & 1 & 1 & -1 \\ -1 & -1 & 5 & -1 & -1 \\ -1 & 1 & 1 & 1 & -1 \\ -1 & 0.5 & 1 & 0.5 & -1 \end{pmatrix}$ | -13 |
| small object killer | $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{pmatrix}$ | 0 | 0 |

Table 1: Templates used in the processing stage.

depends on the actual implementation, so it is treated below. For the pre-processing we used a so-called "small-object-killer" template (see table 1) to make a preliminary cleaning and binarisation of the acquired image (figure 1 (a,b)).

The calculation of the parameters of lines visible in an image is universally performed by the Hough transform [16]. Such approach is a very effective technique, yet computationally intensive, for obtaining direction and positions of all lines existing in an image. However, it becomes unnecessarily complicated when only few lines are present in the image.

Our particular case demands of a reasonably fast response with our limited hardware. For this reason, together with the fact that the Hough transform is a global operation that does not lend itself to efficient implementation on the CNN, we chose to devise a different approach based on CNNs.

In order to perform direction and position evaluation, we need to get a thin line first. A "skeletonisation" algorithm, would be rather slow because it involves many iterations of an eight-step routine (*i.e.* cyclical application of eight cloning templates). For this reason, we resorted to the design [17] of a cloning template that extracts only one of the two edges of a stripe (it actually extracts only those parts of edges of black-and-white objects that lie on the left of the object itself). This template is given in table 1, and its effect is shown in figure 1 (c). Of course, when dealing with approximately horizontal lines, we use a rotated version of this template, which extracts the upper edge.

The line position and orientation computation is based on two steps. We assume that a maximum of two, approximately orthogonal lines are within sight of the camera. This is not very restrictive, because we chose a setup in which the camera is oriented at angle that was chosen as a compromise between looking a reasonably long distance forward, and avoiding a large deformation due to perspective.

The first step is a directional filtering that extracts lines approximately oriented along two orthogonal directions. During normal operation these directions are the vertical and horizontal ones, corresponding to the line being followed and a possible orthogonal line following a bend or crossing. However, when the robot is turning or is largely displaced from all lines, it is necessary to switch to diagonal directions. As this situation is known to the controller, it will signal to the image processing stage which filters should be used.

Operation of a vertically tuned filter (table 1) is depicted in figures 2 and 3. Other direction-selective filters are obtaining by rotation of this cloning template. After the tuned filters have been applied, we get two images containing at most one line. Direction and position of the line is then extracted by performing an horizontal and vertical projection in order to read the positions of the first and last black pixels of the two projections. These four numbers, together with the information about which extreme of the line is closer to one of the borders of the image, are enough to compute direction and position of the line to the maximum precision allowed by image definition.

Extraction of the needed projections can be done

by means of the so-called "connected-component-detector" cloning template (table 1). Operation of such template at an intermediate and final stage of processing is depicted in figure 1(d,e). It is apparent that besides obtaining the desired projections, also the information about which extreme is closer to the border can be obtained from examination of intermediate results.

Using the CNN operation described above, by simple trigonometry, it is easy to obtain the parameters that are necessary for navigation, namely, angle and distance. Such parameters are passed to the controller.

## 5 Fuzzy navigation module

For the sake of simplicity, we chose to implement a quite standard fuzzy controller for the line following task. The fuzzy rule base is built according to the Sugeno model [18]. This scheme allows for a simpler on-line implementation since requires less calculations. The controller, fed with distance $X_c$ and angle $A_{th}$ to the line uses 15 rules to deliver the steering angle $A_{st}$. They are listed in table 2

The global control strategy works as follows. With a single line in sight, the robot tries to align itself as fast as possible on it. When a second line gets into the view, the type of bend or crossing is assessed, by examining the relative positions of the two lines. The possible actions are, then, evaluated. If more than one is possible (e.g. go straight on or turn right), than a command from a higher order level is called for (in our experiments we just implemented a pre-defined list of actions). If a turn is called for, the robot continues on the current line until the crossing is at a pre-defined optimal distance (determined by the maximum steering angle). Then, the algorithm just switches the line to be followed. This involves also switching the directional filter that is selected for line following, and correctly interpreting angular displacement in order to choose the correct turn (left or right).

## 6 Hardware implementation

We realised a small autonomous robot that is guided by the algorithms proposed in this paper. It is a three-wheeled cart driven by a motor fitted on the single front wheel, that also steers by means of a second motor. The cart is approximately 27cm long, 18cm wide, 16cm tall, and weights about 4kg. A PAL camera is fitted on the front of the robot, oriented downwards, 30° from the horizontal. Images are grabbed and digitalised by dedicated circuitry implemented in a CPLD (ALTERA EPM7128STC100), which also performs image decimation to reduce unnecessary definition. Image processing (CNN simulation) is performed in a DSP (TMS320c32), and the control system (fuzzy rule base) is implemented in a 386-microprocessor-based microcontroller. Images size is $60 \times 45$ pixels and lines on the floor are obtained using black tape 2cm wide. A power board feeds the motors, and batteries are carried on board. This way the robot it is completely autonomous. The image processing stage can currently process one image per second. We estimated by simulation that this allows the robot to move smoothly at a speed of 2.5 cm/s. Of course if a CNN chip were used instead of the DSP, it would be possible to reach a much higher speed.

## 7 Simulation results

The robot layout and the algorithms have been thoroughly tested by employing a realistic simulation of the vehicle realised in Working Model, interfaced with a simulation of the fuzzy control and image processing system realised in Matlab. All mechanical and physical parameters of the actual robot were taken into account (going from size and weight to wheel and land materials), as well as actual processing times of the mounted board, that has already been successfully tested. Mechanical mount and electrical testing of the robot is currently being completed, and we expect the robot to be fully functional soon. Meanwhile, figure 4 shows a simulation of the path followed by the robot, starting a little displaced ($A_{th} = 10°$,$X_c = 4$cm) and turning at a crossing. For each position of the robot, a camera shot is assumed and the geometrical parameters of the robot position are calculated and passed to the controller. Some snapshots of the model of the robot on its path are displayed in figure 5.

## 8 Future Directions

In the light of the previous simulations, harder robot vision tasks using CNNs are being envisaged. Subjects such as curved line following or obstacle

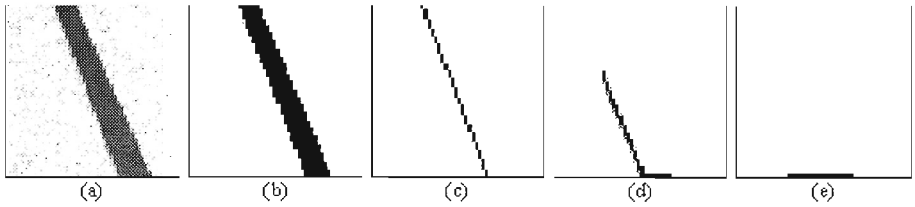


Figure 1: Original image, taken from the actual camera with size $60 \times 45$ pixels (a), cleaning and binarisation (b), left edge (c), connected-component detector -intermediate result (d), final result (e).

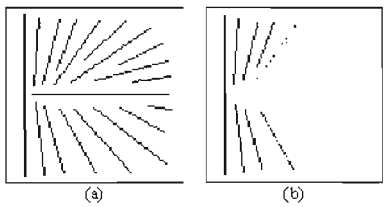


Figure 2: Vertical line extracting filter: original image (a), result (b).

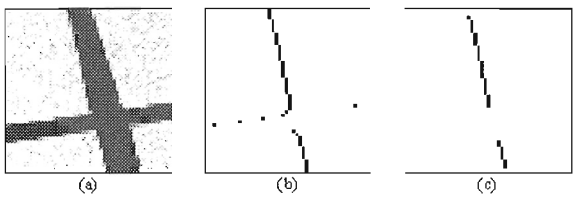


Figure 3: Vertical line extracting filter applied to the image of a crossing (a) and after edge extraction (b): result (c).

| | | $A_{th}$ | | | | |
|---|---|---|---|---|---|---|
| | | Large Positive | Small Positive | Almost Zero | Small Negative | Large Negative |
| $X_c$ | Left | Medium Left | Small Right | Medium Right | Medium Right | Large Right |
| | Centre | Medium Left | Small Left | Zero | Small Right | Medium Right |
| | Right | Large Left | Medium Left | Medium Left | Small Left | Medium Right |

Table 2: Fuzzy control rule set for $A_{st}$.

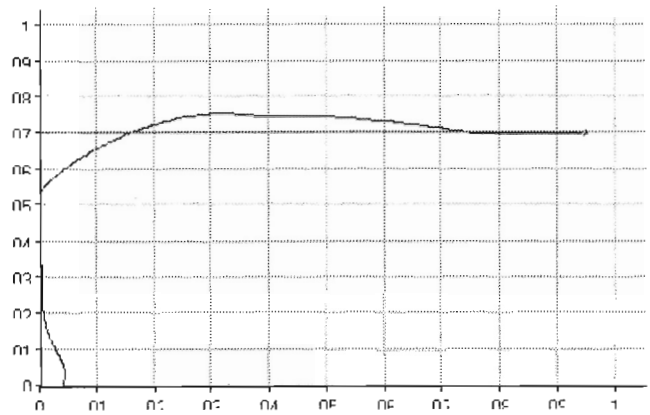

Figure 4: Path followed by the front wheel of the robot. Axis marking in meters.

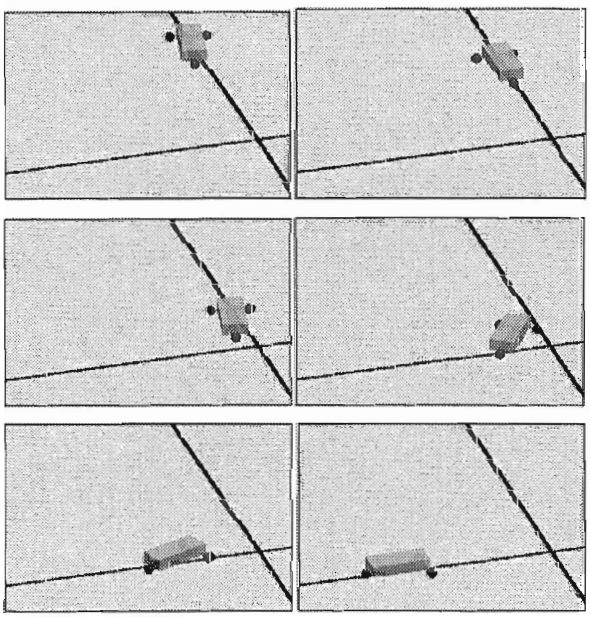

Figure 5: Snapshots of the robot simulation at time $t = 0, 8, 25, 32, 41, 53$s.

avoidance are currently being formulated in terms of CNNs.

The line following algorithm described above can be extended to the case of curved lines by a slight enhancement of the image processing stage, and a modification of the fuzzy rule base. In fact, following a curved line resorts to following the tangent at the closest point. However, in the long term the secant taken between the closest and the farthest point is a better guideline for the required manoeuvring Anyhow, the parameter extraction for either the tangent or the secant line is performed by the very same signal processing used for the straight line.

Regarding obstacle avoidance, we consider as "obstacle" anything that stands out as different from the floor background, assumed to be uniform. This approximation applies fairly well to outdoor paved roads and to most indoor floors and is consistent with the approach successfully taken by other authors [19, 20]. Our current approach is based on detecting edges in the field of view by CNN-based processing. When both objects and floor are quite smooth,and clearly distinguishable from each other, a simple edge detecting operation based on a Sobel-like operator is appropriate. More complicated environments, involving textured surfaces, and more confused object-background separation situations will call for more sophisticated algorithms, which are also available in the CNN framework, such as anisotropic diffusion [21] or snakes [22]. Closed edges are filled, and again projections are used to determine obstacle position. This information is then passed to an obstacle-avoidance guidance module that decides on the path to be followed.

## 9 Conclusions

In this paper we describe the design of a fully autonomous visually guided robot, which is able to follow a line by means of CNN-based image processing and fuzzy control. The robot has been realised and is currently being tested. Accurate and realistic simulation demonstrate the effectiveness of the approach. We are currently working at a development of the image processing stage that will allow the robot to deal with broken/dashed and curved lines, and with vertical obstacles. The design presented allows straightforward integration of a CNN Universal Chip in place of the DSP-simulated CNN, that would allow for highly increased speed. We believe this work is a step in the direction of proving that CNNs are a good candidate for a silicon retina in robot guidance tasks.

**Acknowledgements** : We thank Prof. De Carli, of the Department of Computer Science of 'La Sapienza' University for his assistance in the simulation of the robot. We also thank Prof. Jordi Margalef of Enginyeria i Arquitectura La Salle for his assessment on the hardware development.

## References

[1] http://nannetta.ce.unipr.it/ARGO/english/index.html (as of Dec. 2000).

[2] K.H. Chen, W.H. Tsai, Vision-Based Autonomous Land Vehicle Guidance in Outdoor Road Environments using Combined Line and Road Following Techniques. *Journal of Robotic Systems*; 14(10) (1997) 711-728.

[3] K.C. Cheok, G.E. Smid, K. Kobayashi, J.L. Overholt, P. Lescoe, A Fuzzy Logic Intelligent Control System Paradigm for an In-Line-of-Sight Leader-Following HMMWV. *Journal of Robotic Systems*; 14(6) (1997) 407-420.

[4] M. Maeda, M. Shimakawa, S. Murakami, Predictive Fuzzy Control of an Autonomous Mobile Robot with Forecast Learning Function. *Fuzzy Sets and Systems*; 72 (1995) 51-60.

[5] L.O. Chua, T. Roska, The CNN Paradigm. *IEEE Transactions on Circuits and Systems, part I*; 40(3) (1993) 147-156.

[6] *IEEE Transactions on Circuits and Systems, part I*, Special Issue on Bio-Inspired Processors and Cellular Neural Networks for Vision, 46(2) (1999).

[7] G. Liñán, S. Espejo, R. Domínguez-Castro, E. Roca, A. Rodríguez-Vázquez, CNNUC3: A Mixed-Signal 64×64 CNN Universal Chip. *Proc. of VII International Conference of Microelectronics for Neural, Fuzzy and Bio-inspired Systems, Granada, Spain*; (1999) 61-68. http://www.imse.cnm.es/Chipcat/linan/chip-1.htm (as of Jan. 2001).

[8] T. Roska, L.O. Chua, The CNN Universal Machine: an Analogic Array Computer. *IEEE Transactions on Circuits and Systems, part II*; 40(3) (1993) 163-173.

[9] M. Salerno, F. Sargeni, V. Bonaiuto, Design of a Dedicated CNN Chip for Autonomous Robot Navigation. *Proc. of 6th IEEE International Workshop on Cellular Neural Networks and Their Applications (CNNA 2000)*, Catania, Italy, May 23-25, (2000) 225-228.

[10] A. Zanela, S. Taraglio, A Cellular Neural Network Stereo Vision System for Autonomous Robot Navigation. *Proc. of 6th IEEE International Workshop on Cellular Neural Networks and Their Applications (CNNA 2000)*, Catania, Italy, May 23-25, (2000) 117-122

[11] B.E. Shi, A One-Dimensional CMOS Focal Plane Array for Gabor-Type Image Filtering. *IEEE Transactions on Circuits and Systems, part I*; 46(2) (1999) 323-327.

[12] M. Balsi, Focal-Plane Optical Flow Computation by Foveated CNNs. *Proc. of 5th IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA'98)*, London, UK, Apr. 14-16, (1998) 149 -154.

[13] P. Szolgay, A. Katona, Gy. Eröss, A. Kiss, An Experimental System for Path Tracking of a Robot using a 16×16 Connected Component Detector CNN Chip with Direct Optical Input. *Proc. of 3rd IEEE International Workshop on Cellular Neural Networks and Their Applications (CNNA'94)*, Rome, Italy, Dec. 18-21, (1994) 261-266.

[14] H. Harrer, J.A. Nossek, Discrete-time Cellular Neural Networks. *International Journal of Circuit Theory and Applications*; 20 (1992) 453-467.

[15] T. Roska, L. Kék, L. Nemes, À. Zaràndy, M. Brendel, CSL-CNN Software Library. *Report of the Analogical and Neural Computing Laboratory, Computer and Automation Institute, Hungarian Academy of Sciences*, Budapest, Hungary, 2000.

[16] P.V.C. Hough, Method and means of recognizing complex patterns, *U.S.Patent 3,069,654* (1962). (See also : A.K.Jain, *Fundamentals of digital Image Processing*, Prentice Hall 1989.)

[17] À. Zaràndy, The Art of CNN Template Design. *International Journal of Circuit Theory and Applications*; 27 (1999) 5-23.

[18] T. Terano, K. Asai, M. Sugeno, *Fuzzy Systems Theory and its Applications*. Academic Press, Boston, 1992.

[19] K.H. Chen, W.H. Tsai, Vision-Based Obstacle Detection and Avoidance for Autonomous Land Vehicle Navigation in Outdoor Roads. *Automation in Construction* 10 (2000) 1-25.

[20] C.H. Ku, W.H. Tsai, Obstacle Avoidance for Autonomous Land Vehicle Navigation in Indoor Environments by Quadratic Classifier. *IEEE Transactions System Man and Cybernetics, part B: Cybernetics*, 29(3) (1999) 416-426.

[21] P. Perona, J. Malik, Scale Space and Edge Detection using Anisotropic Diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7) (1999) 629-639.

[22] L. Vilariño, D. Cabello, M. Balsi, V.M. Brea, Image Segmentation based on Active Contours using Discrete Time Cellular Neural Networks. *Proc. of Fifth IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA-98)*, London, UK, Apr. 14-17, (1998) 331-336.