Cellular Neural Networks and Mobile Robot Vision

Marco Balsi[1] and Xavier Vilasís-Cardona[2]

[1]Dipartimento di Ingegneria Elettronica, Università "La Sapienza",
via Eudossiana 18, 00184 Rome, Italy - balsi@uniroma1.it
[2] Departament d'Electrònica, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull,
Pg. Bonanova 8, 08022 Barcelona, Spain - xvilasis@salleURL.edu

Abstract

We propose Cellular Neural Networks (CNNs) for solving the visual-feedback problem in robot guidance. In particular, we show how CNNs provide the necessary image processing to guide an autonomous mobile robot in a maze made of black lines on a light surface. The system consists of a fuzzy controller performing the elementary navigation tasks fed by the result of processing the image only by CNN techniques. We use this solution to make some considerations on more difficult problems such as curved line following and obstacle avoidance.

keywords: Cellular Neural Networks, Robot Vision, Fuzzy Logic.

1 Introduction

A challenge in automous robotics is navigation in unstructured environments with vision-based algorithms. This problem has been tackled with sophisticated image processing techniques, ranging from mathematical morphology to artificial neural networks, and partial solutions have been achieved: algorithms exist to solve obstacle avoidance, cliff detection, autonomous highway driving or active stereo vision for tracking. However, these methods require a large amount of computing effort, forcing a trade-off between real autonomy and real-time operation. To break this compromise, we propose using Cellular Neural Networks (CNNs) [1, 2, 3], which are arrays of locally connected cells or neurons. CNNs have two good points for solving the vision-based navigation problem : first of all, they show a natural suitability for image processing and second, they have direct VLSI implementation, allowing for real-time parallel computation. Still, one needs to prove that CNNs have enough image processing capacity to enable robot navigation. Therefore, our goal is to start with a simple problem, yet with real life relevance: line following and landmark recognition in a maze [4, 5]. The solution shows to be a first step towards tackling more complex problems such as curved line following or obstacle avoidance.

Line (or equivalent marking) following is a classical problem in robot navigation. Solutions found in the literature are either simple but with limited capability, either complex, facing realistic problems and using rather complicated hardware. An example of simple system is the ARGO partially autonomous vehicle [6]. 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. More complex systems are generally based on a computer which is either on board (for large vehicles, e.g. [7, 8]), or remotely connected (e.g. [9]). In these examples images are processed by an additional unit, due to the necessity of high computing power. Besides line following, these vehichles are capable of performing unstructured road navigation, target following and obstacle avoidance.

The paper is organised as follows. First of all, we make a fast review of Cellular Neural Networks and previous uses in robotics. Then, we present our solution for guidance in a maze: a fuzzy controller fed by the information coming from the CNN processing. We describe, then, the image processing in CNN terms and we give some possibilities for hardware implementation and some experimental results. Finally, we study how can we use the maze solution to deal with curved line tracking and obstacle avoidance.

2 Cellular Neural Networks

2.1 The CNN paradigm

Cellular Neural Networks (CNNs) [1, 2, 3] are arrays of dynamical artificial neurons (cells) with local connections only. This essential point has made hardware implementation of large networks possible on a single VLSI chip [10, 11].

Typically, cells are organised in a fairly large two-dimensional grid and an image pixel is assocated to one cell. In this way, CNNs can be used for massively parallel focal-plane processing, avoiding the bottleneck caused by electronic image loading. Moreover, distributed memory and a global control module can be present on chip. In this way the CNN becomes a stored-program massively parallel processor known as the *CNN Universal Machine* (CNN-UM) [12], which is capable of executing a complex sequence of operations (*analogic* programs) in real time. Therefore, CNNs are very good candidates as hardware platforms for robot vision.

In this paper, we shall refer to the Discrete-Time CNN model (DTCNN hereafter) [3]. However, all operations described in the following can also be achieved using continuous-time CNNs. The choice for DTCNNs relies on the limited availability of CNN chips. We are then forced to resort to simulated solutions, on DSPs or either on FPGAs and simulations are faster if DTCNNs are used. The DTCNN core operation is described by the following system of iterative equations:

$$x_{ij}(n+1) = \left(\sum_{kl \in N(ij)} A_{k-i,l-j} \operatorname{sign} x_{ij}(n) + \sum_{kl \in N(ij)} B_{k-i,l-j} u_{ij}(n) + I \right) \quad (1)$$

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;

A is a matrix representing the interaction between cells, which is local (as specified by the fact that summations are taken over the set N of indexes neighbour cells) and space-invariant (as implied by the fact that weights depend on the difference between cell indexes, rather than their absolute values);

B is a matrix representing forward connections issuing from a neighbourhood of inputs.

I is a bias.

$N(i,j)$ is the set of indexes corresponding to cell ij itself and a small neighbourhood (e.g. cell ij and its 8 nearest neighbours). Due to the locality of the computation, implied by the summation over this neighbourhood, and space-invariance, implied by the differences of indexes in matrices A and B in equation (1), it is sufficient to define A and B for a few instances of the indexes, so that they may be represented by small matrices.

The operation performed by the network is fully defined by the so-called *cloning template* $\{A, B, I\}$ (see examples in Table 2). Moreover, under suitable conditions, and with time-invariant input u, a steady state is reached. This steady state depends, in general, on initial state values and input u. Images to be processed are 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 time steps (ranging normally from 10 to 100 according to the task).

Equation (1) can be generalised by including nonlinear and delayed interactions [12]. While non-linearity in the feed-forward (connection to input) path can be exchanged for a suitable linear-connection-based algorithm [13], nonlinear feedback connections allow for real additional functionality.

Unlike most neural networks, very few parameters determine the CNN functionality. So, it is possible to explicitly design [14] a cloning template for a specific task (besides learning it, e.g. [15]), and to broadcast such parameters to all neurons. The cloning template can be understood as an *analogic instruction* for the CNN. *Analogic* instructions can then be combined into *analogic* (analog/logic) programs, using the global controller and local memory. Sophisticated image processing can then be achieved in real time by the suitable sequence cloning templates. Many cloning templates and analog algorithms have been designed for the most diverse tasks [16].

2.2 CNN in Robotics : Previous Approaches

The previous applications of Cellular Neural Networks in robotics can be classified in three main groups: optical flow and time-to-contact estimation, depth estimation by stereo vision and simple line following.

Optical flow estimation is a very rich source of information for assessing the relative motion with respect to the surrounding environment or for tracking a moving object. Although usual algorithms are computationally costly, regularisation-based approach to optical flow can be cast into a CNN-like architecture to achieve local parallel computation [17]. Also, a few one-dimensional arrays are sufficient to make an estimation of the velocity field source (focus of expansion) and magnitude permits evaluation of direction and distance objects for obstacle avoidance purposes [18]. However, the CNN models used for this task are not standard and do not accomodate into general purpose CNN-UM chips.

Stereo vision is widely used for global estimation of depth information. is also a computationally-intensive task, very demanding for traditional architectures. Cellular Neural Networks can be applied to real-time solution of the stereo-matching problem [19]. Hardware implementation of such algorithms [20] is very efficient, but it can be necessary to resort to a multi-chip assembly deal with high-definition images [21].

A simple line-following task has been addressed in reference [22]. Even the task is not complex, simplicity and robustness of the solution is remarkable. Very limited CNN-based hardware resources are proved to perform efficiently in real-time.

3 Navigation in a maze

3.1 The Visual Control System

We have decided to start our robot vision CNN programme by dealing with a simple problem such as driving a robot in a maze made of black lines on light background. Despite its simplicity, this problem involves key features guidance such as navigation (line following) and landmark recognition (crossings and turns). Although there exist simpler solutions, we consider it to be significant benchmark to prove the efficiency of CNNs.

The mobile robot architecture considered is very close to a that of a standard car, with one driving and one steering wheel. In this case, the line following problem can be solved by adapting a well established fuzzy controller solving

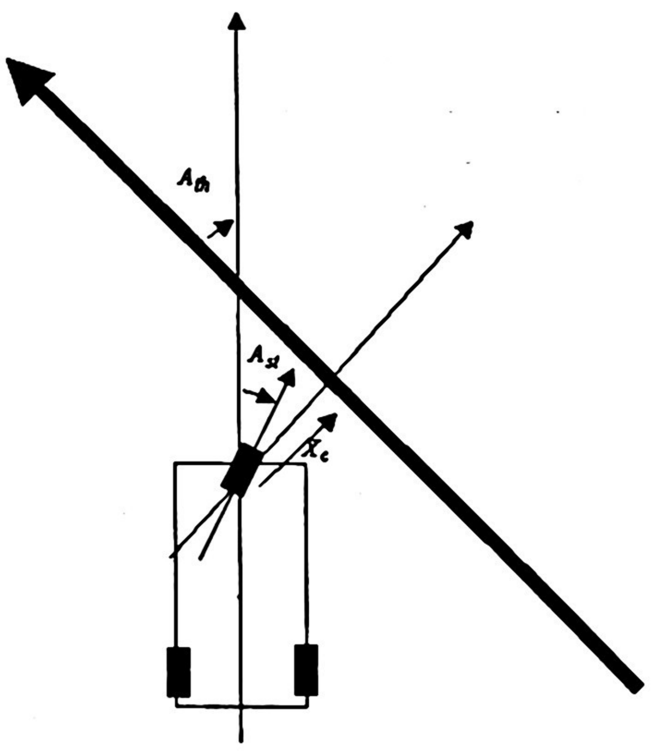

Figure 1: Line parameters. In the situation portrayed A_{st} is positive, A_{th} is positive and X_c is negative (*i.e.* left)

the car backer-upper problem [23]. This fuzzy controller delivers the steering angle given the distance X_c of the centre of the steering wheel from the line to be followed and the angle A_{th} made by the axis of the robot and the line. This is portrayed in Figure 1, where A_{st} is the steering angle.

The robot guidance problem is then split into two processes: a visual processing module and the fuzzy controller. The first is in charge of exctracting the relevant information from the image, namely the number of lines in sight, and its mathematical parameters, and take a decision on what line to follow. The parameters of this line (distance and angle) shall be delivered to the fuzzy controller. The fuzzy rule base is built out of driving experience inspired in reference [23]. The consequent sets are simplified to singletons: this scheme allows for a simpler on-line implementation because it requires fewer calculations. The values for the singletons are adjusted according to the robot platform steering capabilities. 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 1. Membership functions for X_c and A_{th} are shown in Figure 2. Consequents for A_{st} are 0 for Zero and ± 20, ± 40, ± 80 for Small, Medium, Large Left/Right respectively.

The global control strategy works as follows. With a single line in sight, the robot tries to align itself on it as fast as possible. When a second line gets into the view, the type of bend or crossing is assessed, by examining the

Table 1: Fuzzy control rule set for A_{st}.

		A_{th}				
		Positive Large	Positive Small	Zero	Negative Small	Negative L
X_c	Left	Medium Left	Small Right	Medium Right	Medium Right	Large Ri
	Centre	Medium Left	Small Left	Zero	Small Right	Medium R
	Right	Large Left	Medium Left	Medium Left	Small Left	Medium R

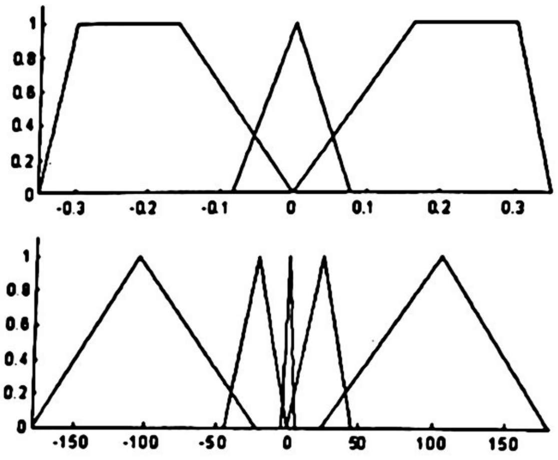

Figure 2: Membership functions for X_c and A_{th}.

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*), then a command from a higher order level is required (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. In this manner, we are capable of guiding the robot over the entire maze.

3.2 Image Processing

In the previous section we have defined the tasks of the image processing module, namely, landmark recognition (crosses and turns) and extraction of line parameters. Actually, landmark recognition just amounts to determine how many lines are in sight. From those, we need to compute X_c and A_{th}. However, prior to line recognition and parameter extraction, the image needs to be acquired and preprocessed. Acquisition shall depend on the actual implementation and will be treated below, while preprocessing will consist in cleaning and enhancing the image contrast. Our goal is to perform all image processing operations using only sequences of CNN templates, in CNN terms, a CNN Universal Machine programme.

For preprocessing we use the so-called *small-object-killer* template (see Ta-

Table 2: Templates used in the processing stage.

	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
connected-component detector	$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{pmatrix}$	0	0

ble 2) to make a preliminary cleaning and binarisation of the acquired image (Figure 3 (a) and (b)).

The calculation of the parameters of lines visible in an image is universally performed by the Hough transform [24], but this is a computationally intensive technique that does not lend itself to efficient implementation on a CNN. We have then to devise a different approach.

In order to perform direction and position evaluation, first we need to get efficiently a thin line. For this purpose, we resort to design [14] 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 2, and its effect is shown in Figure 3 (c). To deal with approximately horizontal lines, we extract the upper edge by using a rotated version of this template.

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 an 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 filter should be used.

Operation of a vertically-tuned filter (Table 2) is depicted in Figures 4 and 5. Other direction-selective filters are obtained 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

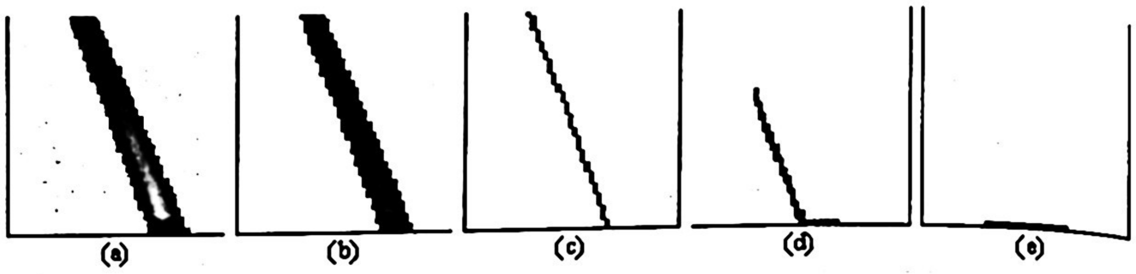
Figure 3: Original image, taken from the actual camera with size 60 × 45 pixels (a), cleaning and binarisation (b), left edge (c), connected-component detector -intermediate result (d), final result (e).

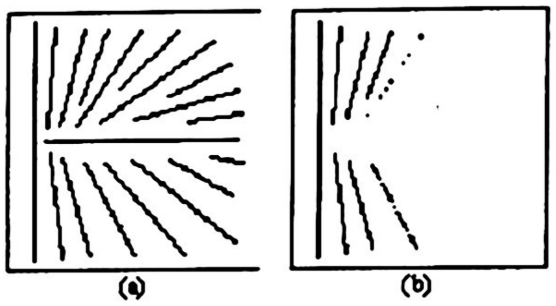
Figure 4: Vertical line extracting filter: original image (a), result (b).

a 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 2). Operation of such template at an intermediate and final stage of processing is depicted in Figure 3(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.

A strong-point of CNN processing is its capability of dealing with low quality or ill defined images, as shown in Figure 6.

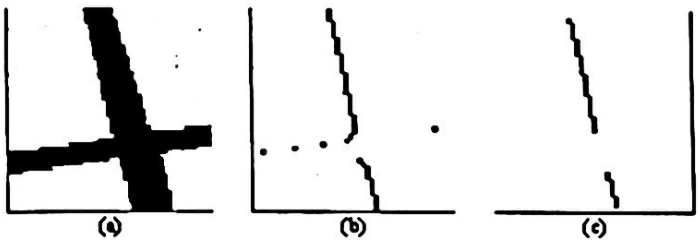

Figure 5: Vertical line extracting filter applied to the image of a crossing (a) and after edge extraction (b): result (c).

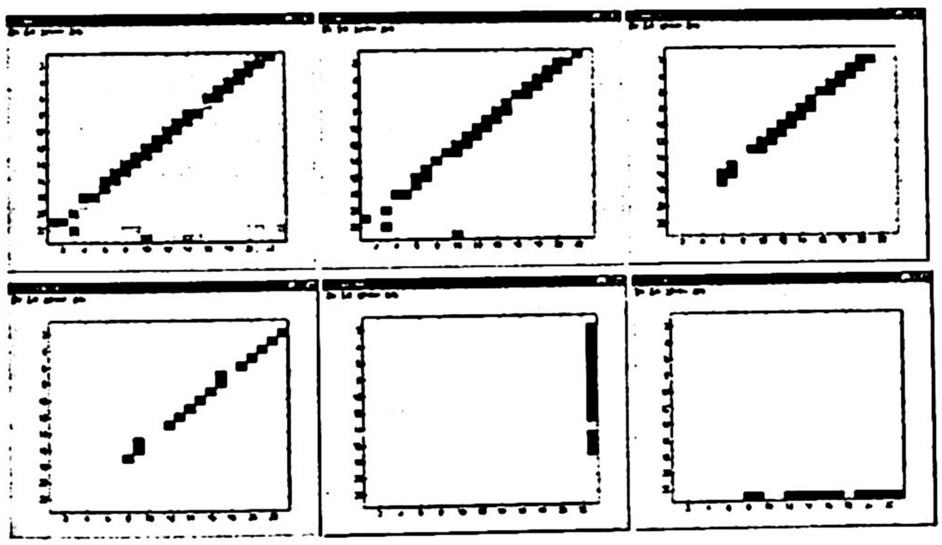

Figure 6: Successful processing of an image with low contrast.

3.3 Hardware Implementation

We have been considering two hardware implementations for a robot to be guided with such algorithms.

The first one is a small autonomous three-wheeled cart driven by a motor mounted 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 digitised 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. Image size is 60×45 pixels and lines on the floor are obtained using a 2cm wide black tape. A power board feeds the motors, and batteries are carried on board. In 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 CNN chip were used instead of the DSP, it would be possible to reach a much higher speed.

The second has similar mechanic characteristics but is controlled by an Xilinx FPGA development board. This solution is not fully autonomous since requires external power supply.

3.4 Experimental 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 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 floor 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 7 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.

4 Further Perspectives

The setup described in section 3 can be extended to other navigation problems provided we are able to define a virtual line to be followed. The parameters of this virtual line are then sent to the fuzzy controller. Bearing this in mind, we can consider more evolved problems such as curved line following or obstacle avoidance.

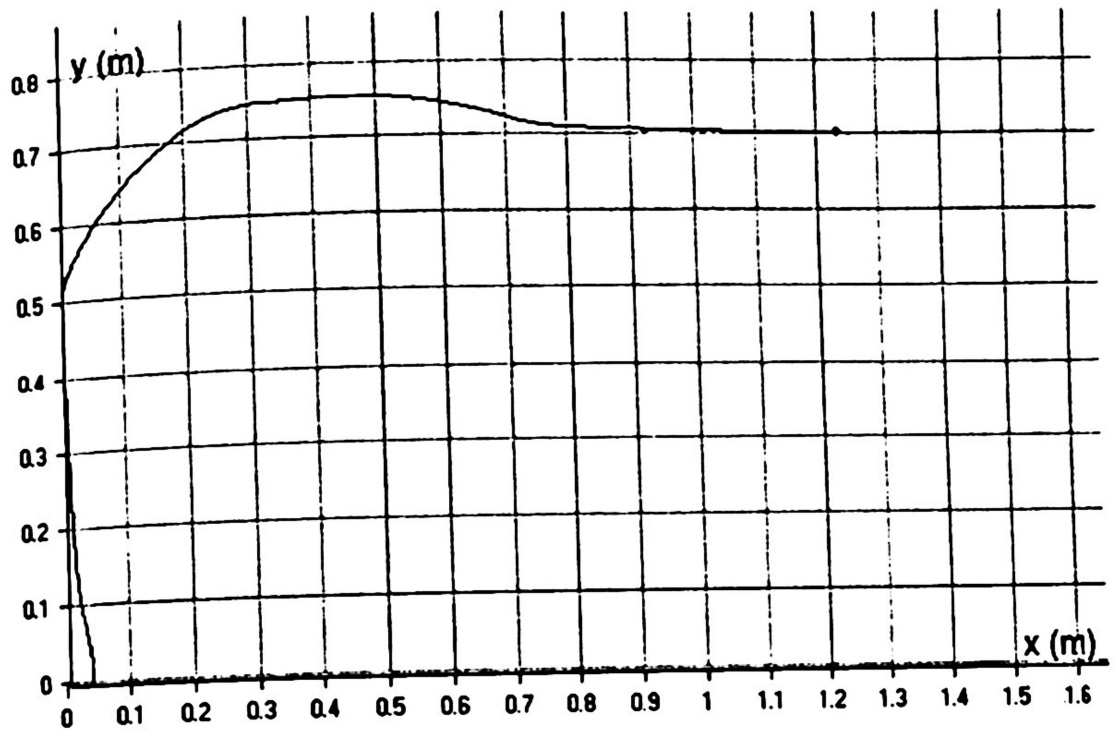

Figure 7: Path followed by the front wheel of the robot. Axis marking in meters.

4.1 Curved Line Following

Curved lines can be addressed by modifying the algorithms used for continuous straight lines. An accurate driving strategy has to take into account the following requirements: small curvature lines should be followed as close as possible, while for large curvature ones, the manoeuvre should be globally optimised, much in the same way as turning at a crossing. For this purpose, we estimate a *short-term* approximation to the curved line which is a the tangent line at the closest end (line *a* in Figure 8), and a *long-term* one which is the secant line joining farthest ends of curve in the image (line *b* in Figure 8). When the long-term approximation is similar to the short-term one, close following is possible using of the tangent line. When the angle difference between the two approximations is large, then substantial correction of the steering angle is necessary. This can be performed by adding, for instance, a supervisory fuzzy control algorithm using the information of both the tangent and the secant. Rules are of the form *if secant is more to the right than tangent, then correct steering angle towards right,* and *if secant is more to the left than tangent, then correct steering angle towards left.*

This correction can also be applied in two other cases: when coarse sampling in time is being used or when speed is relatively high so that the robot has moved a long path before the evaluation of the following image. The tangent line can then be estimated by the projection approach described in Section 3, but just projecting an area of few lines near the bottom of the image instead of the

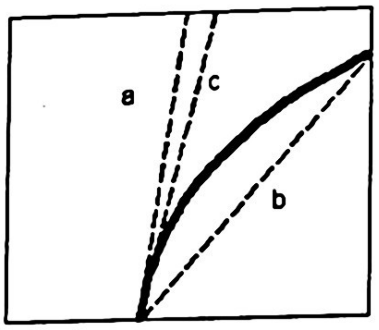

Figure 8: Tangent and secant to the curve.

whole image, so as to obtain a short-term secant, resembling the tangent (line c in Figure 8). This solution, however, is very sensitive to the image noise quantisation.

An alternative approach is based on computing one or more intermediate projections. It is equivalent to making a piece-wise-linear approximation of curve. The bottom projection delivers the short-term approximate line, while other projections can be used for correction.

4.2 Obstacle Avoidance

Obstacle avoidance involves two main tasks which are recognising obstacles such and, then, driving to avoid collision.

Obstacle recognition depends strongly on the characteristics of the environment. For our purposes, we consider an obstacle anything that develops in vertical dimension, departing from the floor level. This means that we shall consider colour, luminosity, texture or shape as indicators. These could just patterns on the floor and make no obstacle for the robot.

Solutions exist for binocular vision by stereo-matching-based range-finding (see for instance references [19, 20]). For high-definition or foveated imaging we can also use an optical-flow-based strategy (e.g. [17, 18]). Still we look solutions using our current hardware based only on monocular low-definition vision. We believe that working with such limitations will prompt us to look for solutions that optimise simplicity and cost.

A vision-only approach is based on checking the deformation of a suitable light pattern projected ahead. For instance, if we project a straight line light it will remain straight and in in a known position. When an obstacle (or level change, such as a step) is present, the line is deformed and displaced (Figure 9 (a)). It is easy to recognise such situation by CNNs, for instance, by projecting shadows of perceived objects on the expected line (Figure 9 (b)). Since obstacle recognition takes a relatively large amount of resources, it may be convenient to track objects once they are recognised rather than performing

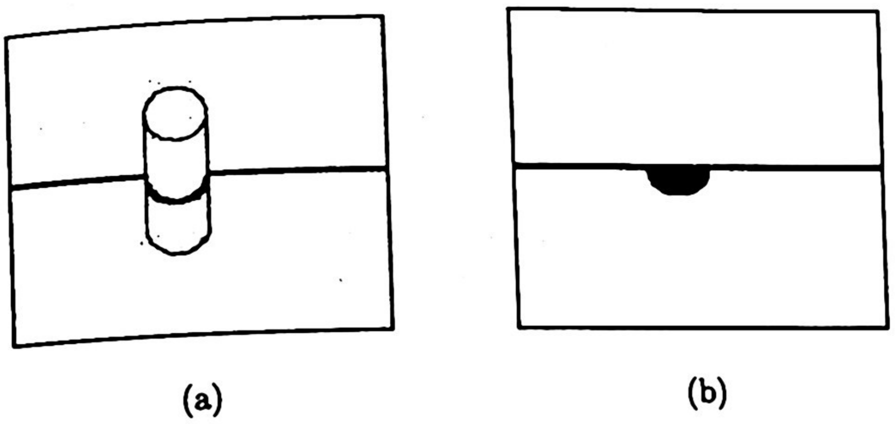
Figure 9: Line deformation in the presence of an object (a) and projection of the deformed line (b).

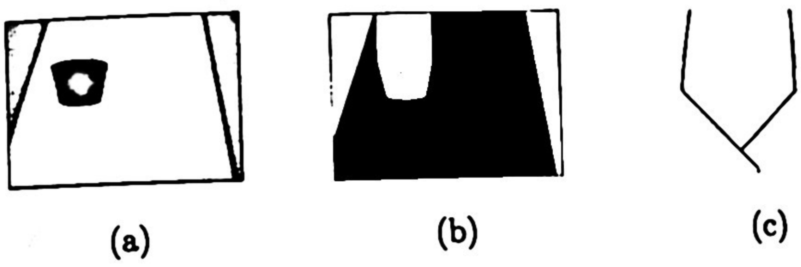
Figure 10: A corridor with an obstacle (a). A CNN template can fill the area available to move the robot (b). A skeletonisation of the available area (c).

recognition from scratch at every step. Efficient CNN-based tracking strategies can be based on active contours [25, 26].

An obstacle-avoiding driving strategy can be built on the line-following approach as follows. Suppose the robot is driving along a corridor with an obstacle in its way (Figure 10 (a)). Once the obstacle is recognised, we can use its contours and boundaries of the hallway as limits for the path. We apply then a CNN template to fill all available area by propagation (Figure 10 (b)). We can then take the skeleton of the shape obtained (Figure 10 (c)), or just draw a line pointing to the centre of the farthest and widest end of the shape obtained.

5 Summary and Conclusions

We have shown how the combination of Cellular Neural Network image processing and a fuzzy control can solve the robot vision problem in a simple case such as the motion in a maze. The image processing steps can be understood as an *analogic programme* in CNN-UM terms.

While the current implementation is based on software realisation of building blocks on programmable hardware, the target implementation is based special-purpose VLSI CNN hardware, yielding substantially enhanced performance.

The solution for this problem gives a hint for studying more complicated problems such as curved line following or obstacle avoidance. Still, this first step towards a wider use of CNNs in robotics for visual feedback. Further contributions should stress the use of CNN chips on a hardware implementation and the formulation of the image processing steps in CNN terms.

Acknowledgements This work is partially supported by Funitec (PGR2001-05). Authors would like to thank Jordi Solsona, Sonia Luengo, Alessandro Maraschini, Giada Apicella, Valentina Giovenale and Ruben Funosas for their collaboration in the project.

References

[1] Chua, L.O., Yang, L. (1988) Cellular Neural Networks: Theory. IEEE Trans. Circ. Syst. **CAS-35** 1257-1272

[2] Chua, L.O., Roska, T. (1993) The CNN Paradigm. IEEE Trans. Circ. Syst. **CAS-I-40** 147-156

[3] Harrer, H., Nossek, J.A. (1992) Discrete-time Cellular Neural Networks. Int. of Circ. Th. Appl.**20** 453-467

[4] Vilasís-Cardona, X., Luengo, S., Solsona, J., Apicella, G., Maraschini, A., Balsi, M. (2002) Guiding a mobile robot with Cellular Neural Networks. *to appear Int. J. of Circ. Th. Appl.*

[5] Balsi, M., Vilasís-Cardona, X. (2002) Robot Vision using Cellular Neural Networks mobile robot vision. *Zhou, C., Maravall, D., Ruan, D. Eds. Autonomous Robotic Systems*, Physica-Verlag 2002.

[6] ARGO Project (2000), http://nannetta.ce.unipr.it/ÃRGO/english/index.html (as of December 2000)

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

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

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

[10] Liñan, L., Espejo, S., Domínguez-Castro, R., Rodríguez-Vázquez, A. (2002) ACE4K: An Analog VO 64x64 Visual MicroProcessor Chip with 7-bit Analog Accuracy. Int. J. of Circ. Th. Appl., in press

[11] Kananen, A., Paasio, A., Laiho, M., Halonen, K. (2002) CNN Applications from the Hardware Point of View: Video Sequence Segmentation. Int. J. of Circ. Th. Appl., in press

[12] Roska, T., Chua, L.O. (1993) The CNN Universal Machine: an Analogic Array Computer . IEEE Trans. Circ. Syst. CAS-I-40 163-173

[13] Kék, L., Zarándy, A. (1998) Implementation of large-neighbourhood non-linear templates on the CNN universal machine. Int. J. of Circ. Th. Appl. **26** 551-566

[14] Zaràndy, À. (1999) The Art of CNN Template Design. Int. J. of Circ. Th. Appl. **27** 5-23

[15] Kozek, T., Roska, T., Chua, L.O. (1993) Genetic algorithm for CNN Template Learning. IEEE Trans. Circ. Syst., CAS-I-40 392-402

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

[17] Balsi, M. (1998), Focal-Plane Optical Flow Computation by Foveated CNNs. Proc. of Fifth IEEE Int. Workshop on Cellular Neural Networks and their Applications (CNNA-98), London, UK, Apr. 14-17, 1998, 149-154

[18] Shi, B.E. (1999) A one-dimensional CMOS focal plane array for Gabor-type image filtering. IEEE Trans. Circ. Syst. CAS-I-46 323-327

[19] Zanela, A., Taraglio, S. (1998) Seeing the Third Dimension in Stereo Vision Systems: a Cellular Neural Network Approach. Eng. Appl. Artif. Intel. **11** 203-213

[20] Sargeni, F., Bonaiuto, V. (2001) CNN Cell for Computing Disparity Map. Elec. Lett. **37** 682-683

[21] Titus, A.H., Drabik, T.J. (2000) Analog VLSI Implementation of the Help if Needed Stereopsis Algorithm. IEEE Trans. Circ. Syst. CAS-II-47 1328-1337

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

[23] Kosko, B. (1992) Neural Fuzzy Systems, Prentice Hall Englewood Cliffs, NJ

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

[25] Vilariño, D.L., Brea, V.M., Cabello, D., Pardo, J.M. (1998) Discrete-time CNN for image segmentation by active contours. Patt. Rec. Lett, **19** 721-734

[26] Vilariño, D.L. (2001) Contornos activos a nivel de pixel: diseño e implementación sobre arquitecturas de redes no lineales celulares, Ph.D. thesis, Univ. of Santiago de Compostela, Spain