CNNA-94 Third IEEE International Workshop on Cellular Neural Networks and their Applications
Rome, Italy, December 18-21,1994

Some Examples of Preprocessing Analog Images with Discrete-Time Cellular Neural Networks

Hubert Harrer*, Péter L. Venetianer†‡, Josef A. Nossek*, Tamás Roska†‡ and Leon O. Chua†

* Institute for Network Theory and Circuit Design, Technical University of Munich, D-80290 Munich, Germany;

† Department of Electrical Engineering and Computer Sciences, ERL, University of California Berkeley, Berkeley, CA 94720, USA

‡ Analogical and Neural Computer Laboratory, Computer and Automation Inst, Hungarian Academy of Sciences, H-1518 Budapest, Hungary

Abstract - *The paper gives two examples, where an analog input image is preprocessed by a sequence of templates, i.e. by analogic CNN algorithms running on the CNN Universal Machine [1]. The examples are: the extraction of horizontal screws with any size in length and the classification of screws according to their size.*

1 Introduction

This paper gives two examples of preprocessing grey-scale input images using the CNN Universal Machine (CNNUM) with analogic multipath CNN algorithms [2]. Let us consider the hypothetical problem, where a robot should select screws of given size and position on an assembly line. This is a typical example, where an analog input image taken by a CCD camera has to be processed in real time with a very high speed, which is determined by the assembly line and the reaction delay time of the robot. For this purpose a Discrete-Time Cellular Neural Network (DTCNN) [4] will be used to process the analog image data and extract specific features. The binary output image can then be postprocessed by a digital signal processor controlling the mechanics of the robot.

Our main concern was to write algorithms that can be implemented on the CNN chips of the present and near future. In our algorithm we use a DTCNN having two local logic memories (LLM I and LLM II) at each cell. Both of them can store the output of an operation or serve as an input [3]. Since the frame rate of an analog hardware implementation [3] is much higher than required by these examples, and since all templates used in the algorithm are local, it is possible to partition the input image into smaller overlapping regions and process them separately on the restricted sized chip.

Section 2 shows how horizontal screws can be extracted. In Section 3 screws are

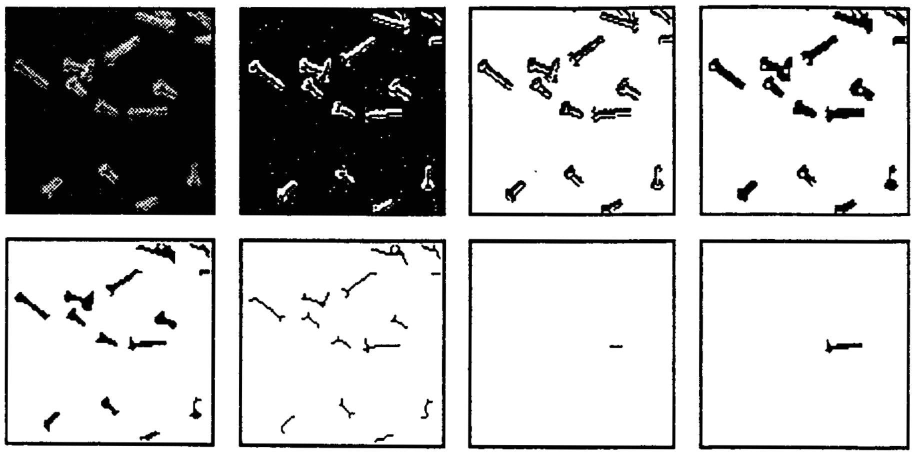

Figure 1: The steps of extracting horizontal screws: (a) input image (b) contour extraction (c) noise removal (d) eliminating light reflections (e) final black-and-white image (f) skeletonization (g) horizontal line extraction (h) final output image.

extracted, which are smaller than a given length.

2 Extracting Horizontal Screws

The task is to extract all horizontal screws in a grey-scale input image. An example is given in Fig. 1, where only a single horizontal screw has been detected. Screws touching each other are not extracted. The input (Fig. 1a) is a 101 by 101 pixel image and has been obtained from a photo scanned with a resolution of 60 dpi. It is emphasized that the input image has to be loaded only once at the very beginning and the sequential templates process the outputs obtained from the previous iteration. If not mentioned separately, all templates operate with the first local logic memory (LLM 1). The algorithm consists of two phases: first (Step 1 - Step 4), the grey-scale input is converted to a black-and-white image, preserving all important information (shape, size, position) of the screws, but suppressing noise. In the second phase the horizontal lines are extracted. Next, the single steps are explained in detail.

Step 1: Contour Extraction

The contours of the screws are extracted using the template

$$\mathrm{a} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad \mathrm{b} = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 4.5 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array} \qquad i = 0.25.$$

The result is shown in Fig. 1b. Since the template performs only a linear thresholding, a single iteration is sufficient to obtain a convergent output image. For more detailed information refer to [4].

Step 2: Noise Removal

The image is inverted and the noise (single white pixels) is eliminated using the following template for a single iteration:

$$\mathbf{a} = \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & -9 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array} \qquad \mathbf{b} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad i = -2.$$

Step 3: Eliminating Light Reflections

The output image in Fig. 1c contains superfluous edges, which arise from light reflections of the metal screws. To eliminate these reflections, a cyclic feedback template (1) of period 2 has been used. It peels off black pixels, if they are surrounded by white pixels either horizontally, or vertically.

$$\mathbf{a}_1 = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad \mathbf{a}_2 = \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \qquad \mathbf{b} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad i = 1. \tag{1}$$

After 3 cycles the result of this step is shown in Fig. 1d.

Step 4: Eliminating Black Spots

The objects contain white holes. Since they are restricted in size to a square of only 4 pixels they can be eliminated by increasing black areas with the template

$$\mathbf{a} = \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline 1 & 1 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \qquad \mathbf{b} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}. \qquad i = 4.$$

in a single iteration. Afterwards, the objects are decreased using the same template with a changed threshold of $i = -4$. The obtained output image is the black-and-white representation of the grey-level image in Fig. 1e. It is stored in the second local logic memory (LLM II).

Remark:

This step has the disadvantage that when increasing the black objects any screws being too close to each other, merge to a single object and cannot be separated by the following decreasing step. This is avoided if a single cycle of the skeletonization template in Step 5 is applied before Step 4. Then, one layer of pixels is already peeled off, so the screws can be increased without merging.

Step 5: Skeletonization

Skeletonization is the central part of this application. The operation is performed using a cyclic template of period 8, peeling off a layer of pixels from one side of the object in a single iteration. For more details refer to [5]. After 3 cycles (24 iterations) the image in Fig. 1f has been obtained. The control template is 0, the current is $i = -0.75$ for each step, while the 8 different feedback templates are as follows:

$$\mathbf{a}_1 = \begin{array}{|r|r|r|}\hline 0.25 & 0.25 & 0 \\ \hline 0.25 & 1.75 & -0.25 \\ \hline 0 & -0.25 & 0 \\ \hline \end{array} \qquad \mathbf{a}_2 = \begin{array}{|r|r|r|}\hline 0.25 & 0.25 & 0.25 \\ \hline 0 & 1.75 & 0 \\ \hline -0.25 & -0.25 & 0 \\ \hline \end{array}$$

$$\mathbf{a}_3 = \begin{array}{|r|r|r|}\hline 0 & 0.25 & 0.25 \\ \hline -0.25 & 1.75 & 0.25 \\ \hline 0 & -0.25 & 0 \\ \hline \end{array} \qquad \mathbf{a}_4 = \begin{array}{|r|r|r|}\hline -0.25 & 0 & 0.25 \\ \hline -0.25 & 1.75 & 0.25 \\ \hline 0 & 0 & 0.25 \\ \hline \end{array}$$

$$\mathbf{a}_5 = \begin{array}{|r|r|r|}\hline 0 & -0.25 & 0 \\ \hline -0.25 & 1.75 & 0.25 \\ \hline 0 & 0.25 & 0.25 \\ \hline \end{array} \qquad \mathbf{a}_6 = \begin{array}{|r|r|r|}\hline 0 & -0.25 & -0.25 \\ \hline 0 & 1.75 & 0 \\ \hline 0.25 & 0.25 & 0.25 \\ \hline \end{array}$$

$$\mathbf{a}_7 = \begin{array}{|r|r|r|}\hline 0 & -0.25 & 0 \\ \hline 0.25 & 1.75 & -0.25 \\ \hline 0.25 & 0.25 & 0 \\ \hline \end{array} \qquad \mathbf{a}_8 = \begin{array}{|r|r|r|}\hline 0.25 & 0 & 0 \\ \hline 0.25 & 1.75 & -0.25 \\ \hline 0.25 & 0 & -0.25 \\ \hline \end{array}$$

Step 6: Extracting Horizontal Lines
Next, all horizontal lines are extracted, which can be performed by the template

$$\mathbf{a} = \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline 1 & 1 & 1 \\ \hline -1 & -1 & -1 \\ \hline \end{array} \qquad \mathbf{b} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad i = -8.$$

The template leaves a black pixel unchanged only if there is a top and bottom white row, and the left and right neighbors are black. Because the template peels off the corner pixels of horizontal lines (no black left or right neighbor), the horizontal lines are decreased from their endpoints in each iteration. In the described example the template is applied for three iterations to delete short horizontal lines not belonging to the body of the screw. The output image is given in Fig. 1g.

Step 7: Restoring Objects
The whole horizontal screw is restored by the template

$$\mathbf{a} = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad \mathbf{b} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 8 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \qquad i = 0, \tag{2}$$

which increases objects of the initial state according to a mark in the input image. Therefore, applying the extracted horizontal lines (Fig. 1g) as initial state and the black-and-white representation of all screws (Fig. 1e) stored in LLM II to the input results in all horizontal screws (Fig. 1h).

Remark:
Because the operations are only global within the size of a single screw, the input image can be partitioned, if there is a minimum overlap determined by the length of the longest screw. The total number of iterations is 54 for the input image in Fig. 1a.

3 Extracting Small Screws

In the next example the task is to extract all screws smaller than a given size independent of their direction. Fig. 2a shows the analog input image consisting of screws of two different sizes. In Fig. 2d only the small screws are shown, if they are not touching another screw.

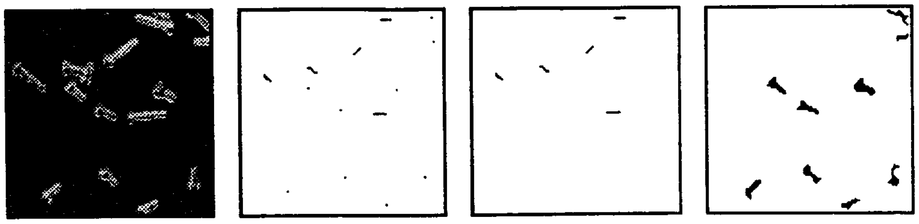

Figure 2: Steps of extracting small screws: (a) analog input image (b) center point detection (c) single black spots deleted (d) final output image.

To obtain a black-and-white representation of the grey-level image, Steps 1 to 4 are identical to the previous example. Here, a single cycle of the skeletonization template in Step 5 has been used before Step 4 to prevent the merging of objects. The templates operate with the first local memory (LLM I), while the black-and-white output of Step 4 is stored in the second local logic memory (LLM II).

Step 5: Center Point Detection

Center point detection is the central part of this application. This operation reduces objects to a single pixel in a time proportional to the size of the object. A cyclic 1-neighborhood template of period 8 has been used for this task. The control template of all steps is 0.

$$\mathbf{a}_1 = \begin{array}{|c|c|c|}\hline 0.25 & 0 & 0 \\ \hline 0.25 & 1.75 & -0.25 \\ \hline 0.25 & 0 & 0 \\ \hline \end{array} \qquad i_1 = -1.0 \qquad \mathbf{a}_2 = \begin{array}{|c|c|c|}\hline -0.25 & 0 & 0.25 \\ \hline 0 & 1.75 & 0.25 \\ \hline 0.25 & 0.25 & 0.25 \\ \hline \end{array} \qquad i_2 = -0.5$$

$$\mathbf{a}_3 = \begin{array}{|c|c|c|}\hline 0.25 & 0.25 & 0.25 \\ \hline 0 & 1.75 & 0 \\ \hline 0 & -0.25 & 0 \\ \hline \end{array} \qquad i_3 = -1.0 \qquad \mathbf{a}_4 = \begin{array}{|c|c|c|}\hline 0.25 & 0 & -0.25 \\ \hline 0.25 & 1.75 & 0 \\ \hline 0.25 & 0.25 & 0.25 \\ \hline \end{array} \qquad i_4 = -0.5$$

$$\mathbf{a}_5 = \begin{array}{|c|c|c|}\hline 0.25 & 0.25 & 0.25 \\ \hline 0 & 1.75 & 0.25 \\ \hline -0.25 & 0 & 0.25 \\ \hline \end{array} \qquad i_5 = -0.5 \qquad \mathbf{a}_6 = \begin{array}{|c|c|c|}\hline 0 & -0.25 & 0 \\ \hline 0 & 1.75 & 0 \\ \hline 0.25 & 0.25 & 0.25 \\ \hline \end{array} \qquad i_6 = -1.0$$

$$\mathbf{a}_7 = \begin{array}{|c|c|c|}\hline 0.25 & 0.25 & 0.25 \\ \hline 0.25 & 1.75 & 0 \\ \hline 0.25 & 0 & -0.25 \\ \hline \end{array} \qquad i_7 = -0.5 \qquad \mathbf{a}_8 = \begin{array}{|c|c|c|}\hline 0 & 0 & 0.25 \\ \hline -0.25 & 1.75 & 0.25 \\ \hline 0 & 0 & 0.25 \\ \hline \end{array} \qquad i_8 = -1.0$$

After three cycles the small, non-touching screws are reduced to the size of a single pixel, while larger objects are still represented by several pixels (Fig. 2b). It is emphasized that the number of cycles determines the maximum length of the screws, which are detected.

Step 6: Deleting Single Black Points

Now, single black points are deleted by applying the noise removal template for one step (Fig. 2c).

$$\mathbf{a} = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 9 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array} \qquad \mathbf{b} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array} \qquad i = -2.$$

Step 7: Restoring the Remaining Objects

The previous step deleted all single pixels, corresponding to smaller screws. Therefore, all pixels of Fig. 2c belong to larger screws. These screws can be restored by template (2).

Step 8: Subtraction of the Large Screws from the Small Screws

To obtain the small screws in the final output image, the result of the previous step is substracted from the black and white image stored on LLM II by applying the zero-neighborhood template $a = -1$ $b = 1$ $i = -1$.

Remark:

Because the operations are only global within the size of a single screw, the input image can be partitioned, if there is a minimum overlap determined by the length of the longest screw. The final output image has been obtained after 71 iterations.

4 Conclusion

The paper has demonstrated two examples how to use analogic CNN algorithms to process grey-level input images with discrete-time cellular neural networks. It is emphasized that the number of iterations has been independent of the image size for all examples. Only the size of the screws is important to determine the total number of iterations. This task could be solved by the hardware implementation described in [3]. The estimated time for processing a single partition of the image amounts to less than 0.1ms. This includes the time for reading the analog input image and writing the binary output image. The templates are always applied to the output image of the previous operation. As future hardware implementations are expected to implement a larger cell density than the chip in [3] and the architecture allows an easy connection of several chips on a board, the required cell size should not be a problem.

References

[1] T.Roska and L.O.Chua, "The CNN Universal Machine: An Analogic Array Computer", IEEE Trans. on Circuits and Systems-I, Vol.40, pp.163-173, 1993.

[2] L.O.Chua, T.Roska, P.L.Venetianer and Á.Zarándy, "Some Novel Capabilities of CNN: Game of Life and Examples of Multipath Algorithms" Proc. of IEEE Intl. Workshop on Cellular Neural Networks and Their Applications, pp.276-281, 1992.

[3] H.Harrer, J.A.Nossek, T.Roska and L.O.Chua, "A Current-mode DTCNN Universal Chip", Proc. of IEEE Intl. Symposium on Circuits and Systems, pp.135-138, 1994.

[4] H.Harrer, "Discrete time cellular neural networks", Dissertation, Shaker Verlag, Aachen, 1992.

[5] P.L.Venetianer, F.Werblin, T.Roska and L.O.Chua, "Analogic CNN Algorithms for some Image Compression and Restoratin Tasks", Technical Report UCB/ERL M94/30, 1994.