# NOVEL TYPES OF ANALOGIC CNN ALGORITHMS FOR RECOGNIZING BANK-NOTES

by

A. Zarándy, F. Werblin, T. Roska, and L. O. Chua

Memorandum No. UCB/ERL M94/29

22 April 1994

# NOVEL TYPES OF ANALOGIC CNN ALGORITHMS FOR RECOGNIZING BANK-NOTES

by

A. Zarándy, F. Werblin, T. Roska, and L. O. Chua

Memorandum No. UCB/ERL M94/29

22 April 1994

**ELECTRONICS RESEARCH LABORATORY**

College of Engineering
University of California, Berkeley
94720

# NOVEL TYPES OF ANALOGIC CNN ALGORITHMS FOR RECOGNIZING BANK-NOTES

by

A. Zarándy, F. Werblin, T. Roska, and L. O. Chua

Memorandum No. UCB/ERL M94/29

22 April 1994

## ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

# Novel Types of Analogic CNN Algorithms for Recognizing Bank-notes

Á. Zarándy+, F. Werblin++, T. Roska+, and L.O. Chua

Electronics Research Laboratory, U.C. Berkeley
+Comp. Aut. Inst. Hungarian Academy of Science
++Molecular Cell Biology, U.C. Berkeley

## 1 INTRODUCTION

Cellular Nonlinear/Neural Networks (CNN) are regular, multiprocessor, nonlinear dynamic arrays with mainly local interconnections [1-2]. The invention of the CNN Universal Machine [3] provides for the application and efficient implementation of analogic CNN algorithms. In these algorithms the various CNN templates [1-3] are combined according to a flowchart. In addition to the global logic of the flowchart, the analog dynamics is combined with logic on the cell level as well.

In this paper we show some novel types of analogic algorithms. These algorithms make complex decisions in images without reading out the CNN chip. This makes them extremely time, area, and power effective. Two crucial effects are emphasized: diffusion type templates are applied during a finite time interval and local logic operates within well defined parts (patches) in the image plane. Hence, a new type of pattern recognition algorithm is introduced. The technique is shown by solving an important and useful problem: recognizing (detecting) color bank-notes.

Recognizing bank-notes became important by the advent of high quality color copiers. The problem is to make built-in protection against illegal copying. Speed and reliability is extremely important. Our solution meets both criteria.

In Section 2 we introduce the problem, in Section 3 the general concept of the algorithms is described, and in Section 4 the necessary new algorithmic elements are described (citing those which was used from the literature as well). In Section 5 we present the solutions for recognizing different US and Canadian bank-notes.

## 2 THE PROBLEM

In the last couple of years the quality of color copying increased significantly. Unfortunately, this very important development also led to the possibility of making very good copies of bank-notes. Thus it has become very important to equip copiers with a real-time built-in protection against counterfeiting.
Solving the problem we have two objectives:

- prevent copying bank-notes;
- do not prevent copying anything else.

These two objectives are slightly contradictory in the sense that we cannot expect 100% reliability for both at the same time. The bank-note can be dirty, scratched, lighter than a brand new one. So we cannot look for a perfect match, and this might result in false alarms.

The speed of the copiers increased as well. At a speed of 4 or 5 color copies per second, it means that within 200 or 250 ms a whole letter sized color image (frame) must be checked. Naturally we have to consider that all of these bank-notes are put on the frame with arbitrary shifts and rotations. This makes the problem even much harder.

In addition, if a user did not try to copy anything illegal in the frame, he or she should not notice any slow down in the copying process.

We have reformulated our problem by considering all these aspects. In most cases, when there is nothing illegal in the frame, we need a very fast decision to allow copying. ur solution can make this decision within the time limit, while maintaining a low false alarm rate.

## 3 GENERAL CONCEPTS OF THE ALGORITHM

Our task is the following. Given a color input image (frame). Make a decision, whether it contains a bank-note or not. Novel types of analogic algorithms with two new features will be used.

- When running these algorithms on a CNN Universal Machine, the analogic chip implements there without reading out the details of the processed image.
- Not just single objects but groups of objects together are processed fully and in parallel. We called these operations the spatial logic.

### 3.1 On-chip decision making

We define a set of image features to be tested by analogic CNN algorithms. These features can be topologic, chromatic,...e.t.c, and may apply to the features directly or their enlargements. The resulting images from these testing algorithms are always binary. They are black (+1, true) in those locations, where the sought feature was found, white (-1, false) everywhere else. If we wish to test for the existence of a known image part (bank-note) within a frame, we have to do carry out the following steps:

- select several characteristic features of the sought image part;
- test forthe existence of the selected features on the frame;
- combine the results by applying appropriate logic operations on them.

The final result is a binary image. If it is completely white, we can be sure that the sought image part was not in the frame. This can be tested with the global line function of the CNN Universal Machine without reading out the image[3].

## 3.2 The spatial logic

Though the inputs are true color pictures, the algorithms are mostly dealing with binary images. In these images there are many objects. The new idea in the spatial logic is that the related objects are collected into groups, and processed together. We consider objects to be related if they are geometrically close to each other in the frame (Figure 1).

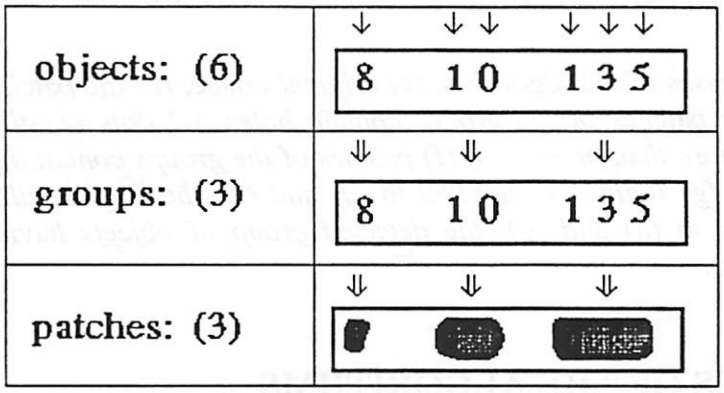


*Figure 1* *Objects, groups, patches.*

The classification of the objects is achived by using a diffusion type template (see Section 4.2). It creates patches which cover totally all objects belonging to the appropriate group. Patches covering different groups must be separated. Figure 1 illustrates the meaning of the objects, the groups, and the patches. To understand why these patches are important, consider an example.

Given a binary image (Figure 2a), suppose we wish to detect those numbers which contain more than one digits <u>and</u> some of those digits contain hole(s). First we generate the patches (Figure 2b). Next we extract the holes (Figure 2c), and restore those patches contaning these holes (Figure 2d). Then, we identify the groups with more than one digits (Figure 2e), and restore those patches also (Figure 2f). Finally, we apply a logic AND to the previous two results, to obtain the final result (Figure 2g). If we need to extract the original object covered by the selected patch, we have to apply a logic AND to the original image, and the final result (Figure 2h).

Observe that without the patches we could not apply logic operation on the two subresults (Figure 2c,e).

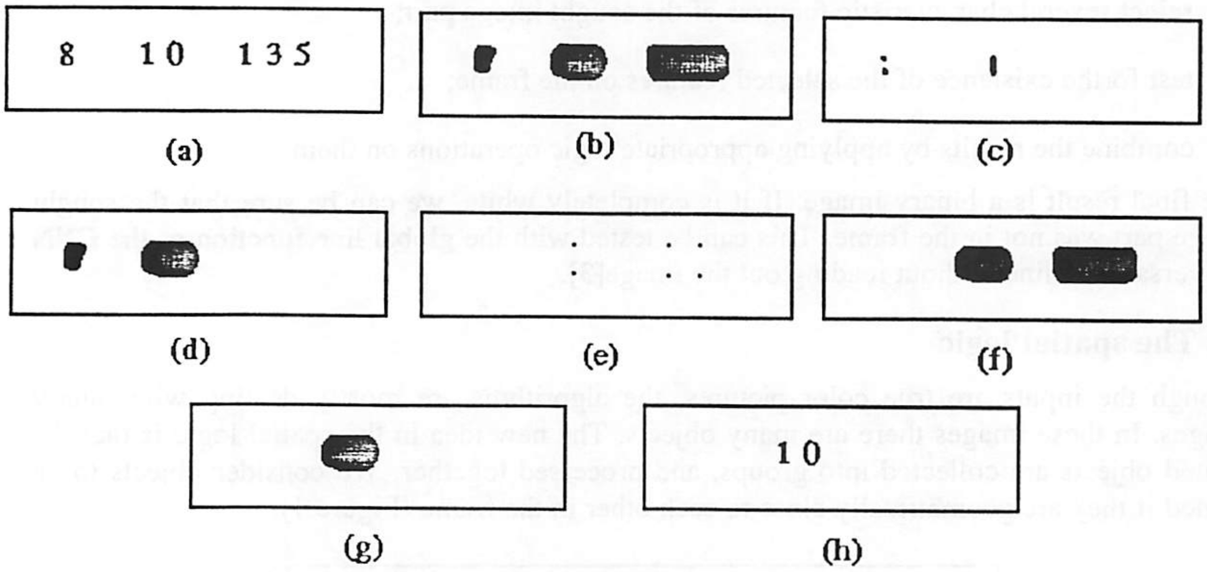


*Figure 2* *The phases of our algorithm. (a) original image, (b) the patches, (c) the holes, (d) the patches of the groups contain holes, (e) dots identifying the groups with more than one digits, (f) patches of the groups containing more than one digits, (g) logic* AND *applied to (d) and (f): the final result, (h) logic* AND *applied to (a) and (g): the detected group of objects having the specified features.*

## 4 THE MODULES OF THE ALGORITHMS

Next, we describe the main functional modules which will be used later. When describing the modules we will also give the approximate running times measured in $\tau$ time constants. $\tau$ varies between 10 and 300ns, depending on the chip. Finally, in Section 5, we will show how the steps can be combined to recognize different bank-notes.

### 4.1 Recalling

Almost all of the following modules use the recall operation. (Similar recall template was reported in [6].) Consider a black-and-white image containing arbitrary black objects against a white background. We want to classify the objects according to their features. Extracting a feature means that some pixels remain from all objects having the desired feature, and none from the others. Thus all images we need are identified, and our goal is to restore the whole object from these signatures. The original image (called base image) is loaded to the input, and the extracted features (the signatures) to the initial state of the CNN. By using the RECALL template the assigned objects will be restored. The template is the following:

$$A = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 4 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad I = 2.5 .$$

In Figure 3 we can see an example. Given a black-and-white image with two objects (Figure 3a) and the signature image (Figure 3b) which identifies one of the objects. After the transient settles down the identified object was restored (Figure 3c).

The running time of this template is proportional to the size of the object to be restored, each layer of pixels requiring $1\tau$. In the examples of Section 5 each recall operation needed approximately $15\tau$.

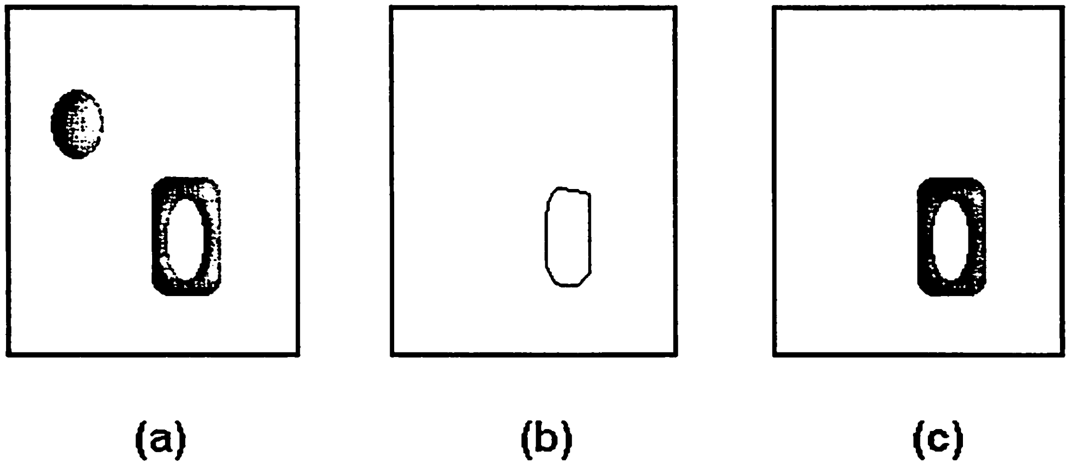


*Figure 3* *The effect of the* R*ECALL template. (a) the original image, (b) the signature, (c) the restored object.*

## 4.2 Creating patches

In this section we show how to create the patches we described in Section 3.2. The PATCHMAKER template is as follows:

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad I = 4.5 .$$

This is a diffusion type template which enlarges each object by growing a layer of pixels in all directions in every $\tau$. So the transient must be stopped after several $\tau$ time, otherwise it will drive all pixels to black. The exact number of $\tau$ can be calculated from the preliminary information we know/assume about the image, i.e. how far the components of a group are from each other. In our examples (see Section 5) it usually took approximately $10\tau$. The process can be seen in Figure 4 with $10\tau$.

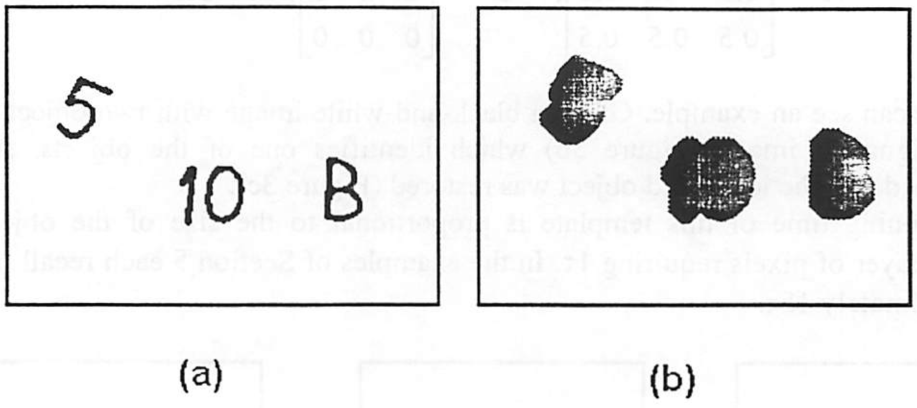


*Figure 4* *The effect of the PATCHMAKER template. (a) the objects, (b) the object groups (patches)*

## 4.3 Extracting objects with a given color shade

The first step of the bank-note recognition algorithm is color filtering. Suppose that we know in advance the color shade of some specific features, e.g. the color of the numbers on the bank-note. Filtering out all other colors already eliminates lot of objects.

Color images are represented by Red, Green, and Blue (RGB) monochromatic maps. Using CNN, we can handle a monochromatic map as any other grey-scale image. It means that in each map there are continues values between -1 and +1, representing the strength of each color component in the actual pixel. In this case a -1 means that the actual color component is missing at that pixel, while a +1 means that it is represented in full power.

The color shade to be extracted is defined by its range in the RGB representation. For example, if we are searching for orange colored objects we can say: a certain color is orange if its red component is between 0.75 and 0.85, the green is between -0.55 and -0.45, and the blue is between -0.95 and -0.85. (These final exact values should of course be tuned to the optoelectronic device we are using.) In the original image all color coordinates can take any value from the (-1,+1) interval, resulting in a color "cube" with a $2*2*2=8$ volume. After the above filtering, the volume of the color cube is reduced to $0.1*0.1*0.1=0.001$. This represents only the $\frac{1}{8000}$th of the original volume, therby greatly reducing the search space.

The color shade extraction algorithm consist of the following steps:

- extract the required interval from the red component;
- extract the required interval from the green component;
- extract the required interval from the blue component;

- apply logic AND to the results of the three previous steps.

The smaller the color cube, the more objects will be eliminated. But due to usage, the color of the patterns on on a used bank-note may fade slightly and some parts of it might fall outside of a narrow color cube, thus missing from the output of the AND operation. To overcome this problem and restore the whole object, we recall them from a reference image. The reference image is a black-and-white image. It must contain all the searched objects separated from the other objects and the background. In our example, we have generated it from one of the monochromatic maps by using the AVERAGE template [1].

Our algorithm for extracting the required interval from a color component contains four steps:

- extract pixels having a value greater than the lower limit of the prescribed interval, using the template

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad I = 2 * L_{imit} \;;$$

  (This template drives to black all the pixels having a value greater than *Limit* and drives to white the others.)

- extract pixels having a value greater than the upper limit of the prescribed interval, using the above template;

- apply logic XOR to the results of the above two steps;

- discard small (less than 4 pixels) objects using the SMALL KILLER template:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad I = 0 \,.$$

  (This diffusion type template drives to white all those black pixels which have more than four white direct neighbors, and reverse: drives to black all those white pixels which have more than four black direct neighbors.)

Extracting the color cube defined above requires approximately $60\tau$ time. The flow-chart of this algorithm is given in Figure 14.

## 4.4 Extracting objects with small holes

This function can be realized in three steps. The first is fill in the holes. Next, extract just the holes by applying a logic XOR operation to the filled image and the original image and finally, recall the appropriate patches associated with the holes.

For hole filling we use a two-step algorithm which is much faster than the original HOLE FILLER template [5]. In the first step we start to fill in the local concavities with the HOLLOW template:

$$A = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 2 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad I = 3.5\,.$$

This diffusion type template drives to black all those white pixels which have at least four black direct neighbors.
We call concave places those white locations which are surrounded by black pixels from at least four of the eight orientations, but still connected to the white background.

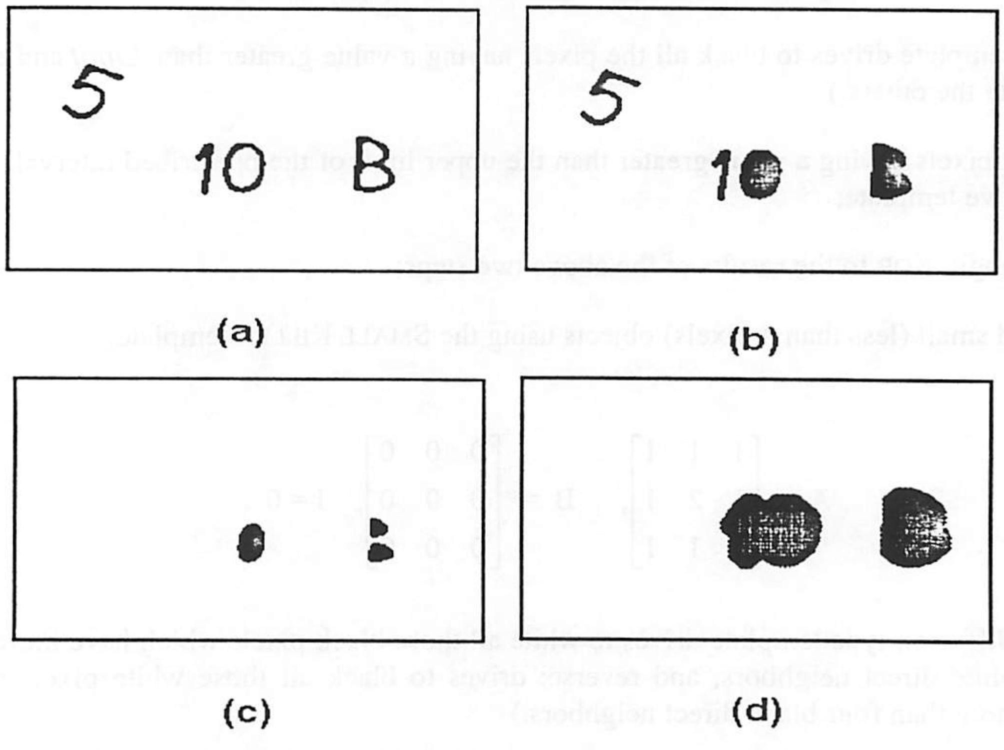


*Figure 5* *Objects with small holes. (a) original image, (b) filled holes, (c) result of XOR, (d) recalled remaining patches.*

The transient must be stopped after 10 - 15τ depending on the size of the largest holes to be filled in. During this time, these holes are definitely filled in. In the second step, we have to get rid of the black pixels that are located not in a hole but in other concavities. By putting the original image on the input, and the filled one on the initial state, the following template discards those newly appeared connected groups of pixels which have at least one white neighbor (this means: not surrounded by black pixels, i.e. they are concavities not holes):

$$A = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 4 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad I = -2.5 \,.$$

Though both of these templates are of a diffusion type, the total transient time is much shorter than that using the traditional HOLE FILLER template, because the filling time depends only on the diameter of the identified holes but not on the size of the frame. The speed of this two-step algorithm is independent of the size of the CNN cell array, it depends only on the size of the holes. In our examples this step required approximately 35τ time. Figure 5 gives an example how this module works.

## 4.5 Extracting concavities of objects

Extracting concavities of objects is a three step algorithm. In the first step, we drive black the pixels located at concave places, with the help of the HOLLOW template (Figure 6a,b).

The second step is applying logic XOR to extract the newly generated black parts of the objects. As it can be seen in Figure 6c, some new pixels appear not just at the real concave places, but also along the convex edges. We can discard the undesired black pixels by using the SMALL KILLER template, and get the final result (Figure 6d). The whole process, shown in Figure 6 requires 25τ time.

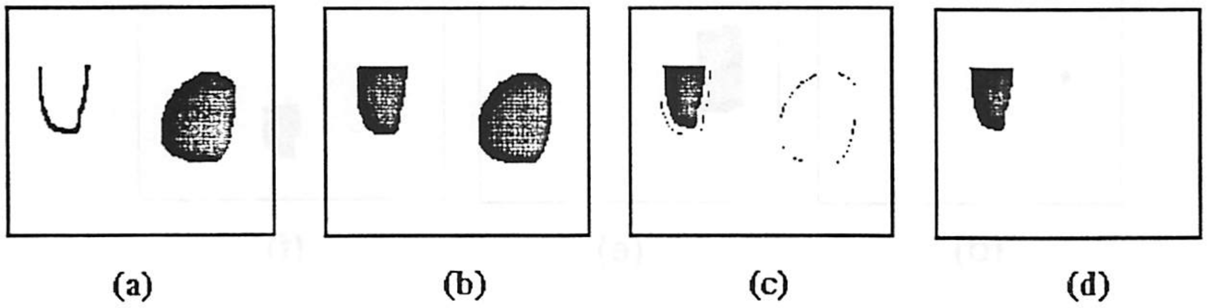


*Figure 6* *Extracting concave locations. (a) original image, (b) concavities, (c) logic XOR of (a) and (b), (d) eliminating small dots on the pheriphery.*

## 4.6 Size classification

Our size measuring is based on peeling. In each step we erase all boundary pixels of the objects. We define the size (measure) of an object as an integer number equal to how many such peeling steps can be made before the last pixel disappears.

For example, we will extract objects larger than size $n$ but smaller than size $m$ with the following analogic algorithm:

- Erase $n$ layers from all objects by using the PEELING template (see below) $n$ times).
- Recall those images that still have some remaining pixels (RECALL template, Section 4.1). This image will contain the desired objects as well as than which are too large.
- Do $m$-$n$ more peelings on the previously peeled image.
- Recall again. In this image only the objects which are too large will remain.
- Apply logic XOR to the two recalled images. The difference between them is the set of the desired objects.

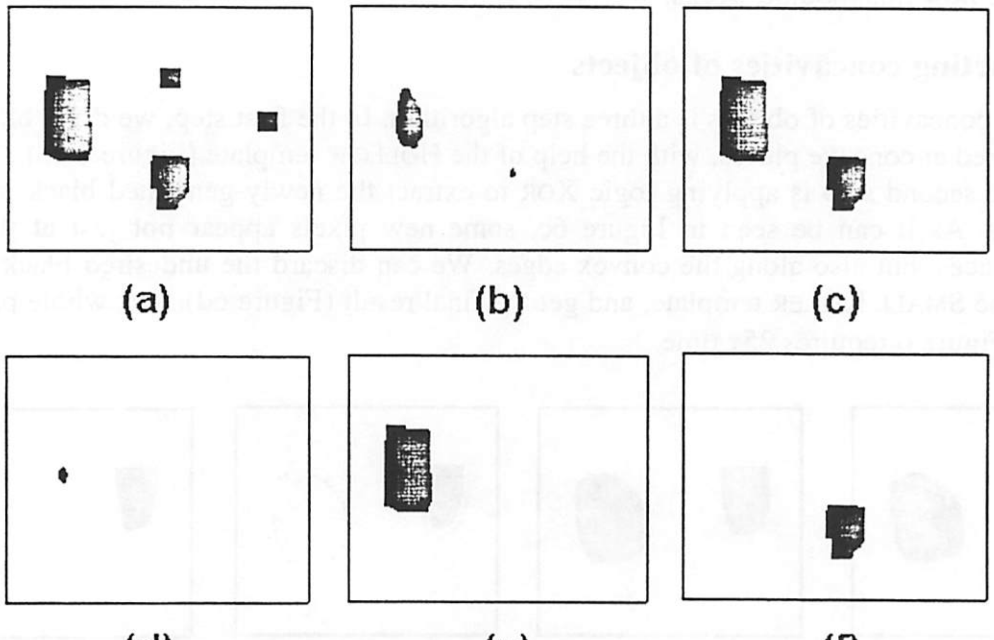


*Figure 7* *The size classification algorithm. (a) original image, (b) peeling off n layers, (c) recalling the result, (d) peel the result of (b) for m-n more steps, (e) recall the result, (e) result: logic XOR of (c) and (e).*

The PEELING template is as follows:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad I = -4.5\ .$$

This template drives to white all those black pixels which have at least one white direct neighbor.
The algorithm is demonstrated in Figure 7. The time demands of this procedure depends on the size of the objects. In our example it took 50τ time.

## 4.7 Skeletonization

Some modules of our algorithm are based on the skeleton of the objects. By the skeleton of the objects we mean a figure which preserve the connectivity and the topology of the input objects having containing only-one pixel wide lines except for the junctions. The skeletonization algorithm is detailed elsewhere [4].

## 4.8 Extracting objects containing more than one hole

This algorithm starts from the skeletonized objects. The skeleton of an object with more than one hole must contain junctions. With the following JUNCTION template, we can extract these points:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 2 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}, \quad I = -1.5\ .$$

This template drives to white all those black pixels which have at less than three black direct neighbors.
After extracting the junctions we can easily restore the extracted objects with the RECALL template.

Unfortunately, the skeleton of an object can contain junctions without containing more than one hole (e.g., the skeleton of the letter P). So before extracting the junctions, we have to get rid of the branches not forming closed loops. This can be done in two steps. The following template erases the pixels forming unclosed lines except the nearest ones to the junction:

$$A = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 2 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad I = 1.5\ .$$

This diffusion type template drives to white all those black pixels which have less than two black direct neighbors.

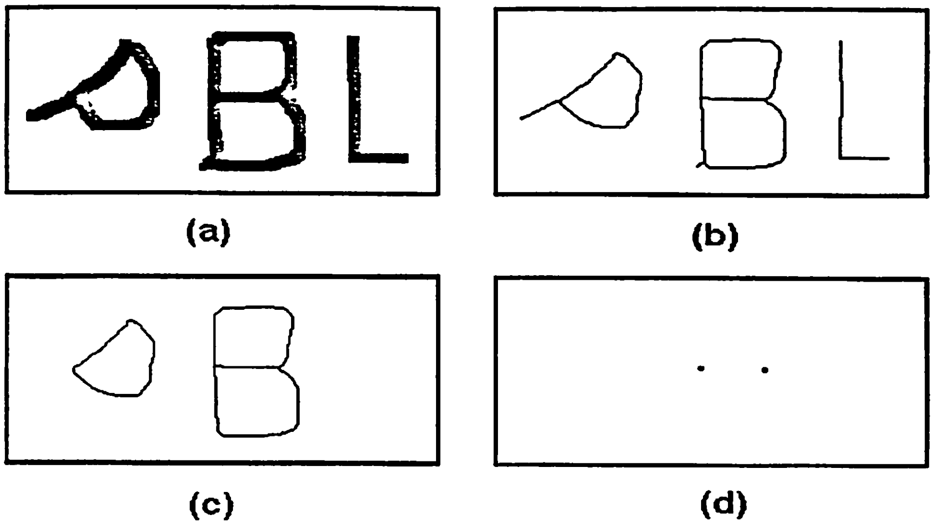


*Figure 8* *Identifying objects containing more than one hole. (a) original image, (b) skeletonized image, (c) eliminating unclosed lines, (d) extracted junctions.*

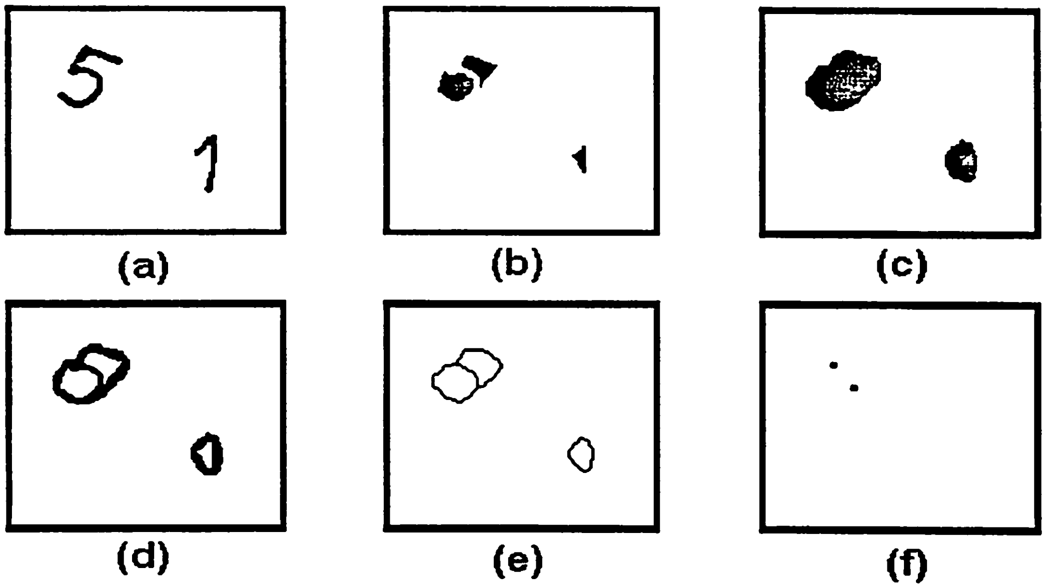


*Figure 9* *Identifying objects containing more than one concavities. (a) original image, (b) concave locations, (c) blowing up concave locations until they melt together, (d) subtract (b) from (c), (e) skeletonization, (f) extracted junctions.*

After eliminating almost the entire unclosed line, the remaining pixels can be erased by a single loop of the skeletonization algorithm. The steps of the algorithm are shown on Figure

8. The running time of the procedure depends on the image, in our examples it required approximately 100τ time.

### 4.9 Extracting objects containing more than one concavity

This algorithm contains four major steps. First, the concave locations are extracted (see Section 4.5), then they are blown up until the ones belonging to the same object melt together, using the PATCHMAKER template. The third step is to subtract the original concavities from the blown up ones by applying the logic XOR, to obtain objects with one or more holes. The last step extracts the objects containing more than one hole, as described in Section 4.7. The steps are shown in Figure 9. This algorithm needs approximately 135τ time.

## 5 EXAMPLES

Here we will demonstrate how the algorithms described above work on 3 different examples. In all examples, we only indicate the steps which result in elimination of some objects. For example, in the case of the US dollar bill recognition, we eliminated the too small objects, but as there are no too large objects, we did not include the step testing this feature in the flow-chart.

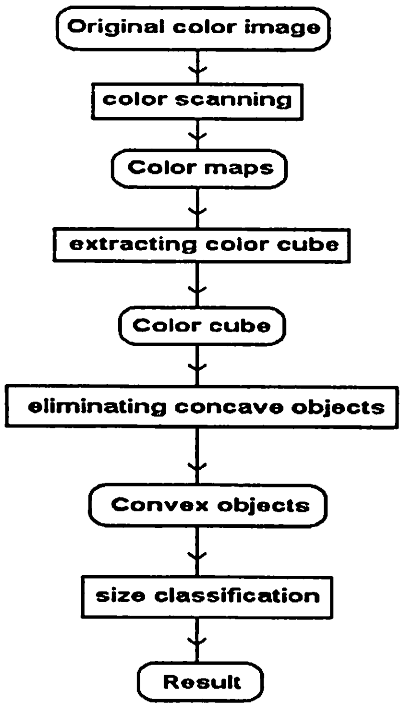
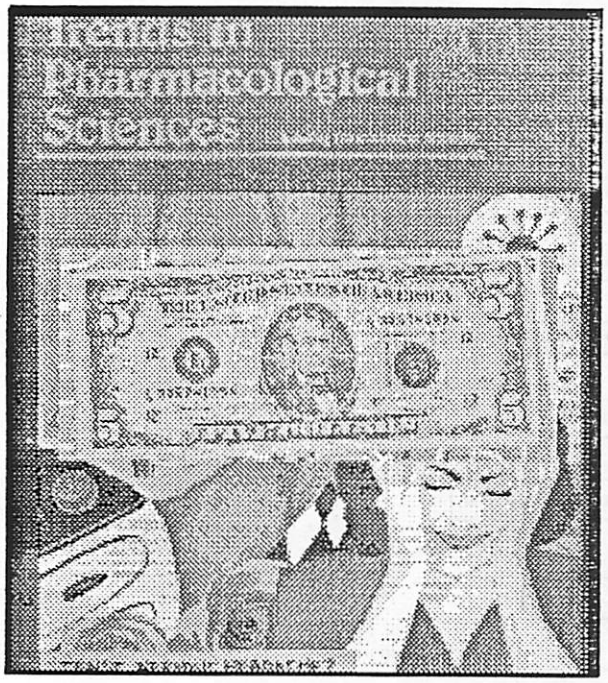


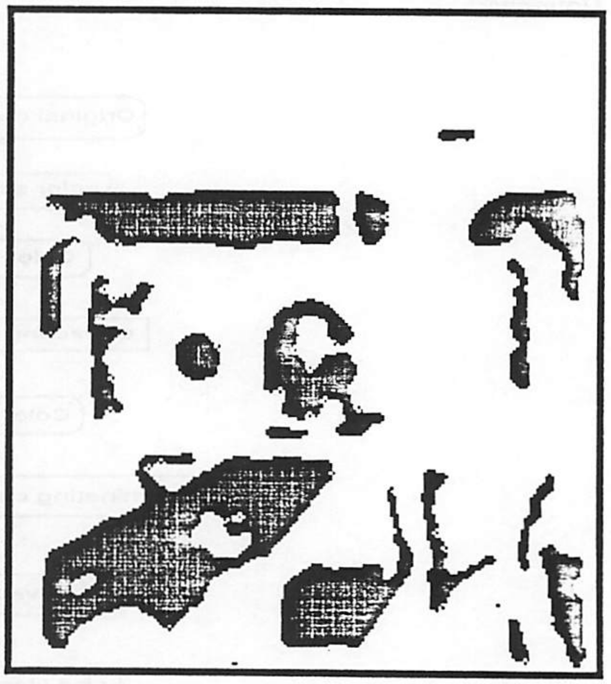
*Figure 10* *The flow-chart of the US dollar bill recognition algorithm. The entire flow-charts can be inplemented in less than 60 μs.*

## 5.1 Recognizing US dollar bills

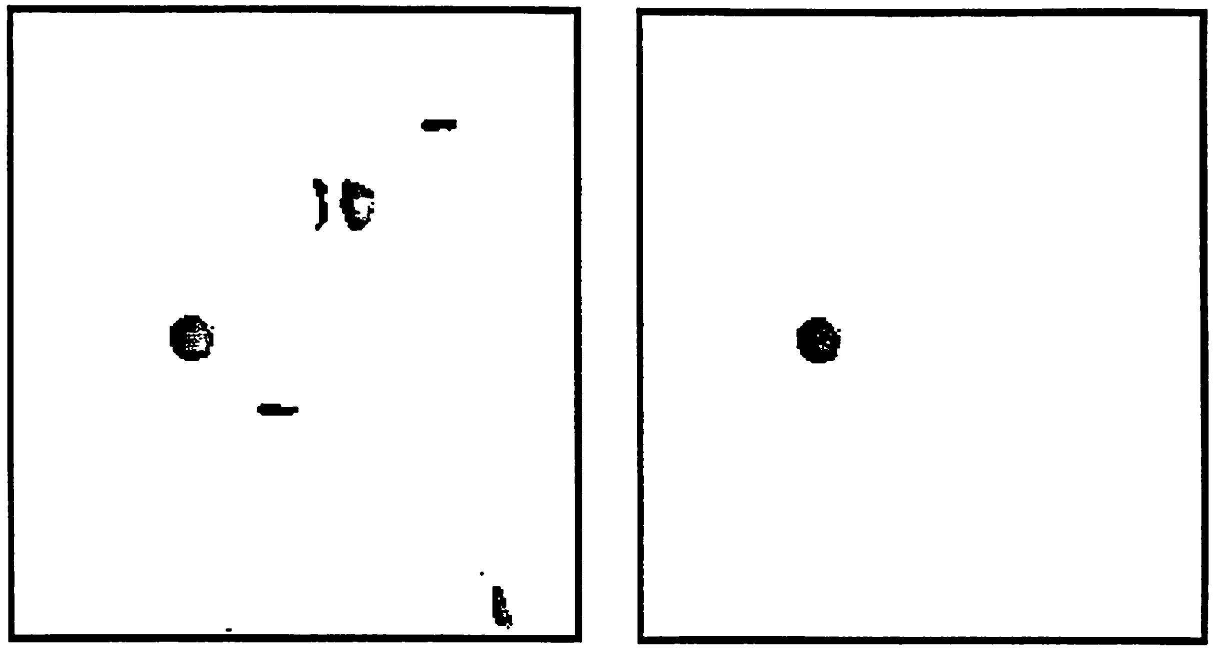
The US dollar bills have the rare but very useful property of having simple characteristic common patterns. Such characteristic patterns are the black and green circles on the left and right side of the bank-note, respectively. These circles have the same size on all denominations. A single algorithm capable of detecting them will work on all US dollar bills. The algorithm contains two phases: first we extract the green circles having the correct size. Next, we run the same procedure for the black circle. If there were no US dollar bill in the frame, in most cases, the algorithms stops at this point. If it found both type of circles the image is read out, and the digital unit of the CNN chip environment measures their distance. If a green and a black circle are extracted and have the appropriate distance from each other, we decide for a bank-note and prevent copying. The flow-chart of the algorithm is shown on Figure 10, while the different steps are illustrated in Figure 11(a)-(d), showing the extraction of the black circle. The same procedure is used for the green circle, followed by measuring the distance between pairs of black and green circles.
The whole process takes approximately 200τ time. This is between 60μs and 2μs, depending on the appropriate CNN chip realization.



(a)



(b)



(c) (d)

*Figure 11* *US dollar recognition: (a) Input image, (b) Color cube, (c) Convex objects, (d) Size classification*

## 5.2 Recognizing Canadian $50 notes

In this section we will demonstrate how Canadian $50 notes can be recognized. In this case we extract the number 50. If anything remains on the screen after the final step, we prevent copying. This idea, that is, extracting numbers works well for several currencies. We have tested it on many other bank-notes including Italian, Dutch, Hungarian notes, etc.

To be able to show several steps, we present the algorithm with two different backgrounds. The first being the color cover page of a journal, proving that with "normal" images the algorithm terminates after just a few steps. In the second example we use an artificial background containing objects of similar color and shape as the digits, but even that could not fool our algorithm.

The following examples should be considered as the first steps of bank-note detecting algorithms. In a real application, we would test much more features before deciding to stop the copying process. We can afford to do this because the CNN chip is so fast we have much unused time to spare.

### 5.2.1 Normal background image

Here we show the recognition of the Canadian $50 note against a natural background image. This demonstrates that in most cases only a very few steps are sufficient to eliminate everything, so we have time to compare the input with many different bank-notes. The flow-chart of the algorithm is shown in Figure 12, while the result of each step is illustrated in Figure 13(a)-(c). These steps need about 150τ time. This is between 45μs and 1.5μs, depending on the appropriate CNN chip realization. In this case we have detected just the correct size of objects with the prescribed color shade.

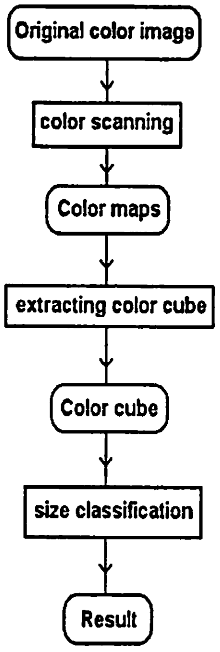
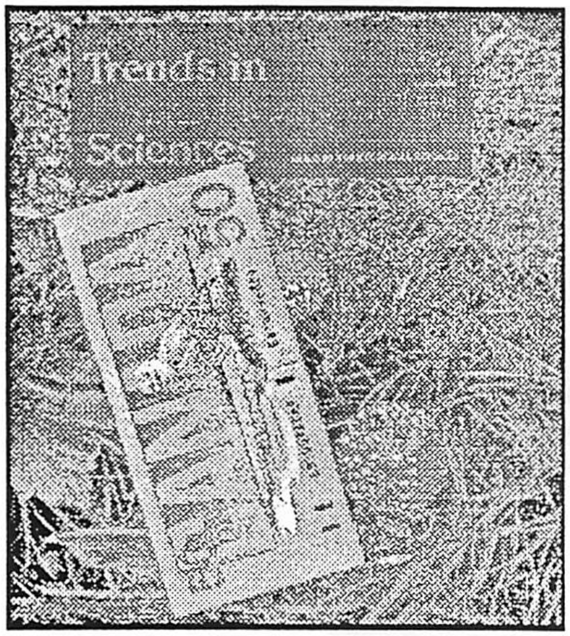


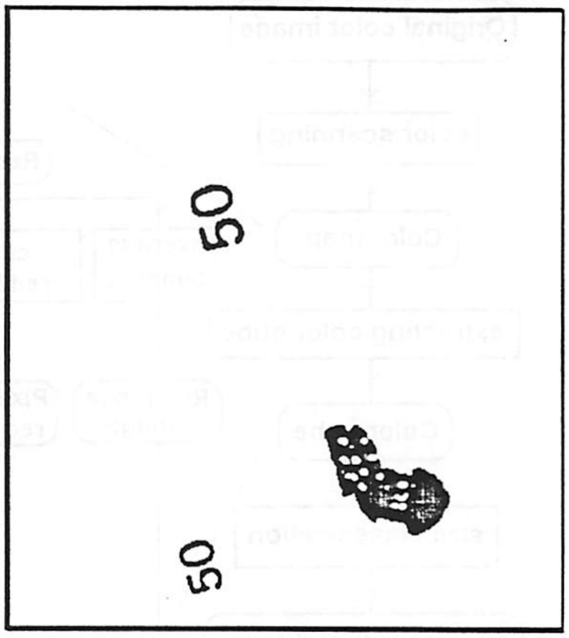
*Figure 12* *The flow-chart of the Canadian $50 recognition algorithm with normal background*

### 5.2.2 Artificial background

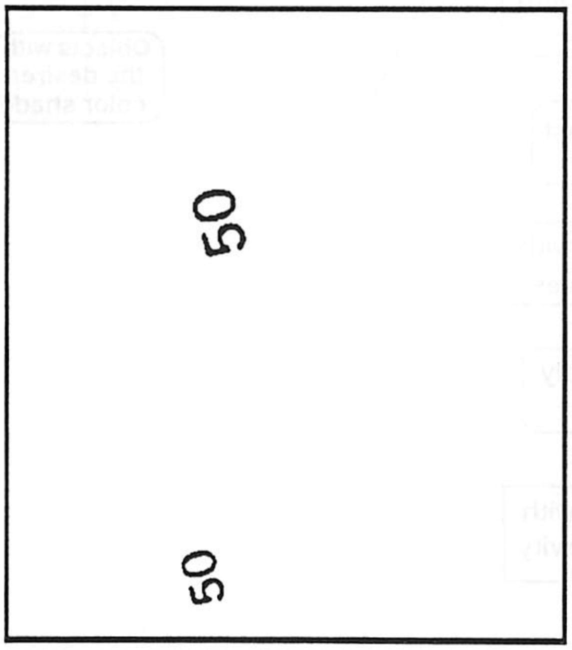
In this example we created an artificial background having objects similar both in color and form to the numbers on the bank-note. The flow-chart of the algorithm is given in Figure 14. To illustrate the complexity of the algorithm, a single step is given in detail as well as on the upper right hand side. This is an analogic subroutine implementing the given step. Then we show the results of steps of the procedures (Figure 15(a)-(e)). This algorithm needs approximately 500τ time running on a CNN Universal Machine. This is between 150μs and 5μs, depending on the appropriate CNN chip realization.



(a)



(b)



(c)

*Figure 13* *Canadian $50 recognition with normal background: (a) Input image, (b) Color cube, (c) Final result: Size classification. (Only these objects having the prescribed size are extracted.)*

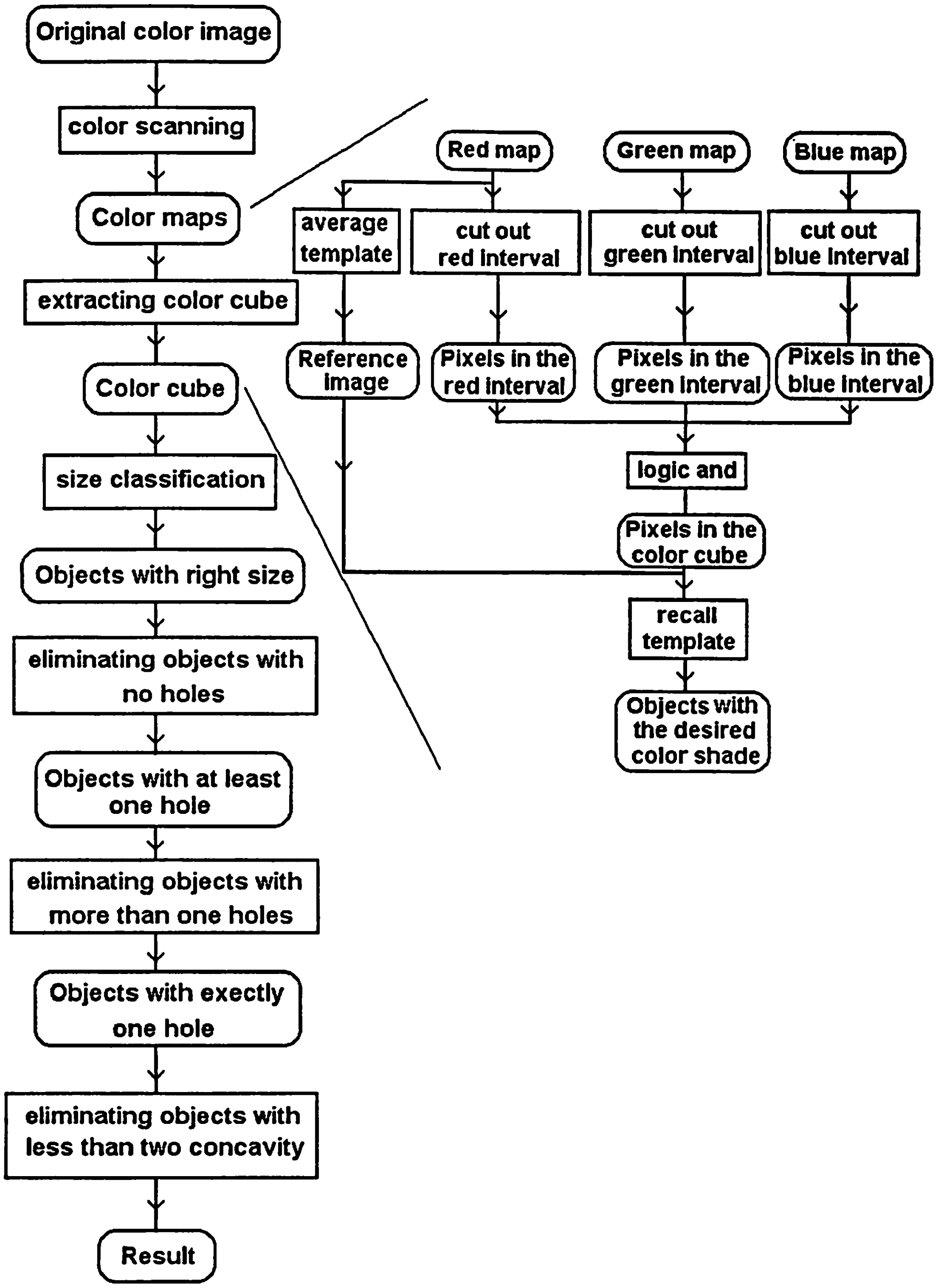
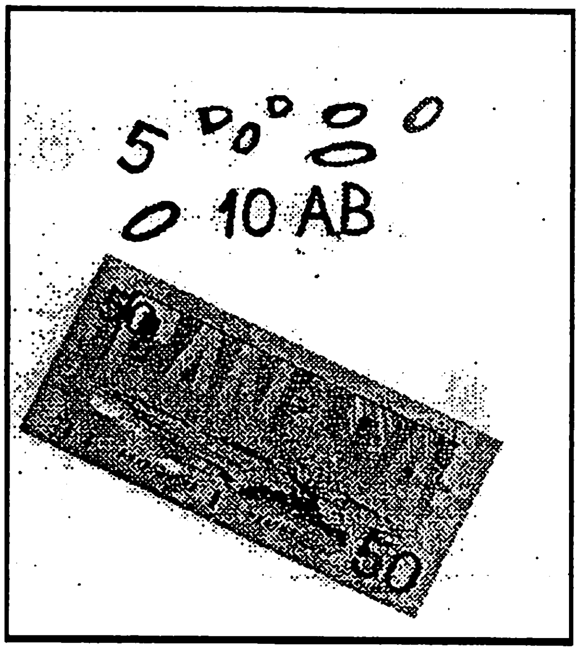
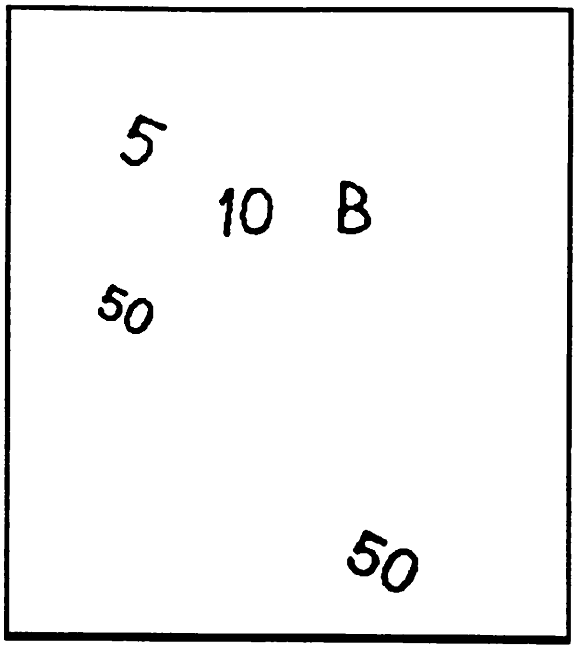

Original color image
color scanning
Color maps
extracting color cube
Color cube
size classification
Objects with right size
eliminating objects with no holes
Objects with at least one hole
eliminating objects with more than one holes
Objects with exectly one hole
eliminating objects with less than two concavity
Result
Red map
Green map
Blue map
average template
cut out red interval
cut out green interval
cut out blue interval
Reference image
Pixels in the red interval
Pixels in the green interval
Pixels in the blue interval
logic and
Pixels in the color cube
recall template
Objects with the desired color shade

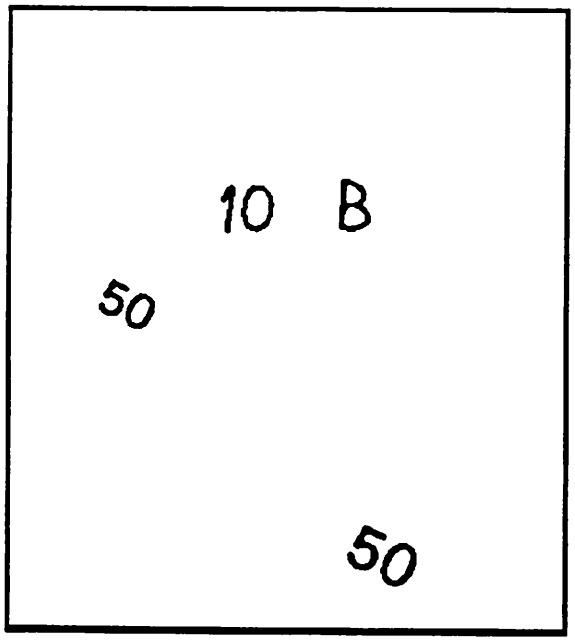
*Figure 14 The flow-chart of the Canadian $50 recognition algorithm with artificial background. One step (extracting color cube) is given in detail.*



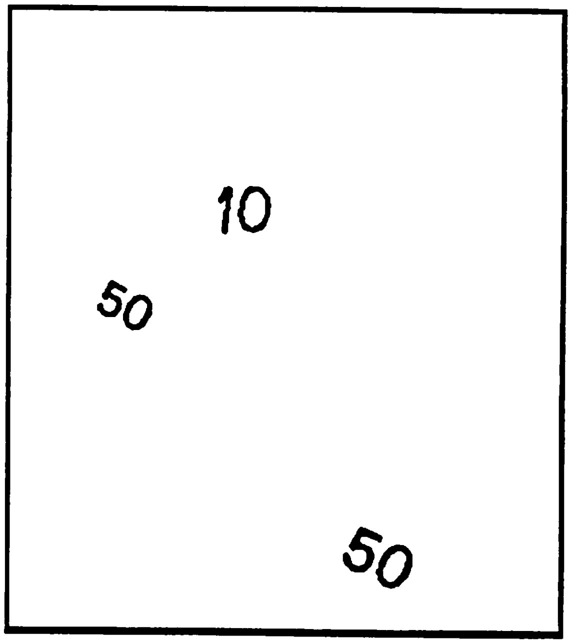
(a)



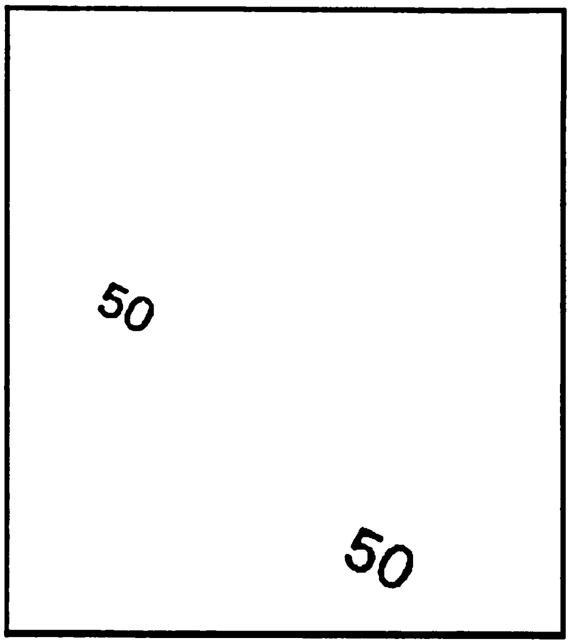
(b)



(c)



(d)



(e)

*Figure 15* *Canadian $50 recognition with artificial background: (a) Input image, (b) Color cube, (c) Objects with holes, (d) Objects with exactly one hole, (e) Final result: Objects with at least two concavities*

## 6 ACKNOWLEDGEMENT

The support of the grant INT 90-01336 of the National Science Foundation incooperation with the Hungarian Academy of Sciences, is acknowledged. Thanks are due to Péter Venetianer for useful discussions and helps.

## REFERENCES

[1] L.O.Chua and L.Yang, "Cellular neural networks: Theory and Applications", IEEE Transactions on Circuits and Systems, Vol.35, pp.1257-1290, 1988.
[2] L.O.Chua, T.Roska, "CNN Paradigm" IEEE. CAS-I 1993 (March)
[3] T.Roska, L.O.Chua, "CNN Universal Machine" IEEE. CAS-II, 1993 (March)
[4] P.Venetianer, F.Werblin, T.Roska, and L.O.Chua, "Analogic CNN algorithms for some image compression, decompression, and restoration tasks", Memo UCB/ERI, Electronics Research Laboratory, University of California at Berkeley, 1994
[5] T.Matsumoto, T.Yokohama, H.Suzuki, R.Furukawa, E.Oshimoto, T.Shimmi, Y.Matsushita, T.Seo, L.O.Chua, "Several Image Processing Example by CNN" IEEE International Workshop on Cellular Neural Networks and their Application, Proceedings.
[6] K.Slot, T.Roska, L.O.Chua,"Optically Realized Feedforward-Only Cellular Neural Networks" AEÜ, Vol.46, (1992) No.3