Learning Binary Shapes as Compression and its Cellular Implementation

Silvano Dal-Zilio Thierry M. Bernard*

Perception System Laboratory, ETCA/CREA/SP
16 bis, Av. Prieur de la Côte-d'Or
F-94114 Arcueil Cedex, FRANCE
vg@etca.fr

Abstract

We present a methodology to learn how to recognize binary shapes, based on the principle that recognition may be understood as a process of information compression.

Our approach, directed towards adaptive target tracking applications, is intended to be well suited to fine-grained parallel architectures, such as cellular automata machines that can be integrated in artificial retinas.

This methodology, fruitfully explained within the frame of mathematical morphology, is then particularized in the perspective of its actual implementation.

1 Introduction

We have shown it feasible and valuable to gather on the same chip a photosensitive array with a programmable array processor, making up what we call a *programmable artificial retina* [3]. Present technological levels allow the integration of a 128×128 grid with each site containing a photodiode, a non-standard analog-to-binary converter, a memory of 8 bits, a logical unit, and connections to closest neighbors. Global SIMD control signals are distributed all over the chip to each pixel.

Several research teams tend to confine to image preprocessing only the contribution of the "retinal concept". Unlikewise, it is our belief that *programmable artificial retinas can and must "handle" shapes in order to provide the vision system they belong to with meaningful enough information*, thus suppressing any communication bottleneck. Instead of outputing full or partial images, particular information reduction mechanisms are to be setup, as detailed hereunder.

Handling shapes in general relates to the field of *pattern recognition.* In the case of retinas, a first stage to reach is exemplified by target tracking applications. This calls for algorithms that are *primitive*, since little a priori information is needed, but *reactive*, as fast adaptation to target appearance evolution – which can be expected thanks to the retinal concept – is desirable.

*Presented at ACCV'95, Singapore, 5-8 december 1995

Target tracking applications imply shape resemblance detection, which usually relies on a total description of the reference shape (by opposition to using features). From this point of view, a key idea exploited in this paper is that *learning and recognition may be understood as processes of information compression* [9]. Following the minimum length description principle [1], the best solution to represent a set of data is the one that minimizes its encoding (with regard to some criteria).

Here, binary images are considered, along with the shapes that they represent. Their encoding is performed thanks to morphological operators. Minimization will be partial as trade-offs must be found with running the algorithms on massively parallel SIMD cellular architectures (especially those that can be integrated on a retina chip).

The paper is organized as follows. In section 2, Mathematical Morphology (MM) [8] is used to present our approach and to formulate some general purpose algorithms for shape learning and recognition. As far as notations are concerned, set difference is denoted by $\setminus$, erosion by $\ominus$, dilation by $\oplus$, and opening by $\odot$. Section 3 introduces an alternative framework for the expression of our algorithms, namely TItBITs. This framework is more practical as it is directly derived from a retinal model of data-parallel computation. Then section 4 presents simulation results about actually running the algorithms on a programmable artificial retina.

2 Learning by compression

Compression is performed by extracting redundancy from a shape. Much like steepest descent minimization procedures, a possible strategy is to look at once for the best matching – *i.e.* the most redundant – pattern $\mathcal{P}$ in shape I. Of course, the search space can be very large. For the moment, let us suppose that $\mathcal{P}$ has been found. One gets a compressed image, the "factor" $F = I \ominus \mathcal{P}$, and a remainder image, $R = I \setminus (I \odot \mathcal{P})$. Exceptionally, R may be empty, corresponding to the ideal situation of pure shape decomposition. But this is hardly ever possi-

```
/* compression of shape $I$ */
tree compress(image $I$) {
  if ($I$ features low redundancy) {
    return $I$; /* $I$ is considered as a pattern */
  } else {
    $\mathcal{P}$ = best matching pattern in $I$;
    tree.pattern = $\mathcal{P}$;
    tree.factor = compress($I \ominus \mathcal{P}$);
    tree.remainder = compress($I \setminus (I \odot \mathcal{P})$);
    return tree;
  }
}

/* recognition of a shape coded by $T$ in $J$ */
image find(tree $T$, image $J$) {
  if ($T$ is a leaf) {
    return ($J \ominus T$.pattern);
  } else {
    return ((find($T$.factor,$J$) $\ominus$ $T$.pattern)
                            $\cap$ find($T$.remainder, $J$));
  }
}
```

Table 1: *Recursive compression and recognition algorithms. The* tree*-structure is expressed using a C-style notation. In order to* **find** *shape I in image J, a recognition tree is used, simply obtained by a depth-first search on the result of* **compress***. This is illustrated in Fig. 2, where tree (b) is turned into tree (c).*

ble. In the general case, the couple $(F, \mathcal{P})$ can be viewed as a coarse description of I, whereas the non-empty remainder R, part of I that cannot be "recovered" from F, keeps a record of some details.

Now the process can be repeated and recursively applied to both F and R. This recursive procedure, presented as **compress** in Tab. 1, yields a pattern-tagged binary tree where every subtree is the compression of an initial shape subpart. Fig. 2 shows (a) the procedure and (b) the output tree for an elementary example. This potentially compact representation of shape I naturally features a hierarchy among the leaves of the tree. Indeed, the contribution of a given leaf to the initial shape is its *cumulated dilation* by the different patterns that seperate it from the root of the tree.

But the main point is that a shape recognition algorithm, presented as procedure **find** in Tab. 1, can be directly derived from this tree, thus *turning compression into a learning process.* The two procedures are bound by the following relationship:

Theorem 1

$$\mathbf{find}(\mathbf{compress}(I), J) \quad = \quad J \ominus I$$

A proof of this result can be obtained through a simple recursion on the depth of the compression tree:

- if **compress**(I) is reduced to a leaf (due to the low redundancy of I), then procedure **find** returns $J \ominus I$ by definition,
- else, I has been compressed into $(F_{[I]} \oplus P_{[I]}) \cup R_{[I]}$. Following our recursion hypothesis:

$$\begin{array}{l} \mathbf{find}(\mathbf{compress}(F_{[I]}), J) = (J \ominus F_{[I]}) \\ \mathbf{find}(\mathbf{compress}(R_{[I]}), J) = (J \ominus R_{[I]}) \end{array}$$

Then, by definition of **find**,

$$\begin{array}{rcl} \mathbf{find}(\mathbf{compress}(I), J) & = & \left[(J \ominus F_{[I]}) \ \ominus \ P_{[I]}\right] \cap \ (J \ominus R_{[I]}) \\ & = & J \ \ominus \ \left[(F_{[I]} \oplus P_{[I]}) \cup R_{[I]}\right] \\ & = & J \ \ominus \ I \end{array}$$

Inspired by a compression approach, **compress** and **find** are generic procedures allowing to compute erosions by complex structuring elements that are only known at run time. They can be very fast if patterns and remainders that are generated have a small size or a small integral. Indeed, the required computations will be efficiently performed on massively parallel SIMD cellular architectures. The importance of these procedures comes from the fundamental role played by erosion as a tool for shape recognition. The rest of the paper will be concerned with actually implementing them.

3 From MM to TItBITs

In order to make the above algorithms more practical, we prefer to express them using the family of **T**ranslation **I**nvarian**t** **B**inary **I**mage **T**ransforms (TItBITs), as defined in previous work of the team [4]. TItBITs can be considered as an operational version of Mathematical Morphology (MM). As a hint for easier comprehension, MM is to rational languages what TItBITs are to finite state automata. TItBITs are grounded on a data-parallel model of computation, which is actually the one supported by our home-made retina [3]. In this model, a set of binary planes interact with each other. Planes can be shifted, complemented or set to a copy of another. Parallel interaction between planes are also possible: given two planes, one can receive the conjunction or disjunction of both. Considering that each plane supports a binary image – which is modeled as a subset of $\mathbf{Z}^2$, $\mathbf{Z}$ the set of naturals – we define a TItBIT counterpart for each retinal operation. Shifting will corresponds to the *translation* of an image I by a vector $v = (v_i, v_j)$, denoted $I[v]$ ($\{(i,j) \in \mathbf{Z}^2,\ (i-v_i, j-v_j) \in I\}$). Likewise, we define *complementation* ($\overline{I} = Z^2 \setminus I$), *conjunction* ($I.J = I \cap J$) and *disjunction* ($I + J = I \cup J$).

Similarities between TItBITs and boolean functions are obvious. The retina architecture can be viewed as a 2-D cellular automaton, and a TItBIT as the boolean formula computed in each place. But, with the help of a graphical representation (see Fig. 1), a deeper similarity appears between TItBITs and images. Indeed points that remains after applying a TItBIT are the "origin" of particular patterns included in the initial image, and totally described by the monomials in the TItBIT "normal disjunctive form". Unlike MM, "black and white" points are symmetrically treated here. Moreover, the TItBIT

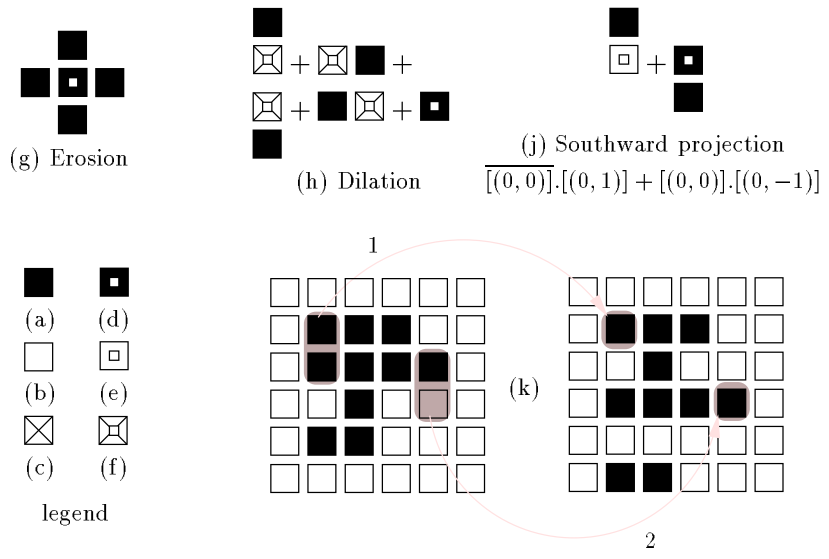

Figure 1: *TItBITs can be simply presented as operators that recognize a certain number (disjunction) of binary masks on an input image. Each mask is made of tiles (conjunction) that identify a '1' (a), a '0' (b) or do not care (c). The binary result of the recognition appears on the output image at the origin of the TItBIT, indicated by a* ▫*-sign, as shown in (d), (e) and (f). These notations are illustrated with the elementary erosion (g) and dilation (h). Then (j) displays an original southward projection TItBIT that lets the image crumble down in (k). Of course, each mask corresponds to a boolean monomial and the arrows show the recognition for each of the two.*

retina model provides a "time and space complexity" measure, related to the size and number of TItBITs, and the number of planes used in a computation.

```
tree compress(image $I$) {
   if ($I$ features low redundancy) {
      return $I$; /* $I$ is considered as a TItBIT,
                  that recognizes mask $I$ */
   } else {
      choose $(i,j)$ such that
                  $|I[(0,0)].I[(i,j)]|$ is maximal
      tree.pattern = $[(0,0)].[(i,j)]$; /* = $\mathcal{P}$ */
      tree.factor = compress($\mathcal{P}(I)$);
      tree.remainder =
           compress ($I \setminus ([(0,0)] + [(-i,-j)])(\mathcal{P}(I))$);
      return tree;
   }
}
```

Table 2: *Using TItBITs to express a possible version of the compression algorithm (an example is given Fig. 2).*

Tab. 2 shows a possible version of the **compress** algorithm formulated within the TItBIT framework. For the sake of simplicity, a particular class of patterns is used, formed by the conjunction of two translations: $[(i_1,j_1)].[(i_2,j_2)]$. Thanks to shift invariance, $[(0,0)].[(i,j)]$ patterns form an equivalent subclass. This choice is dictated by the combinatorial explosion of an exhaustive search through the TItBIT space (isomorphic to the boolean functions space). It certainly does not lead to the best compression rates. However every shape can be compressed using this family.

But let us come back to the central issue of this paper. The motivation for designing algorithms like **compress** and **find** is to allow programmable artificial retinas to manipulate shapes, such that they can output really meaningful information. Are these algorithms suitable ? The version of **compress** presented in Tab. 2 is mostly composed of TItBIT computations, that can be easily performed. However, there are other necessary facilities regarding information communication:

- Checking whether $|I[(0,0)].I[(i,j)]|$ is maximal or not requires comparisons between correlation measures that must be computed. Likewise, an integral measure is useful for evaluating the redundancy of I. Fortunately, the power supply of our retina can be opportunistically exploited [2] as an analog bus or as an analog adder connected to each pixel, to compute *global measures* on the image. Operating the chip in some peculiar way, current consumption measurements can yield the integral or correlation values that are needed.
- The patterns that form the compression tree produced by **compress** are going to be used by **find**. In particular, all the remainders featuring low redundancy (cf bottom of Fig. 3) are images that hopefully contain a few dots only, and that have to be output to be used as TItBITs. Instead of the

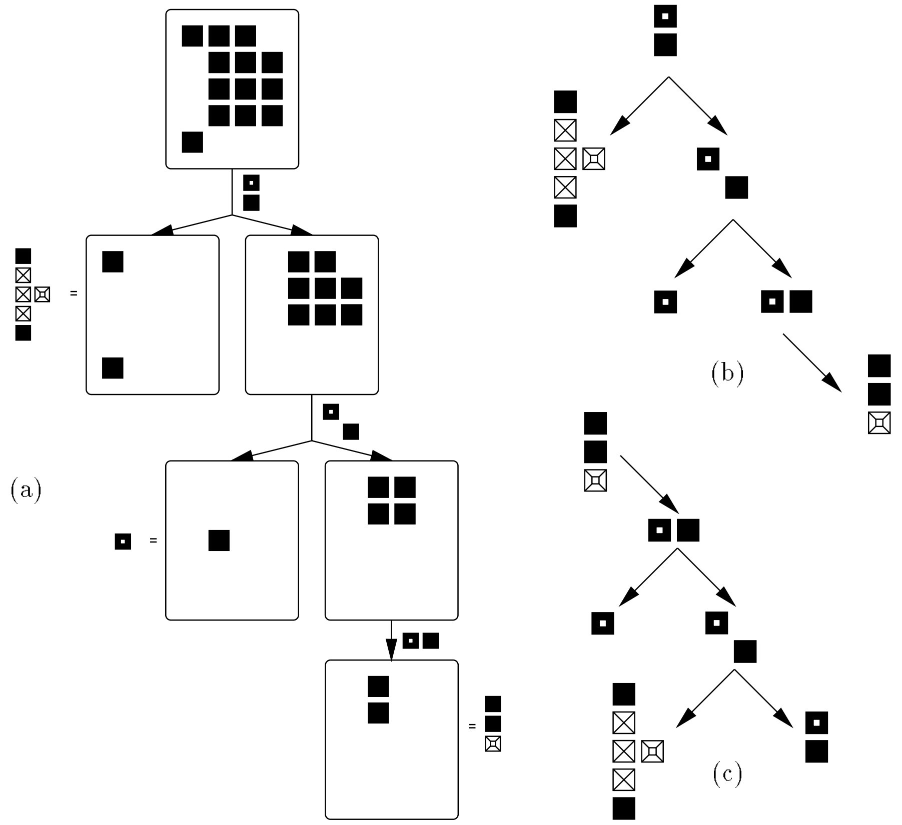

Figure 2: *The compression process corresponding to the algorithm in Tab. 2 is illustrated in (a). The root contains the image to be compressed. Each node gives birth to a compressed image (right), called the factor, and to a remainder (left). Vertices are tagged with the TItBIT that performs the erosion of the shape by the best matching pattern. At the leaves, images can be themselves handled as TItBITs. Then, on the right side, only the compression tree (b) is kept, and a depth first search on it yields the recognition tree (c).*

whole image, only the adresses of these dots should be output. An associative communication protocol is needed. It turns out that the above global measures allow to *emulate* such a protocol with an acceptable efficiency. Yet, it is also possible to provide dedicated hardware for this functionality [6].

Another important point concerns the number of binary planes (TItBIT model) that are necessary to run the algorithms. Procedure **compress** only needs 4 of them for an efficient implementation. Procedure **find** may require the storage of several intermediate images due to the depth of the remainder branches in the recognition tree (a remainder that was still redundant has generated another remainder). If d is the maximum of all these depths, then $d+3$ binary planes have to be used. In the case of a linear tree, like that shown on Fig. 2, $d = 1$. The version of **compress** shown in Tab. 2 does not always produce linear tree. However, redundancy tend to disappear very quickly from successive remainders, so d will generally remain small. Finally, these algorithms do fit the memory shortage of programmable artificial retinas.

4 Simulations

As mentionned before, there are trade-offs to find between an intrinsic compression strategy and its actual exploitation for learning and recognizing shapes at run time with a programmable retina. For the simulation result presented in Fig. 3, the compression strategy is reduced to one of its simplest form. Indeed, patterns that used to be chosen in a certain search space are now fixed (known at compilation time). Plain shapes like silhouettes can be efficiently described as the union of a reduced set of disks of variable centers and radius. Refering to Tab. 2, this can be approximately obtained by making (i, j) rotate among the 8 closest neighbors of the origin $(0, 0)$: disks actually become octogons. Using this techique, remainders are generally made of a few sparse dots, sometimes gathered into curve pieces. When segments (straight pieces) are encountered (cf top center of Fig. 3), they can be themselves reduced to their extremities and possibly some intermediate dots. This is because an octogon is already the cumulated dilation of 4 segments. This transformation is easily supported within the TItBIT retina model.

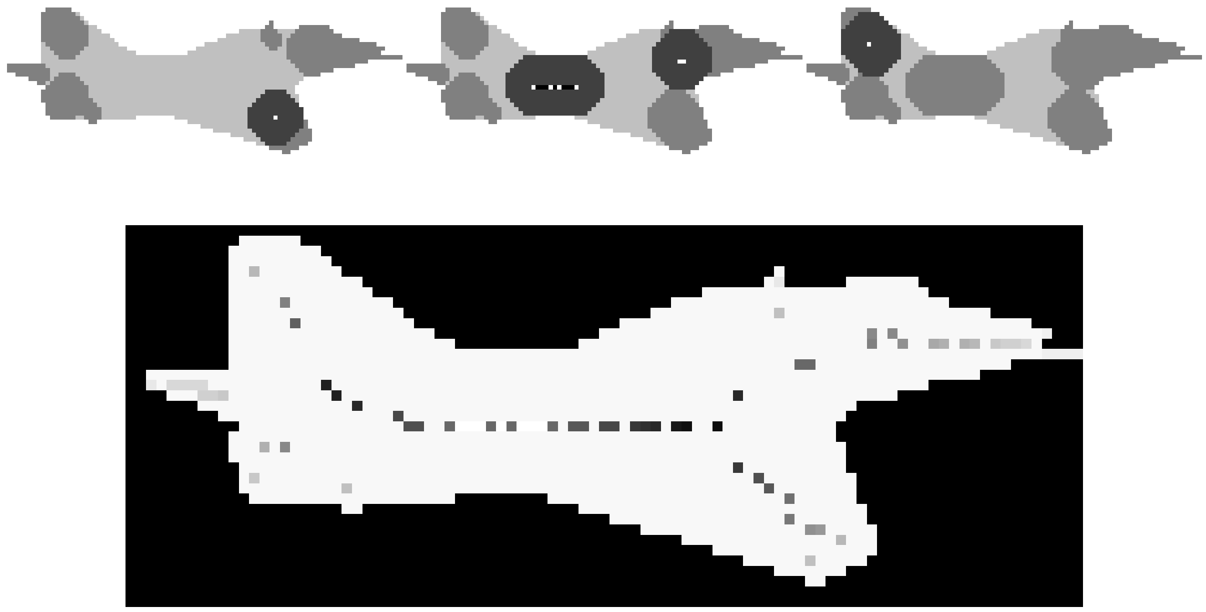

Figure 3: *Compression of a silhouette within the TItBIT retina model. Unlike the version presented in Tab. 2, patterns here are predefined such that their cumulated dilation is as close as possible to a regular growing octogon.* **Above***, 3 successive steps are presented. The legend is as follows: white (dots) for the remainder, dark grey (areas) for the contribution of the remainder to the description of the shape, medium grey for parts of the initial shape already contributed to by previous remainders. In the center image, note that a whole segment of the remainder has been replaced by 4 white dots. Suppressed points appear in black.* **Under***, all the remainders are represented on a single image. The later the remainders have been obtained during compression, the darker they appear.*

5 Conclusion

What we have presented in this paper is a methodology for fast shape learning and recognition that fits the organization of vision systems based on programmable artificial retinas [3]. The emphasis has been put on the gereral foundations and trade-offs rather than on the design of operational algorithms. There are obvious relationships with existing techniques of "shape decomposition into meaningful parts" as well as decomposition into particular structuring elements (e.g. maximal convex polygons [5] or 3×3 element [7]). From this point of view, *compression stands out as a unifying concept.*

As far as retinal adaptive target tracking is concerned, the tree representation we have come up with now has to be improved to become robust with regard to shape deformation, occlusion and rotation. Also, the inside and outside of a shape have to be learned simultaneously. A whole program for further research...

References

[1] T. Bellotti and A. Gammerman. A minimal length encoding system. Technical Report NC-TR-95-035, NeuroCOLT, May 1995.

[2] T. M. Bernard and P. E. Nguyen. Vision through the power supply of the NCP Retina. In *IST/SPIE Symp. on Electronic Imaging: Science and Technology / CCDs and solid state optical sensors V*, February 1995.

[3] T. M. Bernard, B. Y. Zavidovique, and F. J. Devos. A programmable artificial retina. *IEEE Journal of Solid-State Circuits*, 28(7):789–798, July 1993.

[4] P. Garda, A. Reichart, H. Rodriguez, F. Devos, and B. Zavidovique. Une rétine électronique automate cellulaire. *Traitement du Signal*, 5(6):435–449, 1988.

[5] A. Held and A. Keiichi. On the decomposition of binary shapes into meaningful parts. *Pattern Recognition*, 27(5):637–47, 1994.

[6] J. Lazzaro et al. Silicon auditory processors as computer peripherals. *IEEE Transactions on Neural Networks*, 4(3):523–528, May 1993.

[7] H. Park and R. T. Chin. Decomposition of arbitrarily shaped morphological structuring element. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(1), January 1995.

[8] J. Serra. *Image Analysis and Mathematical Morphology.* San Diego: Academic Press, 1982.

[9] G. J. Wolff. Towards a new concept of software. *Software Engineering Journal*, 9(1):27–38, July 1994.