Quick Construction of Efficient Morphological Operators by Computational Learning

Hae Yong Kim

Electronics Letters, vol. 33, no. 4, pp. 286-287, 13th Feb. 1997.

Quick Construction of Efficient Morphological Operators by Computational Learning

H. Y. Kim

Indexing terms: Mathematical morphology, image processing, operator decomposition, PAC learning.

The author introduces a new technique to construct generic morphological operators quickly and automatically using input-output pairs of images as training samples. This process can be modeled as generalized PAC (Probably Approximately Correct) learning. He also introduces a new representation for morphological operators, much more efficient than the traditional one.

Introduction: Mathematical Morphology is one of the techniques used in Image Processing. We must solve two problems to use Mathematical Morphology in practice.

The first problem is the construction of the desired morphological operator, that is, the choice of the most adequate operator among a great number of candidates. Barrera et al. [1] represent the desired operator as input-output pairs of images. In that work, the user has to obtain some typical input images of the application with corresponding output images. These images feed a system that constructs automatically the desired morphological operator. But the algorithms described in [1] are too slow and they are for binary images only. In this letter, we present a new algorithm that constructs automatically generic operators (binary, gray-level and color). Moreover, in some practical tests, when used to construct binary operators, the new algorithm was 11000 times faster than the old ones, spending 3 times less memory (see table 1). Automatic construction of morphological operators can be modeled as a generalized PAC learning (see [2]).

The second problem is the need of complex operators, frequently composed of hundreds of thousands of elementary operators, to process gray-scale and color images. Using conventional techniques, we spend a lot of computational time to apply such a complex operator. An elementary morphological operator, such as an erosion, spends some seconds to process an image with, for example, 512 × 480 pixels. To apply 100000 erosions to that image, we would spend at least 1 day. New algorithm can apply 100000 sup-generators (the infimum of an erosion and an antidilation and consequently more powerful than a simple erosion) in only 15 seconds.

Mathematical Morphology: Heijmans [3] explains the complete lattice theory in details. A partially ordered set $(\mathcal{L}, \leq)$ is a *complete lattice* if all subsets of $\mathcal{L}$ have an infimum and a supremum. A *morphological operator* is a function from a complete lattice into another complete lattice. Banon et al. [4] describe a decomposition of a morphological operator, called *constructive decomposition*. To use it in practice, we

had to modify it and we named this modified one as "interval decomposition". This is the decomposition used in our algorithms.

Cut tree: We will represent a morphological operator as a binary tree, called *cut tree*. Any morphological operator $\psi \in \mathcal{L}^i \to \mathcal{L}^o$ can be represented as a cut tree, whenever $\mathcal{L}^i$ is a product of d totally ordered finite sets and $\mathcal{L}^o$ is an arbitrary complete lattice. Let us say that we know the output of ψ in m points, that is,

$$\vec{a} = (a_1, \ldots, a_m) = ((a_1^i, a_1^o), \ldots, (a_m^i, a_m^o)) .$$

where, $\psi(a_j^i) = a_j^o$, $\forall j \in [1,m]$. The sequence $\vec{a}$ will be called *training sample*. The unknown points will be considered as "don't cares". The *cut algorithm* will construct a cut tree that represents an operator $\hat{\psi}$, similar to ψ. The outputs of $\hat{\psi}$ and ψ are the same in m known points, that is, $\psi(a_j^i) = \hat{\psi}(a_j^i)$, $\forall j \in [1,m]$. But they can differ in "don't care" points.

In the cut algorithm, a hyperplane parallel to the axes of the coordinate system splits the space $\mathcal{L}^i$. This hyperplane divides the known points in two sets with the same cardinality. For each semi-space obtained, the splitting process continues recursively, generating hyper-parallelepipeds. This process stops when each hyper-parallelepiped contains only known points with the same output value. Each hyperplane corresponds to an internal node of the cut tree and each hyper-parallelepiped corresponds to a leaf. A cut tree has the following properties:

a) Each hyper-parallelepiped H represents a sup-generator λ. If $X \in H$ then $\lambda(X)$ is equal to the output value of the known points that belong to H. If $X \notin H$ then $\lambda(X) = \mathbf{0}$. Let us denote the sup-generators of the cut tree as $\lambda_1, \lambda_2, \ldots, \lambda_v$.

b) The supremum of all sup-generators is $\hat{\psi}$.

c) Given a point $X \in \mathcal{L}^i$, $\hat{\psi}(X)$ is equal to the output of one of the sup-generators, say $\lambda_e(X)$, because for all $j \in [1,v], j \neq e$, $\lambda_j(X) = \mathbf{0}$.

d) Given a point $X \in \mathcal{L}^i$, the binary tree structure allows quick search of $\lambda_e(X)$.

The construction of a cut tree spends $O(dm \log m)$ time and uses $O(dm)$ space. The application of an operator $\hat{\psi}$ (represented as a cut tree) in an image with k pixels spends $O(dk \log m)$ time.

Construction of cut tree from sample images: Training sample $\vec{a}$ can be obtained from input-output pairs of sample images. Let us suppose that the sample consists of only one pair of images, say A^i and A^o. From A^i and A^o, we have to find a morphological operator $\hat{\psi}$ that, given an image "to be processed" Q^i, generates a "processed image" $Q^p = \hat{\psi}(Q^i)$ that is "similar" to an unknown ideal image Q^o (see figures 1-5). In

practice, the interesting operators are translation invariant and locally defined within a window W.

Each pixel p of A^i with its neighbor pixels that belong to the window W translated to p form a point in the space $\mathcal{L}^i$. To each point $a_j^i \in \mathcal{L}^i$ so obtained, there is an associated output value $a_j^o \in \mathcal{L}^o$, the corresponding color in image A^o. Let us denote these data as:

$$\vec{a} = (a_1, \ldots, a_m) = ((a_1^i, a_1^o), \ldots, (a_m^i, a_m^o)) .$$

This sequence can be used as a training sample of the cut algorithm. Note that in sequences obtained from the sample images there can be two or more points of $\mathcal{L}^i$ with the same coordinates but with different output values. We must modify the cut algorithm slightly to solve this problem properly.

Applications: We wrote three programs (for binary, gray-scale and color images) that implement the ideas developed in this letter. Table 1 shows the performance of these programs and of ISI-2, the best algorithm described in [1]. The performance of ISI-2, measured in a Sparc Station 2, was transcribed from [1]. The performance of cut tree was measured in a PC-clone computer with AMD 586 processor.

The cut algorithm can be used in noise elimination, pattern recognition, texture recognition, enhancement of edges, etc. A color application that emulates the "trace contour" filter of Adobe Photo Shop is depicted in figures 1-5. From images 1 and 2, the cut algorithm learns the behavior of the "trace contour" filter and constructs a cut tree that emulates it. Applying this cut tree in figure 3, we obtain the figure 4. The figure 5 is the image generated by original "trace contour".

Conclusion: In this letter a new technique to construct a generic morphological operator automatically by computational learning was presented. The cut tree, a new representation for morphological operators, was introduced. A cut tree is a very efficient representation for a morphological operator and it can be constructed and applied quickly using the algorithms described.

H. Y. Kim (Instituto de Matemática e Estatística, Universidade de São Paulo).

References:

[1] Barrera, J., Tomita, N. S., Silva, F. S. C., and Terada, R.: 'Automatic Programming of Binary Morphological Machines by PAC Learning', Proc. SPIE, 1995, **2568**, pp. 233-244.

[2] Haussler, D.: 'Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications', *Information and Computation,* 1992, **100**, pp. 78-150.

[3] Heijmans, H. J. A. M.: 'Morphological Image Operators', Academic Press Inc., 1994.

[4] Banon, G. J. F. and Barrera, J.: 'Decomposition of Mappings Between Complete Lattices by Mathematical Morphology, Part I. General Lattices', *Signal Processing,* 1993, **30**, pp. 299-327.

Algorithm	|W| (size)	Examples (with repetitions)	Examples (without repetitions)	Size of operator (bytes)	Training (seconds)	Application (seconds)
Cut (binary)	12	65341	1747	14076	2	4*
ISI-2 (binary)	12	?	1386	55672	5	?
Cut (binary)	49	65341	27267	218532	5	6*
ISI-2 (binary)	49	?	13966	620352	56858	?
Cut (binary)	25	65341	15855	127044	3	4*
Cut (gray)	4	65341	37543	680576	9	5*
Cut (color)	4	65341	38963	701904	14	9*

Table 1: Performance comparison of different algorithms.
* Time spent to apply the operator in an image 480 × 512.

Figure 1: Input example (A^i)

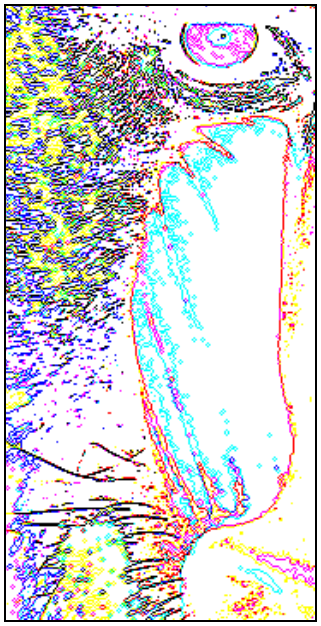

Figure 2: Output example (A^o)

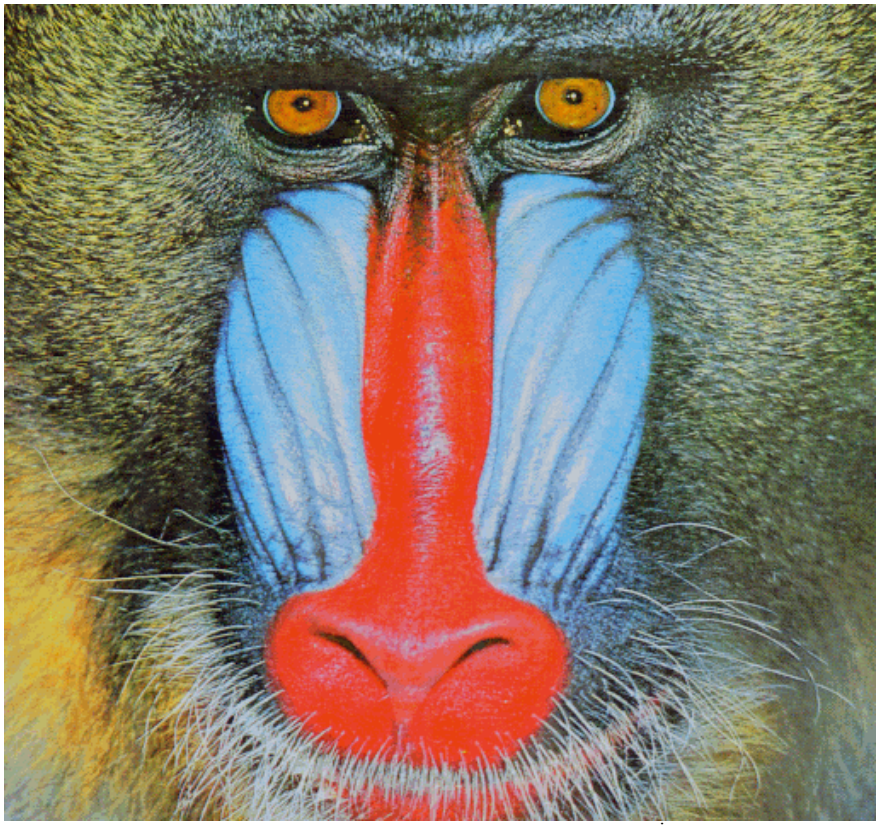

Figure 3: Image to be processed (Q^i)

Figure 4: Processed image (Q^p)

Figure 5: Unknown ideal image (Q^o)