MORPHOLOGICAL HAT-TRANSFORM SCALE SPACES AND THEIR USE IN TEXTURE CLASSIFICATION

A. C. Jalba, J. B. T. M. Roerdink and M. H. F. Wilkinson

Institute of Mathematics and Computing Science
University of Groningen, P.O. Box 800
9700 AV, Groningen, The Netherlands

ABSTRACT

In this paper we present a multi-scale morphological method for use in texture classification. A connected operator similar to the morphological hat-transform is defined, and two scale-space representations are built. The most important features are extracted from the scale spaces by unsupervised cluster analysis, and the resulting pattern vectors provide the input of a decision tree classifier. We obtain 93.5 % correct classification for the Brodatz texture database.

1. INTRODUCTION

Several techniques for multi-scale morphological analysis exist, such as pyramids [1], size distributions, or granulometries [2], which are used to quantify the amount of detail in an image at different scales. A similar method, based on sequential alternating filters, has been proposed by Bangham and coworkers [3]. Their method is used on 1-D signals, though they discuss extensions to higher dimensions. A different multi-scale approach to the analysis of 1-D signals was presented by Leymarie and Levine [4]. They constructed a morphological curvature scale space for shape analysis, based on sequences of morphological top-hat or bottom-hat filters with increasing size of the structuring element.

In [5] we modified the initial technique of Leymarie and Levine to allow for nested structures, and included a method by which features in the scale space may be clustered in an unsupervised way, resulting in a small set of rotation, translation and scale-invariant shape parameters. In this paper we generalize the hat scale spaces to n-dimensional signals, give a fast algorithm for computing these scale spaces, and apply them to pattern classification. We report results for texture classification, using the *Brodatz* texture database.

2. THEORY

Connected operators [6] are characterized by the powerful property of preserving contours, and they only transform an image by selectively altering the grey values of connected sets of pixels. There are several ways of defining the notions of connectivity and connected operators. As is usual in mathematical morphology, a binary image X is considered a subset of some domain E, usually $E \subseteq \mathbb{Z}^2$. Connectivity is defined using either 4-adjacency or 8-adjacency in the square grid of pixels.

A *connected component* of X is a connected set $C(X)$ which is maximal. A *flat zone* L_h at level h of a grey-scale image f is a connected component $C(X_h(f))$ of the level set $X_h(f) = \{p \in E | f(p) = h\}$. A *peak component* P_h at level h is a connected component of the threshold set $T_h(f) = \{p \in E | f(p) \geq h\}$. At each level h there may exist several such components (flat zones, peak components), indexed as L_h^i, P_h^j, respectively, with i, j from two index sets.

A flexible way of defining connected operators for functions is via partitions. Let $\mathcal{P}(E)$ be the set of all subsets of E. A partition $P : E \rightarrow \mathcal{P}(E)$ of E is defined such that (i) $x \in P(x)$, $x \in E$, and (ii) $P(x) = P(y)$ or $P(x) \cap P(y) = \emptyset$, for $x, y \in E$. In words, a partition is a subdivision of the underlying space into disjoint zones. The partition P of E is said to be *coarser* than the partition P' (or P' is *finer* than P) if $P'(x) \subseteq P(x)$ for every $x \in E$. The *partition of flat zones* [6] $C(f)$ of f is used to define connected operators.

Definition 1 *An operator γ acting on a grey-level function f is said to be* connected *if $C(\gamma(f))$, the partition of flat zones of $\gamma(f)$, is coarser than $C(f)$.*

Definition 2 *The* connected opening $\Gamma_x(X)$ *of a set X at a point x is the connected component of X containing x if $x \in X$, and $\emptyset$ otherwise.*

Given a set A, the geodesic distance $d_A(p, q)$ between two pixels p and q is the length of the shortest path joining p and q which is included in A. This distance is highly dependent on the type of connectivity used. The geodesic distance between a point $p \in A$ and a set $D \subseteq A$ is defined as

0-7803-7750-8/03/$17.00

$d_A(p, D) = \min_{d \in D} d_A(p, d)$. One important morphological operator based on the geodesic distance is the *geodesic dilation* which is defined as follows.

Definition 3 *Let $X \subseteq E$ be a subset of E and $Y \subseteq X$. The* geodesic dilation *of integer size $n \geq 0$ of Y within X is the set of pixels of X whose geodesic distance to Y is smaller or equal to n:*

$$\delta_X^{(n)}(Y) = \{p \in X \mid d_X(p, Y) \leq n\}.$$

In the binary case, the *reconstruction* $\rho_X(Y)$ of a set X from a set $Y \subseteq X$ is obtained by iterating geodesic dilations of Y inside X until stability is obtained, i.e.,

$$\rho_X(Y) = \bigcup_{n \geq 1} \delta_X^{(n)}(Y). \tag{1}$$

Similarly, using the threshold superposition principle [7], the grey-scale reconstruction can be defined. Let f and g be two grey-scale images defined on the same domain, such that $g \leq f$ for each pixel.

Definition 4 *The* grey-scale reconstruction $\rho_f(g)$ *of f from g at a point x is given by:*

$$(\rho_f(g))(x) = \max\{h \mid x \in \rho_{T_h(f)}(T_h(g))\}.$$

2.1. Definition of the hat-transform scale spaces

We start by defining a connected operator θ acting on grey-scale functions, which will be used to define the hat scale spaces. Given a grey-scale function f, the value of θ applied to f at a point x is given by

$$(\theta(f))(x) = \max\{h' < f(x) \mid Q_{x,h'}(f)\} \tag{2}$$

where $Q_{x,h'}(f)$ is the following criterion:

$$Q_{x,h'}(f) \equiv \delta^{(1)}_{T_{h'}(f)}(\Gamma_x(T_{f(x)}(f))) \subset \Gamma_x(T_{h'}(f)). \tag{3}$$

In words, the value of $\theta(f)$ at a point x is given by the maximum grey level h' smaller than $f(x)$ for which the criterion in (3) holds. The criterion $Q_{x,h'}$ is fulfilled when the geodesic dilation of size one of the connected opening at point x of the threshold set $T_{f(x)}(f)$ is *strictly* included in the connected opening of $T_{h'}(f)$. When the input function f is constant we use the convention $\theta(f) := f$. An example of application of this operator on a 1-D signal is shown in Fig. 1. Notice that this formulation is applicable without any modification to n-D functions.

To extract the scale space features from f from top to bottom we must use grey-scale reconstruction. The result of reconstructing f from $g = \theta(f)$ is shown in the middle picture in Fig. 1, and is denoted by r. The desired detail is simply $f - r = f - \rho_f(\theta(f))$.

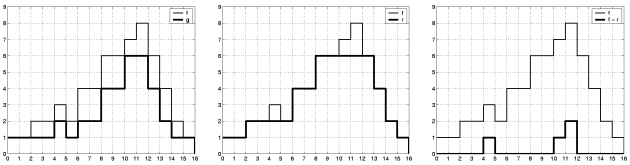

Fig. 1. *Left*: original signal f (thin) and $g = \theta(f)$ (thick); *center*: f and $r = \rho_f(g)$ (thick); *right*: f and the detail signal τ_0 (thick).

Definition 5 *The* connected top-hat transform *of a grey-scale image f at a point x is given by:*

$$(\tau(f))(x) = (f - \rho_f(\theta(f)))(x). \tag{4}$$

The top hat scale space can be obtained by iterating, (4):

Definition 6 *The* top-hat scale space *of a grey-scale image f is given by the sequence $(\tau_0, \tau_1, \ldots, \tau_K)$ defined by the iteration*

$$\begin{aligned} f_{k+1} &= \rho_{f_k}(\theta(f_k)) \\ \tau_k &= f_k - f_{k+1} \end{aligned} \tag{5}$$

where $f_0 := f$ and $k > 0$.

Eq. (5) is iterated until $f_K = f_{min}$ for all pixels, where f_{min} is the minimum value of f. Using $f \leftrightarrow -f$, dual operators of those in (2), (4) and a bottom-hat scale space can be obtained .

3. CONSTRUCTING N-DIMENSIONAL HAT SCALE SPACES

The construction of the hat scale spaces in two or more dimensions relies on a modified version of Salembier's max-tree data structure [8], which can be constructed in linear time.

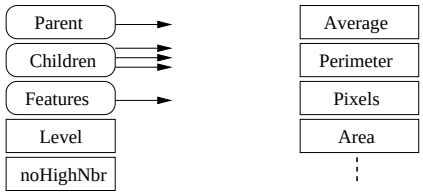

Fig. 2. Data structures. *Left*: a max-tree node; *right*: its corresponding features.

A max-tree is a rooted tree, in which each of the nodes C_h^k at grey-level h corresponds to a peak component P_h^k. However, C_h^k contains only those pixels in P_h^k which have grey level h. We have modified this representation such that it permits bidirectional traversal. Figure 2 shows the node

data structure, along with its corresponding features; the arrows represent pointers. The node structure also contains: (i) *Level* - the grey level of the peak component; (ii) *Features* - a pointer to a feature structure shown in Fig. 2 (see section 3.1); (iii) *noHighNbr* (no high neighbour) - a boolean value; its use will be explained later in this section.

Definition 7 *A node at level h of the max-tree may have zero, one, or more than one child. We call a node:*

- *a* leaf*, if it has no children (i.e. a regional maximum);*
- *a* simple node*, if it has exactly one child;*
- *a* compound node*, if it has more than one child.*

One can construct the top-hat scale space (see (5)) from simple and compound nodes of the max-tree in the recursively:

- All child components of a compound node represent entries in the scale space at the grey level of the compound node. It is easy to see that in this case the criterion in (3) holds.
- The child of a simple node represents an entry in the scale space at the grey level h of the component if the component has at least one pixel with no neighbour at a grey level strictly higher than h. This is because the geodesic dilation of size one of the child component within the component is strictly included in the component, and the criterion in (3) holds.

The variable $noHighNbr$ shown in Fig. 2 indicates if the last case holds for a given node. It is initialized with *false*, but it becomes *true* (and it remains *true*) if the condition is satisfied for any pixel of the component.

3.1. Scale space features

Fig. 2 shows the basic attributes, maintained in each node of the tree, which are used to compute features such as: *compactness*, *complexity*, *moment of inertia*, *average height* and *entropy*. These data sets can be updated when a new pixel, which belongs to a peak component, is found. They can be merged with other data sets of child components, and permit efficient computation of the desired features. The average height and entropy features are computed in the second step, when the scale space is built. All other features can be computed incrementally, when the tree is built.

Direct use of scale-space features as pattern vectors is problematic for many statistical methods, because pattern vectors of different images would differ in length. One way to solve this problem is to set the boundaries between classes of scale-space features from the data themselves. This is done by mean-shift cluster analysis as in [5]. After clustering is performed, the pattern vector is given by the centroids of the first six clusters with the largest areas.

3.2. 2-D hat scale-space implementation

The pseudo-code of the recursive function which builds the scale space in the 2-D case is shown in Algorithm 1. The function *HatScaleSpace* must be called for the root node of the tree. The variable $edata$, used to compute the entropy, is an array of integers of size $Levels$, where $Levels$ represents the number of grey levels present in the image (usually 256). The variable $pixels$ is an integer which must be initialized to 0 when the procedure is called for the root node. At the end of this call, all entries in the scale space are kept in the $sspace$ list. All these variables must in fact be references or pointers to the specified types.

Algorithm 1 2-D hat scale-space computation

Function **HatScaleSpace**(n, $edata$, $pixels$, $sspace$)

```
1:  for each child c of node n do
2:    HatScaleSpace( c, edata, pixels, sspace )
3:    n.Features.Average :=
      n.Features.Average + c.Features.Average
4:    if n.noHighNbr or n has more than one child then
5:      cavg := c.Features.Average
6:      carea := c.Features.Area
7:      n.Features.Average :=
        n.Features.Average − cavg, entropy := 0
8:      for k := 0 to Levels do {Compute entropy}
9:        p := edata[k]/pixels
10:       entropy := entropy + p ∗ log(p)
11:     Clear(edata, 0), pixels := 0
12:     AddEntry(sspace, Entry(cavg/carea, entropy, ...) )
13: if n.noHighNbr or n has more than one child then
14:   edata[n.Level] := n.Features.Area
15:   pixels := n.Features.Area
16: else
17:   edata[n.Level] :=
      edata[n.Level] + n.Features.Pixels
18:   pixels := pixels + n.Features.Pixels
```

The function proceeds by calling itself for each child node c of the parent node n. After the function returns from recursion, the variable $Average$ is updated such that it contains the sum of all $Average$ values of an entire branch of the tree. The test in line 4 is true when one of the cases specified in Definition 7 holds for the node n. If true, a new entry in the scale space is added (line 12). In this case, the $Average$ value of the child c is subtracted from the value of its parent (line 7). After computing the entropy of the grey-level distribution of the entry (lines $8-10$), and resetting the variables $edata$ and $pixels$, the entry is added to the scale space (line 12). If the test in line 13 is true, i.e. new entries in the scale space were added, both the level of the array $edata$ (which corresponds to the grey-level of the node n) and the variable $pixels$ are set to the area of the node (lines $14 - 15$). This is because all children of n were lowered to the grey-level of the node n, and now at this grey-level

there is a flat zone with the same area as n. Otherwise the function simply updates the $edata$ and $pixels$ variables by adding the number of pixels of the component represented by the node n (lines $17-18$).

A similar approach can be followed to compute the bottom hat scale space by constructing a min-tree (see [8]) and using the same procedure as in Algorithm 1. Because each node in the tree is visited at most twice this procedure is linear in the number of nodes. The computation can be extended to arbitrary dimension by defining the associated adjacency and building the max/min-trees.

4. EXPERIMENTAL RESULTS

In our experiments we have used the Brodatz texture database, using the C4.5 algorithm [9] for constructing decision trees, with bagging [10] as a method of improving the accuracy of the classifier. The performance was evaluated using the *holdout* [11] method. The mean performance (in terms of correctly identified textures) of the method was 93.5 ± 1.5 %. This compares favourably with other methods as published in [12]. Those experiments showed that reduced multidimensional histograms provided higher classification accuracies (93.9 %) than those produced using channel histograms (90.4 %) and wavelet packet signatures (85.1 %). A direct comparison cannot be performed, due to the different classifiers involved. We have used bagging in order to improve the holdout estimate of accuracy, while in [12] a selection of features was used which minimized the leave-one-out classification error. Next, they used a genetic algorithm which further improved the classification performance by minimizing the error rate produced by the selected features. However, the methods in [12] were tailored towards texture classification, while our method can also handle other types of classification problems, e.g. automatic identification of diatoms [5].

5. CONCLUSIONS

We have proposed a method for classification tasks based on morphological hat scale spaces, combined with unsupervised cluster analysis, which can be used for texture feature extraction. The classification performance (93.5%) is comparable with the result of one of the best methods for texture classification: reduced channel histograms.

The advantages of using the proposed hat scale-space representations are: (i) a small number of scale space entries, compared with the number of peak components; (ii) all the extracted scales are important because major changes in the topology of the signal occur at these scales; (iii) once some entries in the scale space are obtained, they can be characterized by computing not only shape and size features, but also features related to the 'height' of each peak component. In this representation, one can use links between components at sequential grey levels in the signal.

6. REFERENCES

[1] J. Goutsias and H. J. A. M. Heijmans, "Nonlinear multiresolution signal decomposition schemes - part one: Morphological pyramids," *IEEE Trans. Image Proc.*, vol. 9, pp. 1862–1876, 2000.

[2] E. J. Breen and R. Jones, "Attribute openings, thinnings and granulometries," *Comput. Vis. Image Understand.*, vol. 64, no. 3, pp. 377–389, 1996.

[3] J. A. Bangham, P. D. Ling, and R. Harvey, "Scale-space from nonlinear filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, pp. 520–528, 1996.

[4] F. Leymarie and M. D. Levine, "Curvature morphology," Tech. Rep. TR-CIM-88-26, Computer Vision and Robotics Laboratory, McGill University, Montreal, Quebec, Canada, 1988.

[5] A.C. Jalba, M.H.F. Wilkinson, J.B.T.M. Roerdink, M.M. Bayer, and S. Juggins, "Automatic diatom identification using contour analysis by morphological scale spaces," *Comput. Vision Image Understand.*, 2001, submitted.

[6] J. Serra and P. Salembier, "Connected operators and pyramids," in *SPIE Image Algebra Morphological Image Processing IV*, 1993, vol. 2030, pp. 65–76.

[7] L. Vincent, "Morphological grayscale reconstruction in image analysis: application and efficient algorithm," *IEEE Trans. Image Proc.*, vol. 2, pp. 176–201, 1993.

[8] P. Salembier, A. Oliveras, and L. Garrido, "Anti-extensive connected operators for image and sequence processing," *IEEE Trans. Image Proc.*, vol. 7, pp. 555–570, 1998.

[9] J. Ross Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.

[10] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24(2), pp. 123–140, 1996.

[11] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. Int. Joint Conf. Artific. Intell.*, 1995, pp. 1137–1145.

[12] K. Valkealahti and E. Oja, "Reduced multidimensional co-occurrence histograms in texture classification," *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 20, pp. 90–94, 1998.