Morphological hat-transform scale spaces and their use in pattern classification

Andrei C. Jalba, Michael H.F. Wilkinson, Jos B.T.M. Roerdink [*]

Institute for 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 method based on mathematical morphology which can successfully be used in pattern classification tasks. 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 report classification results obtained using contour features, texture features, and a combination of these. The method has been tested on two large sets, a database of diatom images and a set of images from the Brodatz texture database. For the diatom images, the method is applied twice, once on the curvature of the outline (contour), and once on the grey-scale image itself.

Key words: mathematical morphology, scale space, top-hat transform, bottom-hat transform, connected operators, pattern classification, decision trees, diatom images, Brodatz textures.

1 Introduction

Multi-scale representation is a very useful tool for handling image structures at different scales in a consistent manner. It was introduced in image analysis and computer vision by Marr, Witkin and others who appreciated that multi-scale analysis offers many benefits [1–4]. The basic idea is to embed the original signal $f : \mathbb{R}^n \to \mathbb{R}$ into a stack of signals filtered at increasing scales, in which the fine details are successively suppressed. The signal filtered at scale $\sigma \in \mathbb{R}$ is a function

[*] Corresponding author. Tel: +31-50-3633931; Fax: +31-50-3633800
Email address: `roe@cs.rug.nl` (Jos B.T.M. Roerdink).

$F : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ defined by $F(x, \sigma) = (\mathcal{O}_\sigma(f))(x)$. Here $\mathcal{O}_\sigma$ is a filter operator depending on σ, F is a function in an $(n+1)$-dimensional space, called *scale space*, and the collection of filtered signals is referred to as the *multi-scale representation* of f. The filter operation $\mathcal{O}_\sigma$ can be a linear operation (e.g. Gaussian smoothing) or a nonlinear operation (e.g. morphological filter). Since the scale-space concept was introduced into image analysis and computer vision, its use has been firmly established, and there has been an emerging interest in incorporating scale-space operators as part of high-level computer vision tasks [5]. The output of the scale-space representation can be used for a variety of early visual tasks, from feature detection and feature classification to shape computation [4].

Several techniques for multi-scale morphological analysis exist, such as pyramids [6], size distributions, or granulometries [7, 8], 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 [9]. Their method is used on 1-D signals, though they discuss extensions to higher dimensions [10]. A different multi-scale approach to the analysis of 1-D signals was presented by Leymarie and Levine [11]. 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 [12] we considered classification of diatoms, which are microscopic, single-celled algae, which build highly ornate silica shells or frustules. Some examples are shown in Figure 5.1. For many purposes, automation of the identification of diatoms by image analysis is highly desirable. The Automatic Diatom Identification and Classification (ADIAC) project [13], of which this research is a part, aimed at automating the process of diatom identification by digital image analysis. 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 [12]. Allowing for nested structures is of paramount importance especially for 2-D signals, when small structures nested within a larger one can be extracted and represented at some levels in the scale space. Other advantages of using these scale-space representations, henceforth referred to as ‘hat scale spaces’, are discussed gradually over the next sections.

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 classification results for (i) diatom identification, using both shape and interior structure information, and (ii) texture classification, using the *Brodatz* texture database. We also briefly discuss further applications in texture segmentation.

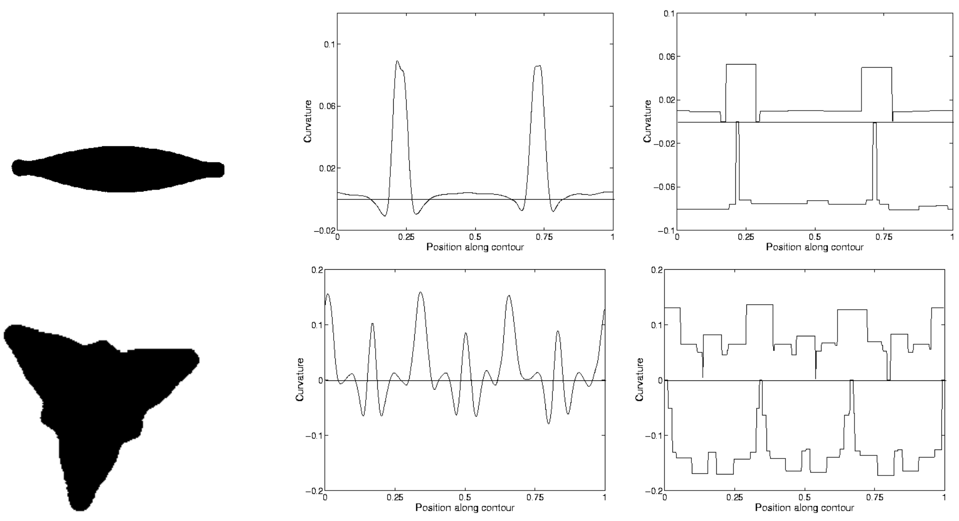
The remainder of this paper is organized as follows. Section 2 briefly describes the construction of 1-D hat scale spaces and the features extracted from these. Section 3 presents the extension to higher dimensions by means of *connected operators*. Section 4 provides a fast implementation of the algorithm for computing 2-D hat scale spaces. In section 5 we report classification results, using a supervised classification technique based on decision trees, on two large sets, a database of diatom images and a set of images from the Brodatz texture database. Conclusions are drawn in section 6.

2 One-dimensional hat-transform scale spaces

Morphological operators [14, 15] can remove structure from a signal and therefore they were found suitable for constructing scale spaces [16–18]. The *hat-transforms* represent an important class of morphological transforms used for detail extraction from signals or images. Assume a signal f and a 1-D structuring element K. The *dilation* computes the maximum signal value over a circular neighborhood of a given radius. Contrary, the *erosion* computes the minimum signal value over the neighborhood. Erosion followed by dilation represents an important morphological transform called *opening*, denoted by $f \circ K$. Its dual, *closing*, denoted by $f \bullet K$, is a dilation followed by an erosion.

The residual of the opening compared to the original signal, i.e., $f - (f \circ K)$ represents the *top-hat* transform. Thus, when the opened signal is subtracted from the original, the desired detail is obtained. Its dual, the *bottom-hat* transform, is defined as the residual of a closing compared to the original signal f, i.e., $f - (f \bullet K)$. Therefore, one can use hat-transforms with increasing size of the structuring element to extract details of increasing size. By performing repeated hat-transforms with increasing size of the structuring element on the signal, we can build the morphological hat scale spaces.

A hat scale space consists of a number of levels $\ell = 1, 2, \ldots, L$, where a level with index ℓ corresponds to a top-hat transform with size δ_ℓ of the structuring element, where δ_ℓ increases with ℓ. Let the signal be stored in an array C, K_ℓ denote the structuring element used at level ℓ, and T_ℓ denote the top-hat $C - (C \circ K_\ell)$. All nonzero elements of T_ℓ are parts of features at scales δ_ℓ or smaller. Starting at level 1, we apply top-hat transforms to extract peaks of the signal. At each level ℓ, T_ℓ is compared to $T_{\ell-1}$. If a peak which is present at level $\ell - 1$ stops increasing at level ℓ, i.e. $T_\ell = T_{\ell-1}$ for all points belonging to the peak, it is removed from the original signal C, and its mean value, extent and location are stored. This process ends when either all elements of array C are zero, or the largest scale L is reached. This yields the top scale space in which every peak is precisely localized. Similarly, bottom-hat transforms are used to obtain the bottom scale space, in which all valleys of the signal are described. At the end of this we have obtained two scale spaces, a top scale space of peaks and a bottom scale space of valleys.



The description just given is only an intuitive interpretation of the construction process of hat scale spaces, and is not used in a real implementation, because such a naive approach results in a time complexity of order $O(L \cdot N)$, where N is the number of values of the signal f. This result is obtained when the opening transform is computed by a linear-time algorithm, insensitive to the size of the structuring element, similar to that of Gil and Werman [19]. Still, when L approaches N, the CPU time taken to construct the hat scale spaces is proportional with N^2. The 1-D hat scale spaces can efficiently be implemented using Tarjan's Union-Find approach for maintaining disjoint sets. For more details, we refer to [12, 20].

The scale spaces can be visualized by plotting each feature as a box of the average height at the appropriate location of the signal. If nested features are present, we can simply stack the features in the plot. An example is shown in Figure 2, where the 1-D function is given by the curvature of the binary images of diatoms [12] in the left-side of the figure.

3 Two-dimensional hat-transform scale spaces

In this section, we show how to extend the one-dimensional hat scale spaces described in section 2 to 2-D signals (images), and more generally to n dimensions, by means of *connected operators*. First, some preliminary definitions are presented which introduce the main concepts. Subsequently, the hat-transform scale spaces are formalized.

3.1 Preliminaries

Connected operators [21] 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. Here we shall follow the definitions of [21, 22].

Let E be an arbitrary nonempty set of vertices, and denote by $\mathcal{P}(E)$ the collection of subsets of E. Also, let $G = (E, \Gamma)$ be an undirected graph, where Γ is a mapping from E to $\mathcal{P}(E)$ which associates to each point $x \in E$ the set $\Gamma(x)$ of points adjacent to x. It is common in image processing to assume that E is a regular grid, i.e. $E \subseteq \mathbb{Z}^n$ ($n = 2$), and Γ corresponds to either 4-adjacency or 8-adjacency in the square grid of pixels. In what follows we assume $E \subseteq \mathbb{Z}^2$.

A *path* π in a graph $G = (E, \Gamma)$ from point x_0 to point x_n is a sequence $(x_0, x_1, ..., x_n)$ of points of E such that (x_i, x_{i+1}) are adjacent for all $i \in [0, n)$. Let $X \subseteq E$ be a subset of E. A set X is *connected* when for each pair (x_0, x_n) of points in X there exists a path of points in X that joins x_0 and x_n. 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 *regional maximum* M_h at level h is a flat zone which has only strictly lower neighbours. 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, regional maxima), indexed as L_h^i, P_h^j, M_h^k, respectively, with i, j, k from three index sets. It should be noted that any regional maximum M_h^k is also a peak component, but the reverse is not true.

A flexible way of defining connected operators for functions is via partitions [22]. A function $P : E \to \mathcal{P}(E)$ is called a *partition* of E if (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. Let P and P' be two partitions of E. Partition P is said to be *coarser* than P' (or P' is *finer* than P) if $P'(x) \subseteq P(x)$ for every $x \in E$.

Grey-level connected operators can be introduced if we define a partition associated to a grey-level function f. It can be shown [21] that the set of flat zones of a function constitutes a partition of the domain of f. In the following, this partition will be called the *partition of flat zones* of f, and will be denoted by $C(f)$.

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 $C(f)$.*

Thus, the only operations a connected operator can do are merging flat zones, and modifying their grey levels.

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 mask), 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 $d_A(p,D) = \min_{d \in D} d_A(p,d)$. 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 [23], 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:*

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

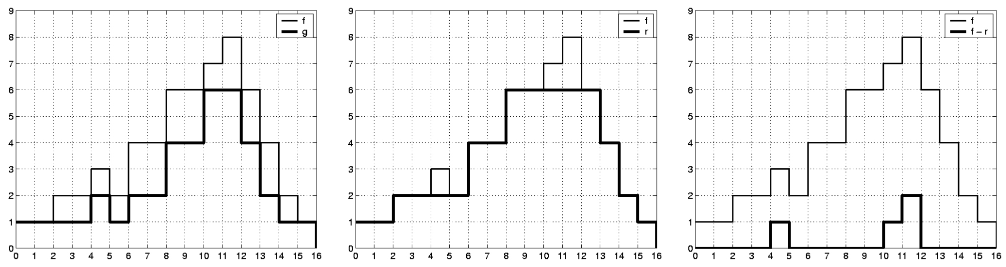
3.2 Definition of the hat-transform scale spaces

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 Eq. (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 $T_{h'}(f)$. Note that this is the n-D analogy to the case when $T_l = T_{l-1}$ for 1-D signals, presented in section 2. 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 Figure 3.2. Notice that this formulation is applicable without any modification to n-D functions. The operator defined in Eq. (2) is neither idempotent nor increasing. The fact that this operator is not idempotent allows it to be iteratively applied on the input signal in order to construct the scale space. Also, it is easy to see that the criterion in Eq. (3) is not increasing, and therefore the operator θ is not increasing either. As shown in [7], when a criterion Q in not increasing, the output (filtered) image may have artificial edges. This is exactly the case for the operator θ, as can be seen in Figure 3.2. This is due to the fact that, although a threshold set may satisfy the criterion Q, sets at lower grey scales may not, and this gives rise to the artificial edges at these grey scales. A possible way to tackle this problem is to process only regional maxima of the image, by descending from high to low grey levels until a threshold set that satisfies criterion Q is found. Then, all threshold sets with grey levels smaller than the grey level of this set are considered to pass the criterion. This is equivalent to imposing an increasing property on Q once the *first* threshold set that satisfies Q is found [7]. This process can be formalized using the notion of grey-scale reconstruction [23], as will be shown next.

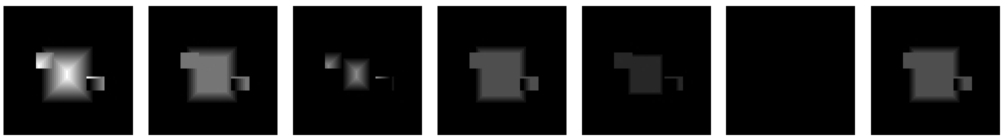
The result of reconstructing f from $g = \theta(f)$ is shown in the middle picture in Figure 3.2, and is denoted by r. Notice that now the desired detail can be extracted (by keeping the residual $f - r$) without introducing any artificial edges.

Finally, a version of the top-hat transform as a connected operator is obtained by retaining the residual of the reconstructed image r, compared to the original image f.

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

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

The top hat scale space can be obtained by *iterating* Eq. (4). This makes sense



(a) (b) (c) (d) (e) (f) (g)

because of the non-idempotent property of the operator defined in Eq. (2).

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*

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

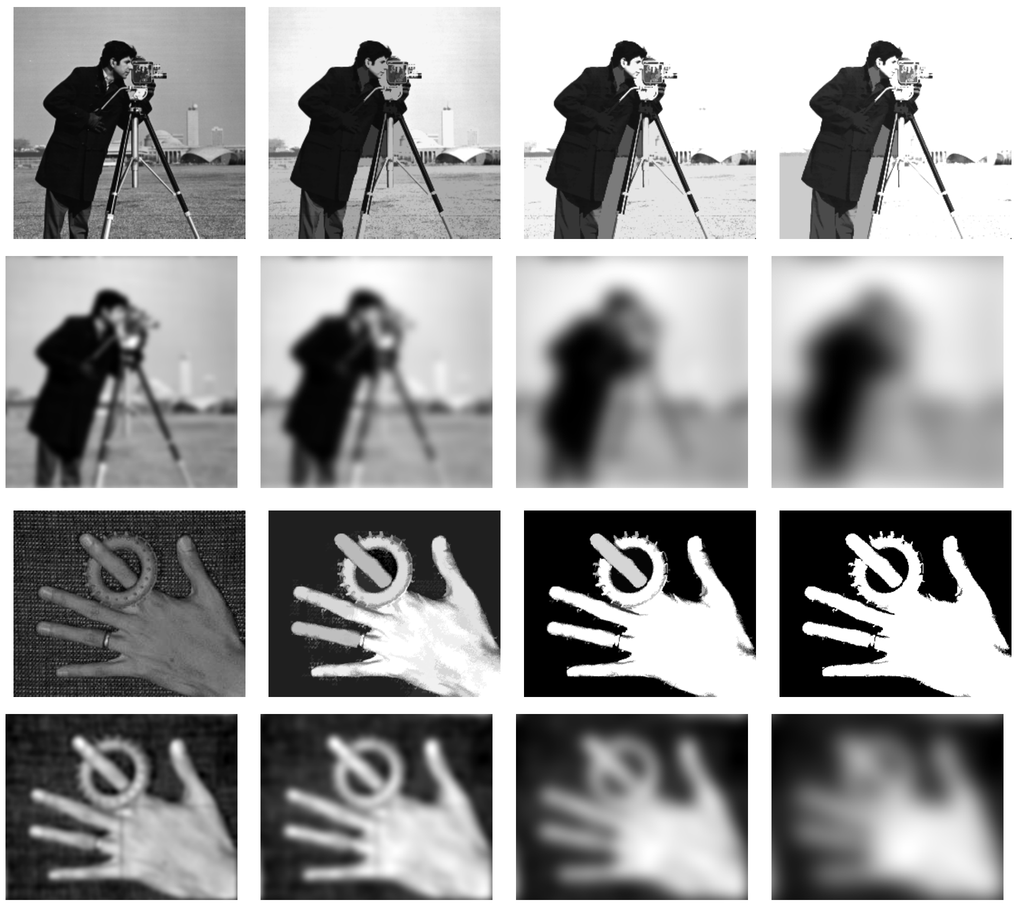
where $f_0 := f$ and $k \geq 0$.

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

It is also possible to formulate a *filtering* process by keeping the result f_k instead of the residual τ_k at each iteration k. An example for a 2-D signal is shown in Figure 3.2. Notice that in the first step the central staircase peak of the signal is removed (see (a)-(c)). In what follows, we will denote by *lowering*, the removal of the detail τ_k from the function f_k, at iteration k.

Additional 2-D filtering examples are shown in Figures 3.2 to 3.2. The filtering results obtained using the hat scale spaces are compared to those produced by Gaussian (linear) scale spaces; for display purposes, all images were enhanced by linear contrast stretching. The results shown in the first figure were obtained using only the top-hat scale space representation (according to Eq. (5)). At each iteration, the image is filtered by removing light-grey peaks. Therefore, when the object(s) of interest in the input image are dark, it is appropriate to use the connected top-hat operator.

The filtering results shown in Figure 3.2 were obtained by alternating connected top-hat and bottom-hat operators. That is, at each iteration k, the image f_k is first filtered by a top-hat operator and then the result is filtered by a bottom-hat operator. The central objects shown in the input image were preserved and can easily be segmented. Alternating top-hat and bottom-hat filters should be performed when no information about the distribution of grey levels in the input image is available a priori.



An important advantage of the hat-transform scale spaces over linear diffusion, as filtering tools, is that they constitute connected operators, hence they do not introduce new contours. Although nonlinear anisotropic diffusion [24] overcomes this problem, the choice of the diffusivity function is not always obvious. Contrary to anisotropic diffusion, the hat-transform scale spaces are parameter-free. Other advantages of using this representation for pattern recognition tasks are discussed in the next sections.

4 Constructing n-dimensional hat scale spaces

It is common to represent a grey-scale image by its level components (connected components, in the binary case). One particular class of methods associates to an image its *tree representation*, where each node in the tree corresponds to a connected component in the binary case, and to a peak or level component in the grey-scale case. Two methods of representing images as trees are widely found in the literature. The first one, called the max-tree representation, was introduced by Salembier *et al.* [25] as a versatile data structure for anti-extensive connected operators. A similar method of representing a grey-scale image is by its component tree, introduced by Jones in [26]. The component tree contains information about each level component and the links that exist between components at sequential grey-levels in the image. A shortcoming of component trees is that the algorithm used to construct them has a worst case running time of $O(N^2 \log N)$, where N is the number of pixels. In [20], the authors proposed an efficient method for implementing connected set openings (see Definition 2) that can be extended to attribute operators [7], based on Tarjan's Union-Find algorithm. Although their method seems to be more efficient than Salembier's, it does not construct a tree representation of the image, which makes it less versatile for filtering tasks.

The construction of the hat scale spaces in two or more dimensions relies on a modified version of Salembier's max-tree data structure, which can be constructed in linear time. The max-tree is a rooted tree, i.e., each *node* has a pointer towards its parent node. In our implementation, we have modified this representation such that it permits bidirectional traversal. Each max-tree node contains two pointers, a *Parent* pointer which allows traversal of the tree from a leaf towards the root, and a pointer to pointer *Children* which allows traversal from the root towards leaves. The node structure also contains: (i) *Level* - the grey level of the peak component represented by the node; (ii) *Features* - a pointer to a feature structure; (iii) *noHighNbr* (no high neighbour) - a boolean value; its use will be explained later in this section. The *Features* structure contains variables needed to compute the extracted features.

Once the max-tree is built, it can be used for processing of the input image, since the tree is its representation. For tasks of filtering, this is a three-step process: construction of the max-tree, criterion assessment and decision, and image restitution. In other words, after the max-tree is created, the filtering step analyzes each node by evaluating a specific criterion and takes a decision on the elimination or preservation of the node. The last step, called here restitution, transforms the filtered max-tree into an output image. For image analysis purposes, this step is not necessary. In this case the max-tree is used only to provide input for a higher abstraction level process, i.e. for a classifier.

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

- *a* leaf*, a component that does not have any child at levels greater than h (i.e. it is a regional maximum M_h^k at level h);*
- *a* simple node*, a component that has exactly one child at a level greater than h;*
- *a* compound node*, a component that has more than one child at levels greater than h;*

One can construct the top-hat scale space (see Eq. (5)) from simple and compound nodes of the max-tree in the following recursive manner. 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 Eq. (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 Eq. (3) holds. The variable $noHighNbr$ indicates whether 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.

The scale space is built in a second step, after the construction of the tree. Because it is a recursive procedure, which resembles *breadth first search*, every node in the tree is processed at most two times, which is linear in the number of nodes. Some advantages emerge from such representation: (i) a small number of scale space entries, compared with the number of peak components; (ii) all these scales are important because some major changes in the topology of the signal occur at these scales; there should be some *edges* separating the nested regions within the parent compound component; (iii) once some entries in the scale space are obtained, they can be characterized by computing not only some *shape* and/or *size* features, but also some features related to the 'height' of an entire branch. In our opinion, this is one of the main advantages of this representation, when compared to *flat filters*. In this representation, one can access and utilize linking between components at sequential grey levels in the signal (image).

By duality, one can construct a min-tree, as explained in [25], and use it in the same manner to construct a bottom-hat scale space. Notice that these representations can be extended to n dimensions by defining the associated connectivity and building the max/min-trees. An advantage of using max/min-trees is that they can handle non-increasing criteria [25], such as that in Eq. (3).

4.1 Scale space features

The selection of scale-space features was guided by the following criteria.

Computational efficiency. The extracted features should be efficient to compute and store. This is especially important for real-time recognition.

Classification performance. The feature set which achieves a higher classification performance is preferred.

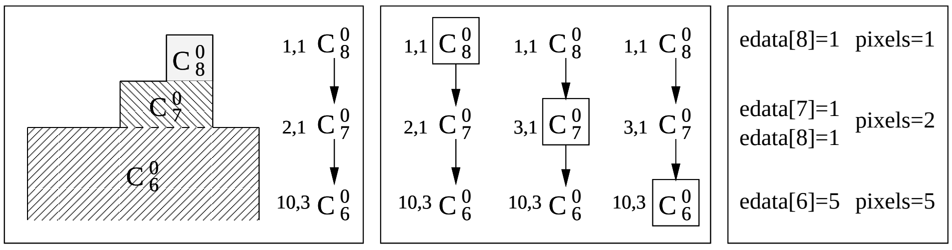
Along with size and shape region descriptors, also descriptors related to the grey-level distribution of the extracted peaks should be included. Some of these descriptors are: *area*, *compactness* (perimeter squared divided by 4π times the area), *complexity* (ratio of perimeter and area), I/A^2 (moment of inertia divided by the square of the area, a strict shape criterion), as region descriptors [27]; *average height* and *entropy*, as descriptors of the distribution of grey levels. Also, as alternatives to the region descriptors given above, moment invariants [28], affine moment invariants [29] and the shape descriptors in [30] were considered. Although these region descriptors satisfy the first criterion (i.e. they can be computed efficiently), the classification results (not shown here) were worse than those obtained with the selected region descriptors, therefore they were not included in the final set of descriptors.

The computation of all features is based on auxiliary data sets maintained for each peak component in the nodes of the tree. The auxiliary data sets can be updated when a new pixel, which belongs to a peak component, is found. They can be merged with other auxiliary data sets of child components, and permit efficient computation of the desired features. Most features can be computed incrementally ('on the fly'). For the perimeter computation a remark is in order. For each pixel p of a node C_h^k, we compute the number of edge-pixels (pixels that have at least one neighbour that does not belong to P_h^k) lower and higher than h, using 4-adjacency. If we denote by $\#L(p)$ and $\#H(p)$ the number of edge-pixels lower, resp. higher, than h, the perimeter of P_h^k is given by

$$Perimeter_{P_h^k} = \sum_{p \in C_h^k} (\#L(p) - \#H(p)) + \sum_i Perimeter_{P_{l_i}^k} \tag{6}$$

where $Perimeter_{P_{l_i}^k}$ is the perimeter of child component $P_{l_i}^k$.

Unlike the area of a set, some features (average height and entropy) cannot be computed 'on the fly' when the max-tree is built. Nevertheless, they can be efficiently computed in a second step after the tree is built, as shown in the next section. In this step also the entries in the scale space are populated, according to the criteria to be discussed in section 4.



4.2 2-D hat scale-space implementation

In this subsection, we present the pseudo-code of the recursive procedure based on Salembier's max-tree which builds the scale space in the 2-D case (see Algorithm 4.2). Unlike the features mentioned in section 4.1, the average height and entropy features cannot be computed incrementally during the construction of the max-tree. However, during tree construction some necessary values needed for computing them are stored in the member variables $Average$ and $Pixels$ of structure *Features*, for each node of the tree. The variable $Average$ holds for each node

n the difference in grey level between the node and its parent, multiplied by the area of the node. The variable $Pixels$ is set to the number of pixels of the component represented by the node n. The difference between $Pixels$ and the area of a node is that the area is incremented by areas of its children, while $Pixels$ is not. A computation example is shown in Figure 4.1. Without loss of generality, let us consider a 1-D signal obtained by thresholding the signal f shown in Figure 3.2 at grey-level 4. In this case, the max-tree contains only three nodes, those from grey-level 6 to 8. The values of both variables $Average$ and $Pixels$ are shown at the left of each node in the tree. Notice that only one entry in the scale space (for the branch which starts with the node C_7^0) should be generated.

The function *HatScaleSpace*, shown in Algorithm 4.2, 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. Notice that all these variables must in fact be references or pointers to the specified types.

```
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 at some node n (see Figure 4.1), the variable $Average$ is updated such that it contains the sum of all $Average$ values of an entire branch of the tree starting with the child c (line 3). The test in line 4 is true when one of the cases specified in Definition 7 holds for the node n. If the test is true, a new entry in the scale space should be added (line 12). In this case, the

$Average$ value of the child c is subtracted from the value of its parent (line 7); this implements the *lowering* of the branch starting at c, and is shown as a dashed-line in the left-side of Figure 4.1. Referring to Figure 4.1, notice that when the current node n is C_6^0 a new entry which corresponds to the branch starting with C_7^0 is to be added, and although the $Average$ value of node C_6^0 was 13, this value is updated to 10. After computing the entropy of the grey-level distribution of the entry (lines $8-10$), and resetting the variables $edata$ and $pixels$ (see right-side of Figure 4.1), 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 (see Figure 4.1). When the test in line 13 evaluates to false (when n is C_8^0 in our example), 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 [25]) and using the same procedure as in Algorithm 4.2. Because each node in the tree is visited at most two times 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.

4.3 Feature normalization

All the extracted features should be normalized before cluster analysis is performed. The area, perimeter and average height of each entry in the scale space are normalized by dividing them by the corresponding maximal values of the components found in the scale space. The compactness and I/A^2 features are inverted. In this way all features are brought in the range $[0...1]$. All shape and size features are therefore translation, rotation and scale invariant. Moreover, the entropy and average height are invariant under linear contrast changes.

Direct use of scale-space features as pattern vectors for classification purposes poses several problems. The first is that the scale space may contain spurious detail caused by noise. To solve this problem, an area opening filter is used to remove all features with areas smaller than a threshold (9 pixels). Furthermore, the pattern vectors of different images would differ in length, which is a problem for many statistical methods. One way to solve this is to set the boundaries between classes of scale-space features from the data themselves. This is done by cluster analysis. Because no assumptions about the number of clusters or the shape of the distribution should be made *a priori*, an unsupervised clustering method will be used.

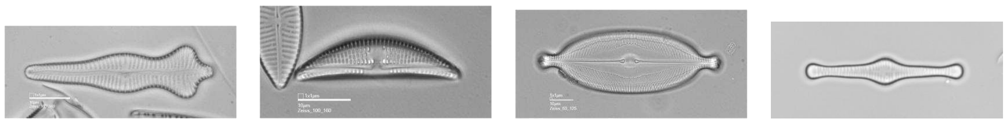
5 Experimental results

In this section we describe the data sets used for experiments, the preprocessing steps of the input signals, the construction of the pattern vectors, and report classification results.

In our classification experiments we have used the C4.5 algorithm [31] for constructing decision trees, with bagging [32] as a method of improving the accuracy of the classifier. The performance was evaluated using the *holdout* [33] method.

5.1 Data sets

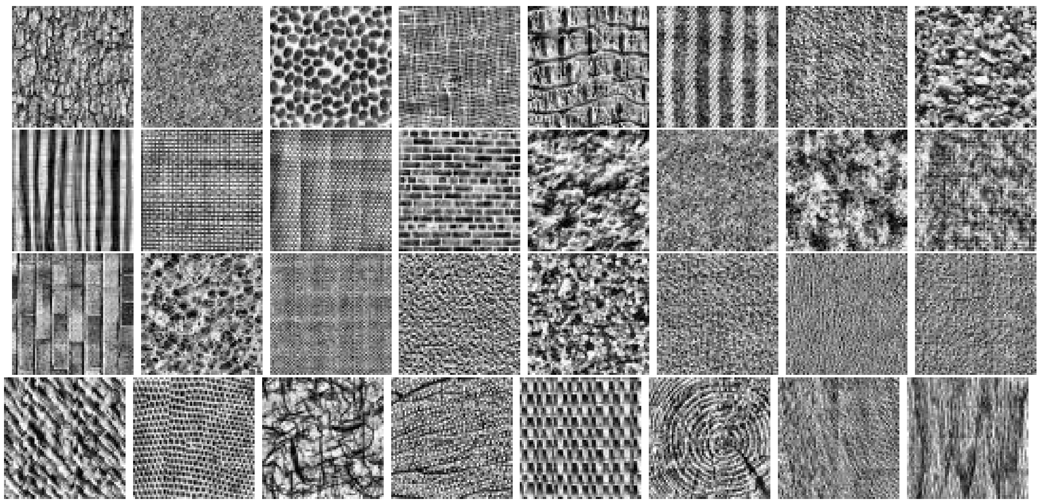
In the experiments, we have used two sets of data, one of them with prominent texture features, and the other with salient shape features. The first set consists of 781 natural images of diatoms. Diatoms are microscopic, single-celled algae, which build highly ornate silica shells or frustules. Each image represents a single shell of a diatom (see Figure 5.1), and each diatom image is accompanied by the outline of its view. For more details, we refer to [13], which contains the results of the Automatic Diatom Identification and Classification (ADIAC) project, aimed at automating the process of diatom identification by digital image analysis. Thus,



both 1-D and 2-D methods described can be used for feature extraction. In the first case, the curvature of the outline (contour) represents the 1-D input signal, and in the last case, the image of the shell itself with its ornamentation is used as input. This set, which we refer to as the *diatom data set*, consists of 37 different taxa of diatoms, and each taxon (class in the pattern recognition sense) has at least 20 representatives.

The second set is obtained from 32 Brodatz textures as shown in Figure 5.1. Each image, which is 256x256 pixels in size, and has 256 grey levels, has been divided in 16 disjoint squares of size 64x64. Each texture sample was transformed, resulting in three additional samples: (i) a sample rotated by 90 degrees, (ii) a 64x64 scaled sample obtained from 45x45 pixels in the middle of the original sample, and (iii) a sample which is both rotated and scaled. The entire data set, which we refer to as *Brodatz data set*, comprises 2048 samples, with 64 representatives for each of the 32 texture categories [34]. For this set we have applied only the 2-D feature extraction method.

[0] Source: `http://www.ee.oulu.fi/research/imag/texture/image_data.`



5.2 Preprocessing

Although the curvature of a curve is invariant under planar rotation and translation [35], it is not invariant under a change in scale. Therefore, various methods to achieve scale invariance of the curvature have been proposed in the literature. These include equal-arclength sampling [36], equal-angle sampling [37] and equal-points sampling [37]. Among these sampling methods, the equal-arclength sampling method apparently achieves the best equal space effect [35]. However, since the diatom database contains objects (diatoms) of different sizes, we first rescale every input contour to the contour with the minimum bounding box among the contours of the diatoms within the same class as the diatom whose contour is given as input; the same scaling factors are also used to scale the diatom image itself. Secondly, 100 points are selected using the equal-points sampling method. If no a priori knowledge about the appropriate scale to be used is available, a multi-scale approach similar to that in [38] may be used.

In the 2-D case, before constructing the scale space(s), an area *open-close* filter with size $\lambda = 49$ is applied. The purpose of this filtering is twofold: noise reduction, and merging of small peak components affected by noise.

5.3 Extraction of the pattern vectors

In the 1-D case, the curvature of each input contour is computed at two different scales (the contour is smoothed with Gaussian kernels of widths $\sigma = 3.0$ and $\sigma = 10.0$). The smaller scale parameter was determined empirically such that the loss of information is small, while unimportant details due to noise are removed. The larger scale was selected such that only salient information about the contour is retained. After both top and bottom hat scale spaces are built at the first scale, each extracted peak is described by its maximum height, average height and extent.

Then, the largest 15 maximum heights are selected, and the process is repeated for the second scale.

Since curvature is a purely local attribute, we supplemented the pattern vector by three global shape descriptors for each scale: *circularity*, *eccentricity* and *bending energy*. Note that similar global shape descriptors were used in conjunction with the Curvature Scale Space (CSS) descriptors in [39–41]. Therefore, the pattern vector in the 1-D case is given by 36 ($2 \cdot (15 + 3)$) numbers. In this case no data reduction (i.e. clustering) is necessary, since in most cases, even at the smaller scale, the peaks which remain after the largest 15 heights are selected have very small maximum heights and can be neglected.

One can choose either to use only one (top or bottom) hat scale space, or to use both of them, depending on the image content. In our experiments we use two data sets. The first set contains natural images of diatom shells in prominent light-grey (see section 5.1). When we use both scale-space representations for the diatom data set, no increase in the classification performance is obtained (results not shown). This is because most diatom images show symmetrical light and dark grey stria patterns of their silica shells, and hence the information extracted from both scale spaces is highly redundant. Hence only one representation is used for this data set. For the Brodatz texture data both scale spaces are used. In general, without any prior knowledge about the image content, both scale-space representations should be used for feature extraction. In the 2-D case, first a cluster analysis is performed on the features by Fukunaga's mean-shift algorithm [42]. The final pattern vector is constructed as follows: (i) for the top scale space we select the first six clusters containing the scale-space features with the largest areas; (ii) for top and bottom scale spaces we select the first three clusters for the top scale space and the first three clusters for the bottom scale space with the largest areas. All these clusters are represented by their centroids. Hence, the size of the pattern vectors is 36 ($6 \cdot 6$, or $2 \cdot 3 \cdot 6$) in both cases. The number of clusters whose centroids are selected as pattern vectors was determined empirically, and represents the optimal choice of the total number of features with respect to identification performance (see section 5.5).

5.4 Classification technique

A decision tree is an example of a multistage decision process. Instead of using the complete set of features jointly to make a decision (as performed by neural networks or statistical classifiers), features are considered one by one, resulting in a sequence of binary decisions. The tree is usually constructed top-down, beginning at the root node, and successively partitioning the feature space. The C4.5 algorithm we have used splits the training set into subsets by choosing the feature that maximizes the information gain [31].

The procedure employed for constructing the bagging predictors resembles that in [32], and is as follows:

- The data set is split into a training set and a test set, such that the test set contains exactly five samples of each class. Then, the size of the test set becomes 25 % of the original set;
- 25 new training sets are constructed using bootstrapping from the initial training set, and a decision tree is built for each of them;
- All 25 decision tree classifiers are evaluated on the test set, and a majority vote is taken on the outcome of each tree;
- All the above steps are repeated 10 times, and the results are averaged.

In the third step of the bagging procedure, all 25 decision tree classifiers are evaluated on the test set, using the holdout method of accuracy estimation. This method is briefly described next.

Let $X = V \times Y$ be the space of labelled instances and $D = \{x_1, x_2, \ldots, x_n\}$ be a dataset consisting of n labelled instances, where $x_i = \langle v_i \in V, y_i \in Y \rangle$; V denotes the space of unlabeled instances and Y the set of possible labels. A *classifier* C maps an unlabeled instance $v \in V$ to a label $y \in Y$ and an *inducer* I maps a given dataset D into a classifier C. This is called *training* of the classifier. Then, $I(D, v) = (I(D))(v)$ denotes the label assigned to an unlabeled instance v by the classifier built by inducer I on a dataset D. Let D_h, the holdout (test) set, be a subset of size h of D, and let D_t, the training set, be $D \setminus D_h$. The holdout estimated accuracy is defined as

$$acc_H = \frac{1}{h} \sum_{\langle v_i, y_i \rangle \in D_h} Match(I(D_t, v_i), y_i), \tag{7}$$

where the binary function $Match(i, j) = 1$ if $i = j$ and 0 otherwise. The *identification performance* is defined as the average of the holdout accuracies over all runs.

5.5 *Identification performance*

Table 5.5 and 5.5 show the identification performances for both data sets using the C4.5 decision tree classifier, with bagging. The identification performance for the diatom data set (Table 5.5) is computed using (i) contour features only (curvature scale space, 36 features), (ii) texture features only (hat scale spaces, 36 features), and (iii) a combination of these. The column '$\bar{x}$' contains the average number of errors; the column 'σ' contains the standard deviation of the number of errors; the columns 'min' and 'max' contain the minimum and maximum number of errors, respectively; the column 'performance' contains the percentage (average with standard deviation about the mean) of samples identified correctly.

Feature set	$\bar{x}$	σ	min	max	performance (%)
contour features (36)	3.2	2.7	0	8	98.3 ± 0.8
texture features (36)	23.5	4.3	15	32	88.1 ± 1.3
combined features (72)	2.2	1.5	0	6	99.6 ± 0.7
MPEG7 descriptor (136)	18.1	3.8	11	26	89.2 ± 1.2
structure tensor (120)	59.2	6.7	35	63	60.4 ± 2.1
combined features (256)	16.5	3.5	10	27	90.5 ± 1.1

It can be seen that for this data set, by combining contour-based features and texture features, the performance reaches almost 100 %. Notice that the standard deviation for the combined feature set decreases to 1.5.

We also carried out a comparison of the morphological hat-transform scale spaces with respect to other scale-space methods, in terms of identification performance. We selected for comparison the curvature scale space (CSS) representation [39, 40, 43], as a contour descriptor, and a texture descriptor [44, 45] based on the the structure (or windowed second moment) tensor [46, 47]. The CSS descriptor is the standard MPEG7 contour-based shape descriptor [41], on which our implementation is based. The classification results using the MPEG7 descriptors, as shown in Table 5.5, were obtained using the same preprocessed contours (see section 5.2) as used for the hat scale spaces; we also tested the preprocessing used in [39, 40], but the results were worse. It can be seen that the result obtained using the hat scale spaces is 9 % higher than that obtained by the CSS descriptor in MPEG7.

The second descriptor, the structure tensor texture descriptor [44], consists of three numbers (polarity, texture anisotropy and contrast), computed for each pixel of the input image. The result is a large set of feature vectors on which data reduction must be applied. Using Fukunaga's mean-shift clustering, the pattern vectors were obtained as the 40 centroids of the most populated (representative) clusters. The result, as shown in Table 5.5, is worse than that obtained by the hat scale space method. Note also that the sizes of the pattern vectors computed by both the MPEG7 and structure tensor descriptors are much larger than those corresponding to the hat scale space descriptors.

The results for the Brodatz data set, using only the texture features extracted from top and bottom hat scale spaces, and those based on the structure tensor, are shown in Table 5.5. As it can be seen, although for this set the result obtained using the structure tensor descriptor is better than that obtained for the diatom set, it is still with 23 % smaller than its counterpart yielded by the hat scale spaces.

Feature set	$\bar{x}$	σ	min	max	performance (%)
texture features (36)	10.3	3.5	7	17	93.5 $\pm$ 1.5
structure tensor (120)	49.5	4.9	32	55	70.1 $\pm$ 1.5

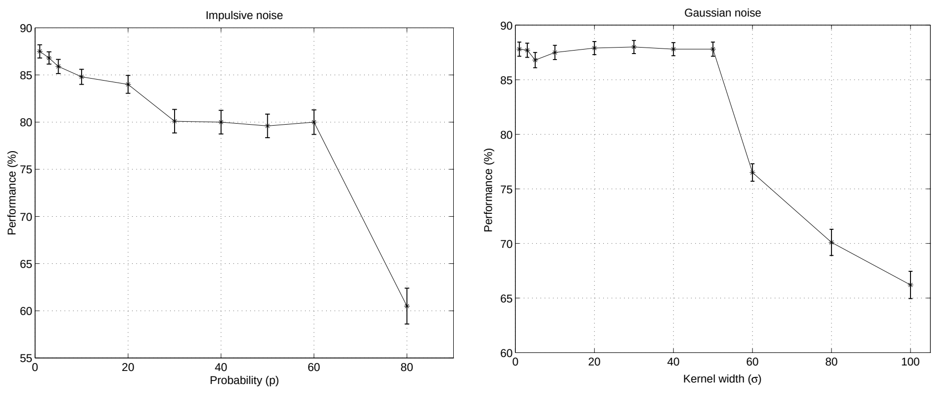
Feature correlation by PCA

Next, we used Principal Component Analysis (PCA) to study how correlated the extracted features are. PCA can be used to remove redundant data, at the expense of some loss of information [48]. In our experiments, PCA was applied: (i) on the features extracted from each image, (ii) on each pattern vector, representing a given image, and (iii) on the whole set of patter vectors; we refer to section 5.3 for computation of the pattern vectors. In each of the cases enumerated above, we selected a number of (principal) components ranging from two up to the total number of components (in the latter case all information is preserved), and projected the initial vectors on the selected components. In each case, the classification performance after applying PCA was worse than the initial result. This can be explained by the fact that decision tree classifiers intrinsically select features, and this selection may contradict that performed by PCA.

In a further experiment, PCA was applied again on the features extracted from each diatom image, but the initial features were projected on the space spanned by the first two principal components; in most cases these first two components explained more than 72 % of variance within the data. Next, we computed two sets of (Pearson's) correlation coefficients between the features projected on each component. Tests of significance of the correlation coefficients revealed that there is a significant correlation (significance level $\alpha = 0.01$) between the projections on the first component of the complexity and entropy features. Similarly, a significant correlation exists between the projections on the second component of the complexity and compactness features. Therefore, we repeated the classification experiment after eliminating the complexity feature. In this case, the classification performance was 2 % smaller than that obtained with the initial feature set; hence, using the initial pattern vector is in our opinion justified.

5.6 Coping with noise

A further experiment was carried out to test the behavior of the feature extraction method in the presence of noise. The two types of noise used to corrupt the images of the diatom data set were additive Gaussian noise and impulsive, or "salt and pepper" noise. The additive noise had a zero-mean Gaussian distribution, with standard deviation (kernel width) σ. That is, each pixel in the corrupted images was the sum of the original pixel value and a random, Gaussian distributed noise value. The "salt and pepper" noise was generated by choosing a probability p to perturb each pixel, and then setting the pixels to be perturbed to random brightnesses of either 0 or 255.



The experiment was carried out for the diatom data set using only the texture features extracted from the top-hat scale space. The results are shown in Figure 5.6. In both graphs, the points represent (averaged) identification performances, as obtained using the bagging procedure, and the error lines represent the standard deviation of the identification performances (in 10 runs) around the mean (see section 5.5). In the presence of impulsive noise, the identification performance did not drop more than 8 %, even when as much as 60 % of the pixels of the input images were corrupted by noise. When Gaussian noise with standard deviation as large as 50.0 was added, the performance remained almost constant around its maximum value. In both cases, only for very large amounts of noise (i.e. more than 60 % of pixels corrupted by impulsive noise, and standard deviations of the Gaussian distribution greater than 50.0), the identification performance dropped dramatically. Therefore, we conclude that the method performs quite well, although just a very simple noise-reduction filter has been used (i.e. area open-close filter). It is likely that more complex noise reduction schemes would further improve the identification performances.

5.7 Comparison to other methods

In [34], the authors proposed a method for texture classification using reduced co-occurrence histograms. Their method uses linear compression, dimension optimization, and vector quantization. A genetic algorithm is used to minimize the computationally expensive leave-one-out classification error. The experiments showed that multidimensional histograms reduced with their method provided higher classification accuracies than those produced using channel histograms and wavelet packet signatures. For detailed explanations of the methods and an ample discussion of the results we refer to [34]. The classification results, taken from [34], are shown in Table 5.7.

Although a direct comparison cannot be performed, due to the different classifiers

Method	performance (%)
Reduced multidimensional histograms	
of DCT coefficients with TSOM	93.9
of DCT coefficients with QVQ	93.4
of mean-removed grey levels with TSOM	92.8
Multidimensional channel histograms	90.4
Wavelet packets	85.1
One-dimensional channel histograms	78.2

involved, the results shown in Table 5.5 are similar to those produced by reduced multidimensional histograms, shown in Table 5.7. We have used bagging in order to improve the holdout estimate of accuracy, while in [34] 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. Note however, that the methods in [34] were tailored towards texture classification, while we have shown that our method can also handle other types of classification problems, e.g. automatic identification of diatoms [12].

6 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 both contour and ornamentation (texture) feature extraction. The sets of feature vectors were used in two classification experiments, using decision trees with bagging. The classification performance for the diatom data set was almost 100 %, and 93.5 % for the Brodatz data set. The first result is among the best obtained in the ADIAC project [13], while the second 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 access and utilize linking between components at sequential grey levels in the signal. Based on this reasoning, we conclude that these representations can be successfully used for image filtering, and we suspect that these representations can readily be adapted for image segmentation.

An important advantage of the 1-D hat scale spaces (used on the curvature signal) as compared to the CSS [39–41] method, is that the extracted features (maximum heights of the peaks of the signal) are not localized along the contour. The descriptors extracted from the CSS representation explicitly use positions along the contour, which means that they are not readily invariant under planar rotation and mirroring, and this results in additional computational overhead needed to match two shapes by the corresponding sets of descriptors (see [40]). Also, all convex objects are identical for the CSS set of descriptors, since it is based on inflection points, and there are none on the contour of a convex object. This implies that a triangle, square, or circle cannot be distinguished using these descriptors [49]. This is not the case for our descriptor, since we extract information about both convexities and concavities. Finally, the MPEG7 descriptor (based on CSS) yielded a worse classification performance than that obtained using the 1-D hat scale spaces. Also, the hat scale space representations can be extended to arbitrary dimensions, while the CSS is defined for 1-D signals only.

Another advantage of the 1-D hat scale spaces over CSS is that they are faster to compute. The computational complexity of constructing the CSS representation is $O(L \cdot N)$, where N is the number of contour points and L is the number of scales. As shown in this paper, the hat scale spaces can be constructed in linear time, i.e. $O(N)$, for arbitrary dimensions, and the number of pattern vectors (scale spaces entries) is usually two orders of magnitude smaller than the size of the input image. The CPU time spent to construct either of the hat scale spaces, for an image of 1000×800 pixels, on a Pentium III at 670 MHz, is under one second.

In conclusion, morphological hat scale spaces can successfully be used in pattern classification tasks, are efficient to compute, and yield very good results.

Acknowledgements

We wish to thank Dr. Kimmo Valkealahti from Nokia Research Center and Professor Erkki Oja from Helsinki University of Technology for providing the images of the Brodatz data sets.

References

[1] D. Marr, Vision, Freeman, San Francisco, 1982.

[2] D. Marr, S. Ullman, T. Poggio, Bandpass channels, zero-crossings, and early visual information processing, J. Optical Soc. Am. 69 (1979) 914–916.

[3] A. P. Witkin, Scale-space filtering, in: Proc. Int. Joint Conf. Artificial Intell., Palo Alto, CA, 1983, pp. 1019–1022.

[4] T. Lindeberg, Scale-space theory: A basic tool for analysing structures ar different scales, Journal of Applied Statistics 21 (1994) 225–270.

[5] T. Koller, G. S. G. Grieg, D. Dettwiler, Multiscale detection of curvilinear structures in 2d and 3d image data, in: Fifth International Conference on Computer Vision, Cambridge, MA, 1995, pp. 864–869.

[6] J. Goutsias, H. J. A. M. Heijmans, Multiresolution signal decomposition schemes. Part 1: Linear and morphological pyramids, IEEE Trans. Image Processing 9 (11) (2000) 1862–1876.

[7] E. J. Breen, R. Jones, Attribute openings, thinnings and granulometries, Computer Vision and Image Understanding 64 (3) (1996) 377–389.

[8] P. F. M. Nacken, Chamfer metrics, the medial axis and mathematical morphology, Journal of Mathematical Imaging and Vision 6 (1996) 235–248.

[9] J. A. Bangham, P. D. Ling, R. Harvey, Scale-space from nonlinear filters, IEEE Trans. Pattern Anal. Machine Intell. 18 (1996) 520–528.

[10] J. A. Bangham, R. Harvey, P. D. Ling, R. V. Aldridge, Morphological scale-space preserving transforms in many dimensions, Journal of Electronic Imaging 5 (1996) 283–299.

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

[12] M. H. F. Wilkinson, A. C. Jalba, E. R. Urbach, J. B. T. M. Roerdink, Identification by mathematical morphology, in: J. M. H. Du Buf, M. M. Bayer (Eds.), Automatic Diatom Identification, Vol. 51 of Series in Machine Perception and Artificial Intelligence, World Scientific Publishing Co. , Singapore, 2002, Ch. 11, pp. 221–244.

[13] H. du Buf, M. M. Bayer (Eds.), Automatic Diatom Identification, World Scientific Publishing, Singapore, 2002.

[14] H. J. A. M. Heijmans, Morphological Image Operators, Vol. 25 of Advances in Electronics and Electron Physics, Supplement, Academic Press, New York, 1994.

[15] J. Serra, Image Analysis and Mathematical Morphology, Academic Press, New York, 1982.

[16] P. T. Jackway, M. Deriche, Scale-space properties of the multiscale morphological dilation-erosion, IEEE Trans. Pattern Anal. Machine Intell. 18 (1996) 38–51.

[17] M. H. Chen, P. F. Yan, A multiscale approach based on morphological filtering, IEEE Trans. Pattern Anal. Machine Intell. 11 (1989) 694–700.

[18] K.-R. Park, C.-N. Lee, Scale-space using mathematical morphology, IEEE Trans. Pattern Anal. Machine Intell. 18 (1996) 1121–1126.

[19] J. Gil, M. Werman, Computing 2-D min, median, and max filters, IEEE Trans. Pattern Anal. Machine Intell. 15 (1993) 504–507.

[20] A. Meijster, M. H. F. Wilkinson, A comparison of algorithms for connected set openings and closings, IEEE Trans. Pattern Anal. Machine Intell. 24 (4) (2002) 484–494.

[21] P. Salembier, J. Serra, Flat zones filtering, connected operators, and filters by reconstruction, IEEE Trans. Image Processing 4 (1995) 1153–1160.

[22] H. J. A. M. Heijmans, Connected morphological operators for binary images, Comput. Vis. Image Understand. 73 (1999) 99–120.

[23] L. Vincent, Morphological grayscale reconstruction in image analysis: application and efficient algorithm, IEEE Trans. Image Processing 2 (1993) 176–201.

[24] J. A. Weickert, Anisotropic Diffusion in Image Processing, Teubner, Stuttgart, 1998.

[25] P. Salembier, A. Oliveras, L. Garrido, Anti-extensive connected operators for image and sequence processing, IEEE Trans. Image Processing 7 (1998) 555–570.

[26] R. Jones, Connected filtering and segmentation using component trees, Computer Vision and Image Understanding 75 (1999) 215–228.

[27] A. K. Jain, Fundamentals of Digital Image Processing, Prentice Hall, Englewood Cliffs, NJ, 1989.

[28] M. K. Hu, Visual pattern recognition by moment invariants, IEEE Trans. Inf. Th. 8 (1962) 179–187.

[29] J. Flusser, T. Suk, Pattern recognition by affine moment invariants, Pattern Recognition 26 (1993) 167–174.

[30] P. L. Rosin, Measuring shape: Ellipticity, rectangularity, and triangularity, in: Proc. 15th Intern. Conf. on Pattern Recognition (ICPR'2000), Barcelona, Spain, Sep. 3-7, 2000, pp. 1952–1995.

[31] J. R. Quinlan, C4. 5: Programs for Machine Learning, Morgan Kaufmann Publishers, 1993.

[32] L. Breiman, Bagging predictors, Machine Learning 24(2) (1996) 123–140.

[33] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: International Joint Conference on Artificial Intelligence, San Mateo, CA, 1995, pp. 1137–1145.

[34] K. Valkealahti, E. Oja, Reduced multidimensional co-occurrence histograms in texture classification, IEEE Trans. Pattern Anal. Machine Intell. 20 (1998) 90–94.

[35] M. P. do Carmo, Differential Geometry of Curves and Surfaces, Prentice Hall, New York, 1976.

[36] F. Mokhtarian, S. Abbasi, J. Kittler, Robust and efficient shape indexing through curvature scale space, in: Proc. British Machine Vision Conference, Edinburgh, UK, 1996, pp. 53–62.

[37] P. J. van Otterloo, A Contour-Oriented Approach to Shape Analysis, Prentice Hall, Hemel Hampstead, 1992.

[38] P. J. Burr, Smart sensing within a pyramid vision machine, in: Proc. of the IEEE, 1988, pp. 1006–1015.

[39] F. Mokhtarian, A. K. Mackworth, Scale-based description and recognition of planar curves and two-dimensional shapes, IEEE Trans. Pattern Anal. Machine Intell. 8 (1986) 34–43.

[40] F. Mokhtarian, S. Abbasi, J. Kittler, Efficient and robust retrieval by shape content through curvature scale space, in: A. W. M. Smeulders, R. Jain (Eds.), Image DataBases and Multi-Media Search, World Scientific Publishing, 1997, pp. 51–58.

[41] M. Bober, MPEG-7 visual shape descriptors, IEEE Trans. on Circ. Syst. Vid. Tech. 11 (2001) 716–719.

[42] K. Fukunaga, L. D. Hostetler, Estimation of the gradient of a density function with applications in pattern recognition, IEEE Trans. Inf. Th. 21 (1975) 32–40.

[43] F. Mokhtarian, A. K. Mackworth, A theory of multiscale, curvature-based shape representation for planar curves, IEEE Trans. Pattern Anal. Machine Intell. 14 (1992) 789–805.

[44] C. Carson, M. Thomas, S. Belongie, J. M. Hellerstein, J. Malik, Blobworld: A system for region-based image indexing and retrieval, in: Third International Conference on Visual Information Systems, Springer, 1999.

[45] C. Carson, S. Belongie, H. Greenspan, J. Malik, Blobworld – image segmentation using expectation maximization and its application to image querying, IEEE Trans. Pattern Anal. Machine Intell. 24 (2002) 1026–1038.

[46] J. Garding, T. Lindeberg, Direct computation of shape cues using scale-adapted spatial derivative operators, Int. J. Comp. Vis. 17 (1995) 163–191.

[47] T. Lindeberg, J. Garding, Shape from texture from a multi-scale perspective, in: 4th International Conference on Computer Vision, 1993, pp. 683–691.

[48] I. T. Jolliffe, Principal Component Analysis, Springer-Verlag, New York, 1986.

[49] L. J. Latecki, R. Lakimper, U. Eckhardt, Shape descriptors for nonrigid shapes with a single closed contour, in: Proc. IEEE CVPR, Hilton Head Island, South Carolina, USA, 2000, pp. 1424–1429.

About the author. ANDREI C. JALBA received his B.Sc. (1998) and M.Sc. (1999) in Applied Electronics and Information Engineering from “Politehnica”University of Bucharest, Romania. He is currently a Ph.D. candidate in the research group “Computing and Imaging”, Department of Mathematics and Computing Science, University of Groningen, The Netherlands. His research interests include computer vision, pattern recognition, image processing, and parallel computing.

About the author. JOS B.T.M. ROERDINK received his M.Sc. (1979) in theoretical physics from the University of Nijmegen, the Netherlands, and a Ph.D. (1983) from the University of Utrecht. From 1983-1985 he was a Postdoctoral Fellow at the University of California, San Diego, and from 1986-1992 he worked at the Centre for Mathematics and Computer Science in Amsterdam. He is currently professor of Scientific visualization and Computer Graphics at the University of Groningen, the Netherlands. His research interests include scientific visualization, mathematical morphology, wavelets, and functional brain imaging.

About the author. MICHAEL H. F. WILKINSON received his M.Sc in astronomy in 1992, and his Ph.D. in computing science from the University of Groningen in 1995. He has worked on digital image analysis of microbes and computer simulation studies of microbial ecosystems. He is currently assistant professor at the Institute for Mathematics and Computing Science, University of Groningen, and works on mathematical morphology and computer simulation in biomedical settings.