Automatic Identification of Diatoms using Multi-scale Mathematical Morphology

Erik R. Urbach
Automatic Identification of Diatoms using Multi-scale Mathematical Morphology

Erik R. Urbach

July 18, 2001
Abstract

Diatoms are unicellular algae with a great ecological importance. They are ornamented with patterns that are characteristic for the species they belong to.

Till now these diatoms are identified by experts, which is a time-consuming and tedious process. For this reason and due to a lack of diatomists, the Automatic Diatom Identification And Classification project ADIAC was started, where various techniques from image analysis are investigated for this use. The goal of this research was to investigate the possibilities of the identification of diatoms using mathematical morphology.

Images are considered by this approach as mathematical entities such as sets or functions on which operators are defined. These operators are used to compute pattern spectra that describe the presence or absence of image details with certain characteristics such as size or shape. The idea of this approach is that the patterns on the diatoms can be described by these pattern spectra, so that an identification can be made.

Although more research is needed to tackle some problems, some first results of diatom identification using different methods give reason for optimism about the usability of this method especially in combination with other strategies. The results were obtained on a set of 781 images, consisting of 37 different taxa, with the C4.5 decision tree classifier. This method identified 81.4% of the test set correctly.
## Contents

1 Diatoms
   1.1 Introduction .......................... 5
      1.1.1 Capturing diatom images ........ 5
      1.1.2 Shape and ornamentation ....... 6
   1.2 Identification and classification ... 7
      1.2.1 Identification based on shape ... 8
      1.2.2 Identification based on ornamentation ... 9
      1.2.3 Identification by diatomists ... 10
   1.3 Applications of diatom identification . 11

2 Mathematical morphology .................. 13
   2.1 Binary images .......................... 13
      2.1.1 Binary image transformations ... 13
   2.2 Grayscale images ...................... 16
      2.2.1 Umbrae ............................ 18
      2.2.2 Grayscale dilation and erosion ... 18
      2.2.3 Grayscale thinning and thickening ... 20

3 Connected operators ...................... 21
   3.1 Connectivity .......................... 21
   3.2 Complete lattices ...................... 22
   3.3 Connected operators ................... 23
   3.4 Attribute opening and thinning ....... 23
   3.5 Grayscale attribute opening and thinning ... 25

4 Multi-scale analysis ..................... 27
   4.1 Image analysis ........................ 27
   4.2 Granulometries ....................... 27
   4.3 Shape distributions .................... 28
   4.4 Size and shape transforms .......... 28
   4.5 Pattern spectra ....................... 29

5 The Max-tree ................................ 31
   5.1 A tree representation ................ 31
   5.2 Filtering ................................ 31
   5.3 The Viterbi rule ....................... 32
   5.4 Image restitution ...................... 34
   5.5 Image decomposition ................... 34
6 Algorithm

6.1 Max-tree implementation

6.1.1 Max-tree construction
6.1.2 Max-tree processing
6.1.3 Image restitution
6.1.4 Max-tree analysis

6.2 Attributes

6.2.1 Computing attributes
6.2.2 Perimeter
6.2.3 Moment of inertia
6.2.4 Mappers

6.3 Computing feature vectors

6.3.1 Mapping a pattern spectrum to a feature vector
6.3.2 Combination

7 Identification of diatoms

7.1 Diatom identification using mathematical morphology
7.2 Feature extraction

7.2.1 Choice of criteria
7.2.2 Mapping to a pattern spectrum
7.2.3 Feature vectors

7.3 Statistics

7.3.1 Leave-one-out
7.3.2 Retrieval efficiency
7.3.3 C4.5

7.4 Results

7.4.1 Performance using leave-one-out
7.4.2 Performance using retrieval efficiency
7.4.3 Performance using C4.5

8 Conclusions

8.1 Future research
8.2 Final notes
List of Figures

1.1 Diploneis heemskerckiana ........................................ 5
1.2 A diatom frustule with its valves indicated ................. 6
1.3 Different kinds of microscopes ................................ 7
1.4 The ornamentation patterns ..................................... 8
1.5 Size and shape changes of diatoms within their life-cycle .. 9
1.6 Symmetry axes of a diatom ..................................... 9
1.7 Symmetry characteristics ....................................... 10
1.8 The ornamentation patterns used for manual identification 10

2.1 Example of a binary image ...................................... 13
2.2 Examples of basic image operations ......................... 14
2.3 Examples of the continuous binary dilation and erosion ... 14
2.4 Examples of the discrete binary dilation and erosion ...... 15
2.5 Examples of the continuous binary opening and closing ... 15
2.6 Examples of the discrete binary opening and closing ..... 16
2.7 Example of the discrete binary thinning .................... 17
2.8 Examples of grayscale image operations .................... 18
2.9 Grayscale dilation and erosion using umbrae ............... 19
2.10 Examples of grayscale opening and closing ............... 20
2.11 Example of the grayscale Hit-or-Miss Transform .......... 20

3.1 The 4- and 8-connected neighbors ............................. 21
3.2 Level components ............................................... 22
3.3 Binary image and its associated partition .................... 22
3.4 Influence of connected operators on an associated partition 23

4.1 Example of a size pattern spectrum ........................... 30
4.2 Example of a shape pattern spectrum .......................... 30

5.1 Building a Max-tree ............................................ 32
5.2 Filtering with different rules .................................. 33
5.3 A simple Max-tree and the corresponding trellis ............ 33
5.4 Grayscale shape decomposition using different rules ....... 35

6.1 Example of using two criteria ................................ 39
6.2 Different approximations of the perimeter .................. 42
Diatoms and mathematical morphology

In mathematical morphology the contents of an image is described using mathematical properties. Usually, multiscale mathematical morphology is used to analyze image details at different scales. In this thesis it will be used to identify diatoms based on the size and the shape of their ornamentation patterns.

Diatoms are unicellular algae with a great ecological importance. An example of a diatom is shown in figure 1.1. Till now, diatoms are identified manually, which is a difficult and time consuming task. Therefore, the Automatic Diatom Identification And Classification project, called ADIAC [2] was started. This graduation project was part of ADIAC. The main goal is to investigate if multiscale mathematical morphology can be used for the identification of diatoms based on their ornamentation patterns.

The outline of this thesis is as follows. Chapter 2 describes the concepts of mathematical morphology. An important class of operators from mathematical morphology are the connected operators, which will be explained in chapter 3. How these operators can be used for texture analysis is the topic of chapter 4. A versatile structure for the computation of these operators is given in chapter 5, while chapter 6 describes how they can be used to compute so-called feature vectors. How these feature vectors can be used for the identification of diatoms is described by chapter 7. The performance of these feature vectors in the identification of diatoms is also covered by that chapter. Finally, this thesis ends with some conclusions and remarks in chapter 8. First, a brief introduction to diatoms is given in the next chapter.
Chapter 1
Diatoms

1.1 Introduction

Diatoms (Bacillariophyceae) are unicellular algae with a silica shell, like the one shown in figure 1.1. They are related to yellow-green (Xanthophyta), golden-brown (Chrysophyta) and brown algae (Phaeophyta, e.g. seaweeds like Fucus and Laminaria), but not related to red, green or blue-green algae. They can live wherever there is water or humidity. Some of these are attached to rocks, sand particles or plants, whereas some can move along the bottom and some live in the body of the water. Most diatoms also need sunlight. They contribute to the world’s carbon fixation for about 20%, which is more than the contribution of all the world’s rain forests.

Figure 1.1: Diploneis heemskeriana (Source: ADIAC)

Diatoms vary greatly in size, most of them have a length of 10 to 100\(\mu m\). Although they are individually invisible for the naked eye, large concentrations can give a surface a brown or golden-brown appearance.

The shell of a diatom can be thought of as a pillbox, consisting of two interlocking halves, called the \(\text{thecae}\), one larger than the other. The larger one is called \(\text{epitheca}\) and the smaller one \(\text{hypotheca}\). The main element of a theca is called a \(\text{valve}\). These thecae are shown in figure 1.2. As can be seen, there are several bands attached to these thecae. Such a band is referred to as a \(\text{cingulum}\). The two cingula between the two thecae are called the \(\text{girdle}\). The whole shell, i.e. the valves and the girdle together, are referred to as the \(\text{frustule}\).

1.1.1 Capturing diatom images

Different kinds of microscopes are used to capture diatom images. The following three are in common use:

...
• **Scanning electron microscope (SEM):** This type of microscope gives perspective images of diatoms. These images are useful for visual inspection, but the perspective effects makes identification very hard. An example of a SEM image is shown in figure 1.2.

• **Interference contrast microscope:** Contrary to a SEM, this one gives a flat image of a diatom, but with shading effects.

• **Brightfield light microscope:** Like above, but without shading effects. This is the one commonly used for identification purposes.

Figure 1.3 shows three images of the same diatom taken with each of these microscopes.

When a light microscope is used, the diatoms are prepared by cleaning them with acid, which results in a disintegration of the frustule in its component parts, i.e. the valves and the girdle. Due to their shape, valves usually settle in such a way that the surface of the valve belonging to the upper or lower exterior of the frustule is visible. This is called the *valve view*.

### 1.1.2 Shape and ornamentation

The valves of the diatom shown in figure 1.1 are ornamented with patterns. The presence or absence and the organization of these patterns, called the *ornamentation*, are specific for each species. There are different kinds of these patterns as is illustrated in figure 1.4. In this figure a richly ornamented diatom valve is shown. The elongated feature along the length of the diatom is called a *raphe*. A raphe is fissure through the valve wall. The regular pattern of lines consisting of pores are called the *striae*. The region in the center of the valve where the raphe and the striae are absent is called the *central area*. The area along the raphe without pores is called the *axial area*.

A diatom shape is influenced by its life-cycle. The diatom life-cycle consists in a sexual and an asexual phase. During the asexual phase a diatom multiplies itself by means of cell division. Inside the mother cell, new valves and girdles are formed. Each daughter cell has one valve from its mother cell and a new smaller valve inside the old valve. This means that by each generation the frustules become smaller. When diatoms are too small to divide, they enter the sexual phase by forming themselves to gametes. These gametes fuse together to form an *auxospore*, which is a specialized cell that swells to the maximum size for the species. The auxospore then divides asexually and the asexual phase is entered again.

Besides a reduction in size, a cell division does often also result in a change of the shape of a diatom. Typically, during a cell division the length of a diatom is reduced more than its width, which means that with each cell division, a diatom becomes less elongated. Furthermore, the outlines tends to get smoother. The influence of the life-cycle on a diatom’s shape is illustrated in figure 1.5. Note that a size reduction usually has practically no effect on the ornamentation, i.e. the size and position of the pattern units, e.g. striae, remain the same; a smaller valve has fewer pattern units than a larger one of the same species.
1.2 Identification and classification

In biology a classification is a logical system of categories, where organisms are grouped into a hierarchical structure based on their characteristics. The term identification is used to refer to the process of finding the identity of an object by reference to a classification. Identification is based on the information known at the time, usually data acquired from that sample, such as an image. However, classification is done using all available evidence, such as morphological, molecular, chemical, ultrastructural and behavioral data. In image analysis, the term classification is also used in the sense of identification. To avoid confusion, these terms will be used here according to the definitions in the biological world.

Diatomists use a classification, where the diatoms are separated into taxa. A subdivision of these taxa into more precise terms is usually made. The most global subdivision used here is the genus. Within each genus there are many species and species can be further divided into infraspecific taxa, i.e. subspecies, varieties, forms, morphotypes, etc. Currently, between 10,000 and 20,000 species are known and published. The real number of diatom species is estimated to be in the order of 200,000 or more. The difference between the number of known species and the estimates is partly due to the existence of still unknown species, but it is mainly caused by differences in the interpretation of the species. An example of a problematic group of taxa is the Sellaphora pupula, which has been treated as a single species, but recently it has become clear that many separate taxa are contained in this group. Since there is currently no classification for this group of diatoms, individual diatoms belonging to this group are separated into demes, where a deme is just a group of individual diatoms with given characteristics.

Although there is some morphological variability which gives each individual diatom unique valves, diatoms can be identified using only information about the shape and ornamentation of its valves, that is the valves of all diatoms belonging to the same species share some morphological...
properties. The distinction between the infraspecific ranks is not possible, because that distinction is usually based on biogeographical instead of morphological grounds. Varieties for example are local variants of a species, that are restricted to a small geographical area.

1.2.1 Identification based on shape

The shape of a diatom frustule has many characteristics which can be used for the identification of that diatom. One important character is the number of symmetry axes a valve has. The axis along the length of the frustule is called the *apical axis*, the one along the width is the *transapical axis*, and the *pervalvar axis* is the one along the height of the frustule. These three axes are illustrated in figure 1.6.

Since identification is usually done using images with the diatom in valve view, the pervalvar axis cannot be used. In figure 1.7 examples of diatom valves are shown with different symmetrical properties.

The valves of most of the diatom genera are symmetrical about both axes, i.e. the apical and transapical axes, like the one shown in figure 1.7a. Others are symmetrical about only one axis as in b) and c). Some genera have a rotational symmetrical property d) and finally, a large group of genera are more or less circular, like e).

Usually, for identification purposes, images are used with the diatom captured in valve view. In
that case, the diatom is surrounded by an outline. This outline or contour has many characteristics that are related to the species the diatom belongs to. This outline can be analyzed mathematically. For example, it can be fitted to a polynomial expansion or a measure of rectangularity can be used. In [2] brief examples of shape analysis of the diatom contour are given. A more elaborate example using the curvature of the diatom contour can be found in [21].

1.2.2 Identification based on ornamentation

The ornamentation of the valves has also many features which can be used to identify the diatoms. However, there are important differences between the way diatomists use the ornamentation for identification and how it could be used for automatic identification by a computer. In this section the use of the ornamentation from a diatomist’s point of view will be described. A method for automatic identification of diatoms is the topic of this research project.

An advantage of using the ornamentation instead of the contour of diatoms is that the ornamentation patterns are size insensitive, i.e. the patterns of smaller valves are equal to the larger ones of the same species, except that the smaller ones contain fewer patterns.

The periodicity and the angle of the striae are very characteristic for each species; some species are for example ornamented with simple lines of pores, while others have double rows of pores. Furthermore, the raphe is important for diatomists. They look whether the raphe is absent or present in the diatom and they investigate many small morphological details of this raphe, such as the raphe endings. Figure 1.8 illustrates the use of some of the ornamentation features for manual
identification by diatomists.

1.2.3 Identification by diatomists

Several methods are used by diatomists to identify an unknown diatom, some of these are:

Picture matching The unknown diatom is matched against pictures of known diatoms, i.e. reference pictures, from a reference book. Although this method is very widely used, it is difficult for the casual observer to perform because a considerable understanding of how the variation works is required. Note that no two diatoms of the same taxa are exactly the same.

Using a dichotomous key This is the equivalent of a decision tree in computing terms and it consists of a series of either/or questions, where each question except the first one, depends on the answer of the previous one. A possible question would be: raphe present/raphe

Figure 1.7: Symmetry characteristics: bilaterally symmetrical about two axes (a) or about only one axis (b) and (c), rotational symmetry (d), and circular (e). (Source: ADIAC)

Figure 1.8: The ornamentation patterns used for manual identification (Source: ADIAC)
absent. Although this is an efficient way to identify diatoms, it is vulnerable to mistakes. If a mistake is made early in the identification process, the resulting identification will probably be wrong.

Character matching The characteristics of the valve are used. In the biological world these are also known as characters. The characters of a sample image are determined; the taxon identified is the one that has the best match with these characters. Contrary to the previous method, a mistake does not necessarily lead to a wrong identification. Note that this method is equal to nearest neighbor classification in computer science.

One such approach is called multi-access keys. In this approach, a string of digits is used, where each digit represents one character of the diatom. The value of the digit is provided by the state of that character. Imagine a key, where the second digit from left represents the raphe. In that case, a diatom with key 023581 the raphe is represented by the digit 2, which could for example mean: "raphe running the full length of the valve". The code of an unknown diatom can then be looked up in the keys' index. When one or more digits are wrong, one can still try to identify the diatom using the other digits.

Usually, a combination of picture and character matching is used for identification.

1.3 Applications of diatom identification

Diatoms have narrow ecological preferences. In other words, the presence of a diatom taxon gives much information about the ecological environment it lives in. Since diatoms have a silica shell, their frustules remain intact long after they die. This means that diatoms can also be used for the reconstruction of the environmental conditions of the past or the present, e.g. palaeoclimate reconstruction. Other examples of practical uses of diatoms are:

- They can be used for measuring the pH and the nutrient status in old sediments.
- In archaeology they are studied to trace ceramic and clothing fragments.
- They are used in forensic research to determine the cause of death in drowning cases. The existence of specific diatom taxa in the victim's body can give clues about the place of drowning, i.e. salt or fresh water, temperature of the water, etc.
- As indicators of water quality.

Till now, the identification of diatoms is done manually by diatomists. This has some disadvantages:

- The identification of diatoms is very time-consuming and tedious. It can take hours to analyze a single diatom sample. Consider that large studies often deal with hundreds of samples.
- Currently, diatoms are identified manually. This can introduce personal views on the identification of diatom samples.
- The identification of diatoms requires a good expertise and understanding of the morphological diversity of the taxa. The identification can depend on subtle morphological differences, like the raphe endings, which makes it hard for newcomers or non-specialists to identify samples.
- There are currently only a few full-time diatom taxonomists worldwide. As a consequence, it is difficult to keep the literature on diatoms up to date; much of the existing literature was published in the 19th or early last century.
Therefore, an European effort called ADIAC was started, which is the Automatic Diatom Identification And Classification project. Its goal is to create image databases and to develop software for feature extraction and identification and to develop methods for automatic microscope slide scanning including autofocussing. The main concern is to study the use of tools from image processing and pattern recognition for the automatic identification of diatoms, using the shape and the ornamentation of the diatoms. In this thesis the use of tools from a branch of image processing, called mathematical morphology, for the automatic identification of diatoms based on their ornamentation is investigated.
Chapter 2

Mathematical morphology

2.1 Binary images

This chapter starts with an explanation of mathematical morphology for binary images and introduces some transformations. In section 2.2 this will be extended to the grayscale case.

An image or picture consists of picture elements, usually abbreviated as *pixels*. In the binary case, an image consists solely of background and foreground pixels. As an example, figure 2.1 shows an image of the letter A; here the gray squares represent the foreground pixels and the • represents the origin. A binary image can be represented by a set. Since such an image consists solely of background and foreground pixels, the set of either foreground or background pixels describes the image fully. An image where each pixel can have a grayvalue from a range of intensities is called a graylevel or grayscale image. A transformation that is used on binary images is called a binary transformation; consequently, grayscale transformations can be applied on grayscale images. First, binary transformations are introduced.

![Figure 2.1: Example of a binary image](image)

2.1.1 Binary image transformations

Let $E$ be the Euclidean space $\mathbb{R}^n$ or the discrete grid $\mathbb{Z}^n$ and $M \subset E$ be the mask, then binary images can be defined as a subset of $M$. In this text, only discrete two dimensional, i.e. $n = 2$, images are used.

The elements of an image, i.e. the pixels, which can be described by pixel coordinates, can be seen as vectors. This way, some basic image processing [3] operations can be defined as:

- **Translation**: $(A)_h = \{\vec{c} | \vec{c} = \vec{a} + \vec{h} \land \vec{a} \in A\}$
- **Reflection**: $\hat{A} = \{\vec{c} | \vec{c} = -\vec{a} \land \vec{a} \in A\}$
- **Complement**: $X^c = \{\vec{x} | \vec{x} \in M \land \vec{x} \notin X\}$
- **Difference**: $X \setminus A = \{\vec{x} | \vec{x} \in X \land \vec{x} \notin A\} = X \cap A^c$
- **Scaling**: $|X|_\lambda = \{\lambda \vec{x} | \vec{x} \in X\}$,
where \( X \subseteq M \) and \( A \subseteq M \) are sets, e.g., a binary image, and \( \vec{h} \in E \). These operations are demonstrated in Figure 2.2 with translation vector \( \vec{h} = (2, 1) \). Some of the morphological transformations make use of small subset of \( E \), called a structuring element. These are used to probe an image to detect certain features. The most common morphological transformations are the dilation and erosion. Let \( X \) be a binary image and \( A \) a structuring element, the dilation and erosion are defined as:

\[
\text{Dilation: } D_A(X) = X \oplus A = \{ \bar{x} + \bar{a} : \bar{x} \in X \land \bar{a} \in A \} = \bigcup_{\bar{a} \in A} X_{\bar{a}} \tag{2.1}
\]

\[
\text{Erosion: } E_A(X) = X \ominus A = \bigcap_{\bar{a} \in A} X_{-\bar{a}} \tag{2.2}
\]

where \( X \oplus A \) and \( X \ominus A \) are known as the Minkowski addition and subtraction respectively. Intuitively, the dilation of \( X \) by structuring element \( A \) can be seen as the set of translations by \( \vec{h} \) of \( A \) along the edges of \( X \), such that \( A \) hits \( X \). This way, the erosion of \( X \) by \( A \) can be seen as the set of translations by \( \vec{h} \) of \( A \) where \( A \) hits \( X \). Using these geometrical interpretations of the dilation and erosion we come to the following definitions of these operations:

\[
\text{Dilation: } D_A(X) = \{ \vec{h} \in E : (\bar{A})_{\vec{h}} \cap X \neq \emptyset \} \tag{2.3}
\]

\[
\text{Erosion: } E_A(X) = \{ \vec{h} \in E : (A)_{\vec{h}} \subseteq X \} \tag{2.4}
\]

Figure 2.3 shows an example of the dilation and erosion with a disk as structuring element. This
Figure 2.4: Examples of the discrete binary dilation and erosion

is known as the duality relation:

\[ Duality: \quad X \ominus A = (X \ominus \bar{A}) \ominus \bar{A}. \] (2.5)

Both the dilation and the erosion transform an image into another, or more formally \( P(E) \rightarrow P(E) \), where \( P(E) \) denotes the power set of \( E \). Unless noted otherwise, any transformation \( \Psi \) satisfies \( \Psi : P(E) \rightarrow P(E) \). The dilation and erosion can be used to define the (structured) opening and closing as follows:

Opening: \( \Gamma_A(X) = X \circ A = (X \ominus A) \oplus A = \bigcup \{(A) \subseteq E : \bar{A} \subseteq X \} \) (2.6)

Closing: \( \Phi_A(X) = X \bullet A = (X \oplus A) \ominus A = \bigcap \{(A) \subseteq E : \bar{A} \subseteq X \} \) (2.7)

As can be seen in figure 2.5 both the opening and the closing smooth the image, but the opening

Figure 2.5: Examples of the continuous binary opening and closing

does this by breaking small bridges and removes small details, where the closing fills small holes and gaps. Figure 2.6 demonstrates this for the discrete case.

A transformation \( \Psi \) can be characterized by the following properties:

- **Extensive**, \( \Psi(X) \supseteq X \): The result can w.r.t. the input only become larger. For example, the closing is extensive.

- **Anti-extensive**, \( \Psi(X) \subseteq X \): The result can w.r.t. the input only become smaller. For example, the opening is anti-extensive.

- **Increasing**, \( X \subseteq Y \Rightarrow \Psi(X) \subseteq \Psi(Y) \): If \( X \) is smaller than \( Y \), than the result of applying a transformation \( \Psi \) to \( X \) can never be larger than the result of applying \( \Psi \) to \( Y \). The aforementioned morphological transformations are increasing. On the contrary, the thinning, which is described below, is usually a non-increasing transformation.
Figure 2.6: Examples of the discrete binary opening and closing

- **Idempotent**, \( \Psi(\Psi(X)) = \Psi(X) \): Repeating the application of a transformation has no effect. Idempotent transformations are also known as filters. Note that the opening and closing are idempotent, unlike the dilation and erosion.

- **Translation-invariant**, \( \forall h \in E : \Psi((X)_h) = (\Psi(X))_h \): The order of executing transformation \( \Psi \) and a translation does not have an effect on the output.

- **Scale-invariant**, \( \forall \lambda > 0 : \Psi([X]_\lambda) = [\Psi(X)]_\lambda \): The result of the application of \( \Psi \) does not depend on the size of \( X \).

- **Rotation-invariant**, The order of performing a rotation and transformation \( \Psi \) makes no difference.

A transformation which is anti-extensive, increasing and idempotent is called an **algebraic opening**, whereas an extensive, increasing and idempotent transformation is called an **algebraic closing**. Note that structured openings and closings are algebraic openings and closings respectively. Since scale-invariant operator is independent of the size of elements in the image, it is therefore sensitive to the shape of these elements. A scale-, rotation- and translation-invariant operator is called a **shape operator** [16].

The intersection of an image \( X \) eroded with \( A_1 \) and the background of \( X \) eroded with \( A_2 \) is called the Hit-or-Miss Transform:

\[
X \circ A = (X \ominus A_1) \cap (X^C \ominus A_2) = \{ h \in E : (A_1)_h \subseteq X \land (A_2)_h \subseteq X^C \},
\]

where \( A = (A_1, A_2) \). This transformation can be used to detect features in an image. Interested readers are referred to [3, 12]. A transformation which is anti-extensive, idempotent, but not necessarily increasing is called a **thinning**. Note that an increasing thinning is an opening. The dual of the thinning is the **thickening**. Likewise, an increasing thickening is a closing. Thinnings and thickenings of image \( X \) by \( A = (A_1, A_2) \) are usually defined using the Hit-Or-Miss Transform as follows:

\[
\begin{align*}
\text{Thinning:} & \quad T_A(X) = X \ominus A = X \setminus (X \ominus A) = X \cap (X \ominus A)^C \\
\text{Thickening:} & \quad \Xi_A(X) = X \circ A = X \cup (X \ominus A),
\end{align*}
\]

where \( A_1 \) and \( A_2 \) are disjoint structuring elements. As can be seen in figure 2.7 a thinning tends to thin image features. Likewise a thickening tends to thicken structures in the image.

### 2.2 Grayscale images

We will now extend the aforementioned morphological transformations to the grayscale case. To distinguish a grayscale image from a binary, we will use the convention to denote a grayscale image with a small letter, usually \( f \), instead of a capital character like \( X \) for binary images. A grayscale
Figure 2.7: Example of the discrete binary thinning.

The image can then be described by a function $f : M \rightarrow G$. The set of gray values $G$ is a subset of $\mathbb{R}$ or $\mathbb{Z}$.

In the previous section, some basic image processing operations were described. These operations can be adapted to the grayscale case as follows:

- **Translation:** \((k)_{k}(\vec{x}) = k(\vec{x} + \vec{h})\)
- **Reflection:** \(k(-\vec{x})\)
- **Complement:** \(f^c(\vec{x}) = m - f(\vec{x})\)
- **Difference:** \(f(\vec{x}) - k(\vec{y})\)
- **Scaling:** \([f]_{\lambda}(\vec{x}) = f(\lambda \vec{x})\)

where \(f : M \rightarrow G\) and \(k : M \rightarrow G\) are functions, e.g. a grayscale image, \(\vec{h} \in E\) and \(\lambda \in \mathbb{R}\). Usually, \(G\) is a range of nonnegative gray values between 0 and a maximum gray value \(m\); in that case, the complement is usually computed as: \(f^c(\vec{x}) = m - f(\vec{x})\). The complement is also known as inversion. These operations are demonstrated for the discrete case in figure 2.8, with \(\vec{h} = (3, 10)\) and \(\lambda = 0.5\). The origin is located in the center of the image. Note that pixels outside the domain \(M\) are considered to be black.

The binary morphological transformations discussed in the previous section modified image features consisting of fore- or background pixels. The grayscale versions of these transformations modify bright or dark image features respectively. In the case of the structuring transformations, the structuring element \(k\) is also a subset of a grayscale image, i.e. a function \(k : K \rightarrow G\), with \(K \subseteq E\). If a structuring element \(k\) consists only of two gray levels, that is \(k : K \rightarrow G\) is constant, \(k\) is called a flat structuring function and the corresponding filter is called a flat filter. In the binary case, a disk as a structuring element was described. If we extend this for grayscale images to a ball as a structuring function, the resulting filter is called a rolling-ball filter.
2.2.1 Umbrae

The set of gray values \( G \) of an \( n \)-dimensional grayscale image \( f : E \rightarrow G \) can also be considered as a dimension. If we add this dimension \( G \) to \( E \), we get the enlarged space \( \bar{E} = E \times G \). In the case of a 2D graylevel image, dimension \( G \) can be seen as the "height" of \( \bar{G} \). Subsets of \( \bar{E} \) are called umbrae. Imagine a light is placed above the graph of an image \( f \) in the space \( \bar{E} \), the shadow cast by the parallel beam of that light is the umbra. This leads to the following two definitions.

Definition 2.1 The umbra of function \( f : E \rightarrow G \) is defined by:

\[
U[f] = \{(x,t) \in \bar{E} : t < f(x)\}. \tag{2.11}
\]

Definition 2.2 A subset \( U \subseteq \bar{E} \) is an umbra if and only if:

\[
(\bar{x},t) \in U \implies \forall (t' \leq t : (\bar{x},t') \in U) \tag{2.12}
\]

Thus, the application of an umbra function on a grayscale image with \( n \) dimensions results in a binary image with \( n + 1 \) dimensions. On this binary image, binary transformations can be applied. The function to transform that binary image back to a grayscale image is called the top.

Definition 2.3 Let \( A \subseteq \bar{E} \) and \( F = \{x \in E : \exists t \in G : (\bar{x},t) \in A\} \), then the top of \( A \) can be defined as the function \( T[A] : F \rightarrow G \) by:

\[
T[A](\bar{x}) = \max\{t \in G : (\bar{x},t) \in A\}. \tag{2.13}
\]

Note that in this definition \( A \) needs to be 'closed from above'. Practically, this is always the case; otherwise the maximum should be replaced by the supremum. The top performs the inverse operation of the umbra, thus they are each others inverse.

\[
T[U[f]] = f \tag{2.14}
\]

\[
U[T[A]] = A, \text{ if } A \text{ is an umbra} \tag{2.15}
\]

2.2.2 Grayscale dilation and erosion

A grayscale morphological transformation can now be performed as follows. First, the grayscale image \( f \) and the grayscale structuring function \( k \) are converted to binary images, \( X \) and \( A \) respectively, using umbrae. Then the binary equivalent of that transformation is applied to \( X \) and \( A \), which results in a binary image \( Y \). Finally, \( Y \) is converted back to a grayscale image using the top function. This process is illustrated for the grayscale structured dilation and erosion in figure 2.9. This is formalized in the following definition:
Definition 2.4 Let $f : F \rightarrow G$ and $k : K \rightarrow G$ be two grayscale images with $F, K \subseteq E$, then the grayscale dilation and erosion are defined as:

**Dilation:**
\[
\delta_k(f) = f \oplus k = T[U[f] \oplus U[k]]
\]

**Erosion:**
\[
\epsilon_k(f) = f \ominus k = T[U[f] \ominus U[k]].
\]

Using this definition, the following theorem can be derived:

**Theorem 2.5**

**Dilation:**
\[
\delta_k(f) = (f \oplus k)(x) = \max_{y \in K, (x-y) \in F} [f(x-y) + k(y)]
\]

**Erosion:**
\[
\epsilon_k(f) = (f \ominus k)(x) = \min_{y \in K, (x+y) \in F} [f(x+y) - k(y)]
\]

This theorem says that the dilation of $f$ by $k$ equals to the maximum of $f(x-y) + k(y)$, where $y$ runs through the elements of the set $K$. Similarly, the dilation equals to the minimum of $f(x+y) - k(y)$. Therefore, these operations are called minimum and maximum filters. The grayscale opening and closing can be defined using the grayscale dilation and erosion.

Definition 2.6 Let $f : F \rightarrow G$ and $k : K \rightarrow G$ be two grayscale images, then the grayscale opening and closing of $f$ by $k$ is defined by:

**Opening:**
\[
\gamma_k(f) = f \circ k = (f \ominus k) \oplus k
\]

**Closing:**
\[
\phi_k(f) = f \bullet k = (f \oplus k) \ominus k.
\]

The results of a grayscale opening and closing are demonstrated in figure 2.10.
2.2.3 Grayscale thinning and thickening

Similar to the binary case, the grayscale thinning and thickening can be defined using the definition of the Hit-or-Miss Transform. The grayscale Hit-or-Miss Transform of an image $f : F \rightarrow G$ can be defined as the minimum of the erosion of $f$ by $k_1$ and the erosion of the inverted image $f^c$ by $k_2$:

$$f \circ k = \min((f \ominus k_1), (f^c \ominus k_2)),$$

where $k = (k_1, k_2)$. Similar to the binary case, $k_1$ and $k_2$ are structuring functions. An example is given in figure 2.11, where the north-east pixels are detected. The structuring elements which were used are grayscale versions of the binary ones depicted in this figure as $A_1$ and $A_2$ and described in [12] on page 32, where the foreground pixels were replaced by white pixels and the background was made black.

Grayscale thinnings and thickenings of image $f$ by $k = (k_1, k_2)$ can now be defined as:

Thinning: $\nu_k(f) = f \circ k = f - (f \circ k) = \min(f, (f \circ k)^c)$  \hspace{1cm} (2.23)

Thickening: $\xi_k(f) = f \circ k = \max(f, (f \circ k))$  \hspace{1cm} (2.24)

Figure 2.11: Example of the grayscale Hit-or-Miss Transform
Chapter 3

Connected operators

3.1 Connectivity

The morphological transformations described in the previous chapter used a structuring element. In this chapter, image transformations are described which use a criterion to process the components of an image. An example would be a filter which removes all small image components. First the notion of connectivity will be explained, which is used later to give a formal definition of image components.

A set of adjacent pixels that belong to each other is called a connected set. Which pixels are adjacent to a certain pixel depends on the connectivity. In the case of a 4-connected connected set, the pixels directly to the north, south, east and west of a pixel $p$ are neighbors of $p$. When the pixels directly to the northeast, northwest, southeast and southwest are also neighbors, that connected set is 8-connected. Figure 3.1 illustrates 4- and 8-connectivity, where the neighbors of the pixel are denoted by black circles.

Figure 3.1: The 4- and 8-connected neighbors (black dots) of the gray square

A connected set of pixels of a binary image is called a connected region. If a connected region of foreground pixels has no neighboring foreground pixels, it is called a connected component. Likewise, a connected region of background pixels with no neighboring background pixels is also a connected component. The grayscale equivalent of a connected component is a flat zone or level component. Thus, a flat zone $L_h$ at level $h$ of a grayscale image $f$ is a connected component of the set of pixels $\{\vec{x} \in M | f(\vec{x}) = h\}$. A regional maximum $M_h$ at level $h$ is a flat zone which has no neighbors with an intensity larger than $h$. The peak component $P_h^k$ of image $f$ at level $h$ is the $k$'th connected component of the threshold set $\Sigma_h(f)$. The threshold $\Sigma_h(f)$ at level $h$ of an image $f$ is defined as:

$$\Sigma_h(f) = \{\vec{x} \in M | f(\vec{x}) \geq h\}. \quad (3.1)$$
Figure 3.2 shows an image consisting of eight level components: a white rectangle, two gray rectangles, an ellipse and its hole, a circle and a triangle. Note that this image has four regional maxima. A threshold set can be considered as a slice taken from an image at level \( h \). To reconstruct the image \( f(\mathcal{E}) \), the inverse can be computed by piling up these slices as follows:

\[
f(\mathcal{E}) = \max(h : \mathcal{E} \in \mathcal{I}_h).
\]  

(3.2)

### 3.2 Complete lattices

As mentioned before, binary images are represented by sets and grayscale images by functions, which are two examples of lattices. Lattices are defined here in the same way as in [15], where a complete lattice is defined as a set of ordered elements (partial or total order) for which each family of elements possesses a supremum (sup.) and an infimum (inf.). In the binary case, the lattice of sets is used, where the order, the sup. and the inf. are defined as respectively: \( \subseteq \), \( U \) and \( \cap \). For grayscale images, the lattice of functions is used, where the order, the sup., and the inf. are respectively: \( \leq \), \( V \) and \( \wedge \).

Salembier and Serra also give the following definition of partitions.

**Definition 3.1** A partition of a space \( E \) is a set of connected regions \( \{A_i\} \) which are disjoint (\( \forall i \neq j : A_i \cap A_j = \emptyset \)) and the union of which is the entire space (\( \bigcup A_i = E \)). Each connected region \( A_i \) is called a partition class.

The partition of a binary image consisting of the connected components of that image, is called its associated partition. This is demonstrated in 3.3, where each connected component of image \( X \) is represented by a different gray value in its associated partition. Furthermore, they call a partition \( \{A_i\} \) to be finer than another partition \( \{B_i\} \), if any pair of points belonging to the same class \( A_j \) also belongs to a unique partition class \( B_j \). This definition of finer can be easily extended for the grayscale case, by using level components instead of connected components.

Figure 3.3: Binary image \( X \) (a) and its associated partition (b)
3.3 Connected operators

The aforementioned structured filters can be used to create a filter by reconstruction [5, 15, 18]. Often, a structured opening is used, in which case it is an opening by reconstruction. In that case, it is done by applying an opening with a connected component, usually a disk for 2D images, to an image and after that restoring all components that were not entirely removed. These operators can be generalized to connected operators. Formally, a binary connected operator can be defined as follows [14]:

**Definition 3.2** A binary operator $\Psi$ is connected when for any image $X$, the set difference $X \setminus \Psi(X)$ is exclusively composed of connected components of $X$ or of its complement $X^c$.

The grayscale connected operator can be defined using the concept of partitions.

**Definition 3.3** Let $\psi$ be a grayscale operator and $f$ a function, then $\psi$ is connected if the partition of flat zones of $f$ is finer than the partition of flat zones of $\psi(f)$.

The effect of the definition is demonstrated in figure 3.4, where a binary image $X$ is filtered with $\Psi$. As can be seen, the original image $X$ is indeed finer than the resulting image $\Psi(X)$.

![Figure 3.4: Influence of connected operators on an associated partition. The associated partition of image $X$ (c) is finer than the associated partition of $\Psi(X)$ (d).](image)

Connected operators are shape preserving, which means that they remove or preserve edges of connected components, but that the preserved ones are not moved, i.e. a connected operator does never introduce new edges. Note that the aforementioned structured operators lack this property, since these can result in that a connected component is split in two or more components, which means that new edges are introduced. An example of a connected filter is the area opening [17], where the connected components with an area less than a certain $\lambda$-value are removed and where area is the Lebesgue measure, which is for discrete images the number of pixels. An overview of connected operators and connectivity can be found in [9].

3.4 Attribute opening and thinning

The idea of filters by reconstruction and area openings can be extended to attribute filters. Salem-bier and Serra [15] gave some hints on this, but the term was introduced later by Breen and Jones [1]. The treatment of Breen and Jones is used here. Their attribute opening is based on the connected opening [17].

**Definition 3.4** The connected opening $\Gamma_\Psi(X)$ of a set $X \subseteq M$ at a point $\vec{x} \in M$ is the connected component of $X$ that contains $\vec{x}$ if $\vec{x} \in X$ and the empty set otherwise.
Thus, the connected opening preserves at most one connected component, namely the one containing $x$. The trivial opening $\Gamma_T$ of a connected set $C$ and an increasing criterion $T$ preserves $C$ only if $C$ satisfies $T$.

**Definition 3.5** Let $C \subseteq M$ be a connected set and $T$ an increasing criterion, then the trivial opening $\Gamma_T$ of $C$ is defined by:

$$
\Gamma_T(\emptyset) = \emptyset \quad (3.3)
$$

$$
\Gamma_T(C) = \begin{cases} 
C & \text{if } C \text{ satisfies criterion } T, \\
\emptyset & \text{otherwise.} 
\end{cases} \quad (3.4)
$$

Note that the criterion $T$ must be increasing, since openings are increasing by definition and a non-increasing criterion would result in a thinning. Examples of increasing criteria of a connected component $C$ are:

- **Area**: The number of pixels of $C$.
- **Length of the diagonal of the bounding box**: The box defined by the coordinates $(x_0, y_0)$ and $(x_1, y_1)$ is the bounding box, where $x_0$ and $y_0$ are respectively the minimum $x$ and the minimum $y$ coordinates of the pixels and $x_1$ and $y_1$ are respectively the maximum $x$ and $y$ coordinates of the pixels. The length of the diagonal of this bounding box is $\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$.
- **Length of the diagonal of the minimum enclosing rectangle**: Idem, but the minimum enclosing rectangle is that smallest rectangle that still contains $C$.
- **Radius, diameter or area of the smallest enclosing circle**: As above, but now the smallest circle instead of rectangle. Radius and area are calculated according to this circle.
- **Radius, diameter or area of the largest circle that fits into $C$**: As above, but now the circle that is still contained in $C$.
- **Area of the largest square that fits into $C$**: As above, but now a square instead of a circle.
- **Maximum ferret diameter**: The distance between the two pixels of $C$ which have the largest distance.
- **Moment of inertia**:  

Note that these examples are related to the size of a component. This is always the case for increasing criteria. Criteria related to size are referred to as size criteria. Examples of non-increasing criteria for a connected component $C$ are:

- **Perimeter**: The number of pixels along the boundary of $C$.
- **Complexity**: perimeter/area.
- **Simplicity**: area/perimeter.
- **Compactness**: $(\text{perimeter})^2/\text{area}$.
- **Eccentricity**: The ratio of the major to the minor axis of the minimum enclosing ellipse.
- **Moment of inertia divided by the square of the area**: This criterion, abbreviated as $I/A^2$, is a measure of the compactness of $C$. Elongated structures results in a large value. For a circle, $I/A^2 = 1$, which is the minimum value for this criterion.
- **Jaggedness**:  

24
- Maximum geodesic distance:
- Length of the minimal skeleton:
- Major and minor axes of the ellipse that best fits $C$:

In addition to these, some more examples can be found in [1, 3, 14]. These examples are mainly related to the shape of a component. A criterion which is related only to the shape, is scale-, rotation- and translation-invariant and is called a shape criterion. An example of this is $I/A^2$.

The connected opening and the trivial opening with an increasing criterion $T$ can be combined to an attribute opening.

**Definition 3.6** Let $X \subseteq M$ be a binary image and $T$ an increasing criterion, then the attribute opening $\Gamma^T$ of $X$ is defined by:

$$\Gamma^T(X) = \bigcup_{x \in X} \Gamma_T(\Gamma_x(X)).$$

(3.5)

An attribute opening removes the connected components of foreground pixels that do not satisfy criterion $T$; all other connected foreground components are preserved. The attribute closing $\Phi^T(X)$, which preserves all connected background components that satisfy $T$ and removes all other background components, which can be defined using the duality relation as: $\Phi^T(X) = (\Gamma^T(X^C))^C$.

If $T$ is a criterion, the trivial thinning $\Gamma_T$ of a connected component $C$ preserves $C$ only if it satisfies $T$.

**Definition 3.7** Let $C \subseteq M$ be a connected set and $T$ a not necessarily increasing criterion, then the trivial thinning $\Gamma_T$ of $C$ is defined by:

$$\Gamma_T(\emptyset) = \emptyset$$

(3.6)

$$\Gamma_T(C) = \begin{cases} C & \text{if } C \text{ satisfies criterion } T, \\ \emptyset & \text{otherwise}. \end{cases}$$

(3.7)

This definition can be used to define the attribute thinning $\Phi^T$ of a binary image $X$.

**Definition 3.8** Let $X \subseteq M$ be a binary image and $T$ a not necessarily increasing criterion, then the attribute thinning $\Phi^T$ of $X$ is defined by:

$$\Phi^T(X) = \bigcup_{x \in X} \Phi_T(\Gamma_x(X)).$$

(3.8)

The dual of the attribute thinning is the attribute thickening $\Xi^T$ which can be defined using this duality relationship as: $\Xi^T(X) = (\Phi^T(X^C))^C$.

### 3.5 Grayscale attribute opening and thinning

The **threshold decomposition principle** can be used to extend a binary transformation $\Phi(X)$ to its grayscale equivalent $\psi(f)$ and is defined as a combination of the threshold and its inverse:

$$\psi(f)(\bar{x}) = \max(h : \bar{x} \in \Phi(\Xi_h(f))).$$

(3.9)

The binary connected opening $\Gamma_x$ can now be extended to the grayscale connected opening $\gamma_x$.

**Definition 3.9** The grayscale connected opening $\gamma_x(f)$ of a grayscale image $f$ at a point $\bar{x} \in M$ is defined as:

$$\gamma_x(f)(\bar{y}) = \max(h : \bar{y} \in \Gamma_x(\Xi_h(f))).$$

(3.10)
Here, M denotes the support of f. In a similar way, the grayscale attribute opening \( \gamma^T \) and the grayscale attribute thinning \( v^T \) can be defined.

**Definition 3.10** Let \( f \) be a grayscale image and \( T \) an increasing criterion, the grayscale attribute opening \( \gamma^T \) is defined as:

\[
\gamma^T(f)(\bar{x}) = \max(h : \bar{x} \in \Gamma^T(\Xi_h(f))).
\]  

(3.11)

**Definition 3.11** The grayscale attribute thinning \( v^T \) of image \( f \) and a not necessarily increasing criterion \( T \) is defined as:

\[
v^T(f)(\bar{x}) = \max(h : \bar{x} \in \Upsilon^T(\Xi_h(f))).
\]  

(3.12)

In chapter 5 several methods are described for the attribute thinning. This definition corresponds to the so-called direct method. Breen and Jones note that these definitions of the grayscale attribute opening and closing would yield inefficient algorithms. Instead they propose to consider only the set of all regional maxima \( R \) of the image \( f \). \( R \) is then defined as: \( R = \bigcup R_c \), where each \( R_c \) is a flat zone in \( f \). A regional maximum \( R_c \) can be represented by a single point \( r_c \). These regional maxima can be selected and filtered using the grayscale trivial opening \( \gamma_{r_c,T} \), which preserves at most one flat zone, namely the one that contains \( \bar{x} \) and satisfies criterion \( T \).

**Definition 3.12** Let \( f \) be a grayscale image, \( T \) an increasing criterion and \( \bar{x} \in M \), then the grayscale trivial opening \( \gamma_{r_c,T} \) is defined as:

\[
\gamma_{r_c,T}(f)(\bar{y}) = \max(h : \bar{y} \in \Gamma_T(\Xi_h(f))).
\]  

(3.13)

The thinning that preserves at most that flat zone that contains \( \bar{x} \) and satisfies a criterion \( T \) and removes all other components of image \( f \) is called the grayscale trivial thinning \( v_{r_c,T} \).

**Definition 3.13** Let \( f \) be a grayscale image, \( T \) a not necessarily increasing criterion and \( \bar{x} \in M \), then the grayscale trivial thinning \( v_{r_c,T} \) is defined as:

\[
v_{r_c,T}(f)(\bar{y}) = \max(h : \bar{y} \in \Upsilon_T(\Xi_h(f))).
\]  

(3.14)

These definitions are used by Breen and Jones to define grayscale attribute opening \( \gamma^T \) and thinning \( v^T \) using the regional maxima of image \( f \).

**Theorem 3.14** A grayscale attribute opening \( \gamma^T \) is given by:

\[
\gamma^T(f) = \max_{r_c} \gamma_{r_c,T}(f).
\]  

(3.15)

**Theorem 3.15** A grayscale attribute thinning \( v^T \) is given by:

\[
v^T(f) = \max_{r_c} v_{r_c,T}(f).
\]  

(3.16)

Connected operators can be computed efficiently using Tarjan’s union-find algorithm. This was described by Meijster and Wilkinson in [8] for morphological area operators and was generalized to attribute operations by Wilkinson and Roerdink in [20]. In chapter 5 an algorithm will be discussed which provides a versatile method for image filtering based on connected operators.
Chapter 4

Multi-scale analysis

4.1 Image analysis

In the previous sections methods were described which process images. They can be used to remove certain features from an image. In this section methods are described which make a decomposition of an image using morphological operators. This image decomposition, which can be based on the size or the shape of image details, can be used to reveal information about the presence or absence of certain features in an image. Since these methods extract image details at different scales, these can be used for multi-scale image analysis.

4.2 Granulometries

Size distributions or granulometries [1, 3, 7, 16, 19] were initially introduced by Matheron [7] (for a recent review of granulometries see [19]) and can intuitively be considered as a set of sieves of different grades, each sieve allowing image details larger than a certain size to pass. This can be used to extract image details of different size classes (scales). A criterion is used to specify the size classes. Usually, the used size criterion is the width of each detail. The size distributions described here are based on the attribute operators of the previous section, so any size criterion can be used. Note that a size criterion means it must be increasing.

More formally, a granulometry can be defined as an ordered set of operators, where each operator processes the image in such a way, that features smaller than a particular size are removed. These operators must be idempotent, anti-extensive and increasing, which means they must be openings. Let \( \{\Gamma_\lambda\} \) be a set of binary openings with size parameter \( \lambda \) and \( X \) an image, then the following absorption law must hold:

\[
V(X \subseteq M \wedge \lambda > 0 \wedge \mu \geq 0 : \Gamma_\lambda + \mu (X) \leq \Gamma_\lambda (X)).
\]

(4.1)

Likewise, this absorption law can be defined for the grayscale case by replacing the \( \Gamma \) and the \( X \) by \( \gamma \) and \( f \) respectively. The binary size distribution can now be defined as follows:

**Definition 4.1** A binary size distribution or granulometry is a set of operators \( \{\alpha_r\} \) with \( r \) from some totally ordered set \( \Lambda \) (usually \( \Lambda \subseteq \mathbb{R} \) or \( \mathbb{Z} \)), with the following three properties:

\[
\begin{align*}
\alpha_r(X) &\subseteq X \quad \text{(4.2)} \\
X \subseteq Y &\Rightarrow \alpha_r(X) \subseteq \alpha_r(Y) \quad \text{(4.3)} \\
\alpha_r(\alpha_s(X)) &= \alpha_{\max(r,s)}(X) \quad \text{(4.4)}
\end{align*}
\]

for all \( r, s \in \Lambda \).
Note that this definitions says that the operators $\alpha_r$ must be openings, since properties 4.2 and 4.3 define $\alpha_r$ to be anti-extensive and increasing and property 4.4 implies idempotence. The grayscale size distribution is defined in a similar way, where $f$ and $g$ are grayscale images with domain $M$.

**Definition 4.2** A grayscale size distribution is an ordered set of operators $\{\alpha_r\}$ with $r$ from some ordered set $\Lambda$, which have the following properties:

\[
(\alpha_r(f))(x) \leq f(x) \quad (4.5)
\]

\[
\forall (x \in M : f(x) \leq g(x)) \Rightarrow (\alpha_r(f))(x) \leq (\alpha_r(g))(x) \quad (4.6)
\]

\[
(\alpha_r(\alpha_s(f)))(x) = (\alpha_{\max(r,s)}(f))(x), \quad (4.7)
\]

for all $r, s \in \Lambda$.

### 4.3 Shape distributions

Thus a size distribution can be used to classify image details of different sizes. If one wants to extract image details based on their shape instead of scale, shape distributions [16] can be used. Like a granulometry, a shape distribution is an ordered set of operators, but these operators remove or preserve image features based on their shape instead of size, which means this operators must be shape operators. The binary and grayscale shape distribution are defined as follows:

**Definition 4.3** A binary shape distribution is a set of operators $\{\beta_r\}$ with $r$ from some totally ordered set $\Lambda$, with the following three properties:

\[
\beta_r(X) \subseteq X \quad (4.8)
\]

\[
\beta_r(X,.) = (\beta_r(X),) \quad (4.9)
\]

\[
\beta_r(\beta_s(X)) = \beta_{\max(r,s)}(X), \quad (4.10)
\]

for all $r, s \in \Lambda$ and $\lambda > 0$.

**Definition 4.4** A grayscale shape distribution is a set of operators $\{\beta_r\}$ with $r$ from some totally ordered set $\Lambda$, with the following three properties:

\[
(\beta_r(f))(x) \leq f(x) \quad (4.11)
\]

\[
\beta_r(f,.) = (\beta_r(f),) \quad (4.12)
\]

\[
\beta_r(\beta_s(f)) = \beta_{\max(r,s)}(f), \quad (4.13)
\]

for all $r, s \in \Lambda$ and $\lambda > 0$.

Remember that these shape operators are scale-invariant thinnings, which means that they are idempotent, anti-extensive, but not increasing. Note that this is specified by the properties of these definitions.

### 4.4 Size and shape transforms

A binary opening transform $\Theta_\alpha$ is a mapping of an image $X \subseteq M$ to a grayscale image using a size distribution $\{\alpha_r\}$ such that for all $x \in M$:

\[
(\Theta_\alpha(X))(x) = \sup \{r \in \Lambda | x \in \alpha_r(X)\}, \quad (4.14)
\]

where $r$ is from some totally ordered set $\Lambda$. In a similar way, the binary shape transform $\Theta_\beta$ is defined as a mapping from a binary image $X$ to a grayscale image using a size distribution $\beta_r$ such that for all $x \in M$:

\[
(\Theta_\beta(X))(x) = \sup \{r \in \Lambda | x \in \beta_r(X)\}, \quad (4.15)
\]
where again \( r \) is from some totally ordered set \( \Lambda \). The opening transform and the shape transform can be extended to the grayscale case as follows:

\[
(\theta_{\alpha}(f))(x) = \sup\{r \in \Lambda | x \in \alpha_r(f)\},
\]

\[
(\theta_{\beta}(f))(x) = \sup\{r \in \Lambda | x \in \beta_r(f)\},
\]

where \( f \) is a grayscale image and \( r \) is from some totally ordered set \( \Lambda \).

### 4.5 Pattern spectra

For purposes of image analysis and classification, a method which extracts information about image details and presents this in function or table might be useful. This can be done by using pattern spectra.

The idea is to compute a size \( s_\alpha \) or shape distribution \( s_\beta \) on which in the binary case the Lebesgue measure is used. This means that larger components contribute more to their class than smaller ones. The result is differentiated to its class number and negated. Formally, this is defined for a binary image \( X \) as follows:

\[
(s_\alpha(X))(u) = \frac{-\partial A(\alpha_r(X))}{\partial r} \bigg|_{r=u},
\]

\[
(s_\beta(X))(u) = \frac{-\partial A(\beta_r(X))}{\partial r} \bigg|_{r=u},
\]

where \( A(X) \) is the Lebesgue measure in \( \mathbb{R}^n \). Often an alternative definition is used for pattern spectra, where the differentiation is left out,

\[
(S_\alpha(X))(r) = A(\alpha_r(X))
\]

\[
(S_\beta(X))(r) = A(\beta_r(X))
\]

These definitions for the size spectrum \( S_\alpha \) and the shape spectrum \( S_\beta \) are commonly used on computers, since differentiations over a size variable are often undesirable for discrete images. Pattern spectra can also be defined using opening and shape transforms,

\[
(S_\alpha(X))(r) = A(\{x \in M | (\theta_{\alpha}(X))(x) > r\})
\]

\[
(S_\beta(X))(r) = A(\{x \in M | (\theta_{\beta}(X))(x) > r\})
\]

For grayscale images, the Lebesgue measure \( A(X) \) in the definitions of the pattern spectrum is replaced by a sum or integral,

\[
(S_\alpha(f))(r) = \int_M (\alpha_r(f))(x) dx
\]

\[
(S_\beta(f))(r) = \int_M (\beta_r(f))(x) dx
\]

Note that taking an integral over a continuous function can be considered as some sort of Lebesgue measure for functions.

Figure 4.1 shows an example of a pattern spectrum using a size criterion. The input image contains circles of three different sizes and the corresponding three size classes of the pattern spectrum have a high value, which is illustrated by the large bars. An example of pattern spectrum using the elongation as its criterion is shown in figure 4.2. Here the input image contains objects with four different kind of shapes at different rotations. In the corresponding pattern spectrum these shapes are represented by only three shape classes, because the circles and the squares are both considered to be compact and in this case are considered to belong to the same shape class. Note that although not all bars do have the same orientation, all belong to the same shape class.
Figure 4.1: Example of a size pattern spectrum: (a) input image and (b) corresponding pattern spectrum using area

Figure 4.2: Example of a shape pattern spectrum: (a) input image and (b) corresponding pattern spectrum using $I/A^2$
Chapter 5

The Max-tree

5.1 A tree representation

In the previous sections methods were discussed which could be used in image processing and analysis. An algorithm to compute these operators in a elegant and versatile way is the topic of this chapter.

An image can be represented by its level components for the grayscale case or by its connected components in the binary case. Salembier et al. [14] show that connected operators in fact process a structured representation consisting of these components. They introduce a new structure, which they refer to as the Max-tree [13, 14, 16]. Another way to perform connected filtering and segmentation using a structured representation called component trees is describes by Jones in [4].

The Max-tree representation is a versatile structure, with which the filtering process can be separated into three steps: tree creation, filtering and restitution. It is a tree where the nodes represent sets of flat zones. The Max-tree node \( C_h^k \) is the set of pixels at level \( h \) of peak component \( P_h^k \). The root node represents the set of pixels belonging to the background, i.e. the set of pixels with the lowest intensity in the image. Each child node represents brighter pixels. The Max-tree is a rooted tree, which means that each node has a pointer to its parent, i.e. the nodes corresponding to the components with the highest intensity are the leaves. Hence the name Max-tree: the leaves correspond to the regional maxima. This means that the Max-tree can be used for attribute openings or thinnings. Conversely, a tree in which the leaves correspond to the minima is called a Min-Tree and can be used for attribute closings or thickenings. A binary image can be considered as a grayscale image, where background consists of black pixels and the foreground components are replaced by white pixels. During the construction phase, the Max-tree is built from the flat zones of the image. This is illustrated in figure 5.1. Algorithms for building the Max-tree will be discussed in chapter 6.

5.2 Filtering

After the Max-tree is built, the tree is processed during the filtering phase. Based on the criterion value \( T(P_h^k) \) of a node \( C_h^k \), the algorithm takes a decision on whether to preserve or to remove it. Two classes of strategies exist [13, 16]: (i) pruning strategies, which remove all descendants of \( C_h^k \) if \( C_h^k \) is removed, and (ii) non-pruning strategies in which the parent pointers of the children of \( C_h^k \) are updated to point at the oldest "surviving" ancestor of \( C_h^k \). The effect of pruning and non-pruning strategies on a tree is demonstrated in figure 5.2, where the Min and Max decisions are pruning strategies and the Direct and Subtractive decisions are non-pruning strategies. These decisions are explained below. As can be seen, a non-pruning strategy removes the nodes that do not satisfy the criterion, but child nodes that do are not removed, where removing a node means that the pixels belonging to that node are lowered in gray level to that of its ancestor. Note that
both of these strategies will yield the same result when an increasing criterion is used, i.e. an opening or closing.

The way a Max-tree is filtered depends on the rule that was used. Salembier [14] describes four rules: the Min, the Direct, the Max and the Viterbi rule. In addition to these, the Subtractive rule was introduced in [16]. The decisions of these rules are as follows:

- **Min**: A node $C^k_h$ is removed if its criterion value $T(P^k) < r$ or if one of its ancestors is removed.
- **Direct**: A node $C^k_h$ is removed if $T(P^k) < r$.
- **Max**: A node $C^k_h$ is removed if $C^k_h$ and all of its descendant nodes do not satisfy a criterion $T$.
- **Viterbi**: The removal and preservation of nodes is considered as an optimization problem. For each leaf node the path with the lowest cost to the root node is taken, where a cost is assigned to each transition. In section sec:viterbi this rule is described in more detail.
- **Subtractive**: Like the Direct rule, but the descendents are lowered by the same amount as $C^k_h$ itself.

The effect of these rules on the peak components is demonstrated in figure 5.2.

### 5.3 The Viterbi rule

The Viterbi rule was described by Salembier et al. in [13, 14]. The idea is to consider the decision process of removing and preserving nodes as an optimization problem. First, each node $C^k_h$ is assigned two states: Preserve $P^k_h$ and Remove $R^k_h$. Each possible state of one node is linked with each possible node of its father by means of transitions and resulting in a trellis structure. An example of a simple tree and its corresponding trellis is shown in figure 5.3. Take for example node $C^0_h$. This node has two states, called $P^0_h$ and $R^0_h$. State $P^0_h$ is linked with the states $P^0_{h-1}$ and $R^0_{h-1}$ of parent node $C^0_{h-1}$. The same holds for state $R^0_h$.
Figure 5.2: Filtering with different rules: (a) peak components, (b) corresponding attribute values, (c-d) result after filtering with non-pruning strategies and (e-g) result after filtering with pruning strategies.

Figure 5.3: A simple Max-tree and the corresponding trellis.

This means that each node $C^k_h$ has four possible transitions with its parent $C^k_{h-1}$: $\mathcal{P}^k_h \rightarrow \mathcal{P}^k_{h-1}$, $\mathcal{P}^k_h \rightarrow \mathcal{R}^k_h$, $\mathcal{P}^k_h \rightarrow \mathcal{R}^k_{h-1}$ and $\mathcal{R}^k_h \rightarrow \mathcal{R}^k_{h-1}$. To each of these transitions a cost can be assigned as follows:

\[
\begin{align*}
\text{Cost}(\mathcal{P}^k_h \rightarrow \mathcal{P}^k_{h-1}) & = r - T(C^k_h) \\
\text{Cost}(\mathcal{P}^k_h \rightarrow \mathcal{R}^k_h) & = r - T(C^k_h) \\
\text{Cost}(\mathcal{P}^k_h \rightarrow \mathcal{R}^k_{h-1}) & = T(C^k_h) - r \\
\text{Cost}(\mathcal{R}^k_h \rightarrow \mathcal{R}^k_{h-1}) & = \infty
\end{align*}
\]

Costs can be assigned in many ways, but the cost should reflect the reliability of a transition in such a way, that a low cost is assigned to a reliable transition and vice versa. As can be seen the costs as defined above do reflect this reliability. The transition $(\mathcal{P}^k_h \rightarrow \mathcal{R}^k_{h-1})$ is often undesirable, because this makes the filter less robust; the output can be similar to the Direct rule. Assigning an infinite cost makes this transition impossible.

Along these transitions paths exists from each leaf node to the root. For each leaf node, the optimum path, which is the path with the lowest cost is chosen. The cost of a path is the sum of the costs of the transitions along that path. Note that $\text{Path}^{k,i}_{h} \subseteq \mathcal{P}$ is the optimum path starting at the leaf node $C^k_i$ at level $i$ and ending at the preserve state of node $C^k_h$. $\text{Path}^{k,i}_{h} \subseteq \mathcal{P}$ is like $\text{Path}^{k,i}_{h} \subseteq \mathcal{P}$.
but ending at the remove state of node $C_h$. If the assumption is made that the root node cannot be removed, then each path ending at the root node should do so in the preserve state. The search for the optimum path can be solved by the Viterbi algorithm, using the following recurrent rules:

If

$$\text{Cost}(\text{Path}_{h+1}^p) + \text{Cost}(\mathcal{P}_{h+1} \rightarrow \mathcal{R}_h) \leq \text{Cost}(\text{Path}_{h+1}^p) + \text{Cost}(\mathcal{R}_{h+1} \rightarrow \mathcal{P}_h)$$

then

$$(\text{Path}_{h+1}^p) = \text{Path}_{h+1}^p \cup \{\mathcal{P}_{h+1} \rightarrow \mathcal{R}_h\}$$

else

$$(\text{Path}_{h+1}^p) = \text{Path}_{h+1}^p \cup \{\mathcal{R}_{h+1} \rightarrow \mathcal{P}_h\}$$

(5.1)

If

$$\text{Cost}(\text{Path}_{h+1}^p) + \text{Cost}(\mathcal{P}_{h+1} \rightarrow \mathcal{P}_h) \leq \text{Cost}(\text{Path}_{h+1}^p) + \text{Cost}(\mathcal{R}_{h+1} \rightarrow \mathcal{P}_h)$$

then

$$(\text{Path}_{h+1}^p) = \text{Path}_{h+1}^p \cup \{\mathcal{P}_{h+1} \rightarrow \mathcal{P}_h\}$$

else

$$(\text{Path}_{h+1}^p) = \text{Path}_{h+1}^p \cup \{\mathcal{R}_{h+1} \rightarrow \mathcal{R}_h\}$$

(5.2)

Here $\text{Path}_{h+1}^p$ denotes the optimum path starting at the leaf node and ending at the preserve state of the node at level $h+1$. When a node $C_h$ has more than one child, more than one optimum paths end at $C_h$. In that case, the optimum path ending at the preserve state of $C_h$ is the union of all optimum paths ending at the preserve state of $C_h$. The cost of this unified path is the sum of all individual costs. The path ending at the remove state of $C_h$ and its cost are computed in the same way. Once the optimum path is found, the nodes corresponding to remove states in this path are removed.

5.4 Image restitution

After this filtering step, the image is updated according to the changes made in the Max-tree during the image restitution phase. This is usually straightforward, since each pixel $p(i,j)$ belongs to a node $C_h$, so $p(i,j)$ is assigned the grayvalue $h$ of $C_h$. Note that if a node was not changed during the filtering process, the corresponding pixels are unaffected, while removing a node means its pixels now belong to its ancestor, which grayvalue will be assigned to these pixels.

However, things get more complicated if a flat zone is defined as all adjacent pixels with intensities within a certain range. This leads to a “softer” binarization, which is less sensitive to small changes such as noise, but in this case, the pixels of a preserved node should not be updated, since this will assign the same value $h$ to all these pixels. A simple solution would then be to update pixels belonging to a removed node in the same as described above.

5.5 Image decomposition

A Max-tree can be filtered using any of the five rules described above. For increasing criteria, all of these rules yield the same result. Which of these rules is the most appropriate for non-increasing criteria depends mainly on the application, e.g., image filtering or decomposition.

Suppose one wants to filter the original image of figure 5.4. From an intuitive point of view, the Min-decision tends to remove too much, while the Max-decision tends to preserve too much and the Direct is not robust, i.e., it creates sharp edges in the image. Finally, the Subtractive decision can darken parts of the image. The Viterbi rule is not used here, because it is a pruning strategy like the Min and the Max rule. As such, it can remove components that should not be removed and preserve components that do not satisfy the criterion.

If the application is a decomposition of image $f$, one would intuitively require that after filtering with a attribute thinning $\mathcal{T}$ no structures which do not meet the criterion $T$ are present in the resulting image. Moreover, the difference image $f - \mathcal{T}(f)$ should only contain structures which do not meet $T$. These intuitive requirements mean that all peak components of $\mathcal{T}(f)$ meet $T$ and all peak components of $f - \mathcal{T}(f)$ do not. More formally:

$$\mathcal{T}(\mathcal{R}_h(\mathcal{T}(f))) = \mathcal{R}_h(\mathcal{T}(f))$$

(5.3)
and

$$\mathcal{T}_h(f - \nu^T(f)) = \emptyset$$  \hspace{1cm} (5.4)$$

for all $h$.

Which of the rules fulfills these requirements, is demonstrated in an example in figure 5.4, where $2\pi I/A^2$ is used as shape criterion. $2\pi I/A^2$ is the $I/A^2$ attribute, but normalized in such a way that an attribute value of 1 corresponds to a perfect circle. As can be seen in the figure, the subtractive rule is the one which works best for decomposition: the filtered image $\nu^T(f)$ contains only elongated structures and $f - \nu^T(f)$ contains only compact structures.

![Fig 5.4: Grayscale shape decomposition of image $f$ using 2$\pi$I/A$^2 \geq r$ thinning with $r = 1.1$. Per column is shown: original image $f$, filtered image $\nu^T(f)$, difference image $f - \nu^T(f)$ and filtered difference image $\nu^T(f - \nu^T(f))$ (the latter two columns have been contrast enhanced for clarity). The four filtering rules are shown per row, The Min filter removes the small bars within the larger circles from the image, whereas the Max pruning strategy leaves the large circles in the filtered image. Of the non-pruning rules, the Direct method has the problem that the difference image contains non-compact details, as can be seen by re-filtering with $\nu^T$ (fourth column).]
Chapter 6

Algorithm

6.1 Max-tree implementation

As mentioned earlier, a tree structure is used here to represent and process images. This way, image processing and analysis consists of three consecutive steps: construction, processing/analysis and restitution. The algorithms of these steps are described in the following sections.

6.1.1 Max-tree construction

First the tree must be constructed. This means that an algorithm is needed to find the flat zones and consequently to use these to build the Max-tree based on peak components. There are several algorithms that can be used. Originally, Salembier et al. [14] used a flood-fill algorithm to find the flat zones. This algorithm constructs the Max-tree, where each node represents a peak component. It should be noted that the component trees as introduced by Jones [4] also use a tree representation of the image. Although their representation is different from the Max-tree, it should be possible to adapt their algorithm to Max-tree. A disadvantage of component trees is their use of Vinçent's algorithm, which for an image of \( N \) pixels has a worst case running time of \( O(N^2 \log N) \) and which is very sensitive to the size of the attribute threshold. On the contrary, Salembier's Max-tree uses a flood-fill algorithm for tree construction, which has a running time linear with \( N \) and does not depend on the size of the attribute threshold.

Wilkinson and Roerdink [20] proposed a method to implement attribute operators based on Tarjan's Union-Find algorithm. Their method seems to be more efficient than Salembier's, but it does not construct a tree currently, which makes it less versatile. It should be noted however that work is in progress to define a method which uses Tarjan's algorithm to build a Max-tree. This would combine the best of both worlds; a versatile and efficient algorithm. During the graduation project Salembier's method was used and extended. It is this algorithm that is described below.

The Max-tree construction algorithm makes use of the following auxiliary functions and data:

**ORI** This is the original gray image. Usually, the image has 256 gray levels, which means that this array can be implemented as a one-dimensional array of \( width \times height \) bytes, where \( width \) and \( height \) refer to the dimensions of that image.

**HQueue** A hierarchical queue, called HQueue is used by the algorithm to store pixels temporarily. It consists of a set of simple queues; for each gray level a separate queue. In the case of an 8-bit per pixel grayscale image, which has 256 gray levels, this means that this HQueue consists of 256 simple queues. HQueue can be accessed by the following three functions: HQueueFirst\((h)\), HQueueAdd\((h,p)\) and HQueueEmpty\((h)\). The last one returns true if HQueue at level \( h \) is empty and false otherwise. HQueueFirst\((h)\) extracts the first pixel of HQueue at level \( h \) HQueueAdd\((h,p)\) adds pixel \( p \) with gray level \( h \) in HQueue at level \( h \).
STATUS  Information about the status of a pixel $p$ is stored in $STATUS[p]$. This array of pixel
states has the same dimensions as $ORI$. If a pixel $p$ is still untouched by the algorithm,
$STATUS[p]$ is Not — analyzed. When the algorithm has added pixel $p$ for processing in
$HQueue$, $STATUS[p] = In — the — queue$. A processed pixel $p$ belongs to a node $P^h_k$. In
that case, $STATUS[p] = k$, which means that $P^h_k = P_{ORI}[p]$

NodeAtLevel The number of elements of this boolean array is equal to the number of gray
levels. This array is used to test quickly if a node has been found at level $h$, in which case
$NodeAtLevel[h]$ is true, otherwise it is false.

numberno nodes This is an array denoting the number of nodes $\#P^h_k$ for each gray level $h$. The
number of black nodes, which is always 1, is denoted by the first element of this array and
the number of white nodes is denoted by the last one.

Before the actual algorithm can be used, the arrays mentioned above need to be initialized.
First, each queue of $HQueue$ is initialized as empty. Then, the pixel $p$ of the input image $ORI$ with
the lowest intensity $h_{min}$ is added to $HQueue$. All entries of the $STATUS$ array are initialized as
Not — Analyzed. The entries of arrays $NodeAtLevel$ and $numberno nodes$ are initially set to $false$
and 0 respectively.

After the initialization, the main routine, called flood, is applied where its parameter $h$
is assigned the value of $h_{min}$. The idea is that flood scans all pixels of $ORI$ at level $h$ belonging to
the peak component $P^h_k$, where $k$ is determined by the pixels stored in $HQueue$ at level $h$. This is
possible, since the pixels stored at a level $h$ in $HQueue$ always belong to the same peak component
$P^h_k$. However, not all pixels at level $h$ belonging to that component are always in $HQueue$ when
this routine is called. Usually, most pixels of a component are added during the call. Note that in
this routine, $h$ refers to the lowest intensity of this peak component, which is not necessarily the
same as the lowest intensity of the whole input image, $h_{min}$, as mentioned above.

Since a peak component $P^h_k$ consists of pixels with intensity $\geq h$, smaller peak components
with intensities $> h$ can be contained in $P^h_k$. This means that flood not only needs to build a
node for $P^h_k$, but also for each peak component contained in $P^h_k$. This means that a subtree needs
to be built. This subtree is built by calling flood for each of these peak components contained in
$P^h_k$. Since these smaller ones may contain even smaller ones, flood is recursive. In other words,
flood generates a subtree representing the set of pixels belonging to the same peak component.
By calling flood with the global $h_{min}$ as argument and a pixel with this intensity previously added
to $HQueue$, the Max-tree representing the whole image is built.

After all pixels of a peak component $P^h_k$ are scanned, the newly created subtree is linked to its
parent. The parent node is the one which corresponds to pixels surrounding $P^h_k$ with intensity lower
than $h$, but higher than all other neighbor pixels of $P^h_k$. Algorithm 6.1.1 shows the pseudocode
Salbemier described in [14] for the flood-algorithm. This was implemented as follows. First, the
flood routine extracts a pixel $p$ from $HQueue$ at level $h_{min}$. The neighboring pixels of $p$ with
status Not — analyzed are then added to the $HQueue$ and thus they are marked in $STATUS$
as In — the — queue. Since these visited pixels belong to other peak components, there will be
nodes at the levels of these, thus the entries of $NodeAtLevel$ corresponding to these intensities
are set to true. If a neighbor has a higher intensity than pixel $p$, this neighbor will belong to a
child node of the node $p$ will belong to. Therefore, a subtree $t$ is built, where each node has a
larger intensity than $h$. This is done by calling recursively flood, now with the intensity of that
neighbor as argument. This procedure of extracting and adding pixels is repeated until $HQueue$
at level $h$ becomes empty.

Now, a new node will be added to the tree, so the entry of $numberno nodes$ belonging to this
level is increased by one. The parent node $P^h_m$ of $P^h_k$ is the brightest neighbor of $P^h_k$. Note that
any neighboring peak component is darker than $P^h_k$ itself, since pixels with the same or higher
intensity belong to $P^h_k$ itself. The intensity $m$ of parent $P^h_m$ can be found using the $NodeAtLevel$
array, since $m$ is the highest entry $i$, for which $NodeAtLevel[i] = true$ still holds. Note that $i$ is
always smaller than $h$. Value $k$ of $P^h_k$ identifies which node at level $h$ it is. The first node created
at level $h$ becomes $P^0_h$, the second has $k = 1$, the third $k = 2$ and so on. Since the algorithm starts
flood(h)
while not hqueue-empty(h)
    p <- hqueue-first(h)
    STATUS(p) <- number-nodes(h)
    for every neighbor q of p
        if STATUS(q) == 'Not-analyzed'
            hqueue-add(ORI(q), q)
            STATUS(q) <- 'In-the-queue'
            node-at-level(ORI(p)) <- true
        if (ORI(q) > ORI(p))
            m = ORI(q)
            repeat
                m <- flood(m)
            until m = h
    number-nodes(h) <- number-nodes(h) + 1

m <- h - 1
while m >= 0 and node-at-level(m) = false
    m <- m - 1
if m >= 0
    i <- number-nodes(h) - 1
    j <- number-nodes(m)
    father of C\_i\_h <- node C\_j\_m
else
    C\_i\_h has no father (C\_i\_h is root node)
node-at-level(h) <- false
return m

Algorithm 6.1.1: Salembier's flood algorithm

counting k at zero, for a new node, k is equal to the number of nodes found so far at this level minus one. Thus for P_h^n it can be computed as follows: k = numbernodes[h] - 1. If m cannot be found, this new node is the root node and k = 0.

The implementation of the flood algorithm uses an array of Max-Tree nodes. Since the number of nodes is unknown when flood is called, the size of that array that is allocated beforehand, should be at least the maximum number of nodes the image can contain, which is just the number of pixels.

6.1.2 Max-tree processing

After the tree is built, it can be used for processing or analysis of the input image, since the Max-tree is a representation of the input image. In both cases an attribute is computed for each node, which value is based on the characteristics of that node. An example of such an attribute is the area of the peak component the node represents. In the current implementation, these attribute values are computed during the construction of the Max-tree and they are stored in the nodes.

Image processing or filtering now consists of traversing the Max-tree and removing nodes that do not satisfy some criterion. In the image restitution step, the image is updated according to these modifications of the Max-tree. In the case of an increasing criterion, processing the Max-tree is straightforward; the nodes that satisfy a criterion are preserved, the others are removed. In the non-increasing case, it sometimes happens that a node that according to its attribute value should be preserved has a parent that should be removed. Solutions to solve this problem were described in section 5.2. However, when one wants to process a Max-tree with two or more criteria, from which at least one is non-increasing, another problem occurs. Suppose one wants to filter a
Max-tree using two criteria. In figure 6.1 an example of this is shown, where the peak components are to be filtered using two criteria. For each peak component, the two corresponding attribute values are shown, enclosed by brackets.

\[
\begin{align*}
\text{(a) Peak components} & \quad \text{(b) Attribute values} \\
\begin{array}{c}
P_3^1 \\
P_2^1 \\
P_1^1 \\
P_0^1 \\
\end{array} & \quad \begin{array}{c}
(30, 6) \\
(20, 1) \\
(10, 9) \\
(50, 8) \\
\end{array}
\end{align*}
\]

(a) Peak components (b) Attribute values

Figure 6.1: Example of using two criteria: the peak components of an image (a) and their attribute values (b)

The problem now is, how should the corresponding Max-tree be filtered. Intuitively, one requires that the result of this should be the same as filtering the tree twice using one criterion, where the order of criteria should be irrelevant. Therefore an operator \( \psi^{T_1,T_2} \) should meet the following requirement:

\[
\psi^{T_1,T_2}(f) = \psi^{T_1}(\psi^{T_2}(f)) = \psi^{T_2}(\psi^{T_1}(f)),
\]

(6.1)

where \( f \) is an image and \( T_1 \) and \( T_2 \) are two criteria. Likewise, this requirement can be extended for any number of criteria.

In section 5.2 different filter rules using one criterion were discussed. The extension of the Viterbi rule using two criteria is still under investigation. Let \( f \) be an image, \( \psi \) a filter and let \( T_1 \) and \( T_2 \) be two criteria, then the other rules can be extended as follows:

- **Min**: Performing \( \psi^{T_1,T_2}_{\min} \) should be equivalent with \( \psi^{T_1}_{\min}(\psi^{T_2}_{\min}(f)) \), which means a node should only be preserved when that node is preserved by both \( \psi^{T_1}_{\min} \) and \( \psi^{T_2}_{\min} \), which is the case when that node and its ancestors satisfy both criteria.

- **Direct**: A node is preserved if it satisfies all the criteria; otherwise it is removed.

- **Max**: A node is removed only if it does not satisfy at least one criterion and all of its descendents are removed as well.

- **Subtractive**: Extended in a similar way as the Direct rule.

The extension of the operations for any number of criteria is straightforward.

### 6.1.3 Image restitution

After processing a Max-tree, the image must be updated according to the changes made to the Max-tree. The way we do this, is by visiting each pixel of the image and updating its value according to the node it belongs to.

### 6.1.4 Max-tree analysis

The criterion values of the Max-tree nodes can also be used to extract information about the image. One usage of using a Max-tree for image analysis is for the computation of pattern spectra based on granulometries, which will be discussed below. Since the computation of these pattern spectra, like other image analysis methods, does not modify the image content, the image restitution step is usually empty, except for some temporary changes in the Max-tree for implementation reasons.

A pattern spectrum can be considered as an array of values, each value representing a size or shape class. For example, if the area is used as the criterion, the corresponding pattern spectrum
consists of an array of size classes. Compactness is an example of a shape criterion and the corresponding shapes classes vary from a very compact, such as a circle, to a very complex class on the other end of the spectrum. Objects belonging to a complex class often have many small features, although a spiral also belongs to a complex class.

A pattern spectrum can be computed by traversing the Max-tree, where for each node its attribute value and the contribution is computed. A contribution \( v_h^k \) for a node \( C_h^k \) with parent \( C_m^k \) is computed as follows: \( v_h^k = \text{area} \times (h - i) \), where area is the area of \( C_h^k \). This contribution is added to the entry of the pattern spectrum that represents the class the criterion belongs to. The multiplication is to give large distinctive image details a larger weight in the pattern spectrum. Often, there are many nodes with criterion values that belong to the same class. In this case, the value in the corresponding entry is the sum of the contributions of all these nodes.

The computation of a pattern spectrum can be considered as finding the criterion threshold for each node. This means the same problem occurs in the case of a granulometry of multiple criteria. In the case of two criteria \( T_1 \) and \( T_2 \), it can be solved as follows:

- **Min**: The contribution of a node \( C_h^k \) is added to the entry with index \( \text{pos} = (p_x, p_y) \), which is computed as follows: \( p_x = \min(T_1(C_h^k), p_x) \land p_y = \min(T_1(C_h^k), p_y) \), where \( p_x \) and \( p_y \) are the minimum criterion values of the descendents of \( C_h^k \).

- **Direct**: Filtering a Max-tree with the Direct Rule means that if a node \( C_h^k \) does not satisfy both criteria the pixels with level \( h \) belonging to \( C_h^k \) are replaced with the intensity of the parent of \( C_h^k \). When using this rule for the computation of pattern spectra, the contribution of the pixels of \( C_h^k \) at level \( h \) are added to the entry with index \( \text{pos} = (T_1(C_h^k), T_2(C_h^k)) \). Pixels belonging to \( P_h^k \) but not to \( C_h^k \), i.e. at graylevel higher than \( h \), contribute to the entry with index \( \text{pos} = (p_x, p_y) \), where \( p_x = \max(T_1(C_h^k), p_x) \) and \( p_y = \max(T_2(C_h^k), p_y) \). \( p_x \) and \( p_y \) are the maximum criterion values of the descendents of \( C_h^k \).

- **Max**: The contribution of a node is added to the entry with index \( \text{pos} = (p_x, p_y) \), where \( p_x \) and \( p_y \) are defined as above.

- **Subtractive**: The Subtractive rule adds the contribution of a node \( C_h^k \) to the entry with index \( \text{pos} = (T_1(P_h^k), T_2(P_h^k)) \).

The extension of the computation of granulometries for any number of criteria is straightforward.

### 6.2 Attributes

In section 3.4 many different kinds of attributes were discussed. Besides the usefulness of a criterion, its computational cost is also very important. This means that it must be possible to compute an attribute incrementally. The Max-tree algorithm provides an efficient structure for an incremental computation of attributes. The following ones were implemented incrementally during the graduation project:

- **Area**: This can be implemented efficiently and easily. The area of a node \( C_h^k \) is the number of pixels belonging to \( C_h^k \) summed with the areas of its children.

- **Area of the minimum enclosing rectangle**: In this implementation, the minimum enclosing rectangle is a bounding box enclosing the object, i.e. the box is not rotated to better fit that object. The minimum \((x_{\text{min}}, y_{\text{min}})\) and the maximum \((x_{\text{max}}, y_{\text{max}})\) coordinate values of each pixel belonging to \( P_h^k \) are computed. For \( x_{\text{min}} \), this is computed as follows. For each pixel \( p \) belonging to this node \( C_h^k \), the minimum of \( x_{\text{min}} \) and the \( x \)-coordinate of \( p \) is assigned to \( x_{\text{min}} \). Furthermore, for each child of \( C_h^k \) the minimum of the current \( x_{\text{min}} \) of \( C_h^k \) and the \( x_{\text{min}} \) of its child is assigned to the current \( x_{\text{min}} \). This computation is done in a similar way for the minimum \( y \)-coordinate \( y_{\text{min}} \) and the maximum \( x \)- and \( y \)-coordinates \( x_{\text{max}} \) and \( y_{\text{max}} \). The area is now computed as: \((x_{\text{max}} - x_{\text{min}}) \times (y_{\text{max}} - y_{\text{min}}) + 1\).
• **Length of the diagonal of the minimum enclosing rectangle**: Same as above, but now the length of its diagonal is used as the criterion: \( \sqrt{(x_{\text{max}} - x_{\text{min}})^2 + (y_{\text{max}} - y_{\text{min}})^2} \).

• **Perimeter**: This is discussed in section 6.2.2.

• **Complexity**: Perimeter and area are computed incrementally. At the end, complexity is computed as: \( \text{perimeter}/\text{area} \).

• **Simplicity**: Inverse of complexity, thus: \( \text{simplicity} = \text{area}/\text{perimeter} \).

• **Compactness**: This is computed as follows: \( \text{compactness} = (\text{perimeter}^2)/(4\pi \text{area}) \). This is a strict shape criterion under the condition that the computed (discrete) perimeter does not differ too much from the theoretical (continuous) perimeter. This condition is not met by the current implementation, see section 6.2.2.

• **Moment of inertia**: Discussed in section 6.2.3.

• **I/A^2**: This is the moment of inertia (I) divided by the square of the area. This is a strict shape criterion.

• **Jaggedness**: This criterion yields higher values for jagged objects and should give 1 for smooth ones. It is implemented as: \( (\text{AP}^2)/(8\pi^2 \text{I}) \), where A denotes the area, P the perimeter and I the moment of inertia. Like the compactness criterion, this one does also suffer from inaccurate results caused by the computation of the perimeter.

• **Entropy**: The entropy is a statistical measure for the amount of information in a signal. It is implemented as follows. First, for each gray value the number of pixels with this intensity and belonging to this node \( C_h^k \) is counted and put in an array. Since each child node has already computed its own array, the pixels belonging to children are not visited again, but the entries of their arrays are added to the array of \( C_h^k \). This array is used to build a histogram of probabilities \( p \), i.e. divide each entry by the number of pixels belonging to \( P_h^k \). Finally, the entropy is computed as: \( \sum_i p[i] \log(p[i] + \epsilon)/\log(2) \), where \( \epsilon \) is a small value to avoid \( \log(0) \) errors. Since an array is created for each node, this criterion, besides being computationally intensive, also uses large amounts of memory. Note that operators using this attribute are not filters.

Since images are often taken at different resolutions, the computed criterion values need to be scaled in such a way as if all images were taken at a uniform resolution. Let this “uniform” resolution be denoted by \( r_u \), let then assume an image taken at a resolution \( r_i \). Furthermore let there be a peak component \( P_h^k \) in this image with criterion value \( v \). How \( v \) should be scaled properly depends on the type of criterion.

The area attribute is quadratic in the sense that if the resolution is multiplied by 2, the area of \( P_h^k \) is increased by a factor of 4. On the contrary, the perimeter scales linearly with the resolution and shape criteria like \( I/A^2 \) are independent of the resolution the image was taken in.

### 6.2.1 Computing attributes

Salembier's Max-tree was extended in such a way that on the one hand increasing and non-increasing criteria can be computed in a flexible way and that they can be computed efficiently on the other hand. It should be noted that this approach is similar to the one proposed by Breen and Jones [1]. The computation is divided over four functions:

• **New**: This function is called once for each attribute. It should allocate and initialize a structure to contain intermediate results of the computation of one attribute. This structure is passed as argument to the other three functions. Furthermore, one pixel of the node is passed as argument to **New**. This avoids problems of initializing attribute data based on empty sets, such as the maximum.
Delete: This function frees all memory allocated by New for this attribute.

AddTo: Each time flood processes a new pixel this function is called. It should update the intermediate results in the structure.

Merge: Pixels belonging to children should also be used for the computation of the attribute of the current node. This function is called by flood with two arguments, the structures of the computation of this node and that of the child node, each time after processing a child.

Attribute: This should return the attribute value based on the computations performed by the previous functions. Sometimes the result needs to be computed from the entries of the attribute structure. These final computations are done by this function.

The extended flood-routine also has three function parameters for the New, the AddTo and the Merge functions. Other routines are also extended with function parameters. Selecting a criterion now consists of passing the corresponding functions as arguments to the routines.

6.2.2 Perimeter

There are many ways to compute the perimeter, but one method, called City-Block for now, was implemented efficiently during the project. In this case, the perimeter of $P_k^h$ is simply the number of pixels along the edge of $P_k^h$. An edge pixel is a pixel that has at least one neighbor that either does not belong to $P_k^h$ or does not belong to the image at all.

This can be computed incrementally. For each pixel $p$ belonging to $C_k^h$ the number of edges lower $\#L(p)$ and higher $\#H(p)$ than $h$ of that pixel are computed. The perimeter of $P_k^h$ is now $\sum_{p \in C_k^h} \#L(p) - \#H(p)$ increased by the perimeters of the children of $C_k^h$. This method of computing the perimeter is nearly as efficient as computing the area. A disadvantage is that the method yields results which are especially for ellipses too large compared with the desired output.

During the project two suggestions came for a better approximation of the perimeter. Contrary to the City-Block method, in these methods the boundary of an object consists besides of horizontal and vertical edges also of diagonal edges. The idea of the first method, called LargePerimeter for now, is that perimeter is the length of a rubber band along the object, where horizontal and vertical edges have a length of one and diagonal edges $\sqrt{2}$. Although a much better approximation than the City-Block, this method still yields results larger than the actual perimeter.

In the other suggestion pixels can be considered as squares with a needle placed in the center of each square. The perimeter is now the length of the boundary of the rubber band along the outer needles. The length of the edges is the same as above. Contrary to the other methods described above, this method tends to give results which are a bit smaller than the actual perimeter.

![Figure 6.2: Different approximations of the perimeter](image)

As can be seen in figure 6.2 the problem of these approximations is caused by the diagonal sections. Compared to the perimeter of an object in the continuous case, in the digitized version of that object some parts are considered by the first suggestion to belong to that object while the
other suggestion does omit some parts which do belong to that object in the continuous case. It seems that compared to the ideal perimeter the deviation of the first is the same as that of the second suggestion. It seems likely that an average of both results will be good approximation of the perimeter. This however is a topic for future work.

6.2.3 Moment of Inertia

The moment of inertia \( I \) for an object in a two-dimensional image is defined by:

\[
I = \sum_{n} (x - x_c)^2 + (y - y_c)^2
\]

The variable \( n \) runs over all pixels \( p = (x, y) \) belonging to the object and \( x_c \) and \( y_c \) are the coordinates of the center of the object. This formula can be rewritten as:

\[
I = \sum_{n} x^2 + \sum_{n} y^2 - \frac{(\sum_{n} x)^2}{n} - \frac{(\sum_{n} y)^2}{n}
\]  
(6.2)

Since we are using discrete images, this equation needs a correction for the moment of inertia of a single pixel. This is computed as follows:

\[
\int_{-0.5}^{0.5} \int_{-0.5}^{0.5} x^2 + y^2 dy dz \approx \frac{1}{6}
\]  
(6.3)

This correction is needed for each of the \( n \) pixels belonging to the object. This yields the following equation:

\[
I = \sum_{n} x^2 + \sum_{n} y^2 - \frac{(\sum_{n} x)^2}{n} - \frac{(\sum_{n} y)^2}{n} + \frac{n}{6}
\]  
(6.4)

This means that for each node five values are required: area, \( \sum_{n} x \), \( \sum_{n} y \), \( \sum_{n} x^2 \), \( \sum_{n} y^2 \). At the end of the computation, the above equation can be used to compute the moment of inertia using these values.

This way, the moment of inertia can be computed incrementally, since these values can be passed on from child nodes to their parents, who can add these to their corresponding values.

6.2.4 Mappers

A pattern spectrum is implemented here as an array. Each entry represents a size or shape class of respectively the size or shape distribution. These entries can be accessed by their index. The implementation was done in the programming language C, where array indices are integers, starting with 0, while a floating point representation was used for the attribute values. The range of possible attribute values can be considered to be continuous and infinite.

The value of each entry of the pattern spectrum is the sum of the individual contributions of the nodes belonging to the size or shape class that entry represents. Let \( C_h^k \) be a node with parent \( C_{h-1}^{i} \), the contribution of \( C_h^k \) to a pattern spectrum entry \( i \) is \( (h - i) \cdot \text{area} \), where \( \text{area} \) is the area of \( C_h^k \). Note that the area of a node is the area of the pixels of \( P_h^k \) at level \( h \).

The continuous and infinite set of criterion values needs to be mapped to the discrete and finite set of array indices. This is done by a mapper function, which takes a criterion value as input and returns the corresponding array index. This criterion value should be in a fixed range \([D_0, D_1]\). Values lower than \( D_0 \) are mapped to the same class \( D_0 \) belongs to and values greater than or equal to \( D_1 \) are mapped to a special "out-of-range" class. Let \( N_c \) be the number of classes, excluding out-of-range classes, the following mapper functions were implemented to compute the array index \( i \) corresponding to the attribute value \( v \):

- **Linear**: The range \([D_0, D_1]\) is divided into equidistant classes: \( i = (v - D_0)N_c/(D_1 - D_0) \).
6.3 Computing feature vectors

Having computed the pattern spectra, the last step before the identification can begin is mapping a pattern spectrum to a feature vector. This discussion is limited to feature vectors based on two dimensional pattern spectra, which is the case in the application of diatom identification described in the next chapter. Due to the large amounts of data it requires, the extension to higher dimensions will probably be rare for identification purposes. The extension to higher dimensions should however be straightforward.

6.3.1 Mapping a pattern spectrum to a feature vector

The mapping of a two dimensional pattern spectrum to a one dimensional feature vector can be done in various ways. The following three methods were implemented:

Matrix The pattern spectrum is divided into 16 sections of equal size, i.e. four columns and four rows. The value of an element of the feature vector is the sum of the entries belonging to the same section.

Moments This method uses moments to compute the feature vector \( x = \{x_0, x_1, \ldots, x_{15}\} \) out of the pattern spectrum. Instead of using the discrete pattern spectrum array, this method uses the criterion values of the Max-tree directly for more precise results. This way, the pattern spectrum can be considered as being continuous. Four kinds of moments [3] are computed:

Moments:

\[
 m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) \, dx \, dy
\]  

(6.5)

Central moments:

\[
 \mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q f(x, y) \, dx \, dy
\]  

(6.6)

Normalized moments:

\[
 n_{pq} = \frac{m_{pq}}{m_{00}}
\]  

(6.7)

Normalized central moments:

\[
 \eta_{pq} = \frac{\mu_{pq}}{\mu_{00}}
\]  

(6.8)

where

\[
 \bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}} \quad \text{and} \quad \gamma = \frac{p + q}{2} + 1
\]

In the implementation the integrations \( \int_{-\infty}^{\infty} \) in these definitions are replaced by \( \sum_{D_{1}}^{D_{2}} \), where \( D_{0} \) and \( D_{1} \) specify the range of the criterion as discussed earlier. The \( p \) and \( q \) specify the order of the moment on respectively the horizontal and the vertical axis of the pattern spectrum.
The Moments method creates a two dimensional array \( A \) of \( 4 \times 4 \) elements using these moment functions, where the \( p \) variable runs over the horizontal and \( q \) over the vertical axis. The normalized central moments are the most appropriate to use, but \( \eta_{00} = 1 \) and \( \eta_{10} = \eta_{01} = 0 \). This method computes \( A \) as follows:

\[
A[p, q] = \begin{cases} 
    m_{00} & \text{if } p = 0 \land q = 0, \\
    \eta_{01} & \text{if } p = 0 \land q = 1, \\
    \eta_{10} & \text{if } p = 1 \land q = 0, \\
    \eta_{pq} & \text{otherwise.}
\end{cases}
\]

The mapping of \( A \) to \( \vec{v} \) is performed as follows: \( \vec{v}(4y + x) = A[x, y] \).

**Selection** The user can select which entries of the pattern spectrum are useful and how they should be mapped to the 16 elements of the feature vector. For our application we used some statistical measures like the variance and the entropy to find the important entries. If more than one entry should be mapped to the same vector element, the result value is the sum of all these entry values.

The feature vector described above is not necessarily the final one as will be explained below.

### 6.3.2 Combination

Till now the assumption was made that pattern spectra were based on size or shape distributions. However, many images are identified using both the bright and the dark patterns. To get information about the dark patterns anti-size and anti-shape distributions are required, which are based on closings and thickenings respectively. Instead of designing new algorithms for the computation of pattern spectra based on these anti-size and anti-shape distributions, these were implemented as the computation of a pattern spectra of an inverted image.

A question now arises if the pattern spectrum based on the normal or the inverted image should be used, or maybe even a combination of these two. Therefore the following combinations were implemented:

- **Normal**: The pattern spectrum based on the normal image is used. This means that only information about the bright structures in the image is known.

- **Inverted**: This one uses the pattern spectrum of the inverted image, which means that the dark structures in the image can be investigated.

- **Add**: Two temporary feature vectors are computed according to the methods Normal and Inverted described above: \( \vec{v}_n \) and \( \vec{v}_i \) respectively. The resulting feature vector \( \vec{v}_r \) is computed as: \( \vec{v}_r = \vec{v}_n + \vec{v}_i \). An advantage of this method is that bright and dark image details are present in this vector. Furthermore, dark and bright details with the same criterion values are represented by the same class. This means that this feature vector is invariant to inversion, i.e. inverting an image does not change the result, which is useful for diatom identification, because choosing a different focus depth can make bright structures turn dark, or vice versa.

- **Join**: Like the previous method, two vectors \( \vec{v}_n \) and \( \vec{v}_i \) are computed. Now the result \( \vec{v}_j \) is the concatenation of \( \vec{v}_n \) and \( \vec{v}_i \). Note that this vector is twice as large as the results of the other methods. Like the previous method bright and dark image details are present in the result, but now bright and dark details with the same criterion values are represented by different classes. The vector \( \vec{v}_j \) is not invariant to inversion, which can be a disadvantage, but on the other hand this also means that \( \vec{v}_j \) now contains more information about the image than with the previous method.
Chapter 7

Identification of diatoms

In this chapter an application of diatom identification using mathematical morphology is given together with some test results.

7.1 Diatom identification using mathematical morphology

The goal of the ADIAC project is to develop software for an automatic identification of diatoms. One part is about capturing images automatically. This part is beyond the scope of this thesis, thus it will be assumed that grayscale images of diatoms in valve view are available.

The main part is the development of feature extraction tools. Different methods are investigated in the ADIAC project for the computation of these features, including the use of: invariant moments, Fourier descriptors, curvature scale space, and size/shape distribution. Some of these use the information about the contour of the valve only, some use only its ornamentation and others make use of all available information of the valve to compute the feature vectors. During this graduation project a size/shape distribution of the valve ornamentation based on mathematical morphology is used to compute the feature vectors.

Since the feature extraction tools need to know the position and the dimensions of the valve in the image, contour extraction tools are also developed. This is done by other partners in the ADIAC project. The idea of these tools is that they output a file containing pixel coordinates of the contour of that diatom. The feature extraction tools now have two inputs: a grayscale image of a diatom and a corresponding contour file. These tools compute then a feature vector which is passed to the integration module.

The task of this module is to glue all components together. The feature extraction tools receive their inputs from this module and output a feature vector. The feature vectors of the extraction tools are combined by the integration module and then passed to a classification tool, like C4.5[11], which uses its input to find the best match.

7.2 Feature extraction

In the previous chapter several parameters were discussed on which the contents of the final feature vector depends. How we used these parameters for the identification of diatoms is described in this section.

7.2.1 Choice of criteria

One such a parameter is the choice of the filtering strategy. Four methods were discussed in chapter 5. It was concluded that the subtractive rule is the most appropriate for image decompositions. For this reason, the subtractive rule was used for the computation of pattern spectra,
Another parameter is the set of criteria to be used for the computation of the size and shape distributions. Since these distributions are based on filters, the corresponding criterion must be idempotent, which for example is not the case with the entropy criterion mentioned in section 6.2. The area is used as the size criterion. Two other size criteria described in the same section are the length and the area of the minimum enclosing rectangle. The advantage of the area as the size criterion over the other two is that the area is invariant to rotation.

In the same section some shape criteria were discussed. Most of them were based on the perimeter. Due to the inaccuracies of the current perimeter implementation, these perimeter-based criteria cannot be used as a strict shape criterion. This means that from the current implementations $I/A^2$ is the only accurately implemented shape attribute. The $I/A^2$ criterion did provide useful shape information. Using this criterion, not just elongated objects are clearly distinguishable from the compact ones, but also circles, rectangles and ellipses were placed in separate shape classes of the pattern spectra. Even circles and squares can be distinguished when needed.

For these reasons, during the project the area was used as the size criterion and $I/A^2$ was used as the shape criterion. Pattern spectra based on multiple criteria were discussed in sections 6.1.2 and 6.1.4. For the case of diatom identification pattern spectra were used based on the area and the $I/A^2$ criteria for the computation of feature vectors.

The diatom images of the ADIAC project were used during the graduation project. These images were taken at different resolutions, so the criterion values were scaled afterwards to avoid resolution dependencies in the results. The resolution a diatom image was taken in is described by its pixel width in an index file. The unit of pixel width in this file is microns. The choice of a uniform pixel width is arbitrary, but in this thesis one micron is used. This means that a peak component consisting of $n$ pixels of an image with pixel width $w$ has an (scaled) area of $n/w^2$.

### 7.2.2 Mapping to a pattern spectrum

Using the parameters described above, combined size/shape distributions can be computed. The information produced by these distributions is mapped into pattern spectra, which are two-dimensional arrays here. These mappings were discussed in section 6.2.4. The area criterion is quadratic, i.e. an object twice as large has four times the number of pixels, which means that a Sqrt-mapper would be the obvious choice. However, smaller areas are the most important ones, thus the Log2-mapper was used instead, where the classes representing the smaller criterion values are more finely binned than they would if a Sqrt-mapper was used instead.

Experiments with the $I/A^2$ criterion shows that perfect circles have the minimum value of $1/(2\pi)$, but that the value increases dramatically when the aspect ratio is changed a little bit. Thus a mapper is desired where the classes representing the smaller values are the most finely binned; so again the Log2-mapper is used here. It should be noted that in our implementation of the $I/A^2$ attribute, we multiplied the result with $2\pi$.

The range $[D_0..D_1)$ was chosen by experiment. A large set of diatom images was used to compute the minimum and maximum attribute values. These values were assigned to $D_0$ and $D_1$ respectively. In most cases $D_0$ would be the minimum value of $I/A^2$, but it can be higher if circular objects are not present or not important in the application.

The number of classes was chosen fairly arbitrarily to be 30. However, low values here would result in a binning which is not fine enough and high values would result in very large pattern spectra that consume large amounts of memory and probably are very sensitive to noise.

### 7.2.3 Feature vectors

The identification of diatoms is done using one-dimensional feature vectors, so the last step is the mapping of a two-dimensional pattern spectrum to a feature vector.

The way this mapping is done depends on the parameters described in the previous chapter: the combination and mapping method. A test was conducted on a large set of diatoms with all possible configurations of these two parameters. This test showed that the configuration of Join
as the combination method and Matrix as the mapping method gave the best results. These and other results can be found in section 7.4. It should be noted however, that the selection array used by the mapping method Selection was far from perfect. A more careful designed selection array can improve the results of this method.

7.3 Statistics

The performance of the feature vectors described in the previous sections was tested in three different ways:

- Leave-one-out
- Retrieval efficiency
- C4.5

These techniques are explained below. Identification results using these methods can be found in section 7.4. All of these methods use a list of feature vectors. For each of the vectors in this list the class it belongs to is known. The feature vector of the sample is compared with each entry in the list. The class identified is the one which is the most similar with the sample. Examples of possible classes are the taxon or the genus of the diatoms.

7.3.1 Leave-one-out

This method requires that all feature vectors in the list and the one from the sample are normalized elements-wise. Let $V(i,j)$ be the $i$th element of the $j$th vector in the list, then $X = \bigcup_{j} \{V(i,j)\}$.

The following four methods were implemented to normalize a value $S_i$:

- Average: $A_{\text{avg}}(X)$
- Min/Max: $\frac{S_i - \text{min}(X)}{\text{max}(X) - \text{min}(X)}$
- Standard deviation: $\frac{S_i - \text{stddev}(X)}{\sigma}$
- Median: $\frac{S_i - \text{median}(X)}{Q_3 - Q_1}$

The value $S_i$ is the $i$th element of a feature vector. Note that $S_i$ can be from a vector in the list or from the sample. For algorithms to compute the median $\text{median}(X)$ and the standard deviation $\text{stddev}(X)$, the reader is referred to [10]. The computation of the first and the third quartile, $Q_1$ and $Q_2$ respectively, is done in a similar way as the computation of the median.

The actual identification process consists of computing a measure of similarity, called the error, between an entry $a$ in the list and the sample $b$. This is done for all entries in the list. The class most similar is the one with the lowest error. Three different methods were used to compute the error:

- Euclidean: $\sqrt{\sum_{i}(a_i - b_i)^2}$
- City-block: $\sum_{i}|a_i - b_i|
- Maximum error: $\max_{i}|a_i - b_i|

The identification performance of this method can be measured using the leave-one-out technique. It consists of iteratively extracting a feature vector from the list and then trying to match this vector with other vectors in the list. The one identified is the one with the lowest error. After matching a vector, that vector is put in the list again and the process continues with the next vector till all vectors are processed. The identification performance is then computed as the number of correctly identified diatoms divided by the number of entries in the list. Note that the list should contain at least two samples per class.

48
7.3.2 Retrieval efficiency

The leave-one-out technique computes for each feature vector in the list the errors with each other entry in that list. The result is a list of matches with each of those entries. After sorting this list of matches, the best match, i.e. the one with the lowest error, is the first match in this list.

The first technique, described above, used only the first match of each entry. This has some disadvantages. First, it can be useful to know how many of the samples in that class were among the best matches. Furthermore, the match with the lowest error is not always the correct class, but probably others are, e.g. the second or the third match. A technique that uses the information from the best \( m \) matches is the retrieval efficiency [6]. Let \( r_1, \ldots, r_m \) be the \( m \) top-ranked matches for the query \( q \) and let \( N_q \) be the number of available prototypes for the class that \( q \) belongs to, then the retrieval efficiency is defined as:

\[
\text{Retrieval efficiency: } \eta(q,m) = \frac{1}{N_q} \sum_{k=1}^{m} \text{Match}(q,r_k),
\]

where \( \text{Match}(q,r_k) = 1 \) if \( r_k \) and \( q \) belong to the same class, and 0 otherwise. When \( m \) is smaller than \( N_q \), the value of \( m \) should instead be used for \( N_q \). The average retrieval efficiency is computed by performing the retrieval efficiency for all entries in the list. Note that when \( m = 1 \) the average retrieval efficiency yields the same result as the leave-one-out technique.

The number of incorrect first rank matches divided by the total number of queries is called the error rate. Note that this is equal to one minus the average retrieval efficiency for \( m = 1 \).

7.3.3 C4.5

The third technique that was used is based on decision trees. A set of feature vectors is computed from a set of diatoms whose identities are known. This set of vectors is split in a training and a test set. Quinlan's C4.5 package [11] is then used to build a decision tree based on the training set; the so-called decision tree classifier. This decision tree can then be used a predictor to identify unknown diatoms. The test set can now be used to test the identification performance of this predictor.

A disadvantage of using decision trees as a classifier is that they are often unstable. A method to solve this problem is called bootstrap aggregating or bagging. The idea is to obtain an aggregated predictor by building multiple predictors, where the outcomes of these predictors are combined in some way. The sets to train these predictors are constructed by bootstrapping from the initial training set, which means that the samples for each training set are drawn at random, but with replacement, i.e. the same sample can occur more than once in a training set. The following steps are taken to use bagging with C4.5:

1. Divide the data into an initial training and test set.
2. Create several new training sets from the initial training set using bootstrapping. A decision tree should be built for each of these training sets.
3. Use the test set to evaluate these classifiers. Take a majority vote on the outcomes of each tree.
4. These steps should be repeated a number of times. Average the results.

7.4 Results

7.4.1 Performance using leave-one-out

The leave-one-out technique was first used on a set of 548 diatom images. This test was performed using all possible configurations of the mapping and the combination methods to find the best
configuration. Based on these results we decided to use the Matrix mapping method and the Join combination method. The identification results are shown in table 7.1. The first two columns show the different configurations of the mapping and the combination methods. The number of diatoms identified correctly are shown in the third and fourth column based on the correct genus and taxon respectively. The last column shows the number of diatoms of which the species was identified correctly, assuming the genus was known. It should be noted that this set of diatoms was far from ideal; often there were at most a few samples per taxon.

### 7.4.2 Performance using retrieval efficiency

Recently a test was performed using retrieval efficiency on a new set containing 808 images that consisted of 21 different genera and 38 different taxa. The results are shown in table 7.2. As above, the first two columns show the different configurations that were used. The third column shows the retrieval efficiency for the combination of the normalization and error methods that gave the best result for that configuration. The last three columns show the retrieval efficiencies the combination of the normalization and error methods for respectively $m = 1$, $m = 2$ and $m = 10$ that gave the best results for the Matrix and Join configuration, which are best results of all these configurations.

### 7.4.3 Performance using C4.5

The identification method discussed in this thesis was one of several strategies for the computation of feature vectors that were implemented by the ADIAC partners. A test performed using C4.5 by Michel Westenberg gave for these strategies the identification results shown in table 7.3. The set of diatom images contained 781 images and consisted of 37 different taxa. The different strategies are
shown in the first column; the results on the training and the test set using 10-fold cross-validation are shown in the second and third column. The last column shows the identification rates when bagging is used. In this case the training set consisted of 185 samples which is about 25% of the data set. The test set contained exactly five samples of each specimen. Furthermore the process of building the 25 trees was repeated 10 times.

The method size/shape distribution is the one discussed in this thesis. Other methods using ornamentation are: raphe, undulation, frequencies, striae and texture. Note that our method is among the best of these, only the texture method gives a slightly better result. It should be noted that the texture strategy uses several methods to compute the feature vector, which results in large vectors. One of these methods uses Gabor energy, which is computed for several filter orientations. This makes this method, contrary to our method, not rotation-invariant and computational intensive.

The results of table 7.4 were produced after combining several of these strategies. Combining

Table 7.3: Results using C4.5

<table>
<thead>
<tr>
<th>Method</th>
<th>training</th>
<th>test</th>
<th>bagging</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bern raphe</td>
<td>3.7%</td>
<td>3.7%</td>
<td>2.7%</td>
</tr>
<tr>
<td>symmetry class</td>
<td>9.3%</td>
<td>8.0%</td>
<td>7.2%</td>
</tr>
<tr>
<td>undulation</td>
<td>13.3%</td>
<td>11.0%</td>
<td>9.7%</td>
</tr>
<tr>
<td>frequencies</td>
<td>70.9%</td>
<td>34.3%</td>
<td>40.3%</td>
</tr>
<tr>
<td>striae</td>
<td>68.1%</td>
<td>38.8%</td>
<td>38.9%</td>
</tr>
<tr>
<td>invariant moments</td>
<td>87.6%</td>
<td>63.0%</td>
<td>67.9%</td>
</tr>
<tr>
<td>geometric</td>
<td>89.7%</td>
<td>68.5%</td>
<td>72.3%</td>
</tr>
<tr>
<td>simple shape</td>
<td>91.8%</td>
<td>68.9%</td>
<td>75.0%</td>
</tr>
<tr>
<td>form</td>
<td>91.5%</td>
<td>69.6%</td>
<td>75.8%</td>
</tr>
<tr>
<td>texture</td>
<td>95.0%</td>
<td>71.5%</td>
<td>84.3%</td>
</tr>
<tr>
<td>Fourier descriptors</td>
<td>95.8%</td>
<td>76.9%</td>
<td>84.1%</td>
</tr>
<tr>
<td>Faro curvature</td>
<td>73.9%</td>
<td>41.4%</td>
<td>43.6%</td>
</tr>
<tr>
<td>mach band</td>
<td>83.7%</td>
<td>43.8%</td>
<td>54.5%</td>
</tr>
<tr>
<td>characteristic profile</td>
<td>91.4%</td>
<td>70.0%</td>
<td>75.0%</td>
</tr>
<tr>
<td>Groningen size/shape distribution</td>
<td>94.1%</td>
<td>71.1%</td>
<td>81.4%</td>
</tr>
<tr>
<td>curvature scale space</td>
<td>93.2%</td>
<td>74.5%</td>
<td>79.0%</td>
</tr>
<tr>
<td>combined</td>
<td>99.0%</td>
<td>89.0%</td>
<td>96.9%</td>
</tr>
</tbody>
</table>

Table 7.4: Combined results

<table>
<thead>
<tr>
<th>Method</th>
<th>Identification Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bern</td>
<td>96.4%</td>
</tr>
<tr>
<td>Faro</td>
<td>83.9%</td>
</tr>
<tr>
<td>Groningen</td>
<td>89.6%</td>
</tr>
<tr>
<td>Contour</td>
<td>92.2%</td>
</tr>
<tr>
<td>Internal</td>
<td>91.6%</td>
</tr>
<tr>
<td>All</td>
<td>96.9%</td>
</tr>
</tbody>
</table>

all strategies based on ornamentation gives an 91.6% identification rate, which is impressive, since diatomists need both the contour and the ornamentation of a diatom for its identification.
Chapter 8

Conclusions

In this thesis we saw how image analysis tools from mathematical morphology can be used for the identification of diatoms. After the introduction of theory of diatoms and mathematical morphology, Salembier's Max-tree for image filtering was described. This was extended to the computation of pattern spectra based on size and shape, which were used to compute the feature vectors for identification.

8.1 Future research

During a project like this one always come across problems for future research, some of which were mentioned in previous chapters. We have seen how parts in a diatom image, like striae and the raphe can be described by their shape, e.g. compact or elongated. One problem is that sometimes these parts are connected in the image; the striae being connected with the contour for example. In that case, our method does not see the elongated structures which the striae usually are, but it just sees a complex structure containing the contour and many of the striae. A solution to this problem could be to remove the contour beforehand or maybe changing the connectivity in such a way that two components which are connected by a small bridge are considered as two separate components.

Furthermore, the feature vectors can be improved. The current mapping and combination methods were mainly chosen because they are fast and easy to implement. One can think of using principal component analysis on the pattern spectrum to find the interesting features.

Finally, new filters and attributes can be developed. In this thesis the current problems with the computation of the perimeter were already mentioned. A good implementation of the perimeter would make it possible to compute the compactness attribute. Currently, two pattern spectra are computed for each diatom image, because the filters we used cannot be used to investigate the dark and the bright structures at the same time. This means that if we have a bright diatom with striae consisting of dark lines with bright dots on it, these line patterns are never be considered as lines by our current filters. Auto-dual filters are known to be sensitive to both dark and bright image details.

8.2 Final notes

The Max-tree is an efficient and flexible way for image analysis and filtering. An area opening on a 1008 x 1018 diatom image required 1.75 seconds on a 233MHz Pentium-II. The required time for the computation of a pattern spectrum on that image is practically the same.

The method proposed for automatic diatom identification using the valve ornamentation is possible. On a test using C4.5 of 781 images 81.4% of the diatoms was identified correctly on the first rank. As diatomists cannot identify diatoms on the basis of its ornamentation alone, it cannot be expected that automatic identification using our method or any other texture method
will be able to do that. It can thus be concluded that diatom identification using mathematical
morphology is possible, but that a combination of diatom contour and ornamentation analysis is
required for the best results.
Bibliography


