ABSTRACT

Title of dissertation:	EFFICIENT IMAGE SEGMENTATION AND SEGMENT-BASED ANALYSIS IN COMPUTER VISION APPLICATIONS
	João V. B. Soares, Doctor of Philosophy, 2015
Dissertation directed by:	Professor David W. Jacobs Department of Computer Science

This dissertation focuses on efficient image segmentation and segment-based object recognition in computer vision applications. Special attention is devoted to analyzing shape, since it is particularly important for our two applications: plant species identification from leaf photos, and object classification in remote sensing images. Additionally, both problems are bound by efficiency, constraining the choice of applicable methods: leaf recognition results are to be used within an interactive system, while remote sensing image analysis must scale well over very large image sets.

Leafsnap was the first mobile app to provide automatic recognition of tree species, currently counting with over 1.7 million downloads. We present an overview of the mobile app and corresponding back end recognition system, as well as a preliminary analysis of user-submitted data. More than 1.7 million valid leaf photos have been uploaded by users, 1.3 million of which are GPS-tagged. We then focus on the problem of segmenting photos of leaves taken against plain light-colored backgrounds. These types of photos are used in practice within Leafsnap for tree species recognition. A good segmentation is essential in order to make use of the distinctive shape of leaves for recognition. We present a comparative experimental evaluation of several segmentation methods, including quantitative and qualitative results. We then introduce a custom-tailored leaf segmentation method that shows superior performance while maintaining computational efficiency.

The other contribution of this work is a set of attributes for analysis of image segments. The set of attributes is designed for use in knowledge-based systems, so they are selected to be intuitive and easily describable. The attributes can also be computed efficiently, to allow applicability across different problems. We experiment with several descriptive measures from the literature and encounter certain limitations, leading us to introduce new attribute formulations and more efficient computational methods. Finally, we experiment with the attribute set on our two applications: plant species identification from leaf photos and object recognition in remote sensing images.

EFFICIENT IMAGE SEGMENTATION AND SEGMENT-BASED ANALYSIS IN COMPUTER VISION APPLICATIONS

by

João V. B. Soares

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Table of Contents

Chapter 1: Introduction

This dissertation focuses on efficient image segmentation and segment-based object recognition in computer vision applications. The first application studied is tree species recognition from leaf photos. Leafsnap was the first mobile app to provide automatic recognition of tree species, and today counts with over 1.7 million downloads. Since its launch in 2011, Leafsnap users have submitted more than 1.7 million valid leaf photos, 1.3 million of which are GPS-tagged. A preliminary analysis of this data, as well as a general overview of Leafsnap, is presented in Chapter 2. Appropriate leaf segmentation is essential for recognition, since it allows the distinctive shapes of leaves to be used. To allow for this, Leafsnap users are asked to photograph leaves against plain, light-colored, backgrounds. The problem of segmenting leaves from these types of photos is studied in Chapter 3. In Chapter 4, we move on to present a set of attributes of image segments. These attributes are by design intuitive and easily describable, so they may be used within knowledge-based systems. At the same time, they are required to be computationally efficient, to be applicable to a range of problems. The main focus of the attribute set has been on describing 2D shape properties. We experiment with the attributes on object classification from satellite images of urban areas, as well as identification of tree species from leaf images. The work is concluded in Chapter 5.

Throughout this dissertation, special attention is devoted to describing shape, which is particularly important for both our applications. Additionally, both our applications are bound by efficiency, constraining our choice of methods: results of leaf recognition are to be used within an interactive application, while remote sensing image analysis must scale well over very large image sets.

Leafsnap is a series of electronic field guide of trees, packaged in the form of user friendly mobile applications [1]. Users can quickly view the names and descriptions of tree species, and navigate high-quality photos of their leaves, flowers, fruit, bark, etc. In order to simplify the process of species identification, it allows users to automatically identify a tree from a photograph of one of its leaves. The fact that mobile devices with cameras and Internet connections are so prevalent has allowed Leafsnap to reach a great number of users. It has had more than 1.7 million downloads to date, being used by scientists, ecologists, foresters, urban planners, amateur botanists, gardening clubs, landscape architects, citizen scientists, educators, and children in classrooms. Given the large number of users, since its launch in 2011, Leafsnap has collected a large amount of user-submitted images. Over 1.7 million valid snaps have been taken, with 1.3 million of these presenting GPS tags. This unique dataset shows potential for use in tracking tree species over geographic location and time. Additionally, over 100 thousand of the GPS-tagged images have user-provided species labels, whose accuracy is being evaluated in an ongoing study. An overview of Leafsnap and a preliminary analysis of its collected data is presented in Chapter 2.

For species identification, Leafsnap requires that a single leaf be photographed against a plain light-colored background. This constraint greatly simplifies segmenting the leaf, allowing its shape, which is known to be very distinctive, to be used for recognition. At the same time, the requirement does not impose too much effort on the part of the user, representing a reasonable compromise. In Chapter 3 of the dissertation, we present work on segmentation of leaf photographs in these semi-controlled conditions. Whereas segmentation as a general problem is usually very difficult and poorly posed, the constrained leaf images allow us to study a well-defined problem, whose ground truth is practically unambiguous. Segmenting a leaf from a solid background may seem like a simple problem at first sight, but in fact it presents a series of practical challenges, including the presence of cast and attached shadows, shiny leaves with specularities, and natural variations in leaf shape and color. Additionally, segmentation must be performed in time compatible with an interactive system, while maintaining sufficient image resolution to preserve details such as serrations and thin leaf tips. These challenges are exemplified and discussed in more detail later in Chapter 3.

A comparative experimental evaluation of several popular segmentation methods is presented on the leaf segmentation task, including quantitative and qualitative results. For the evaluation, three large real-world datasets of leaf images are used, with manually segmented images serving as ground truth. We observe that the traditional methods tested are not immediately applicable to the problem: they are either too slow or would require that important modifications be introduced in order to obtain competitive results. Many of these methods are dedicated to general purpose segmentation and have their own inherent biases, making it difficult to apply them to the leaf segmentation problem. Given this difficulty, we introduce a custom-tailored leaf segmentation method that shows superior performance while being computationally efficient. The method is based on pixel clustering in an appropriate color space, followed by a graph cut step guided by image edges. We also make use of a training set of manual segmentations in order to adjust important method parameters.

In Chapter 4, we propose a set of attributes for analysis of image segments. These attributes focus mostly on shape, though there are also attributes describing photometric image properties, such as average intensity, or contrast strength along segment boundaries. The attribute set is designed for use in knowledge-based systems, such that they must be intuitive and easily describable. They are also required to be computationally efficient, so that they may be applied to a wider range of tasks. Several attributes from the literature were explored. We encountered limitations of some of these attributes, leading us to introduce new formulations, as well as more efficient computational methods. Illustrative examples are presented to demonstrate the usefulness of the proposed attribute set as well as the most significant changes introduced. We experiment with remote sensing images from urban areas to show how the attributes discriminate between different classes of structures, as well as on leaf images, in order to help discriminate between different tree species.

Chapter 2: The Leafsnap System and Analysis of Its User-Collected Data

2.1 Overview

Leafsnap was the first mobile application for identification of plant species using automatic visual recognition. It has received much public interest, with over 1.7 million downloads [1]. Besides the visual recognition engine, the mobile app serves as an electronic field guide, containing high-quality photographs of species (including, leaf, flower, fruit, bark, etc.) and textual descriptions of their characteristics. An overview of Leafsnap is presented below in Section 2.2. In Section 2.3 a brief analysis of its user-collected data is presented. A large dataset of user-collected leaf images (over 1.7 million) is examined, many of which have GPS tags and user-provided species labels. A collaboration with botanists from the Smithsonian Institution is underway to assess the potential of this data in tracking tree species over geographic location and time. Later in Chapter 3 a study on segmentation of leaves in semi-controlled conditions is presented. The study focuses on the type of leaf images that are collected by Leafsnap users, where single leaves are photographed against plain light-colored backgrounds. Leafsnap's recognition engine works by analyzing the distinctive shape of leaves, relying on the quality of the segmentations obtained to do so.

2.2 The Leafsnap System

Leafsnap is a visual recognition system for automatic plant species identification [1]. It is distributed in the form of mobile apps that help users identify trees from photographs of their leaves. Leafsnap has been launched in the US and UK, with each version containing tree species data from their respective countries. The US version of Leafsnap has coverage for all of the 184 tree species of the Northeastern United States, while Leafsnap UK covers 156 species from across the entire UK. Leafsnap US was launched in April 2011 and as of March 2015 counted with 1.73 million installs on iPhones and iPads. Leafsnap UK was launched in May 2014 and as of March 2015 counted with 28 thousand installs of its iPhone app.

Leafsnap is now being used by scientists, ecologists, foresters, urban planners, amateur botanists, gardening clubs, landscape architects, citizen scientists, educators, and school children in classrooms. It was developed to greatly speed up the manual process of plant species identification, collection, and monitoring. Without visual recognition tools such as Leafsnap, a dichotomous key (decision tree) must be manually navigated to search the many branches and seemingly endless nodes of the taxonomic tree. Identifying a single species using this process – by answering dozens of often-ambiguous questions, such as, "are the leaves flat and thin?" – may take several minutes or even hours. This is difficult for experts, and exceedingly so (or even impossible) for amateurs.

Automatic species identification has been an area of recent but growing interest in computer vision. Branson et al. [2] describe a system that combines human input with computer vision results to assist in the identification of birds. In the plant world, Nilsback and Zisserman [3] describe a system that can automatically identify plant species using images of flowers. While this system shows impressive results, its concerns are largely complementary to ours. Identification of species (or varietals) from flowers is of great interest to gardeners and flower enthusiasts. However, flowers are of limited value in systems used in biodiversity studies or for identifying local trees, since flowers are not present throughout most of the year. A summary of recent work on algorithms for plant identification and their respective evaluations are described in the ImageCLEF plant images classification task [4].

The development of Leafsnap stemmed from the initial works of Agarwal et al. [5] and Belhumeur et al. [6], which describe a much earlier version of the system. (Other related works on plant identification can be found in these two references.) More generally, recognition of tree species via leaves is an example of fine-grained visual categorization: the discrimination of instances into classes that are more specific than basic level categorization (e.g., leaves or animals) yet not as specific as the identification of individuals (e.g., face recognition).

To allow the general public to use the results of this research, we have implemented a complete end-to-end recognition system and packaged it as an electronic field guide called Leafsnap. The recognition engine consists of a back end server that accepts input images from various front-end clients. Currently, we have front-end apps for the iPhone and iPad devices, with work on Android devices in progress. The Leafsnap app contains high-quality photographs [1] and botanist-curated descriptions of tree species from the Northeastern US and UK. This already represents a significant step up from a traditional field guide: as a software application, operations such as browsing, sorting, and textual searching are trivial. Our automatic visual recognition system further improves on this by providing the user with the most likely candidate species for a given query input image. The user makes the final classification decision by visually comparing the actual plant to the high-quality photographs in the app, which span all aspects of the species – the leaf, flower, fruit, petiole, bark, etc.

Figure 2.1 shows screenshots from the iPhone app to illustrate the typical user experience. In scanline order: (a) the home screen, with a randomly-chosen image cycling every few seconds and access to educational games, (b) the browse screen, with a sortable and searchable list of all the species contained in the system, (c) the search functionality for finding particular species by scientific or common name, (d) the detail view for a particular species, showing the different images available for viewing, (e) the Snap It! screen, for performing automatic identification, (f) the returned identification results, in sorted order, (g) to (i) the manual verification stage as the user explores the images and textual descriptions of one of the results to confirm it as the correct match, (j) labeling the correct match, (k) the addition of that leaf to the user's collection for future reference, and (l) a map view showing

[1]The high-quality photographs in Leafsnap US were taken by the non-profit group Finding Species, while the ones in Leafsnap UK were taken by the Natural History Museum, London.

where that leaf was collected.

Our automatic system requires that a single leaf specimen is photographed on a solid light-colored background. The complete recognition process [1] consists of:

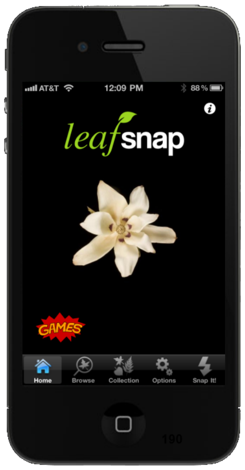
1. Classifying whether the image is of a valid leaf, to decide if it is worth processing further, using a binary classifier applied to gist features [7].

2. Segmenting the image to obtain a binary image separating the leaf from the background. Our segmentation approach is described later in Chapter 3. It consists of a custom-tailored pixel clustering in the saturation-value space of the HSV colorspace, followed by a graph cut step. Leafsnap requires fast and effective leaf segmentation, as it relies on the shapes of leaves for recognition. We have identified leaf segmentation as an important step to be improved in order to obtain better results on the recognition task.

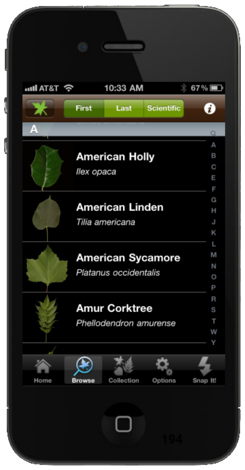
3. Extracting curvature features from the binarized image for compactly and discriminatively representing the shape of the leaf. We robustly compute histograms of curvature over multiple scales using integral measures of curvature.

4. Comparing the curvature features to those from a labeled database of leaf images and returning the species with the closest matches. Due to the discriminative power of the features and the size of our labeled dataset, we use a simple nearest neighbor approach with histogram intersection as the distance metric.

Users are then shown the top matches and make the final identification themselves,



(a) Home



A

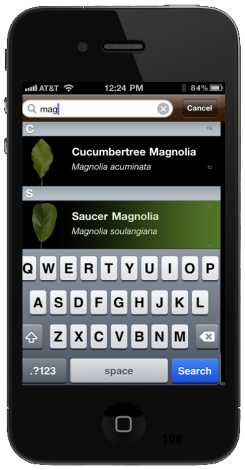
American Holly
Ilex opaca

American Linden
Tilia americana

American Sycamore
Platanus occidentalis

Amur Corktree
Phellodendron amurense

(b) Browse



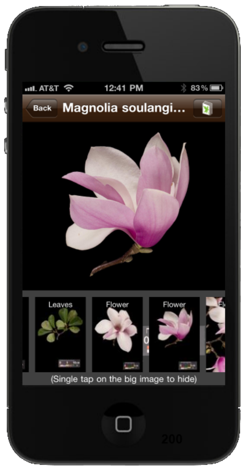
C

Cucumbertree Magnolia
Magnolia acuminata

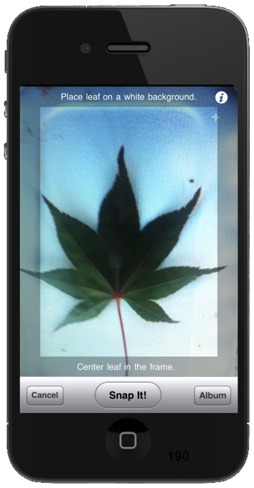
S

Saucer Magnolia
Magnolia soulangiana

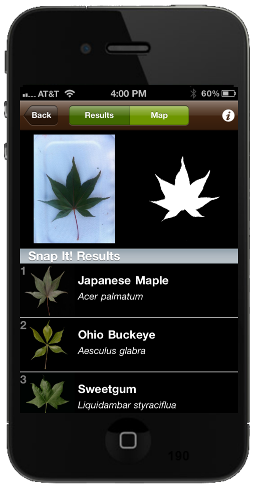
(c) Search



(d) Detail



(e) Snap It!



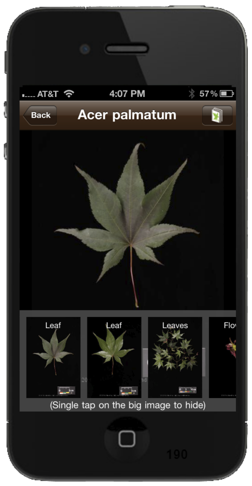
Snap It! Results

1 Japanese Maple
Acer palmatum

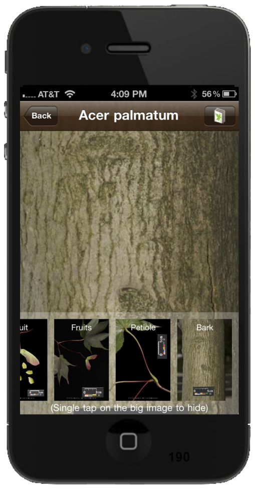
2 Ohio Buckeye
Aesculus glabra

3 Sweetgum
Liquidambar styraciflua

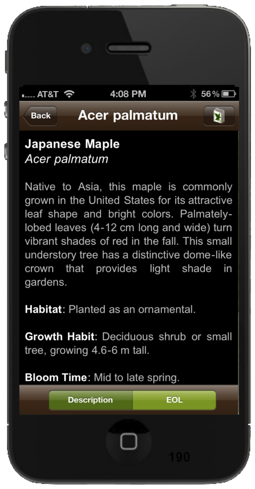
(f) Results



(g) Verification 1



(h) Verification 2



Japanese Maple
Acer palmatum

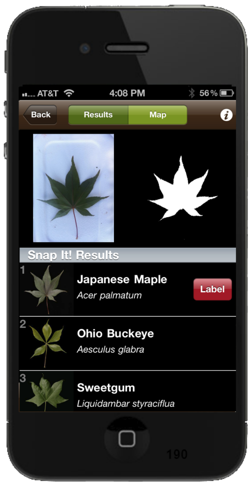
Native to Asia, this maple is commonly grown in the United States for its attractive leaf shape and bright colors. Palmately-lobed leaves (4-12 cm long and wide) turn vibrant shades of red in the fall. This small understory tree has a distinctive dome-like crown that provides light shade in gardens.

Habitat: Planted as an ornamental.

Growth Habit: Deciduous shrub or small tree, growing 4.6-6 m tall.

Bloom Time: Mid to late spring.

(i) Verification 3



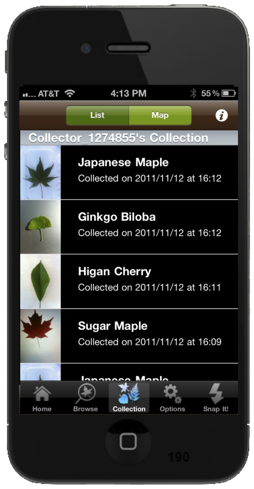
Snap It! Results

1 Japanese Maple
Acer palmatum

2 Ohio Buckeye
Aesculus glabra

3 Sweetgum
Liquidambar styraciflua

(j) Label



Collector 1274855's Collection

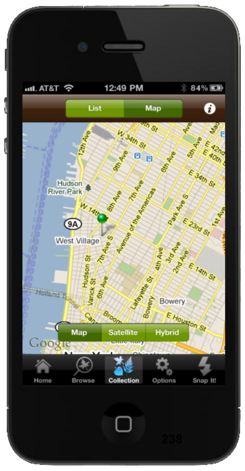
Japanese Maple
Collected on 2011/11/12 at 16:12

Ginkgo Biloba
Collected on 2011/11/12 at 16:12

Higan Cherry
Collected on 2011/11/12 at 16:11

Sugar Maple
Collected on 2011/11/12 at 16:09

(k) Collection



(l) Map

Figure 2.1: Tour of the iPhone version of Leafsnap. See text for details.

by examining additional content present in the app, such as the high-quality images of the species and textual descriptions of their characteristics.

All computation required for recognition is completed on average in 3 seconds, not including the time required for image upload. The back end server is currently a single Intel Xeon machine with 2 quad-core processors running at 2.33 Ghz each, and 16 GB of RAM. Aside from high-resolution versions of some images, which are served via Amazon's Simple Storage Service (S3), all other operations are handled by the server.

2.3 Data Analysis

We are collaborating with the Smithsonian Institution to analyze the data collected by Leafsnap's US users. Since its launch in 2011, Leafsnap US has had over 1.7 million downloads and users have uploaded over 1.7 million valid leaf images. Given this large amount of images, the objective of an ongoing study is to assess the potential of this data for tracking tree species over geographic location and time.

This ongoing analysis uses Leafsnap's data collected since it's launch in May 2011 up to November of 2013. During this period, there have been about 1.76 *valid* snaps submitted by users. We consider a snap to be *valid* if it was considered appropriately taken by the leaf/non-leaf classifier (see Section 2.2) and at the same time not later deleted by the user. Many user-submitted images are in fact not considered valid, according to the breakdown below.

- 32.4% of uploaded images are considered inadequate by our leaf/non-leaf clas-

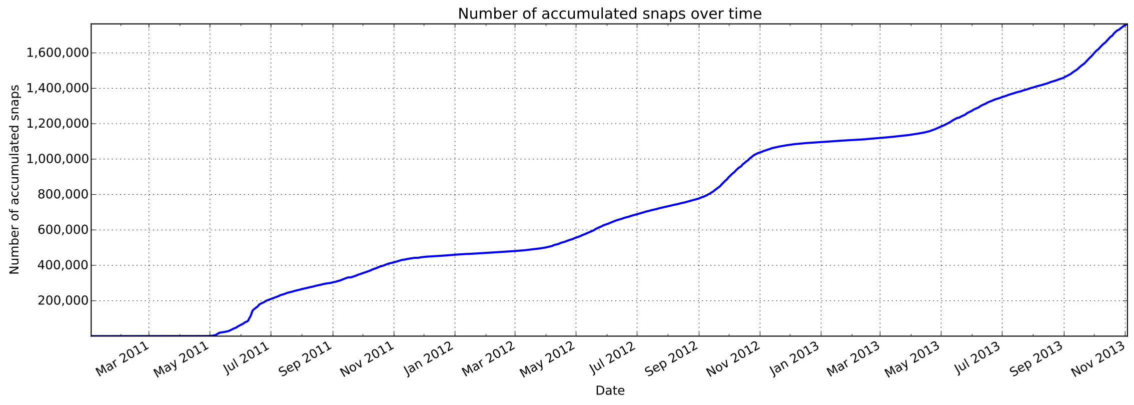


Figure 2.2: Accumulated *valid* snaps over time. Note that the rate at which valid snaps were added was larger in 2013 than in 2012. Note also that during the US winter, the number of new snaps decreases significantly, as would be expected.

sifier (see Section 2.2).

- 13.0% are eventually deleted by the user.
- 54.6% (1,763,945 snaps) are considered valid leaf images.

Figure 2.2 shows the number of accumulated valid snaps over the period. Note that the rate at which valid snaps were added was larger in 2013 than 2012. This rate could further increase with the launch of an Android app for Leafsnap. Currently, Leafsnap US offers apps for the iPhone and iPad, with work on an Android version underway. Note also that during the US winter, the number of new snaps decreases significantly, as would be expected.

For this study, we are particularly interested in leaf photos with GPS tags, so that the we may evaluate the potential for tracking the geographical span of species. We are also interested in photos that have user-provided species labels. A preliminary analysis shows that, for certain species, the user-provided labels are accurate enough to provide useful geographic distributions. For others, reliable species labels could be obtained through other sources: either from the work of botanists or from crowd-sourcing efforts using non-experts [8].

Of the 1.76 million valid snaps, 1.35 million have GPS tags, of which around 102 thousand have user-provided labels. Thus, about 7.6% of valid GPS-tagged images having been labeled by users. The reasons for this level of engagement are not clear at the moment, but one likely cause is the apps' user interface. The interface currently does not make the labeling functionality obvious to users, such that only more careful users might take the time to understand how to label leaves. Nonetheless, the current set of user-labeled GPS-tagged images is already large enough to be useful, as specified below.

Leafsnap US contains 184 tree species. Of these, the current dataset has

- 149 species with at least 100 user-labeled GPS-tagged images each;
- 68 species with at least 500 user-labeled GPS-tagged images each;
- and 27 species with at least 1,000 user-labeled GPS-tagged images each.

The most commonly labeled species are shown in Figure 2.3. The large number of images per species demonstrates the potential of the dataset for determining geographic distributions.

Preliminary results indicate that there are dozens of species for which users have an accuracy of 90% or higher[2]. In the case of native species, there is prior documentation available on their native geographic extent, which dates back to the work

[2]Reports on this analysis are currently in preparation, but have not yet been published.

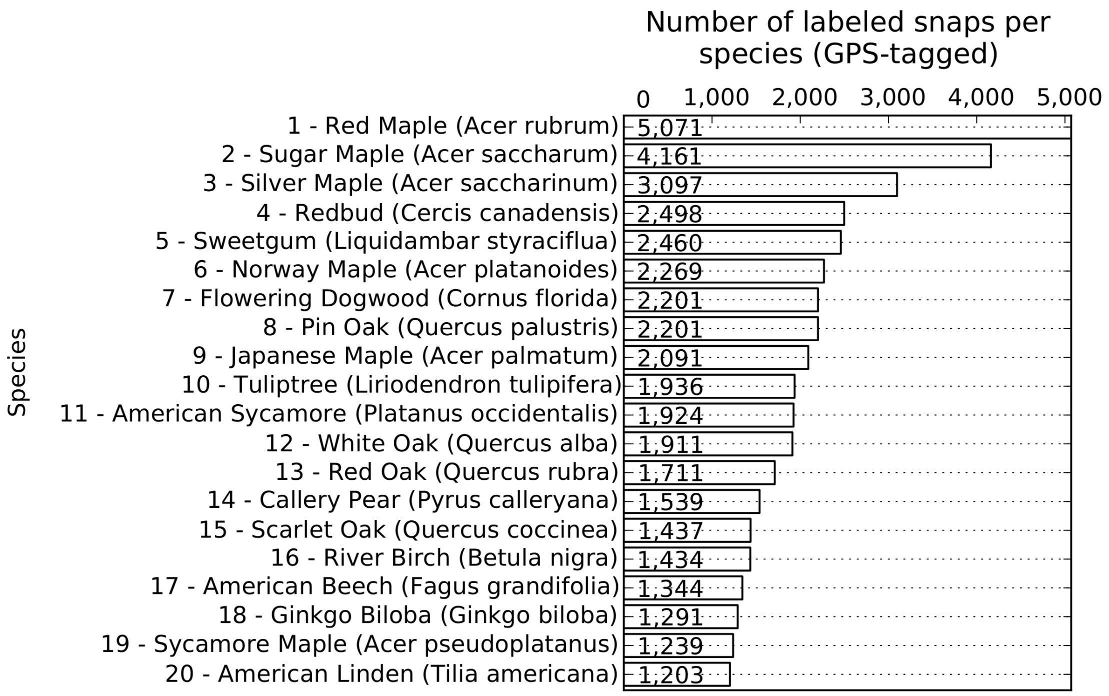
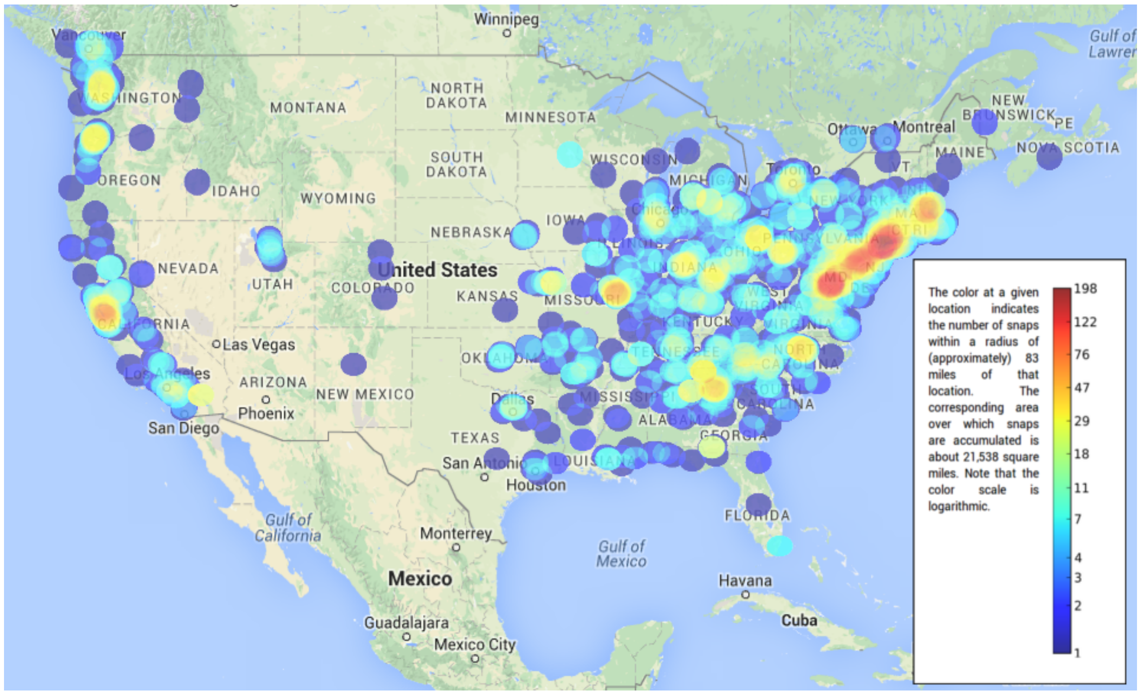


Figure 2.3: Counts of user-labeled GPS-tagged images per species in Leafsnap US are shown for the top species, in descending order of number of images.

of Elbert Little and colleagues in the 1970's [9,10]. Our dataset thus makes it possible to compare the geographic extent of species today with the native documented range in order to determine species migration over time. For introduced (exotic) species, the dataset provides a valuable reference of the species' geographic distributions in the US. A map showing snap densities for an exotic species is presented in Figure 2.4.

Finally, the dataset can be extremely helpful for species discovery. Individual leaf images with known geographic location can be quickly viewed by botanists. Having this data readily available can preempt several field studies, allowing botanists to only focus on more promising areas for specimen collection.



The color at a given location indicates the number of snaps within a radius of (approximately) 83 miles of that location. The corresponding area over which snaps are accumulated is about 21,538 square miles. Note that the color scale is logarithmic.

Figure 2.4: Map displaying density of snaps for a species exotic to the US, the Japanese Maple (*Acer Palmatum*). The map was generated from over 2,000 user-labeled GPS-tagged snaps.

Chapter 3: Efficient Segmentation of Leaves in Semi-Controlled Conditions

3.1 Overview

This chapter presents a study on segmentation of leaf images restricted to semi-controlled conditions, in which leaves are photographed against a solid light-colored background. The Leafsnap [1] system uses this type of image in practice for plant species identification, by analyzing the distinctive shapes of leaves. An overview of Leafsnap was presented in presented in Chapter 2, which explains the context in which our leaf segmentation method is used.

By restricting our attention to segmentation in a semi-controlled condition, we focus on a well-defined problem, which at the same time presents several challenges. The most important of these challenges are: the variety of leaf shapes, inevitable presence of shadows and specularities, and the time constraints required by interactive species identification applications. The segmentation problem is introduced and the difficulties exemplified in Section 3.2. Only very recently have larger efforts been directed towards leaf segmentation for plant species identification. Section 3.3 overviews previous work, with special attention to the ImageCLEF plant identification task [4].

We evaluate several popular segmentation algorithms on the task and observe that many of the methods are not immediately applicable: they are either too slow or would require that important modifications be introduced. In Section 3.4, we thus present a new segmentation method based on pixel clustering in color space followed by a graph cut step guided by image edges. We also make use of a training set of manual segmentations in order to adjust important segmentation parameters. The proposed method is fast enough for an interactive application, while producing state-of-the-art results. For comparative evaluation between different methods, three datasets of leaf images are used, with manually segmented images serving as ground truth. Section 3.5 gives an overview of our datasets, the different methods compared, and the evaluation protocol used. Results for all methods are presented in Section 3.6, including both quantitative and qualitative analyzes. The conclusion and directions for future work are presented in Section 3.7.

3.2 Leaf segmentation

Leafsnap, described previously in Chapter 2, uses the shape of a leaf in order to identify its tree species. Leaf shapes are very distinctive for identification, while other features such as the color of the leaf, its venation pattern, or images of the flowers are not suitable for various reasons – they are either too highly variable across different leaves of the same species, undetectable due to the poor imaging capabilities of most mobile phone cameras, or only present at limited times of year. Reliable leaf segmentation is thus crucial in order to obtain shape descriptions that are sufficiently accurate for recognition. The Leafsnap system requires that users photograph leaves against a light, untextured background. Large datasets with this type of image were collected, giving us the opportunity to study a useful semi-controlled segmentation problem that at the same time allows for objective assessments of different methods.

In contrast, much research in image segmentation has been dedicated to the problem of general segmentation in uncontrolled settings. For example, Arbelaez et al. present a comparison of methods on images of several different types [11], which constitute the Berkeley Segmentation Dataset and Benchmark [12]. Similarly, Alpert et al. [13] collected a database with a variety of different image types. Thus, much previous work has been dedicated to designing general-purpose methods that have high agreement with users' subjectively defined hand-drawn regions. The leaf segmentation problem we address here is different in that the environment is much more constrained. Furthermore, there is very little subjectivity involved in defining the ground truth for these images. Finally, we are also concerned in precisely locating segmentation boundaries. Precise boundary detection is important for plant identification since the leaf shape will be used as the main recognition cue. Again, this is in contrast to previous work, as precise boundary detection is usually not a prerequisite for general-purpose segmentation methods.

Though at first sight segmenting a leaf against a plain light-colored background seems easy due to the somewhat controlled conditions, in fact it poses significant challenges. The task requires that segmentations be produced in time that is suitable for an interactive application, and whose boundaries are faithful to the true leaf boundaries, enough as to enable correct recognition.

Figure 3.1 presents examples of leaf images from the three datasets we experiment with in this study, to illustrate the variety of leaves and acquisition conditions that must be dealt with. In Figure 3.2, we demonstrate the difficulty of traditional methods on this problem. A series of undesirable results are shown, resulting from the GrabCut method (see Section 3.5.3) when applied in difficult scenarios. The GrabCut method was chosen here for illustration purposes since it is very well known and was one of the best performing on our datasets (see results in Section 3.6). Below we more generally identify a series of significant practical challenges.

- Speed is a major challenge for our application, since the segmentation method should work as part of an interactive system. Coupled with this, high-resolution images are required in order to capture fine-scale details, such as leaf serrations.

- The datasets present a large variety of leaf shapes, due to the diversity among the different species. In particular, compound leaves are especially difficult to segment for traditional methods, due to their complex segmentation boundaries, some of which include several concavities and holes. Examples are presented in Figures 3.2c and 3.2d.

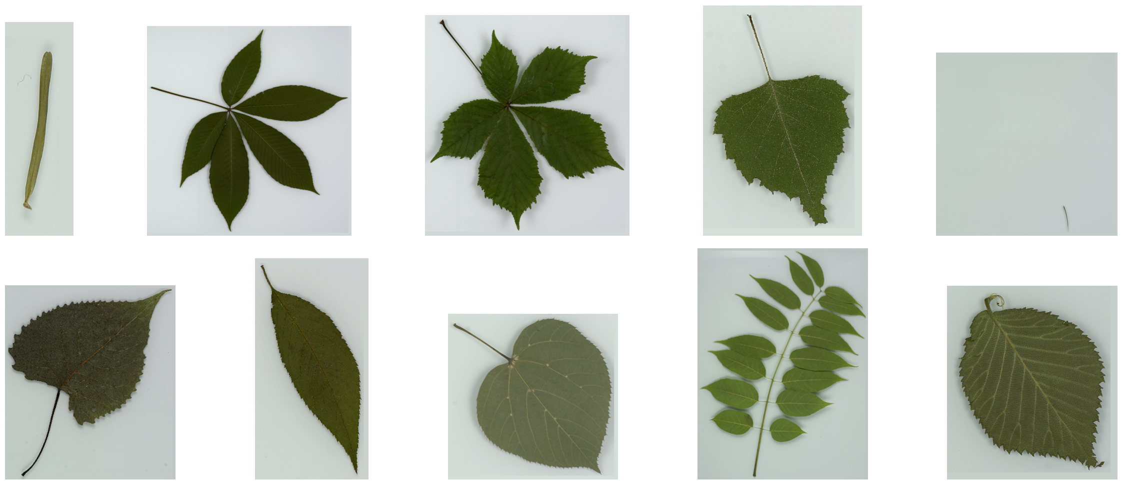
- Pine leaves were identified to present a special difficulty, since most of them occupy only a small fraction of the image. Many methods fail on these because they are biased towards producing larger segments. Figure 3.9 in Section 3.6 illustrates this situation well, by showing results of applying the GrabCut method to several photos of pines leaves.

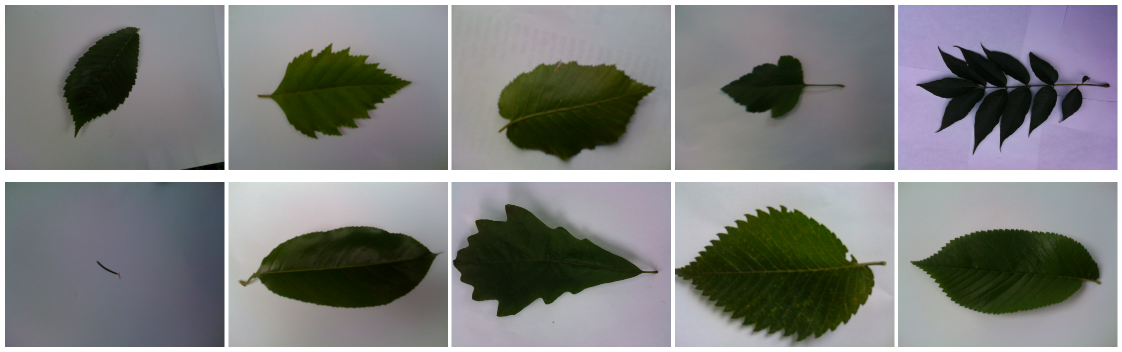
- The leaf images present natural variations in lighting. One of the most difficult problems on these datasets has been correctly segmenting out the shadows that the leaves cast on the background. Refer to Figures 3.2a, 3.2b, and 3.2c.

- The color of a leaf can vary, producing difficulties as presented in Figure 3.2h. In addition, some tree species have shiny leaves, occasionally producing specular highlights which can confuse segmentation methods, as in Figures 3.2e and 3.2f.

- The venation patterns on leaves can be very light-colored, presenting a strong contrast and creating a strong edge with the rest of the leaf, as in Figure 3.2g.

In this work, we experiment with several available implementations of popular segmentation methods. In order to provide an objective evaluation, we manually segment images from two leaf datasets. We also present qualitative observations on these two datasets, as well as an additional third dataset, composed of images uploaded by users of the mobile leaf identification system. We observe that the segmentation methods do not immediately work well for leaf segmentation. First of all, several methods were not developed with speed as a requirement and, for reasonably sized images, cannot produce results in sufficient time. Other methods have their own specific inherent biases, and would require the introduction of important modifications in order to work well throughout the different leaf species. We thus present a new segmentation method based on pixel clustering in color space followed


(a) Leaf images from the *Lab* dataset.


(b) Leaf images from the *Field* dataset.

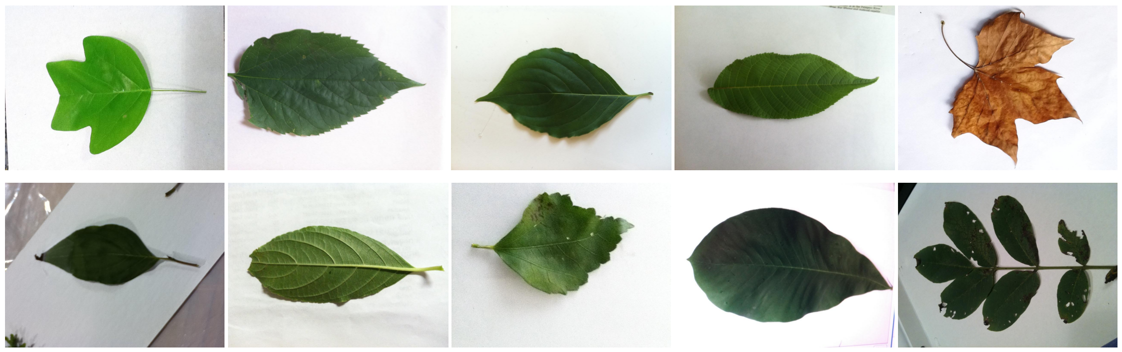
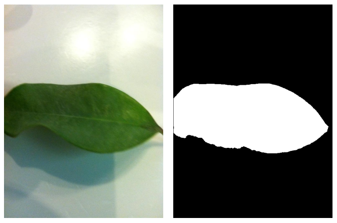
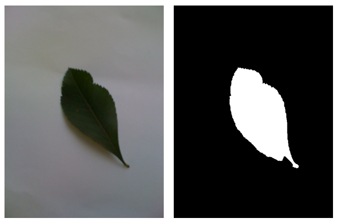

(c) Leaf images from the *User* dataset.

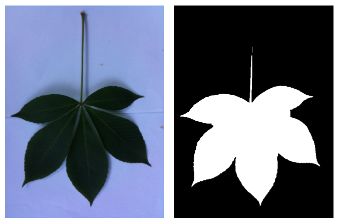
Figure 3.1: Images illustrating the three datasets used in this study. Refer to Section 3.5.1 for descriptions of each dataset. For the *Lab* and *Field* datasets, leaf species were randomly selected to have their images shown. For the *User* dataset, images were randomly sampled without regards to species, since this dataset was not labeled.



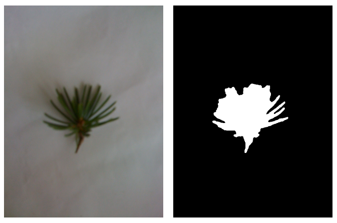
(a) The dark shadow cast by the leaf is incorrectly segmented due to how similar its color is to the leaf's.



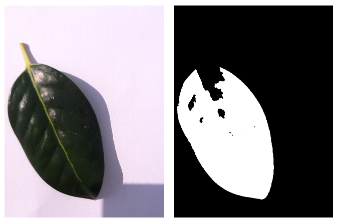
(b) As in the example in (a), a cast shadow is incorrectly segmented.



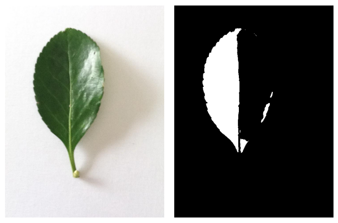
(c) Due to the shadows and the bias of GrabCut towards simpler boundaries, the segmentation result does not reflect the compound nature of the leaf, appearing instead as if the leaf simply had multiple lobes.



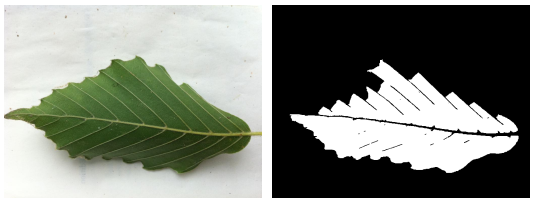
(d) The complex and long segmentation boundaries of this cluster were not correctly captured due to a segmentation bias towards simpler boundaries.



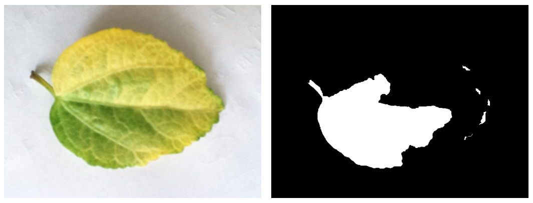
(e) The specularities on this leaf have a color that is similar to the background's, showing up as false negatives in the segmentation result.



(f) The large specularity has a color that resembles the background. In addition, the venation along the center of the leaf forms strong image edges. These effects combined lead to the erroneous result.



(g) The venation patterns on this leaf are bright, being reasonably similar in color to the background. They also form strong edges with the remainder of the leaf.



(h) The color of leaves can vary to a large extent. Note that the green tips of the leaf are segmented, while the yellow body of the leaf is not. Figure 3.1 contains more examples of leaves with varying colors.

Figure 3.2: Some of the undesirable results obtained when applying GrabCut to a challenging set of images. See the text for a discussion, and Section 3.5.3 for a review of the GrabCut method and the setup used.

by a graph cut step guided by image edges. We also incorporate learning of several parameters of the method, via a training a set of manually segmented images. The new approach is fast, while producing improved segmentation results.

3.3 Related Work

Several plant species identification approaches were presented for the ImageCLEF plant identification task [4]. The task's datasets are composed of leaf images, along with their respective metadata. Some of the identification approaches proposed have a leaf segmentation step, while others do not. The *scan* and *scan-like* datasets are photographed against a uniform background and usually segmented using some variant of Otsu's thresholding method [14]. For literature reviews of plant identification approaches using images of leaves, we refer the reader to [4, 6].

A more sophisticated leaf segmentation approach was presented by Cerutti et al. [15], in which polygonal shape models were used to constrain an active contour. Due to the variety of leaf shapes, modeling becomes a complex problem. The authors presented a model for leaves with multiple numbers of lobes, while a separate model was used for compound leaves. The approach seems to be especially beneficial when dealing with unconstrained environments (see below). Though it should work well on leaves with known shape, it could be prone to failure on leaves with shapes that were not modeled. This lead the authors to only consider Angiosperm species for segmentation in ImageCLEF [15]. A similar approach (though somewhat more simple) was presented by Manh et al. [16], in which a single shape model was used with the goal of segmenting leaves of a particular species of weed.

In the present work, we do not adopt any kind of shape prior, due to the difficulty of the modeling problem. Leaves can be simple, compound, or found grouped into clusters. More direct strategies such as that of Cerutti et al. [15], Nilsback and Zisserman [17] for flowers, or by Kumar et al. for other object classes [18], do not appear to be sufficient for our problem. These kinds of shape priors involve a somewhat limited amount of flexibility, which makes it difficult to model the variety and complexity of leaf shapes.

Leaf segmentation is much more difficult on the *photo* ImageCLEF dataset, in which photographs are taken in unconstrained conditions. In these photos, leaves can be found in a variety of circumstances: picked (against unconstrained background, usually on the floor), attached to the branch, or even within more dense foliage. Some authors have attempted to use completely automatic approaches, but with limited success. Casanova et al. [19] experimented with k-means to cluster the image pixels into *leaf* and *background*. Yanikoglu et al. [20] assume the central region of the image contains the leaf, so that the largest cluster found from this central region (when performing histogram clustering) defines the leaf's color. With this representative color in hand, an over-segmentation is found using watershed, whose segments are assigned to leaf or background according to the distance of their color to the reference leaf cluster color. Camargo Neto et al. [21] also approached the unconstrained problem, in the context of image analysis for weed control. Their approach begins by finding leaf pixel candidates based on a color index. The final segmentation is found by creating fragmentations of the set of leaf pixels, which are then merged in a procedure that favors the formation of convex shapes.

Most works, however, applied traditional *interactive* segmentation techniques in order to obtain reasonable results on ImageCLEF's *photo* images. Casanova et al. [19] used mean shift [22] to produce an over-segmentation of the image. A user would then indicate some background and foreground segments, which were used in a merging phase as to label the remaining segments. Cerutti et al. [15] used an interactive version of their polygonal model method described previsouly, in which a user initially marks a region inside the leaf, providing a color model for leaf pixels. Yanikoglu et al. [20] use the marker based version of the watershed transform, while Arora et al. [23] adopted the interactive GrabCut system [24]. Though user interaction is interesting in certain situations, it becomes impractical for large datasets with thousands of images or more [4]. In this study we focused only on fully automatic approaches.

A system capable of working with photos of leaves on uncontrolled backgrounds would be very appealing from the perspective of a user. However, it is not clear to what extent such a system should rely on the resulting segmentation. As noted by Bakić et al. [25], shape boundary features of the segmentation become unreliable in this scenario, leading several authors to work with interactive segmentation techniques or avoid the problem altogether [4]. It is important to note that dealing with uncontrolled conditions requires approaches that are different in nature to the ones we study here, which are more appropriate in our semi-controlled scenario.

We should note that for the task of plant identification, leaf shape is an extremely important cue [5, 6]. In order to use the leaf's shape for recognition, it is not sufficient to provide only a coarse description of where the leaf is, but it is required that segmentation boundaries faithfully represent the true leaf shape. Due to this challenge, leaf datasets have been collected in which leaves were photographed against a plain light-colored background [1, 4]. We approach the leaf segmentation problem under this semi-controlled condition. The current problem is thus substantially different from previous work on segmentation in uncontrolled environments [3, 4, 26], in which a precise segmentation boundary is not as crucial.

In our experiments, we use images taken in semi-controlled conditions, either in a laboratory setting, or taken with mobile devices against plain light-colored backgrounds (see Section 3.5.1). Our datasets were collected from a real-world functioning system for plant species identification via mobile devices [1]. Images from our *Lab* and *Field* datasets are used in practice as labeled training data within the system's recognition engine, while the *User* images were taken by users of the mobile system, presenting us with a series of real challenges. We would like to point out that the leaf datasets from the ImageCLEF plant identification task [4] are also relevant for the segmentation task, but we did not experiment with them. However, the datasets we have used are comprehensive and allow us to arrive at relevant conclusions, since they include a wide variety of species and acquisition conditions, and were taken from a real-world functioning system. It is important to note that images from our *Lab* dataset, which contain pressed leaves taken under controlled conditions, are representative of those in ImageCLEF's *scan* photos. At the same time, our *Field* and *User* images, which were taken with mobile devices with leaves placed against plain light-colored backgrounds, represent to a good extent those in the *scan-like* photos from ImageCLEF.

The segmentation method we present here begins with a custom-tailored color space clustering procedure to define foreground and background distributions. A graph cut [27] step follows, applied on the pixel probabilities provided by the clustering method. The graph cut technique is widely used and well established in image segmentation. Its attractiveness comes from its global optimization formulation along with the fast methods available to find its solution. We estimate model parameters via a learning phase, using manual segmentations. This removes the need for manual parameter tuning, though it requires the availability of a dataset of manually segmented images.

GrabCut was initially developed by Rother et al. for interactive image segmentation [24]. However, given an appropriate initialization, the method is shown to be very useful for image segmentation in general, side-stepping the need for user interaction. The GrabCut approach has some similarity to the approach we adopt here, in that we perform color space clustering, followed by a graph cut step. However, there are two important differences. First, GrabCut makes use of multiple iterations, which alternate between model estimation in pixel color space and graph cut in image space. Our approach is faster, defining the model in a single step, after which graph cut is applied. Second, GrabCut uses a Gaussian mixture to model each class, instead of single Gaussians as we do here, thus we have a more specific model of what distributions are expected, which also also require less computation. In Section 3.6, we present a comparative evaluation of GrabCut and our method, demonstrating these differences.

In this work we learn certain graph cut parameters by selecting a range of parameter values and choosing the one that produces the lowest error on a training set of manually segmented images. This training strategy is able to work in a reasonable amount of time, since we end up only having two parameters to estimate. It has been noted – among others by Kumar and Hebert [28] – that training of random fields using pseudo-likelihood inference is known to produce unreliable parameter estimates. Using more precise inference techniques becomes very time-consuming, so that more recent work takes an alternative, two-phase approach, similar to what we adopt here. In this two-phase approach, first unary potential parameters are learned in isolation. Then, the parameters describing binary or higher order interactions are learned by cross-validation (or hold out validation). An illustrative example of this type of approach was presented by Kohli et al. [29].

A problem somewhat related to leaf segmentation is that of flower segmentation for species recognition. Nilsback and Zisserman [3] developed a flower recognition approach which relied on segmentation prior to feature extraction. The flower segmentation problem is particularly challenging because image backgrounds are uncontrolled, while at the same time the appearance of different classes of flowers varies to some extent. The problem was dealt with by learning foreground and background color distributions, and by defining a flexible shape prior that captured the structure of petals.

Chai et al. [30] later proposed a co-segmentation method for flower segmentation, which did not require manual segmentations or modeling of a shape prior, yet showed increased performance. In the current work, we make use of manually segmented leaf images, following a more traditional supervised line of work.

3.4 Proposed Segmentation Method

Our method consists of the following steps, illustrated in Figure 3.3 and described in more detail in the next sections. First, the image pixels are clustered into two groups in saturation-value colorspace, so that one group corresponds to leaf pixels, and the other, to background pixels. The clusters are found using the Expectation-Maximization (EM) algorithm to fit two normal distributions to the pixel data. This provides us with a probability indicating how likely each pixel is to belong to each cluster, according to the model from the normal distributions. The probabilities define the energies used in the graph cut step that follows. The graph cut formulation also allows us to incorporate edge information, encouraging the segmentation boundaries towards strong image edges. Finally, false-positive regions are removed from the segmentation via a post-processing procedure. An optional stem removal step may be applied in order to obtain more standardized shapes during recognition.

3.4.1 Expectation-Maximization in Saturation-Value space

The first step of our segmentation method uses a modified version of the Expectation-Maximization (EM) clustering procedure in saturation-value color space. A mixture model composed of two normal distributions is fit to the pixel data in saturation-value space. If we denote a pixel's saturation and value as $\boldsymbol{x}$, the mixture

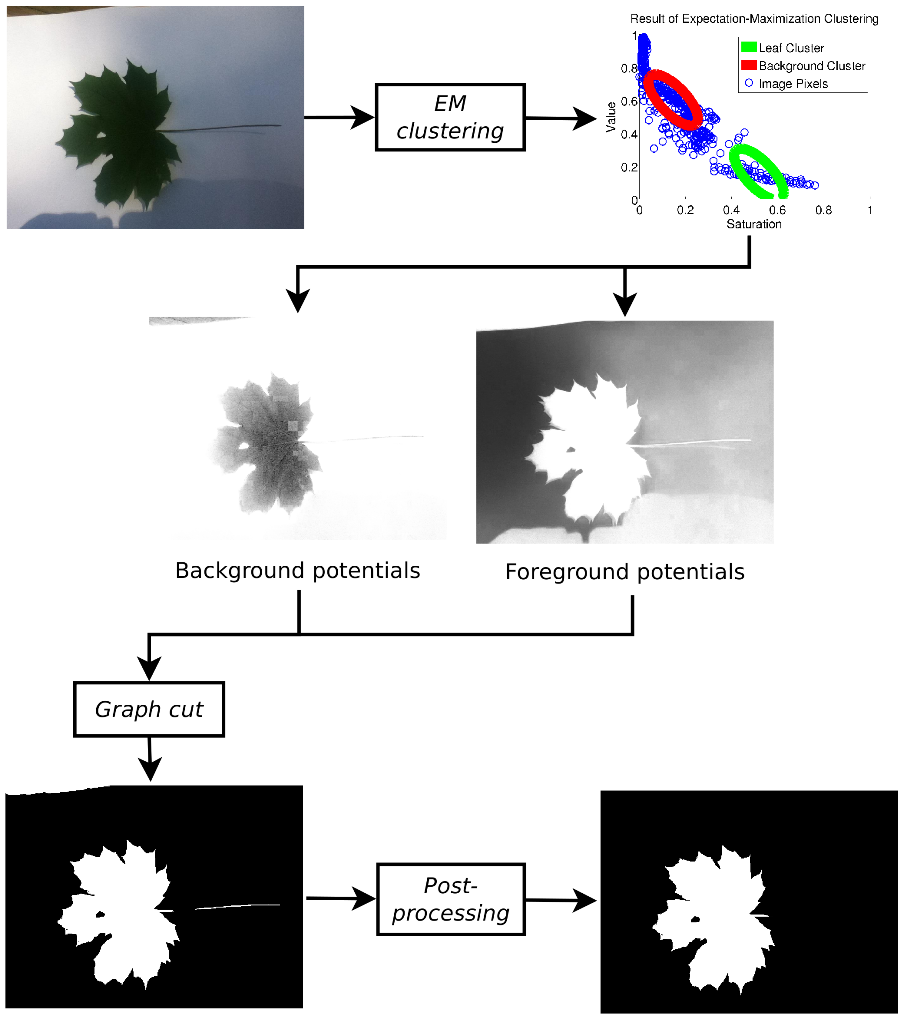


Figure 3.3: Segmentation pipeline. The first step consists of EM clustering in saturation-value space. This provides us with probabilities that are used for defining energies in the graph cut formulation. After the graph cut step, a post-processing procedure is applied in order to eliminate false positive regions.

model is

$$Pr(\boldsymbol{x}|\boldsymbol{\Theta}) = \frac{1}{2}Pr(\boldsymbol{x}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}) + \frac{1}{2}Pr(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}), \tag{3.1}$$

where $Pr(\boldsymbol{x}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma})$ and $Pr(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$ are normal distributions. $\boldsymbol{\mu}_0$ represents the mean of the background distribution, while $\boldsymbol{\mu}_1$ is the mean of the foreground (leaf) distribution. A common shared covariance matrix $\boldsymbol{\Sigma}$ is used. The set of model parameters is $\boldsymbol{\Theta} = \{\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}\}$. Note above that each normal distribution is assigned an equal weight of 1/2.

The model is fit using the EM algorithm. From an initial estimate of the model parameters $\boldsymbol{\Theta}$, EM proceeds to find a local maximum of the data's likelihood (see e.g. [31]). This can lead to undesirable solutions if the initial parameters are not set carefully. The means for the normal distributions are thus initialized near their expected values, so that they converge to the correct clusters when provided with pixel data from a new image. The covariance matrix is simply initialized as a multiple of the identity matrix.

Pixel weighting is introduced in order to correctly segment pine leaves in which only a small fraction of the total image pixels are leaf. A region in saturation-value space likely to contain leaf pixels is defined a priori. Given a new leaf image, the pixels inside and outside the region are weighted so that the two sets of pixels end up having the same weight. These pixel weights are then used during the EM procedure in order to change the relative density/importance of pixels.

The region in saturation-value space that is more likely to contain leaf pixels is learned from a training set of manually labeled leaf and background pixels. In a preliminary version of this method, presented in [1], these regions were delineated manually, which, as we will show, leads to somewhat worse results. The manually segmented images provide ground truth labels for each pixel, which can either be *leaf* or *background*. For each of the leaf and background classes of pixels, this allows us to estimate a probability density function in saturation-value space, using kernel density estimation [31]. From the kernel density estimates, the regions more likely to contain leaf versus background pixels are determined.

3.4.1.1 Speed Optimizations

We would like to note that it is possible to optimize our EM procedure to obtain very fast speeds while handling reasonably large images. In the comparative analysis presented in this chapter, we do not experiment with the speed optimized versions. However, we do employ the speed optimized version on the Leafsnap server which attends to recognition requests from users. Below we explain the optimizations used.

First, the fact that the covariance matrix is shared between the two normal distributions brings a significant speed advantage. Assuming the model with normal distributions and a shared covariance matrix from Equation 3.1, in the two-class case, the posterior function defining cluster memberships takes on a linear logistic form [32], which can be efficiently computed. Specifically, if we denote the label of pixel $\boldsymbol{x}$ as $y \in \{0, 1\}$, such that 0 denotes background, and 1 denotes foreground, then we can write

$$Pr(y = 0|\boldsymbol{x}) = 1/[1 + \exp(\beta_0 + \boldsymbol{\beta}^T \boldsymbol{x})], \text{ where} \tag{3.2}$$

$$\boldsymbol{\beta} \equiv (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}^{-1}, \text{ and} \tag{3.3}$$

$$\beta_0 \equiv -\frac{1}{2} (\boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 - \boldsymbol{\mu}_0^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_0). \tag{3.4}$$

After computing Equation 3.2 for each pixel $\boldsymbol{x}$, we can quickly obtain $Pr(y = 1|\boldsymbol{x}) = 1 - Pr(y = 0|\boldsymbol{x})$.

A second optimization comes from the observation that it is possible to obtain reliable estimates of the Gaussian parameters using only a fraction of the pixels in the image. We thus use a downsampled version of the original image during EM. Once the procedure has converged, the labels for each pixel can be quickly computed over the entire original image using Equation 3.2.

Using these optimizations, this part of the segmentation method runs on average in 0.062 seconds on a 700×525 image, when using 25% of pixels during EM.

3.4.2 Graph Cut

After running the EM procedure, we are able to compute an estimated probability that each pixel belongs to either leaf or background. We then include a graph cut step that uses these probabilities to determine the image segmentation, following the work of Boykov and Jolly [27]. For quickly finding the optimal graph cut, we employ the optimized algorithm by Boykov and Kolmogorov [33], whose implementation is provided by the authors.

Let $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ denote the binary segmentation we wish to solve for, with all $y_i \in \{0, 1\}$, and N the total number of image pixels. The graph cut minimizes the energy function (or cost function)

$$E(\mathbf{y}) \;=\; \lambda R(\mathbf{y}) + B(\mathbf{y}). \tag{3.5}$$

R is the regional term defined by the unary potentials of the underlying Markov Random Field (MRF) formulation and B denotes the boundary term, which penalizes discontinuities in the cut, and corresponds to binary or higher-order potentials in the MRF formulation. First, for the regional term, we will use the posterior probabilities provided by the EM procedure. Let $Pr(y_i = 0|\boldsymbol{x}_i)$ and $Pr(y_i = 1|\boldsymbol{x}_i)$ denote respectively the probability that pixel i is background and leaf, given its features $\boldsymbol{x}_i$. These probabilities are assigned according to the model estimated by EM. Then the regional term is written

$$\begin{aligned} R(\mathbf{y}) &= \sum_{i=1}^{N} R_i(y_i), \text{ with} && (3.6)\\ R_i(0) &= -\ln Pr(y_i = 0|\boldsymbol{x}_i), && (3.7)\\ R_i(1) &= -\ln Pr(y_i = 1|\boldsymbol{x}_i). && (3.8) \end{aligned}$$

Following [27], the boundary term is given by

$$B(\mathbf{y}) = \sum_{\{i,j\} \in \mathcal{N}} B_{\{i,j\}} \cdot \delta(y_i, y_j), \text{ where} \tag{3.9}$$

$$\delta(y_i, y_j) = \begin{cases} 1, & \text{if } y_i \neq y_j \\ 0, & \text{otherwise,} \end{cases} \tag{3.10}$$

and $\mathcal{N}$ denotes the set of neighboring pairs of pixels in the image, which in our case we take to be the pairs of 4-neighbors. From a grayscale version G of the image, we adopt the commonly used boundary coefficients

$$B_{\{i,j\}} \propto \exp\left(-\frac{(G_i - G_j)^2}{2\sigma^2}\right), \tag{3.11}$$

where G_i and G_j are respectively the gray values for pixels i and j.

We work with a set of manually segmented training images, which allow us to estimate the graph cut parameters λ and σ (defined respectively in Equations 3.5 and 3.11). This is done by selecting a grid of (λ, σ) pairs and choosing the one that produces the highest average accuracy when applied to the images in the training set.

3.4.3 Post-processing

After the graph cut step is applied to an image, we follow a straightforward post-processing procedure to remove undesired false positive regions. The procedure removes two type of false positives. The first type, which is very common to these images, consists of regions along the image boundaries which fall outside the light-colored sheet of paper or background used to create a contrast against the leaf. The second type consists of isolated regions present due to shadows, uneven backgrounds, or extraneous objects.

Post-processing consists of first performing a morphological dilation, then computing the connected components of the result. A connected component that has a large intersection with the image border relative to its area is then excluded as a false positive. Of the remaining connected components, the largest one is taken to be the leaf.

3.4.4 Stem Removal

For the sake of comparison, we do not apply stem removal in the comparative analysis of segmentation methods that follows in the current work. However, for accurate leaf recognition, stem removal can be beneficial. Below we explain the stem removal step, which we use when we are interested in performing leaf recognition (again, though not when comparing different segmentation methods).

After the previously described post-processing procedure, the stem of the leaf may or may not be present in the segmentation. The original leaf might not have had a stem to begin with, or it might have been lost by the segmentation method, which may occur when the stem has a lighter color than the leaf. Even when the stem is correctly segmented, it may vary in length depending on how the user picked the leaf. To standardize the final shape for recognition, it is desirable to remove the stems from the segmentations prior to feature extraction. Stem removal is done through a series of morphological operations.

First, the set of all thin structures that protrude from the leaf is determined. This is done by taking the top-hat transformation of the segmentation [34], using as structuring element a disc with diameter larger than the width of any potential stem. For a binary image B and structuring element s, the top-hat is defined as

$$T_{\text{hat}}(B) = B - B \circ s, \tag{3.12}$$

where $\circ$ denotes the opening operation and is defined as an erosion followed by a dilation.

Next, we determine which of these candidate structures is most likely to be the stem. First, we note that removing the stem should not change the number of connected components in the segmented leaf or in the background; thus, only such candidates are considered as possible stems. Of the possible stem candidates, we only consider those of appropriate size, and choose to remove the one that is most elongated (i.e., with the highest ratio of its two principal moments).

3.5 Experimental Evaluation

The datasets used in our experiments are presented in Section 3.5.1 below. In Section 3.5.2, we explain how methods that produce multiple segments (over-segmentations) are evaluated and compared to those that produce binary segmentations. All of the compared methods are listed along with their relevant settings in Section 3.5.3. The performance metrics used in the quantitative evaluation are discussed in Section 3.5.4, which are computed against ground-truth manually segmented images. To compute statistically significant differences between the performances of methods, we use the sign test, as described in Section 3.5.5. Later on, we present the experimental results in Section 3.6.

3.5.1 Datasets

We experiment on three datasets of leaf images, which are illustrated in Figure 3.1.

1. The first dataset, which we refer to as *Lab*, consists of leaves collected from trees in the Northeastern US by field botanists. These images have important particularities: their leaves were flattened by pressing prior to being photographed, and they were photographed under controlled lighting with a high-quality camera. The complete dataset has 4,221 images. The original images were manually cropped close to the leaves and then resized so that the size of their maximum dimension (width or height) would be 512 pixels. Of the 4,221 images, 30 were randomly selected to be manually segmented.

2. The second dataset, which we call *Field*, consists of 1,042 images collected by researchers in the field using different mobile devices. These present varying acquisition poses, illumination conditions and amounts of blur, though an effort was made by the researchers for the images to be reasonably uniform. Of the 1,042 images, 786 present a size of 1600×1200 pixels, while the remaining

256 have a size of 2048×1536 pixels. 56 of the images were randomly selected to be manually segmented.

3. Finally, the third dataset is from images uploaded by users of the mobile leaf identification system, which we will call *User*. 1,000 uploaded images were randomly selected in the period from July to October 2011, from users near New York City or Washington D.C. Only images that were appropriately taken according to instructions provided to users were accepted, leaving only 497 images. These images present more challenges than the ones from the *Field* dataset, since they contain an even greater variety of conditions. Users could choose to upload images in three different sizes: 640×480 pixels (small), 960×720 pixels (medium) and 1024×768 pixels (large). Of the 497 images analyzed, 317 were large, 147 were medium, and 33 were small. Since we did not manually segment these images, this dataset is only used for qualitative observations.

3.5.2 Evaluation of Over-Segmentations

Initially, we expected all methods we experimented with to be able to produce two segments, such that one would correspond to leaf and the other to background. In practice, however, we found that some methods, when set to produce just two segments, do not produce meaningful results on our images. Since this was so common, instead of abandoning these methods altogether, we resorted to experimenting with them by producing over-segmentations.

The over-segmentations are evaluated in two different manners. In the first evaluation, to obtain an upper bound on a method's performance, we assign each segment to leaf or background as to yield the highest agreement with the ground truth as measured by pixel accuracy. This provides an optimistic evaluation, or upper-bound, on the method's performance. Secondly, for a more realistic evaluation, we use a simple heuristic strategy to assign each of the segments to leaf or background, as follows. First we compute the median pixel saturation of each segment. Then, we take the mid-point between the minimum and maximum of these median values. Segments that have median saturation that are larger than the mid-point are assigned *leaf*, while those with median saturation below the mid-point are assigned *background*. The heuristic provides a lower bound for a method's performance, though in practice the heuristic works well when the number of segments is small, resulting in performance that is very similar to that obtained from the best possible assignment strategy.

For a fair overall comparison between methods, we would like to somehow assess how well they are solving the original leaf segmentation problem. The most relevant situation for us is thus when the number of segments produced by the over-segmentations is small. Conversely, when the number of segments produced is large, the segmentation method itself is not solving the original problem, but only part of it, so we don't find these results relevant for comparison purposes. As we will see in Section 3.6, when methods that produce over-segmentations are set to produce a small number of segments, the heuristic assignment strategy gives results that are very similar to the results from the best assignment. Thus, when performing comparisons to methods that produce binary segmentations, we will always use the results obtained from the heuristic strategy.

Though the upper bounds obtained by the best possible assignment strategy can be very high when the number of segments produced is large, for small and intermediate numbers of segments, they provide us with relevant information. First, when the number of segments is small, the best assignment strategy provides similar performance to the heuristic assignment strategy. This lets us know that the heuristic strategy is performing as well as possible. Second, if we believe that for some particular intermediate numbers of segments, it is feasible to design a perfect assignment strategy, then for that number of segments, the best assignment's performance should be interpreted as realistic.

3.5.3 Methods and Settings

This section reviews the methods that were compared on our datasets. Any relevant adjustments and settings of each particular method are also presented. The methods to be tested were chosen due to their popularity and availability of implementation. The reader familiar with the previous approaches we use may skip to Sections 3.5.3.8, 3.5.3.9, and 3.5.3.10, where we list the new proposed approaches that we experiment with.

3.5.3.1 Otsu

As a baseline method, we convert the images to grayscale and apply a threshold, resulting in an image segmentation. The threshold is found using the method of Otsu [14]. We use the implementation that is provided in Matlab's Image Processing Toolbox.

In order to understand Otsu's threshold selection method, let us consider that the pixels in the image form an empirical probability distribution. The distribution can be computed from the image's gray level histogram by simply dividing all of the histogram frequency counts by the total number of pixels in the image. Otsu's method considers classical criteria used in discriminant analysis [35] on the distribution in order to define a good threshold. A threshold on the gray level divides the image pixels into two classes. Intuitively, the criteria favor maximizing the resulting *between*-class variance while at the same time minimizing the *within*-class variance.

Suppose we have fixed a threshold, as to divide the image pixels into two classes. Let us denote the total density of the pixels in each class by ω_0 and ω_1 (so that $\omega_0 + \omega_1 = 1$), their means by μ_0 and μ_1, and their variances by σ_0^2 and σ_1^2. Additionally, denote the mean and variance of the complete data distribution by μ_T and σ_T^2. Then we can define the between-class variance as $\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2$. The criterion function that Otsu's method maximizes is $\eta = \sigma_B^2/\sigma_T^2$. In practice, the optimal threshold is found by simply computing the criterion function η for each possible gray level and choosing the one that maximizes it.

3.5.3.2 Graph-based image segmentation (GBIS)

Felzenszwalb and Huttenlocher proposed an efficient graph-based image segmentation method [36], and have provided their own implementation, which we experiment with here.

The method begins by defining a graph where each pixel is a vertex and edges connect nearby pixels together. Each edge is assigned a weight that indicates the dissimilarity between its pixels. The graph implicitly defines an initial segmentation, such that each image pixel forms its own segment. The segmentation algorithm proceeds to analyze the edges in order of increasing weight, so that the most similar pixel pairs are analyzed first. For each edge being analyzed, if its pixels currently belong to different segments, we consider whether or not we should merge these segments. Segments are merged whenever the edge weight measuring pixel dissimilarity is small relative to the *minimum internal difference* (defined below) between the segments. Intuitively, we merge segments when the pair of pixels analyzed does not indicate that there is an image edge between them, by presenting low dissimilarity relative to their minimum internal differences, which measure the natural within-segment variations.

Here, we review the criterion used by the method for deciding when to merge segments. Denote V the set of graph nodes, which is initially the set of all image pixels. Denote the set of m edges linking pixels together as E, and the weight of an edge $e \in E$ by $w(e)$. The *internal difference* of a segment $C \subseteq V$ is defined as the largest weight in the minimum spanning tree $MST(C, E)$ of the segment:

$$Int(C) = \max_{e \in MST(C,E)} w(e).$$

Given a pair of segments (C_1, C_2) being considered, their minimum internal difference is defined as

$$MInt(C_1, C_2) = \min\left(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)\right),$$

where the threshold function τ is defined as $\tau(C) = k/|C|$, with $|C|$ denoting the number of pixels in segment C. The original internal difference $Int(C)$ can occasionally (by chance) be very small, especially when dealing with smaller segments. Without the $\tau(C)$ term within $MInt$, this would impede certain small segments from ever merging. The $\tau(C)$ term corrects for this effect.

The most relevant parameter of the method is k above which defines the threshold function τ, roughly controlling the minimum size of a segment. We experimented with the range $k \in \{1, 10, 100, 1000, 4000\}$.

In the implementation we use, there are edges between 4-neighboring pixels and their dissimilarity is defined as the L_2 (Euclidean) distance between their feature vectors. A feature vector for a pixel is defined as (x, y, r, g, b), where (x, y) is the location of the pixel in the image, and (r, g, b) is the color of the pixel.

Throughout, we set the remaining relevant settings as follows. These settings have shown to work well on our different sets of leaf images. The GBIS method applies a Gaussian filter to smooth the image slightly, in order to compensate for digitization artifacts without significantly affecting the content. We set the standard deviation of the Gaussian used to smooth the image to $\sigma = 0.5$ pixels. The final step of the method is to apply a post-processing, where very small segments are merged to their most similar neighboring segments. We have set the minimum size a segment is allowed to have (so that it is not merged to a neighbor) to 0.5% of the image size.

3.5.3.3 Mean shift

Comaniciu and Meer proposed the use of mean shift [22] as a general clustering procedure, having applied it to image segmentation. We used the speed optimized implementation of mean shift provided in the EDISON system [37]. Clustering is performed jointly in spatial coordinates and in the L*u*v* colorspace.

For each pixel in an image, the mean shift procedure finds a local maximum of a density function defined from the image data. Each pixel is then associated to its respective maximum. The density function is defined as a kernel density estimate, computed from the image data, and thus requires a kernel bandwidth to be defined. When mean shift is used for image segmentation, two bandwidths are used: h_s, which is defined in geometric image space, and h_r, defined in the feature space, which in this case is the L*u*v* colorspace. The main advantage of the mean shift procedure is that the local maximum of the density function associated with each pixel can be found relatively quickly.

More specifically, a given image pixel $\mathbf{x}$ can be described by its spatial coordinates $\mathbf{x}^s$ and L*u*v* features $\mathbf{x}^r$, such that $\mathbf{x} = (\mathbf{x}^s, \mathbf{x}^r)$. An image with N pixels $\mathbf{x}_1, \ldots, \mathbf{x}_N$ defines the density function of interest

$$\hat{f}(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} K(\mathbf{x} - \mathbf{x}_i),$$

where K is the kernel function. The kernel function we adopt is

$$K_{h_s,h_r}(\mathbf{x}) = \frac{C}{h_s^2 h_r^3} k\left(\left\|\frac{\mathbf{x}^s}{h_s}\right\|^2\right) k\left(\left\|\frac{\mathbf{x}^r}{h_r}\right\|^2\right),$$

where k is a function of a single variable called *profile*, which defines the shape of the kernel, and C is a normalizing constant.

Using large image space bandwidths, h_s, is beneficial, resulting in more correct and complete segmentations. In practice, however, the use of a large spatial bandwidth is very time consuming, so that we have fixed this bandwidth at $h_s = 30$ pixels. After having fixed this parameter, varying the feature space bandwidth h_r will determine the number of segments in the result. We have experimented with the range $h_r \in \{5, 10, 20\}$.

Figure 3.4 presents mean shift execution times as a function of the bandwidth parameters h_s and h_r. The times shown are averages taken over the manually labeled images from the *Lab* dataset, after having resized them so that their largest dimension was 700 pixels. See Section 3.6 for specifications of the machine used.

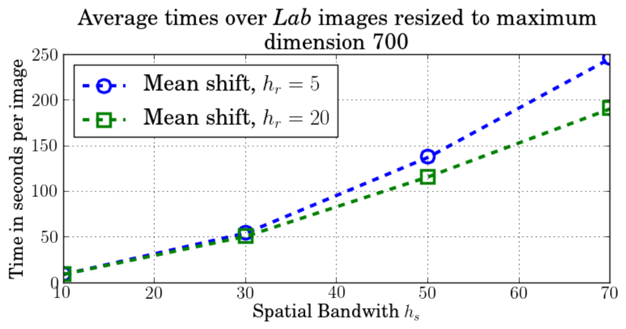


Figure 3.4: Average execution times of the mean shift algorithm on images from the *Lab* dataset as a function of the bandwidth parameters h_s and h_r.

3.5.3.4 GrabCut

Rother et al. presented the GrabCut [24] system for the purpose of interactive image segmentation. An open source implementation of the method is available within the OpenCV library [38], which we use in this study.

Though our application is not interactive, we take advantage of GrabCut's sophisticated segmentation algorithm. The segmentation approach can be seen as an extension to the graph cut algorithm, which was reviewed in Section 3.4.2. The important difference is that GrabCut assumes that the color distributions of the foreground and background classes are unknown. The distributions are modeled in color space each by a Gaussian Mixture Model (GMM). The segmentation problem then consists of jointly finding the best set of pixel labels (which define the segmentation), along with the best set of GMM parameters, as to minimize a Gibbs energy function. The energy defined is analogous to the cost from Equations 3.5 to 3.11, with the main difference being the addition of the GMM terms.

More specifically, the energy to be minimized is

$$\mathbf{E}(\mathbf{y}, \mathbf{k}, \boldsymbol{\theta}, \mathbf{z}) = U(\mathbf{y}, \mathbf{k}, \boldsymbol{\theta}, \mathbf{z}) + V(\mathbf{y}, \mathbf{z}),$$

where U denotes the data term, and will be defined as the the log-likelihood of the pixels' GMM probabilities, and V denotes a smoothness term, based on neighboring pixels, as described further below. $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ denotes the binary segmentation, with all $y_i \in \{0, 1\}$, and N the total number of image pixels. Similarly, $\mathbf{z} = (z_1, z_2, \ldots, z_N)$ with each z_i containing the RGB values for pixel i, and $\mathbf{k} = (k_1, k_2, \ldots, k_N)$ with each k_i indicating the unique GMM component to which pixel i is assigned. (GrabCut uses hard component assignments for speed purposes.) Finally, $\boldsymbol{\theta}$ contains the set of GMM parameters of both classes.

The smoothness term $V(\mathbf{y}, \mathbf{z})$ is analogous to the boundary term in Equation 3.11, but defined in color space:

$$V(\mathbf{y}, \mathbf{z}) = \gamma \sum_{\{i,j\} \in \mathcal{N}} \delta(y_i, y_j) \exp\left(-\beta \|z_i - z_j\|^2\right).$$

Each of the terms above is described in turn. γ is a model parameter indicating the relative importance of the data and smoothness terms U and V. $\mathcal{N}$ denotes the set of neighboring pairs of pixels in the image, which is taken to be the pairs of 4-neighbors. δ is defined as in Equation 3.10. z_i and z_j are respectively the RGB values for pixels i and j. β is another method parameter, which is computed on a per-image basis from an estimate of the image noise. On the other hand, γ above assumes a fixed value, which must be set manually. We set $\gamma = 2$ so that the method works well across the different varieties of leaf images.

In order to minimize the energy function, an iterative scheme was devised [24]. Minimization simply consists of alternating between updating the GMM models according to the running set of labels, and then using the updated models to compute new class likelihoods for each pixel. The class likelihoods are then used to find a new set of labels via the graph cut method, and the whole process is iterated until convergence.

The method requires an initial estimate of the GMM parameters to be supplied. This estimate is computed in practice by setting an initial image map, which provides one of four possible labels for each pixel: "certainly foreground", "probably foreground", "certainly background", and "probably background". We carefully initialize the image map in the following manner: pixels on the image border are labeled as "certainly background"; pixels with high saturation and low value are labeled as "probably foreground"; and the remaining pixels are labeled as "probably background".

3.5.3.5 Multiscale Normalized Cut (MSNcut)

The normalized cut (Ncut) criterion for image segmentation was proposed by Shi and Malik [39], along with a method to approximate its solution. The normalized cut approach treats the segmentation problem as a node partitioning problem, where image pixels are considered as nodes in a graph. Thus, the partition also defines a cut of the graph edges. The novelty of the normalized cut lies in allowing such a graph-based approach to not only consider the dissimilarity between different groups of pixels, but to also compensate for the within group similarities.

We experimented with the normalized cut implementation provided by its authors. The method seems to work better as the graph connection radius increases, which adds more edges to the graph being cut. However, increasing the connection radius is prohibitively slow, rendering this method impractical for our application. Thus, we did not include experiments with normalized cut, and instead worked with the multiscale version, described below.

Cour et al. presented a multiscale algorithm to solve a constrained version of the normalized cut problem (MSNcut) [40]. Their method allowed the use of graphs with longer-range dependencies, in effect allowing for the use of larger images with improved results. The method decomposes the weight matrix, which represents all graph edge weights, as a sum over multiple scales. At smaller scales, more pixels are considered for forming edges, but the connection radius is limited, while at larger scales, pixels are sub-sampled more sparingly, allowing the connection radius to increase. The problem formulation considers the graphs at different scales simultaneously, with an added explicit constraint that the segmentation obtained be consistent across scales. The problem of finding the optimal partition is then approximated by a constrained optimization problem, whose solution can be found in time linear in the number of pixels.

The authors provide an implementation on their website, which we use here. The implementation allows graph edge weights to be defined by a combination of two terms: pixel color similarity, and an intervening contours [41] similarity. Using the intervening contour produced worse results on our images, so we only use pixel color similarities to define the edge weights.

Increasing the graph connection radii improved results and we have set them as large as possible, up to the memory limitation of our machine. As an example of the memory consumption of the algorithm, if we are working with an image of size 700×525, we can choose to use four different scales to work with, as done in the original authors' implementation. Let us denote the set of connection radii associated to each scale as $R = \{r_1, r_2, r_3, r_4\}$, where r_i is measured in pixels. Under this setting, if we set $R_1 = \{6, 9, 12, 18\}$, the method consumes a peak of 2.1 GB, whereas for a larger $R_2 = \{10, 15, 20, 30\}$, the memory peak is of 4.9 GB. In practice,when dealing with images of this size, we use the set of radii R_2. We varied the number k of segments output by the method within the range $k \in \{2, 10, 20, 30, 40\}$.

3.5.3.6 Segmentation by Weighted Aggregation (SWA)

Sharon et al. introduced segmentation by weighted aggregation (SWA) [42]. SWA is a multiscale graph-based approach that forms pixel aggregates, represented as nodes in graphs. Edges initially link neighboring pixels, with weights that represent pixel similarity. The overall objective of the method is to segment an image into multiple salient regions. A saliency criterion is defined for a given group of graph nodes, which measures the weight of edges that leave the group of nodes relative to the weight of edges within the group of nodes. This is in the same spirit of the normalized cut criterion (see Section 3.5.3.5), though the exact formulated criterion is different.

The main part of the method proceeds in a bottom-up fashion. It begins by assigning every pixel to a small aggregate, each aggregate containing a representative node. These assignments are done based on pixel affinities, which are stored in the graph edge weights. Aggregates are then recursively aggregated together, leading to a series of graphs, each at a different level of coarseness. The result is a pyramid of graphs, from which the segmentations will be later obtained. At any given level of the pyramid, aggregation is done by selecting a set of representative nodes, such that each of the non-representative nodes are strongly connected to at least one representative. The relationship between the aggregates at successive levels is stored in an interpolation matrix, computed from the edge weights. The main observation here is that the saliency computed at a given level can be approximately represented by the saliency at its corresponding successive coarser level. The aggregation process is such that at any given level of the pyramid, aggregates are are allowed to have pixels that overlap.

After building the pyramid of aggregates, a segmentation can finally be obtained through a top-down procedure. First, from the pyramid, a set of the most salient aggregates is selected. Note that in principle the selected salient aggregates can be from different levels of the pyramid. Each aggregate in this set is repeatedly projected down onto finer levels via the interpolation matrices that were computed when building the pyramid as to compute weights that link the finer level aggregates with their higher level representatives. At the finest level, each image pixel will be assigned to the aggregate with which it has the highest linking weight.

Compared to other methods, an important advantage of this type of approach is that the larger aggregates allow for an appropriate extraction of texture features, which would not be possible at finer levels [43]. At the same time, descending to lower levels allows the fine details of the segments to be preserved. Also of importance is that the multiscale strategy allows the method to be linear time in the number of pixels.

The authors have provided an implementation of the method on their website. It has has a series of adjustable parameters, though we have found reasonable results by using the default parameter settings. The algorithm is capable of producing a collection of segmentations, each with a different level of coarseness, corresponding to a different level of the pyramid. We denote the pyramid level here by c, whose range starts at 1 (which produces the coarsest segmentation, usually containing only two segments). We experimented with $c \in \{1, 2, 4, 6, 8\}$ and left the other parameters at their default values.

3.5.3.7 Global Pb with Oriented Watershed Transform and Ultrametric Contour Map (gPb+OWT+UCM)

Arbelaez et al. [11] presented an approach for segmentation with a series of steps. It starts out by computing a multiscale version of the probabilistic boundary detector due to Martin et al. [44]. From this initial boundary map, a global boundary map is computed using a spectral clustering formulation. This global boundary map is used to to produce a super-segmentation of the image via the Oriented Watershed Transform. Finally, a hierarchical collection of segmentations, represented using the Ultrametric Contour Map, is computed from the Oriented Watershed Transform. We have experimented with the implementation provided by the authors of the method. Due to its memory requirements and our required image resolution, we were not able not perform comprehensive experiments with this method. See Section 3.6 for a discussion.

3.5.3.8 Expectation-Maximization (EM)

We experiment with our segmentation method proposed in Section 3.4, using our non-optimized implementation in Matlab. However, this version of the method has two important modifications that make it simpler. First, pixel weights for EM are determined via a manually delineated region in saturation-value colorspace, as in the preliminary version of our method [1]. Second, no graph cut step is applied. For consistency between all methods, we never include the stem removal step from Section 3.4.4. It should also be noted that we also do not include the speed optimizations from Section 3.4.1.1, so that the times reported here are significantly larger than those of an optimized version.

3.5.3.9 Expectation-Maximization with trained pixel weighting (EM+-TW)

This version of the method includes the trained pixel weighting used during EM, as opposed to using a hand-drawn delineation for the weighting scheme (see Section 3.4). Since this method requires a training set of manually segmented images, it is evaluated via two-fold cross-validation. Note, though, that here we still do not apply the graph cut step.

3.5.3.10 Expectation-Maximization with trained pixel weighting and graph cut (EM+TW+GC)

The method we propose here, based on Expectation-Maximization, and followed by a graph cut step, is described in Section 3.4. As with the previous method (EM+TW), it requires training data and is evaluated via two-fold cross-validation.

3.5.4 Performance Metrics

We evaluate the following performance measures for each of the different methods compared. For the quantitative analysis, we experimented on the images from the *Lab* and *Field* datasets that have manual segmentations. This allows us to compute measures of the following three important segmentation characteristics: how well the pixels from a segmentation agree with the ones from its respective manual segmentation; how well its boundary matches the boundary from the manual segmentation; and how similar the features computed from a given segmentation are to the ones computed from the manual segmentation. We will describe the metrics used to assess these characteristics further below. We also time the methods, since we are concerned with using them in an interactive application.

For quantifying the degree of agreement between pixels from a method's segmentation and a manual segmentation, accuracy is a very intuitive measure. However, when only a fraction of the pixels belong to the leaf class, accuracy is very insensitive. We also measure pixel precision, recall and F-measure (F_1 score) in this case, with the F-measure acting as a reasonable overall summary of performance [13].

The pixel agreement measures defined above are not sensitive to important image features that have only a small number of pixels, such as leaf serrations, or thin leaf tips. Thus, we also include measures of boundary agreement. Again, we use precision, recall and the F-measure, but computed over the boundary pixels. For deciding whether a point on the boundary is considered a true positive, false positive, true negative or false negative, we find a correspondence between the points of the boundaries produced by the method and the ground truth. The correspondence is done using the assignment procedure described by Martin et al. in the appendix of [44], whose code is provided along with the Berkeley Segmentation and Boundary Detection Benchmark and Dataset [12].

Finally, to give us an idea of the effect of the different methods on system performance, we use shape features computed from the segmentations. The shape feature we use is the histograms of curvature over scale (HoCS) [1]. After normalizing the histograms, the similarity between the features provided by a segmentation method $a = (a_1, a_2, \ldots, a_n)$ and those from the corresponding manual segmentation $b = (b_1, b_2, \ldots, b_n)$ is defined by their histogram intersection as $s(a, b) = \sum_{i=1}^{n} \min(a_i, b_i)$.

3.5.5 Testing for Statistically Significant Differences

We use hypothesis tests in order to compare a given pair of methods according to a performance metric. In general, a reasonable idea about which method performs better can be obtained by simply comparing the means or medians of the methods on a given dataset. However, especially on smaller datasets, there is some variance associated to these mean and median values. Hypothesis testing allows us to attach a confidence to a comparison, by computing the probability of the observed outcome under the (null) hypothesis that the two methods in fact have equivalent performance.

In our quantitative experiments, we adopt the sign test. The discussion and notation below follow Dixon and Mood [45]. Given a metric and a pair of methods to be compared, we treat the value of the metric obtained by each method when applied to leaf images as a random variable. It is difficult to make any strong assumptions about the probability distributions of our metrics, due to their complex nature. Thus, we resort to the sign test, which makes very few assumptions about its underlying distributions. A downside of the sign test is that it has reduced statistical power relative to others such as the paired t test (i.e., it is more conservative).

Suppose an observed leaf dataset has n images. For each image $i \in \{1, \ldots, n\}$, we compute a pair of metrics produced by the two segmentation methods, which we denote (x_i, y_i). The sign test takes into consideration only the signs of the differences $x_i - y_i$. The main assumptions of the test are the following. First, it assumes that there is a fixed (and unknown) probability $p = Pr(x_i > y_i)$, with $0 < p < 1$. In other words, p is the probability that, for any pair of observations (x_i, y_i), we will have $x_i > y_i$. Second, it is assumed that the different observation pairs $(x_i, y_i), i = 1, \ldots, n$ are independent of each other. These weak assumptions contrast with, for example, those of the paired t test, which requires that the differences between paired observations be normally distributed.

Our null hypothesis is that $p = 1/2$, which is equivalent to assuming that the median difference in the metrics resulting from the two methods is zero. Given the nature of our metrics, it is safe to assume that $Pr(x_i = y_i) = 0$, so that $Pr(x_i < y_i) = 1 - p$. Thus, if we denote by w the number of pairs for which $x_i > y_i$, then under the null hypothesis w will follow a binomial distribution of probability 1/2.

All tests we perform are two-tailed, since we cannot make any prior assumptions about which of the two methods being compared is better. To perform the test, we count the number of pairs w for which $x_i > y_i$ and $n - w$ for which $x_i < y_i$. Let r denote the smaller of the two counts, i.e. $r = \min\{w, n - w\}$. Given an observed value of r, the corresponding p-value is $Pr(R \leq r)$, where R denotes a random variable from the same distribution that generated r. Computation of the p-value $Pr(R \leq r)$ is done by adding up the values of the binomial distribution that correspond to $R \leq r$, which will span both of its tails [45]. We set the significance value to the commonly adopted $\alpha = 0.05$. That is, if we obtain a p-value smaller than 0.05, we reject the null hypothesis and we call a given difference between methods significant.

In our quantitative experiments, we study two different leaf populations in turn: the first is that of images taken in laboratory settings (for which we experiment with the *Lab* dataset), and the second is that of images taken by researchers in the field (for which we use the *Field* dataset). These datasets are described in Section 3.5.1.

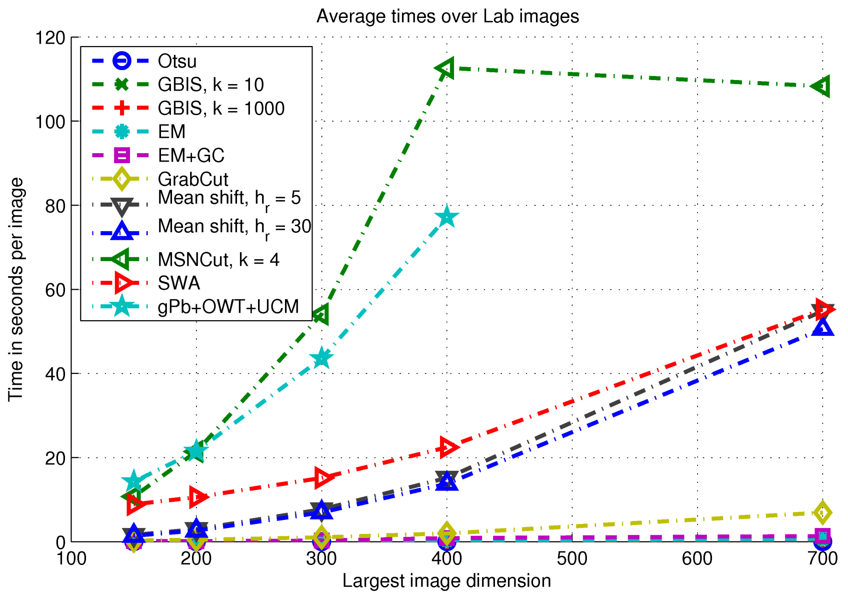
3.6 Results

We first present a comparison of execution times. All experiments were performed on a machine with 2 quad-core Intel Xeon CPUs, at 2.13 GHz clock speed and 4 MB cache. The machine had 12 GB RAM. All manually labeled images from the *Lab* dataset were used for measuring average times. Each image was resized so that its maximum dimension (either height or width) was set to a predetermined value, while preserving its aspect ratio. Figure 3.5a presents the average execution time per image as a function of image size. In order to better visualize the times for the faster methods, these are again plotted in greater detail in Figure 3.5b. Observe in particular that, when the largest image dimension is set to 700 pixels, GrabCut takes around 7 seconds per image. Though GrabCut is among the fastest tested methods, this speed will not be satisfactory for many interactive applications. For Mean shift, MSNcut, SWA, and gPB+OWT+UCM, the average computation times for images with their largest dimension set to 700 are above 50 seconds, which restricts their applicability to our problem.

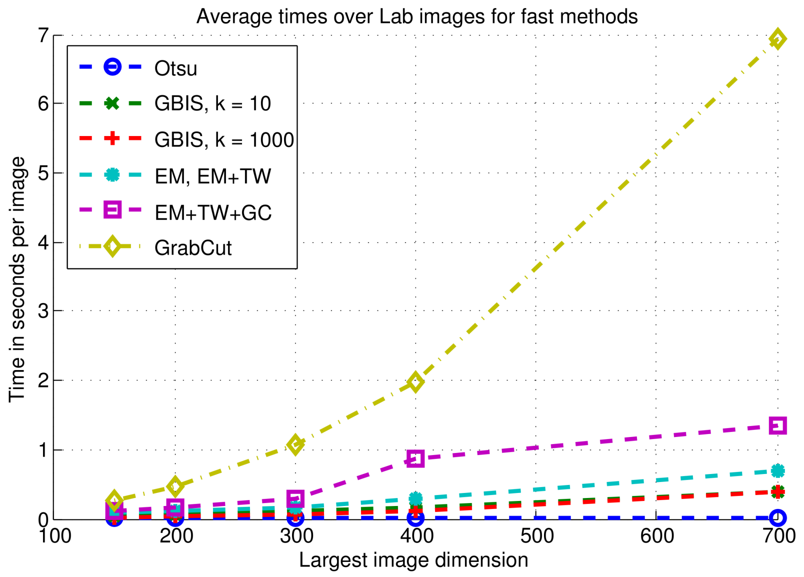
It is important to point out some particularities of the previous experiment when the largest image dimension was set to 700 pixels. Note first that, at 700 pixels, the execution times are not available for gPB+OWT+UCM. This was due to the memory requirements of the method, which were not met by the machine. Note also that in general the time for MSNcut increases with image size. However, for the case when the largest image size is 700 pixels, due to our limited memory, we decreased the graph connection radii across scales. This had the side effect of not increasing the method's execution times, though the results with smaller radii tend to be worse.

For the remaining experiments that follow, we resized all images so that their maximum dimension was 700 pixels. This resolution preserves most of the leaf image details, allowing us to capture thin stems and small-scale leaf serrations. We have excluded the gPB+OWT+UCM method from the remaining analysis, since it would require introducing some major modifications in order to run at the desired resolution without running out of memory.

As a principal metric to summarize performance, we measure the boundary agreement between the segmentation produced by a given method and the corresponding ground-truth manual segmentation. This agreement can be quantified by the F-measure, computed as described in Section 3.5.4. A more complete set of results is presented in the supplementary material of [46], though usually all of the measures follow the same trends. The boundary agreement F-measure has the ad-



(a) Execution times for all methods.

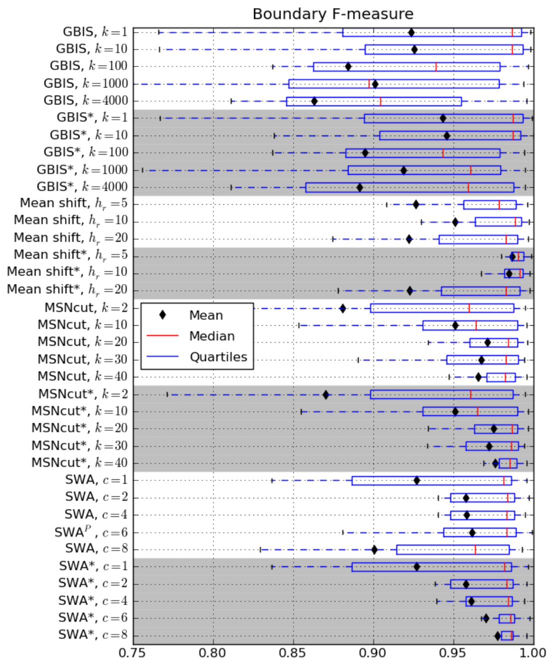


(b) Detail showing execution times only for fast methods.

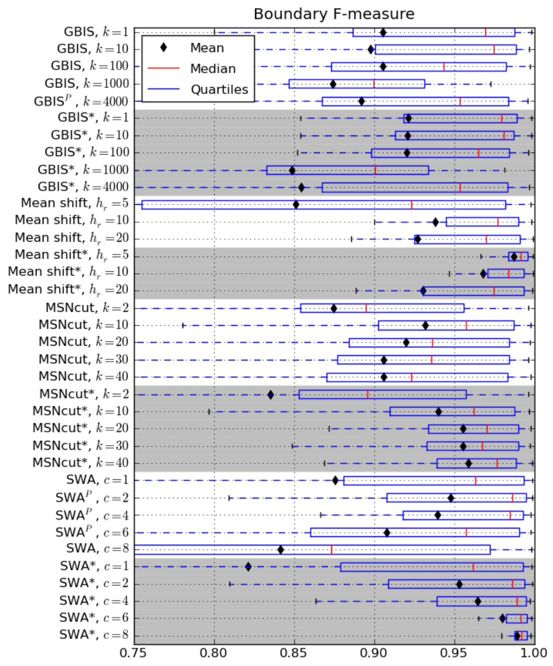
Figure 3.5: Average execution times per image on the *Lab* dataset as a function of image size. (a) shows the times for all methods, while (b) shows a detail with only the fastest methods.

vantage of being sensitive to differences in the shapes of the segmentations, which will finally be used for leaf identification. A brief discussion on the merits of the different metrics was presented in Section 3.5.4.

Figures 3.6 and 3.7 show the distribution of F-measures on the *Lab* and *Field* datasets. These were computed on the 30 *Lab* images and 56 *Field* images for which manual segmentations were available. In Figure 3.6, when a method's name is marked with an asterisk, it indicates that the method was used to produce over-segmentations of the images, which were then evaluated according to the best possible assignment of the segments to leaf and background. The best possible assignment was determined using the ground truth manual segmentations, and provides an upper-bound on the method's performance (see Section 3.5.2). On the other hand, the absence of an asterisk on a method that produces over-segmentations indicates that the segments were assigned according to the heuristic strategy described in Section 3.5.3. In order to obtain a fair comparison with methods that produce binary segmentations, in Figure 3.7, methods that produce over-segmentations always had their segments assigned according to the heuristic strategy (denoted without an asterisk). All methods were run with and without the post-processing procedure described in Section 3.4.3. We report only the result that produced the best mean boundary F-measure: in the figures, when a method is marked with a superscript P, it indicates post-processing improved the result and is therefore reported, whereas the absence of the P indicates post-processing did not improve the results, so that the result without post-processing is reported. This gives us a more meaningful comparison between methods. In any case, it is important to point out that



(a) Boundary F-measures on *Lab* dataset.

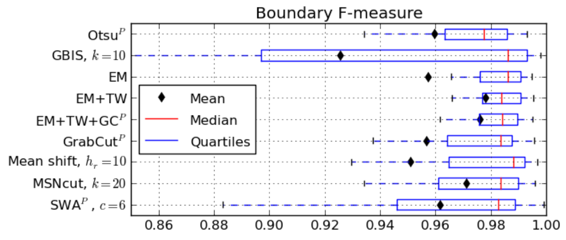


(b) Boundary F-measures on *Field* dataset.

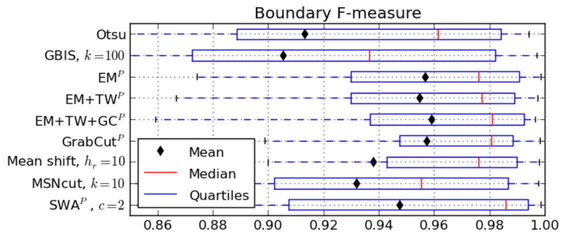
Figure 3.6: Performance of methods which produce over-segmentations and require a parameter to be chosen. (For a summary of the results of all methods, including those without parameters to be chosen, refer to Figure 3.7.) Boundary agreement F-measures are shown for images from the *Lab* and *Field* datasets. Higher values indicate better performance. There were a total of 30 *Lab* images and 56 *Field* images. Methods marked with an asterisk (whose over-segmentations were evaluated according to the best possible assignment of segments to leaf and background) have been shaded in gray. The boxes in the plots contain the second and third quartiles, while the vertical red line indicates the median value.

the post-processing procedure can be very beneficial for certain methods, as shown later in Figure 3.12.

Figure 3.6 shows only the methods which produce over-segmentations, having a parameter that is varied throughout a range. Table 3.1 shows the average number of segments produced by these methods for each of their parameter settings. In Figure 3.6, note first that the methods marked with an asterisk improve with the number of segments that they produce, up to the point of achieving very high



(a) Boundary F-measures on *Lab* dataset.



(b) Boundary F-measures on *Field* dataset.

Figure 3.7: Boundary agreement F-measures for images from the *Lab* and *Field* datasets. Higher values indicate better performance. There were a total of 30 *Lab* images and 56 *Field* images. The boxes in the plots contain the second and third quartiles, while the vertical red line indicates the median value.

F-measures.[1] For very fine segmentations, though, the methods are not really solving the original leaf segmentation problem, but only part of it. Thus, we are not interested in comparing its results in this case to those of methods that directly produce binary segmentations. On the other hand, when the number of segments produced is small, note from Figure 3.6 that the performance of the heuristic assignment is very close to the upper bound performance given by the best assignment. This lets us know that in this case the heuristic is working as well as possible. This is the most interesting case for us for comparison purposes: a fairer comparison to methods that produce binary segmentations is obtained when the over-segmentation methods are set to produce few segments as to try, as much as possible, to solve the original segmentation problem. Finally, note that, for the methods evaluated with heuristic assignments, the performance has a peak at a certain parameter setting, at which point using either a finer or coarser segmentation will result in a performance decrease.

[1]GBIS is an exception, since even at the smallest observation scale ($k = 1$), it still does not produce a fine enough segmentation.

Table 3.1: Mean number of segments produced by over-segmentation methods with different parameter settings. The mean number of segments generated is indicated along with its respective standard deviation. The methods were run on the subsets of *Lab* and *Field* images for which manual segmentations were available.

Method	Mean number of segments on *Lab*	Mean number of segments on *Field*
GBIS, $k = 1$	35.9 ± 4.6	35.7 ± 5.6
GBIS, $k = 10$	35.7 ± 8.2	47.7 ± 4.7
GBIS, $k = 100$	14.3 ± 7.8	5.9 ± 3.7
GBIS, $k = 1000$	6.8 ± 4.5	4.3 ± 2.1
GBIS, $k = 4000$	4.0 ± 2.4	3.1 ± 1.2
Mean shift, $h_r = 5$	67.4 ± 72.4	97.0 ± 51.0
Mean shift, $h_r = 10$	17.5 ± 24.5	14.3 ± 8.3
Mean shift, $h_r = 20$	10.8 ± 16.5	4.5 ± 3.1
MSNcut, $k = 2$	2.0 ± 0.0	2.0 ± 0.0
MSNcut, $k = 10$	10.0 ± 0.0	10.0 ± 0.0
MSNcut, $k = 20$	20.0 ± 0.0	20.0 ± 0.0
MSNcut, $k = 30$	30.0 ± 0.0	30.0 ± 0.0
MSNcut, $k = 40$	40.0 ± 0.0	40.0 ± 0.0
SWA, $c = 1$	2.1 ± 0.3	2.4 ± 0.6
SWA, $c = 2$	3.7 ± 1.0	5.0 ± 2.0
SWA, $c = 4$	14.9 ± 7.9	25.4 ± 14.7
SWA, $c = 6$	74.7 ± 46.3	157.9 ± 115.4
SWA, $c = 8$	455.4 ± 323.0	1189.3 ± 1001.3

In Figure 3.7, all methods are present, with only the best performing parameter settings reported, chosen according the highest mean boundary F-measure. In the figure, the methods that produce over-segmentations were evaluated using the heuristic assignment strategy, as to allow for a fair comparison. Figure 3.7 shows that EM+TW, and EM+TW+GC are consistently the highest scoring on both datasets. They are followed closely by EM, GrabCut, MSNcut and SWA. As was previously shown in Figure 3.5, MSNcut and SWA are significantly slower than other methods, which puts them at a practical disadvantage. On the other hand, EM and EM+TW are the fastest of the best performing methods, followed closely by EM+TW+GC, then GrabCut. Otsu, GBIS, and Mean shift in general produce worse results.

We performed several sign tests to compare different pairs of methods, as described in Section 3.5.4. The p-values for the tests comparing boundary F-measures are shown in Tables 3.2 and 3.3, respectively for the *Lab* and *Field* datasets. Again, here the methods that produce over-segmentations were evaluated using the heuristic assignment strategy. The performance differences follow the same trends on both datasets, but note that statistical significance appears much more frequently on the *Field* dataset. The corresponding tables for the other performance measures are presented in the supplementary material of [46]

Figure 3.8 presents the results from Table 3.3 in the form of a graph. When there is a statistically significant difference between a pair of methods, as measured by boundary agreement F-measures on the *Field* dataset, there is an arrow going from the better method to the worst. The absence of an arrow indicates that the difference between methods was not significant. Methods are grouped together

Table 3.2: P-values of two-sided sign tests comparing boundary F-measures between different methods. These values were computed on the *Lab* dataset using manual segmentations. Bold values indicate a significant difference at $\alpha = 0.05$. A plus-sign (+) after the p-value indicates that the method on the row is better than the method on the column, while a minus-sign (−) indicates the converse is true.

	Otsu^P	GBIS, $k = 10$	EM	EM+TW	EM+TW+GC^P	GrabCut^P	Mean shift, $h_r = 10$	MSNcut, $k = 20$	SWA^P, $c = 6$
Otsu^P	–	0.3616(−)	**0.0428(−)**	0.0987(−)	**0.0161(−)**	0.0987(−)	0.0987(−)	0.4583(−)	0.5847(−)
GBIS, $k = 10$	0.3616(+)	–	0.3616(+)	0.8555(−)	0.8555(−)	0.8555(+)	1.0000(+)	1.0000(−)	0.8555(+)
EM	**0.0428(+)**	0.3616(−)	–	0.7111(+)	0.1360(+)	0.3616(+)	0.3616(−)	0.2005(+)	0.8555(+)
EM+TW	0.0987(+)	0.8555(+)	0.7111(−)	–	0.8555(+)	0.2005(+)	0.3616(−)	0.8555(+)	0.3616(+)
EM+TW+GC^P	**0.0161(+)**	0.8555(+)	0.1360(−)	0.8555(−)	–	0.5847(+)	0.8555(−)	1.0000	0.3616(+)
GrabCut^P	0.0987(+)	0.8555(−)	0.3616(−)	0.2005(−)	0.5847(−)	–	0.2005(−)	1.0000(−)	0.8555(−)
Mean shift, $h_r = 10$	0.0987(+)	1.0000(−)	0.3616(+)	0.3616(+)	0.8555(+)	0.2005(+)	–	0.2649(+)	**0.0428(+)**
MSNcut, $k = 20$	0.4583(+)	1.0000(+)	0.2005(−)	0.8555(−)	1.0000	1.0000(+)	0.2649(−)	–	0.2005(+)
SWA^P, $c = 6$	0.5847(+)	0.8555(−)	0.8555(−)	0.3616(−)	0.3616(−)	0.8555(+)	**0.0428(−)**	0.2005(−)	–

Table 3.3: P-values of two-sided sign tests comparing boundary F-measures between different methods. These values were computed on the *Field* dataset using manual segmentations. Bold values indicate a significant difference at $\alpha = 0.05$. A plus-sign (+) after the p-value indicates that the method on the row is better than the method on the column, while a minus-sign (−) indicates the converse is true.

	Otsu	GBIS, $k = 100$	EM^P	EM+TW^P	EM+TW+GC^P	GrabCut^P	Mean shift, $h_r = 10$	MSNcut, $k = 10$	SWA^P, $c = 2$
Otsu	–	0.6889(−)	**0.0000(−)**	**0.0000(−)**	**0.0000(−)**	**0.0018(−)**	**0.0018(−)**	0.1770(−)	**0.0007(−)**
GBIS, $k = 100$	0.6889(+)	–	**0.0105(−)**	**0.0105(−)**	**0.0046(−)**	**0.0007(−)**	**0.0018(−)**	1.0000	**0.0001(−)**
EM^P	**0.0000(+)**	**0.0105(+)**	–	0.2806(+)	0.6889(−)	0.6889(−)	0.8939(+)	**0.0222(+)**	**0.0440(−)**
EM+TW^P	**0.0000(+)**	**0.0105(+)**	0.2806(−)	–	0.6835(−)	1.0000	0.8939(−)	**0.0222(+)**	**0.0440(−)**
EM+TW+GC^P	**0.0000(+)**	**0.0046(+)**	0.6889(+)	0.6835(+)	–	0.1409(+)	0.8939(+)	**0.0046(+)**	0.1409(−)
GrabCut^P	**0.0018(+)**	**0.0007(+)**	0.6889(+)	1.0000	0.1409(−)	–	0.6889(−)	**0.0440(+)**	**0.0440(−)**
Mean shift, $h_r = 10$	**0.0018(+)**	**0.0018(+)**	0.8939(−)	0.8939(+)	0.8939(−)	0.6889(+)	–	**0.0105(+)**	0.5044(−)
MSNcut, $k = 10$	0.1770(+)	1.0000	**0.0222(−)**	**0.0222(−)**	**0.0046(−)**	**0.0440(−)**	**0.0105(−)**	–	**0.0222(−)**
SWA^P, $c = 2$	**0.0007(+)**	**0.0001(+)**	**0.0440(+)**	**0.0440(+)**	0.1409(+)	**0.0440(+)**	0.5044(+)	**0.0222(+)**	–

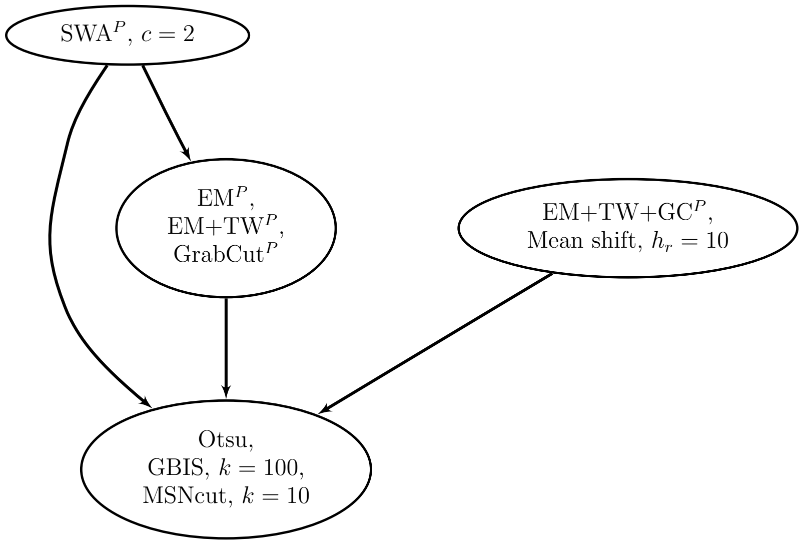

SWA^P, $c = 2$
EM^P, EM+TW^P, GrabCut^P
EM+TW+GC^P, Mean shift, $h_r = 10$
Otsu, GBIS, $k = 100$, MSNcut, $k = 10$

Figure 3.8: Graph showing statistically significant differences between boundary agreement F-measures on the *Field* dataset. An arrow indicates that there was a statistically significant difference between a pair of methods. Methods are grouped together into the same node when they present the same set of differences between other methods and no difference amongst themselves.

into the same node when they present the exact same set of differences between other methods and no difference amongst themselves. Note in Figure 3.8 that SWA was better than many of the other methods. Upon further investigation, it was noted that, most of the time, SWA produces results that are only slightly superior. However, there are some few images for which SWA produces large mistakes that would interfere with recognition, whereas other methods do not. This is reflected in the lower average performance of SWA, which is shown in Figure 3.7. The sign test ignores the severity of these mistakes, as it avoids making any assumptions about the distributions of the performance metrics.

We would also like to point out the following regarding the hypothesis tests. On the *Field* dataset, among all four metrics (pixel accuracy, pixel F-measure, boundary F-measure, and HoCS feature similarity), at significance $\alpha = 0.05$, there is a fairly consistent separation of the methods into two groups. The first group contains the better performing methods, and is composed of EM, EM+TW, EM+TW+GC, GrabCut, Mean shift and SWA. Their performances are not consistently different amongst themselves and are fairly consistently better than the methods in the second group. The second group is composed of Otsu, GBIS, and MSNcut, which can also be seen grouped into the bottom node of Figure 3.8.

Finally, we experimented on the larger datasets: all of the *Lab* images (excluding the ones that have manual segmentations, which were used for parameter adjustment); all of the *Field* images (again excluding those with manual segmentations); and all of the *User* images, whose segmentation parameters were set using images with manual segmentations from the *Field* dataset. Due to the large number of images in these datasets, it was only practical to experiment with the fastest methods, namely Otsu, GBIS, EM, EM+TW, EM+TW+GC, and GrabCut. The other methods showed themselves to be too slow for our application (see Figure 3.5).

In order to make differences between methods evident on these large datasets, we adopt the following procedure. Given a pair of methods that we would like to compare, we order all the images in the dataset by how similar the segmentations produced by both methods are. For example, given methods A and B, the image for which the segmentations produced by A and B are most *dissimilar* should appear first, while the image which produces the most similar segmentations should appear last. Here, we measure similarity between segmentations using their overlap ratio (see e.g. [47]), defined simply as the number of pixels in the intersection of the segmentations, divided by the number of pixels in their union.

The above procedure for comparing pairs of methods is motivated by the following observations. It would be prohibitively time-consuming to provide manual segmentations for very large sets of images, given that manually segmenting out the complex shapes of leaves is a labor intensive process. At the same time, given a pair of methods, by viewing the images for which the resulting segmentations are most dissimilar, we are able to quickly understand some of their major differences. Examples of this behavior will be shown in the figures that follow. In particular, in the majority of cases, simply by looking at the most dissimilar results on a given set of images, it is easy to see which method is performing better.

A major mode of failure for all methods is small pine leaves, which only occupy a small fraction of the image. EM, EM+TW, and EM+TW+GC perform much better on this type of image due to the pixel weighting procedure, though there is still some room for improvement. Figure 3.9 compares EM+TW+GC^P with GrabCut^P on the complete *Lab* dataset, making evident the difficulty of traditional methods on small pine leaves. Next, we would like to note that on the *Lab* dataset, trained pixel weighting brings an important improvement over weighting using a manually delineated region in saturation-value space. Figure 3.10 illustrates this by comparing EM and EM+TW on the *Lab* images. We would also like to note the effect of adding a graph cut step to EM. Graph cut improves the results by requiring more compact segmentations and by being able to position the segmentation boundaries over image edges. This tends to fix errors such as those due to specularities, cast shadows, or leaves with uneven colors. Figure 3.11 illustrates this by showing a comparison between EM+TW and EM+TW+GC on the *User* dataset, where the

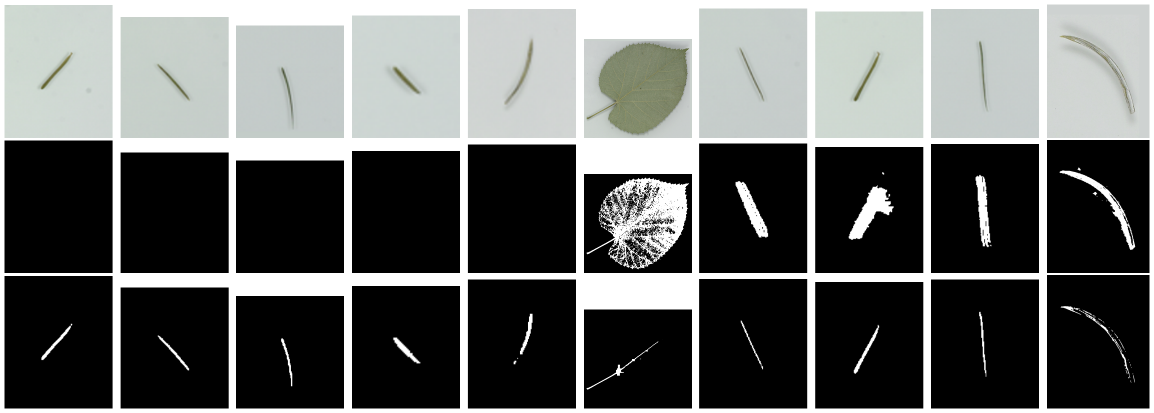


Figure 3.9: Segmentation results on the *Lab* dataset illustrating the difficulty of traditional methods with pine leaves. From top to bottom: original image, result of GrabCut^P (GrabCut, plus post-processing), and result of EM+TW+GC^P (EM with trained pixel weighting and graph cut, followed by post-processing). The images are ordered by overlap ratio between the two segmentations, so that the left-most image has the most dissimilar segmentations, with the similarity increasing as we move right. In order to better illustrate the range of differences, every eighth image is shown. Most images have been cropped closely to the leaves after segmentation, for better visualization. The original image sizes range from about two to five times larger in each dimension.

difference between the two methods is more pronounced. Finally, another common issue is the presence of false positives in the outer regions of the images. These are caused either by poor illumination or an unexpected absence of the light-colored background. The post-processing step we add to the various methods is able to fix this problem in most cases, as exemplified in Figure 3.12.

After visually assessing all of the results, we have the following qualitative observations. As noted in Section 3.2, the following general difficulties were noted on these datasets: images with small pine leaves; complex compound leaves; uneven illuminations; cast shadows; specularities; natural variations in color; and venations. Overall, even for EM+TW+GC^P, which performs very well, there appears to be a good amount of room for improvement due to these difficulties. The *User* images

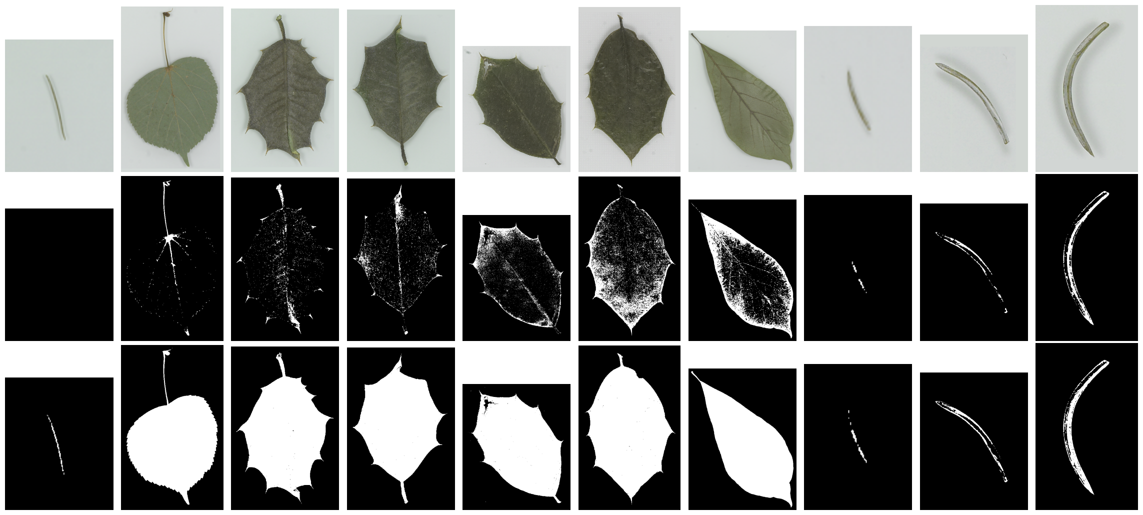


Figure 3.10: Segmentation results on the *Lab* dataset illustrating the effects of training for EM pixel weighting. From top to bottom: original image, result of EM (with pixel weighting using a manually delineated region), and result of EM+TW (EM with trained pixel weighting). The images are ordered by overlap ratio between the two segmentations, so that the left-most image has the most dissimilar segmentations, with the similarity increasing as we move right. In order to better illustrate the range of differences, every eighth image is shown. The first image and the third to last image show zoomed-in details of the originals for better visualization.

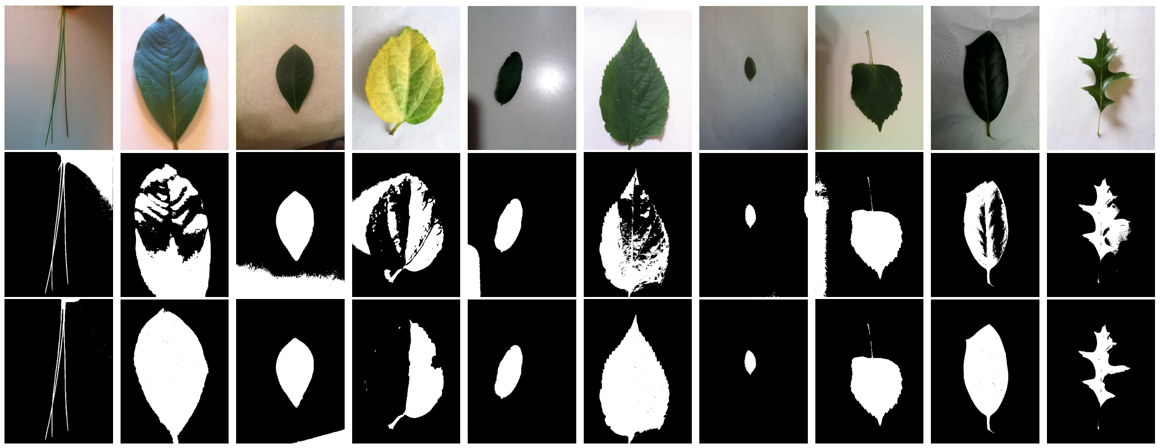


Figure 3.11: Segmentation results on the *User* dataset illustrating the effects of adding the graph cut step. From top to bottom: original image, result of EM+TW, and result of EM+TW+GC. The images are ordered by the overlap ratio between the two segmentations, so that the left-most image has the most dissimilar segmentations, with the similarity increasing as we move right.

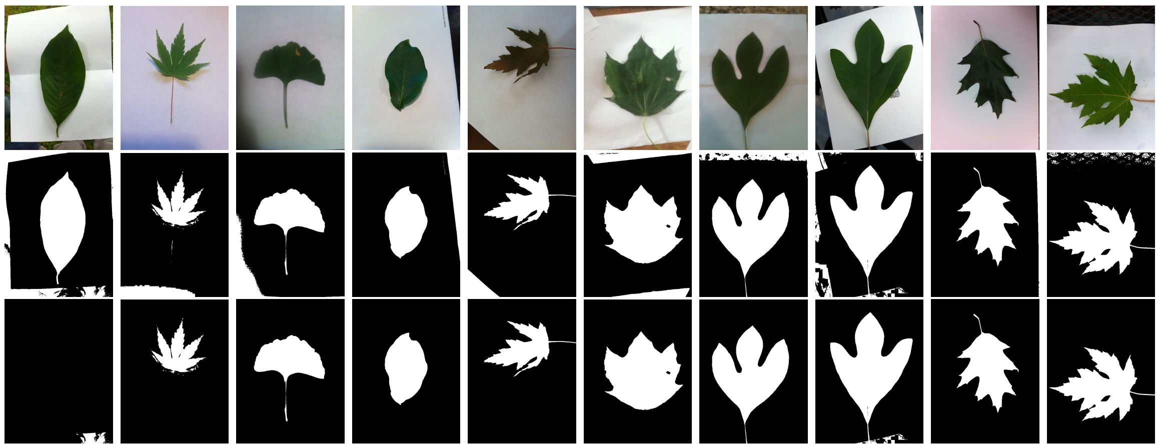


Figure 3.12: Segmentation results on the *User* dataset illustrating the effects of post-processing. From top to bottom: original image, result of EM, and result of EM^P (EM with post-processing). The images are ordered by the overlap ratio between the two segmentations, so that the left-most has the most dissimilar segmentations, with the similarity increasing as we move right. In order to better illustrate the range of differences, every eighth image is shown. Note the post-processing procedure was in error on the left-most image, though in all the other images it improved the segmentations. The post-processing error on the leftmost image occurred due to the following. In post-processing, we begin by dilating the segmentation, so that close by segments get merged together. This caused the leaf part of the segmentation to merge to the large segment that surrounds it. This merger was not expected by our algorithm and resulted in the error. The surrounding segment with which the leaf segment merged falls outside the sheet of paper in the original image, and ideally should be eliminated. However, in this case the complete merged segment was later judged to be lying along the image boundary, and was eliminated as a whole.

proved to be much more challenging than the *Field* and *Lab*, due to the large variety of imaging conditions.

3.7 Conclusion

This chapter presented a study on efficient segmentation of leaves in semi-controlled conditions. The leaf images experimented with were acquired from Leafsnap – an interactive application for plant species identification that employs computer vision. The results showed that several general-purpose segmentation algorithms do not work satisfactorily on this problem when they are tested over large datasets containing a variety of species. Some of the methods experimented with are too slow, or have steep memory requirements. Other methods were able to work reasonably well, but would require important modifications in order to produce competitive results across the different leaf species.

In order to address the segmentation problem, we have proposed a method based on color space clustering with a pixel weighting strategy. A graph cut step is then applied, allowing us to incorporate image edges as an important cue. This helps overcome problems due to shadows and specularities. We adjust the important parameters of the method using training. By introducing training with manual segmentations, the method was able to work well on different leaf datasets with minimal manual parameter adjustment. In practice, the resulting method is fast and presents state-of-the-art results. In our quantitative experiments, it showed to consistently be among the top performing methods, while the qualitative results clearly showed the benefits of the new proposed extensions.

Considering that segmentation will be applied within an interactive application, one line of improvement is to allow a user to guide the segmentation process, by interactively indicating leaf and background regions such as in the work of Boykov and Jolly [27]. Another line of improvement could consider more complex models, by adding pixel classes corresponding to cast shadows, specularities, or venations, and restrictions on the relative colors and positions between different classes, or on their shapes.

Chapter 4: Operational Estimation of a Comprehensive Set of Complementary Shape, Size, and Photometric Attributes of Image-Objects

4.1 Introduction

This chapter presents an original selection and implementation of a comprehensive set of complementary polygon (image-object, segment)-specific feature estimators, eligible for use in the popular object-based image analysis (OBIA) paradigm [48]. These feature estimators are: (i) provided with an intuitive meaning – perhaps physical, or perceptual – to be easy to deal with by human experts, (ii) scale invariant, (iii) computationally efficient and (iv) equivalent to multiple sources of complementary information. In particular, selected per-segment variables encompass both photometric attributes (e.g., mean intensity, intensity variance, mean contrast along the boundary, etc.) and geometric (i.e., shape and size) properties. To enhance their usability by human experts, whenever possible, segment-specific descriptors are conceived as scalar variables provided with no unit of measure and a normalized (percentage) range of change.

Our working hypothesis is that, together with an input image, an image-derived segmentation map is made available on an *a priori* basis, i.e., an image segmentation map is provided as input together with the test image for estimation of the segment-specific descriptive variables. *Image segmentation* is an inherently ill-posed early vision problem [49], whose output is a mutually exclusive and totally exhaustive partition of an input image into segments, otherwise called regions, polygons, image-objects, 2-D objects, image-parts, plane figures, patches, connected components, *etc.* [50–52]. Throughout this chapter, these terms will be used interchangeably. Hence, by definition, an image-object is a connected 2-D region, whether or not simply connected, provided with a unique identifier, e.g., an integer value. The prior availability of the image segmentation allows a series of attributes describing each segment to be computed. In particular the shape of the segments may be described, which is one of the main focuses of this chapter due to its importance to a variety of applications. We also note that we allow segments to contain holes, which oftentimes define important object properties [53].

The proposed set of per-segment attributes finds inspiration in existing works on object-based image analysis (OBIA) [54–56] and computer vision [57–60]. Unfortunately, during our experiments, several per-segment attribute implementations found in the literature scored low in one or more metrological/statistically-based quality indexes of operativeness to be community-agreed upon, such as effectiveness (e.g., accuracy), efficiency (e.g., computation time), robustness to changes in the input data set (encompassing robustness to noise), robustness to changes in input parameters, if any, and scale invariance. To overcome these operational drawbacks, our selected and implemented set of estimators is novel in terms of: (a) individual formulations and/or implementations and (b) overall combination of individuals. A major aim of the set is comprehensiveness, which is tantamount to saying application independence. Additionally, redundancy between attributes should be avoided, although a degree of correlation can be tolerated.

During experiments, we encountered limitations in certain attributes found in the literature. This motivated their improvement, leading to the following contributions.

- **Approximate rectangularity** (see Section 4.4.2). Shackelford and Davis [54] provide a fuzzy rule to compute a segment's approximate rectangularity. The rule acts upon a simplified polygonal representation of the segment to determine how rectangular it is. The authors obtained the polygonal representation by using as polygon vertices the endpoints from the region's skeleton. Important problems have been noted with polygons resulting from this approach. Thus, in the current work, an alternative polygonal approximation method is used, namely the classic Ramer-Douglas-Peucker (RDP) algorithm.

- **Elongatedness** (see Section 4.4.8). Nagao and Matsuyama [56] estimate a region's elongatedness as the ratio between the length of the longest path along its skeleton and the region's average width along that path. Prior to skeletonization, the authors opted to fill in the region's holes, if any. Instead, we propose a new measure of elongatedness that accounts for the complete skeleton (instead of just its longest path) and that does not require holes to be filled in beforehand. Additionally, Nagao and Matsuyama suggested using a thinning-based skeletonization procedure, whereas we compute the filtered Euclidean skeleton of the region [61], which is better-defined and faster to compute.

- **Simple-connectivity** (see Section 4.4.9). To quantify the presence of holes in a region, an original shape index, called simple-connectivity, is implemented. Previous works have assessed the presence of holes using measures like the absolute number of holes or the area of the holes relative to the area of the region. The new proposed definition of simple-connectivity takes into consideration the length of the boundary of the holes relative to the length of the complete boundary of the region. In order to be sensitive to both the presence of multiple holes (which may have an overall small area) and holes with large areas (which my have an overall limited boundary length), our final simple-connectivity measure is defined as a fuzzy AND (minimum) combination of the boundary-based simple-connectivity with a traditional area-based measure.

- **Straightness of boundary** (see Section 4.4.5). The straightness of a region's boundary is computed in agreement with Nagao and Matsuyama [56]. However, in order to deal with multiple resolutions and structures of different sizes, an adaptive scale selection procedure is incorporated.

- **Morphological multiscale characteristic** (see Sections 4.4.6 and 4.4.7). Pesaresi and Benediktsson proposed to segment images using morphological reconstruction granulometries [62]. In the process, the authors defined the *morphological multiscale characteristic* as a characteristic scale to be associated to each image pixel. The morphological multiscale characteristic provides an estimate of the size of the main image structure that each pixel belongs to. We have adopted the average value of the characteristic over each segment as an attribute in our set. A practical challenge with such an approach is that computing the morphological multiscale characteristic requires morphological operations that can be prohibitively slow for large images. In order to speed up this process, we make use of decomposable filters for dilation and erosion [63], as well as Robinson and Whelan's downhill filter for reconstruction [64].

Eligible for use in a wide range of computer vision problems, the proposed set of per-segment attribute estimators is tested in two domains. In the first test case, man-made objects, e.g., roads, buildings, impervious surfaces, etc. [54] must be detected in a satellite optical image of an urban area. This type of analysis requires an approach that scales well over the very large satellite image. To accomplish this task in a two-stage OBIA framework, the Satellite Image Automatic Mapper (SIAM) is selected as the first stage. SIAM provides a segmentation of the image, from which image-object attributes may then be computed. The SIAM software product is a system for prior knowledge-based multi-level discretization of a continuous color space [65–67]. By means of a two-pass connected-component multi-level image labeling algorithm [68], implemented in series with a prior knowledge-based decision tree, SIAM is capable of generating, automatically (without user's interactions) and in near real-time, multi-level preliminary classification (pre-classification) maps and multi-scale segmentation maps of a radiometrically calibrated spaceborne/airborne multi-spectral image.

The second test data set consists of leaf images, acquired with consumer-level digital cameras from Leafsnap users, to be identified in terms of tree species (see Chapter 2). Leafsnap provides mobile applications that allow users to photograph tree leaves and quickly receive a short list of matched tree species [1, 4, 6]. The interactive nature of the applications requires results to be returned in a matter of seconds. Shape is an extremely distinctive feature for leaf identification, having in fact been successfully used as a sole identification cue by Leafsnap's leaf image identification system [1]. Leaf identification is thus a very interesting testbed for the proposed set of feature extractors, whose main focus is on shape properties to be computed efficiently, so that, in the recognition (classification) phase, a prior knowledge-based decision rule set may be applied.

This chapter is organized as follows. Section 4.2 discusses related works on shape representation and recognition as well as the use of attributes in image retrieval. Section 4.3 summarizes the proposed set of per-segment attributes. These are described in more details in Section 4.4, whose special emphasis is on segment-specific shape and size attributes, considered as our main concern. Section 4.5 discusses experimental results collected in the two aforementioned OBIA test cases. Conclusions and directions for future work are reported in Section 4.6.

4.2 Related Works

Some previous works have focused on the development, assessment and comparison of intuitive shape descriptors [58, 60, 69]. However, these works dealt with relatively small sets of descriptors. It remains a challenge to generate basic segment-specific properties transferable to multiple application domains. As discussed in Section 4.1, during the development of the present study, operational limitations of several existing descriptors have been registered, leading to our improved formulations and/or implementations, proposed hereafter.

Ruiz et al. [70] developed a set of object-based features for analysis of remote sensing images of agricultural sites. Though their work includes basic shape features, their main focus is on describing texture and appearance, more relevant in agricultural sites. A traditional supervised learning classification approach is adopted in series with high-dimensional feature extraction. The present work, whose objective is the implementation of a comprehensive set of intuitive and complementary segment-specific attributes, focuses on a problem that is quite different.

According to Section 4.1, the image information primitive of interest is the *image-object*, defined as a connected set of pixels. In the context of geographic information systems, Goodchild *et al.* [51] discuss general representations for geospatial data. These authors categorize geospatial information into dimensionless *geo-atoms*, continuous *geo-fields* (eventually discretized into so-called *field-objects*) and discrete *geo-objects*. According to the terminology introduced by the Open Geospatial Consortium (OGC) [52], geo-objects can be either (0-D) points, multipoints, (1-D) polylines, multipart polylines, (2-D) polygons or multipart polygons. Hence, geo-objects may be disconnected (multipart) and may contain holes or enclaves. Image-objects, as defined here, constitute a subclass of discrete geo-objects, being equal to either a single point, a single polyline or a single polygon, detected on an *a priori* basis, i.e., detected before any estimation of segment-specific attributes begins.

According to the project requirements specification proposed in Section 4.1, selected per-segment attributes must be complementary and intuitive to deal with, to become a suitable input to a two-stage OBIA system where, at the second stage, a deductive (static, non-adaptive to input data) decision-tree classifier can be implemented by modeling prior knowledge of domain experts [54,55,71,72]. These project requirements are in line with those suggested by Peura and Iivarinen, who advocate, whenever possible, the use of basic semantically simple shape descriptors [57]. Our attributes are required to be complementary, in agreement with the authors' statement that, though some amount of correlation between descriptors is acceptable, it is desirable that combining descriptors should always introduce a new perspective. Zhang and Lu [59] duly observe that these semantically simple shape descriptors are not suitable to be used in isolation as standalone descriptors, but a combination of descriptors is necessary in order to accurately describe shapes. This strategy of combining descriptors, which is also adopted in the present work, copes with the well-known non-injective property of any summary statistic or quality index, which implies that no universal quality index can exist [73].

Some recent works have focused on the use of attributes for image search via keywords, as well as object recognition via textual descriptions [74–77]. The use of attributes provides more modularity if compared to pure data-based learning approaches: attributes can be shared across different categories, and new object categories can be learned from very few examples, or even solely from textual descriptions. The use of attributes also allows data from varying sources to be more easily combined [71, 72], by decoupling prior domain knowledge from specific features. From a design standpoint, we require our attribute set to be intuitive to deal with, so that it may be used within knowledge-based systems. This also allows for easily validating the correctness and suitability of attributes, as well as simplifying the process of setting control parameters. A potential limitation of attribute-based approaches is that they usually do not provide a complete representation, eventually missing object properties that are difficult to be described in words. This can be mitigated in practice by adopting hybrid inference systems (combined learning-by-rule and learning-from-examples, i.e., combined deductive and inductive) in which attributes are combined with more traditional sets of features, to be used with classifiers learned from training examples [66].

A large body of literature focuses on 2-D shape features suitable for the OBIA paradigm [59, 77–79]. Some works aim to perform recognition directly using high dimensional representations, which are oftentimes complete [59]. Among these high dimensional representations, some are composed of global coefficients such as Fourier or wavelet descriptors, or geometric, Legendre and Zernike moments. Other representations are more structural in nature, such as the medial axis transform, the shape's polygonal approximation, or the chain code that describes its contour. There has been particular interest in the signature obtained from the curvature of a shape's boundary, which naturally allows for multiscale analysis [1, 80]. Though these representations are not directly suitable for knowledge-based systems, they have the advantage of often being complete and making evident certain relevant shape characteristics. The representations are better suited to example-based frameworks in which an example prototype or a set of training examples is available. In contrast, our current work, based on per-segment attributes, allows classes of image-objects to be directly derived by partitioning the space of attributes based on specialists' prior knowledge.[1]

Another relevant segment of the computer vision literature defines distances between pairs of shapes via shape matching. The distances may then be used within a framework containing a model or a set of training examples. Most works within the matching paradigm find non-rigid deformations that match a pair of shapes to each other, based on dense point-to-point correspondences. Given a pair of shapes, a distance can be defined as a sum of two costs: one arising from the non-rigid deformation (the deformation term), and the other from the similarity of the point-to-point matches (the data term). In this direction, we point out, first, the work of Chui and Rangarajan [81], who develop an optimization scheme to simultaneously determine point correspondences and a thin plate spline spatial deformation mapping. Notably, later Belongie et al. [82, 83] developed a local descriptor called the shape context in order to help find better correspondences. The shape context descriptor is used to reliably compute local shape distances in order to accurately

[1]Nonetheless, in the process of computing our attributes, different types of high dimensional intermediate representations are used, such as the shape's medial axis transform, its polygonal approximation, convex hull, etc. For a description of how the shape attributes are computed, see Section 4.4.

measure local dissimilarities. A similar line of work is that by Sebastian et al. [84], who used graphs to describe the medial axis transforms of shapes. Shape deformations were then encoded as edits to the graphs, providing a matching cost based more on object structure. Finally, also somewhat related to the medial axis transform, Felzenszwalb represented shapes using particular triangulations of polygons [85]. The triangulations allowed for efficiently finding optimal matchings that minimized (exactly) a certain class of energy functions.

Alternative OBIA approaches avoid the segmentation problem and directly extract either localized shape descriptors or fragments of contours from images. Usually, edge intensities or a binary edge map must be computed to allow extraction of local shape descriptors. The shape context is an extremely popular example of a local shape descriptor, which allows distances between pairs of parts of shapes to be computed [82]. These distances have been used for shape matching [82,83] and matching of scenes [86], as well as for shape-based object recognition using various different strategies [87]. The original shape context formulation is modified somewhat in most of the recent works so that it also takes into account edge orientations (besides their positions), resulting in a more informative representation [83]. Other shape descriptors also based on histogram binning of edges have been explored by Mikolajczyk et al. [88], Carmichael and Hebert [89] and Bosch et al. [90].

A recent trend in computer vision attempts to use contour fragments instead of the more localized shape descriptors. Shotton et al. [91, 92] and Opelt et al. [93] extract contour fragments from training images and use them later as templates for chamfer matching. When trying to detect an object in a new image, for each training contour template, the chamfer matching process provides matching distances for image pixels. Actual detection of the object accumulates votes cast by each contour detector towards the expected centroid of the object being detected. A common problem with these approaches is that the learned contour fragments are collected from windows in the training images that will also contain contours from background objects. In order to obtain cleaner contour fragments – avoiding the contours from nearby clutter – Ferrari et al. [94] proposed using chains of connected, roughly straight, contour segments.

All these shape analysis approaches are in contrast to our two-stage OBIA paradigm where, at the first stage, an image segmentation map is generated, whatever the adopted image pre-classification and/or segmentation algorithm is (e.g., SIAM), and, at the second stage, a segment-based classification system, employing either learning-from-examples (inductive, bottom-up), learning-by-rule (deductive, top-down) or hybrid (combined deductive and inductive) inference mechanisms, is implemented.

4.3 Overview of Selected Attributes

The list of implemented segment-specific attributes is provided below, following the taxonomy of shape features proposed by Zhang and Lu [59]. According to this taxonomy, every shape attribute below is *global*, since they represent the shape as a whole (as opposed to *structural*, which represent a shape by sections, or primitives). The taxonomy further distinguishes between *contour-based* and *region-based* representations. The contour-based shape attributes in the proposed set are the following.

- Angle (orientation) of the segment's minimum enclosing rectangle (see [56,95]).
- Convexity (Section 4.4.1).
- Polygon-based approximate rectangularity (Sections 4.4.2 and 4.4.3).
- Roundness (Section 4.4.4).
- Straightness of boundary (Section 4.4.5).

Region-based shape attributes are listed below.

- Area (size).
- Average characteristic scale, computed as the morphological multiscale characteristic, equivalent to an edge-preserving local autocorrelation value estimation (Sections 4.4.6 and 4.4.7).
- Elongatedness (Section 4.4.8).
- Simple-connectivity (Section 4.4.9).

Finally, photometric attributes include the following.

- Average contrast along the boundary (Section 4.4.10).
- Statistics of achromatic (e.g., brightness) or chromatic (e.g., multispectral) values, e.g., mean, minimum, maximum and standard deviation of each available spectral channel over the pixels of each segment.

4.4 Description of Attributes

4.4.1 Convexity

A commonly used shape property in image analysis is convexity, also known as convexity ratio or solidity. It is defined as follows:

$$\text{ConvexityAndNoHole} = \frac{A}{A_{\text{convex}}}, \tag{4.1}$$

where the area value, A, is defined as the number of pixels that belong to the region, excluding those belonging to holes, if any, while A_{convex} is the area of the convex hull of the region, which by definition does not contain any inner hole. Since the area of the convex hull is never inferior to that of the original segment, the ConvexityAndNoHole variable always belongs to range $[0, 1]$. It takes on high values when the original segment is convex or close to being convex and, at the same time, it does not present holes. Hence, for the sake of clarity, this shape indicator is called ConvexityAndNoHole.

The first step in computing the area of the convex hull (A_{convex}) is to find a representation of the hull. To do so, we begin by tracing the boundary of the original region using the standard boundary tracing algorithm attributed to Moore [34, 96]. This results in a sequence of pixels describing a walk along the region's boundary. From this sequence, the subset of the boundary pixels that form the set of vertices of the region's convex hull is selected using the algorithm independently discovered by Melkman [97] and Tor and Middleditch [98], which runs in $O(n)$ time, with n equal to the original number of boundary points.

After finding the vertices of the convex hull, the area A_{convex} can be computed from the vertices using the surveyor's (or shoelace) formula [99]. In common practice, due to discretization, this approach can lead to undesirable values of the ConvexityAndNoHole variable. For example, a direct algebraic calculation of A_{convex} for a region consisting of a straight line would be 0, whereas A would simply be the number of pixels in the line. It seems complicated to try to adapt the surveyor's formula to be able to cope with all possible scenarios of this type in digital images. Rather, here A_{convex} is estimated with a discretization procedure, where a binary image is generated from the convex hull, such that pixels are set to 1 if they belong to the convex hull and 0 otherwise. A_{convex} is then estimated as the number of pixels whose values is equal to 1.

4.4.2 Polygonal Representation

Shackelford and Davis adopt a simplified polygonal representation of image-objects, to which a set of fuzzy logic rules is applied in order to reason about their shapes [54]. As an example rule, one might state that (fuzzy) rectangles always have (around) 4 (fuzzy) vertices, each with a (fuzzy) inner angle of about 90°. An intermediate polygonal representation can be used to extract a series of different properties of interest [79]. Our focus, however, is the implementation of a measure of an object's approximate rectangularity, described further in Section 4.4.3.

Shackelford and Davis obtain a polygonal representation of a region from the endpoints of the region's skeleton, computed via a morphological thinning skeletonization. In common practice, this approach turns out to be problematic when the endpoints do not give a good description of an object's boundary (see Figure 4.1). This seems to hold independently of the skeletonization algorithm used. To avoid this problem, a different strategy is implemented here. After tracing an object's boundary, the boundary representation is treated as a polygon, which is then simplified using the Ramer-Douglas-Peucker (RDP) algorithm [100][2]. In addition to giving a better approximation of the region's boundary, this algorithm is also faster than skeletonization.

At each step of the RDP algorithm, two points are considered. The points determine a line segment, which can be thought of as a rough representation of some part of the polygon. The point in that part of the original polygon which is the farthest from this line segment is then found. If the distance from this point to the line segment is smaller than a tolerance ϵ, then the line segment is accepted as is. Otherwise, the problematic point is added to the representation, generating two new line segments which are recursively analyzed using the same procedure. We set the approximation tolerance ϵ to the scale corresponding to the maximum value of Straightness_s from Section 4.4.5. In other words, $\epsilon = \text{argmax}_s \text{Straightness}_s$. Figure 4.1 shows toy regions for which the skeleton-based polygon does not provide a good representation, whereas simplification with the RDP algorithm does. For

[2]We have used the implementation of the Ramer-Douglas-Peucker algorithm from the OpenCV library [38]

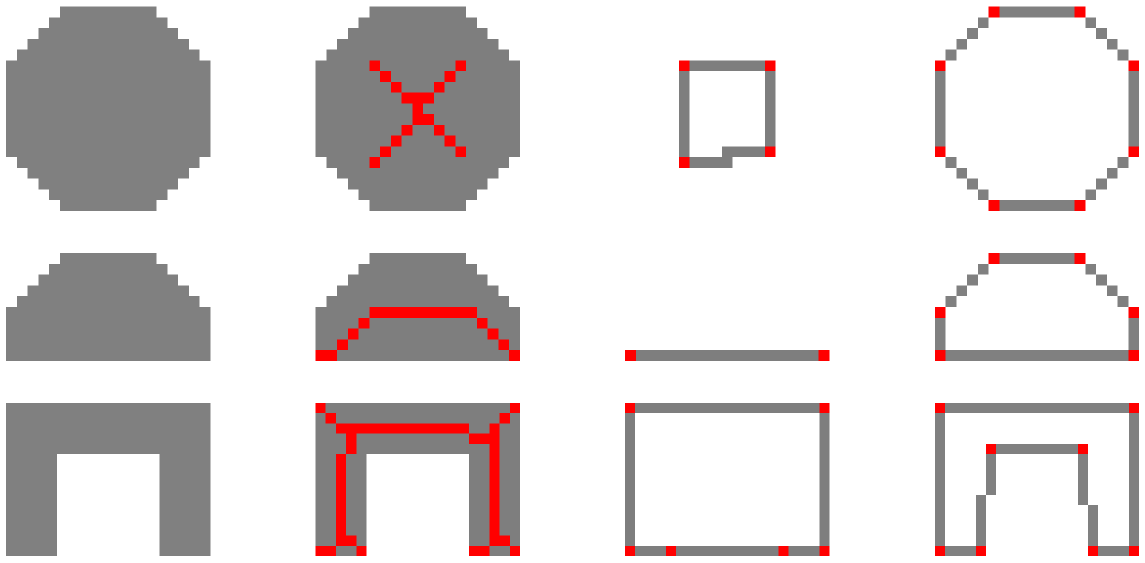


Figure 4.1: Regions for which the skeleton-based polygons fail to provide good approximations. From left to right, the columns are: original region; skeleton obtained using thinning; polygonal approximation from skeleton endpoints; polygonal approximation using the RDP algorithm. The skeleton-based approximation shows problems in the first and second rows due to the rounded region boundaries not creating skeleton endpoints, while a similar situation appears in the third row for the case of concavities.

this figure, the morphological thinning skeletonization method described in [54] was used.

4.4.3 Polygon-Based Approximate Rectangularity

Shackelford and Davis define a measure of “approximate rectangularity” using a set of fuzzy rules [54]. These rules are applied to a polygonal representation obtained from a region’s skeleton. (The method used to obtain the polygonal representation was reviewed in Section 4.4.2). Here, this measure of approximate rectangularity is subject to two important changes. First, the polygon to which the rule set is applied is not obtained by skeletonization, but with the Ramer-Douglas-Peucker algorithm, described in Section 4.4.2. Second, in order to deal with noisy segments several of the rules' parameters are changed. Though the same set of rules is adopted, by relaxing the rule parameters, the rule set becomes much less strict.

Other measures of rectangularity have been used in the literature. A standard rectangularity measure is the area of the region relative to that of its minimum enclosing rectangle [56]. Rosin [58] further improved on this measure by developing robust rectangle fitting procedures. Additionally, he presented a measure of rectangularity defined using the difference in moments between the region and its best fitting rectangle. However, in the current work, the polygon-based approximate rectangularity measure of Shackelford and Davis is used due to its capability of accurately expressing the class of rectangular shapes. By using fuzzy rules to specify the expected number of vertices of a rectangle, their angles, and relative distances, the measure becomes robust to a series of common shape variations, whether or not due to image noise.

4.4.4 Roundness

A very popular measure of the compactness, or complexity, of a region is A/P^2, with A denoting the region's area and P its perimeter. The measure takes its maximum value of $1/4\pi$ when the region in question is a circle. This consideration motivates the definition of a measure of roundness, or circularity, as $4\pi A/P^2$, which will always lie in the $[0, 1]$ interval. Other authors also directly define a measure of noncompactness as the inverse of compactness, i.e. P^2/A [56, 101].

The segment's area, A, can be estimated simply as the number of pixels in the region, excluding pixels that belong to holes, if any. A question arises, though: how to compute P, since different definitions of perimeter exist when dealing with images? [101] Our implementation estimates the perimeter P as the 4-adjacency cross-aura measure of the region's total boundary. The precise definition of this cross-aura measure is presented in Section 4.4.9. In brief, it is computed by visiting each of the region's boundary pixels and counting its number of 4-neighbors that do not belong to the region. The counts are added up over all boundary pixels, resulting in the perimeter measure.

With these definitions of area A and perimeter P, our measure of roundness becomes

$$\text{RoundnessAndNoHole} = \frac{4\sqrt{A}}{P}. \tag{4.2}$$

The normalization by 4 above guarantees that RoundnessAndNoHole is always between 0 and 1. It can be easily proved that the proposed implementation of the RoundnessAndNoHole equation is approximately scale invariant. Noteworthy, this measure treats holes as intrinsic properties of the object. This is reflected in the way both A and P are computed. The area measure A does not count pixels that belong to holes, while the perimeter measure P also takes into account the boundary that the region forms with its holes. This measure scores high for regions that are round and, at the same time, do not have holes. This behavior justifies its name, RoundnessAndNoHole.

Although measures of compactness or roundness should be maximum for circles, Rosenfeld pointed out that, in images, depending on how the perimeter is measured, compactness measures could turn out to be larger for squares or octagons than for digitized circles [101]. Our measure is no exception and takes maximum values for squares, which have maximum area for a given fixed value of the 4-adjacency cross-aura measure.

4.4.5 Straightness of Boundary

A measure of the straightness of a region's boundary is especially discriminative for the analysis of remote sensing images of urban areas or agricultural land. In general, man-made structures tend to present straight boundaries, independently of whether their overall shape is simple or more complex.

Nagao and Matsuyama proposed the following procedure to compute the straightness of a region's boundary [56]. The boundary of the region is first traced. In this case, the standard boundary tracing method attributed to Moore [34, 96] can be adopted. This results in a sequence of pixels describing a closed walk along the region's boundary, which we denote p_i ($i = 1, \ldots, n$), where $p_1 = p_n$ (since the boundary is closed), so that the total number of boundary pixels is $n - 1$. For each pixel p_i on the boundary, the angle Δ_i between the two lines connecting p_i with p_{i-s} and p_{i+s} is calculated, where $i + s$ and $i - s$ are modulo n. Variable s is referred to as the step size. A pixel i is counted as "straight" if $|\Delta_i| \leq \alpha$ for some angle threshold α, which is given as a parameter to the method. Let $\hat{n}_s$ denote the number of straight pixels in the boundary, measured using a step size s. Then, the straightness of boundary for step size s is $\text{Straightness}_s = \hat{n}_s/(n - 1)$. Found to work well in practice across a range of s values, an angle threshold $\alpha = 30°$ is chosen. Noteworthy, the straightness measure always lies in the $[0, 1]$ interval, being maximum when all boundary pixels are considered to be straight.

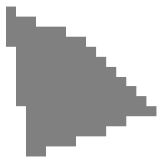
In practice, the step size s acts as an observation scale. To deal with images of different resolutions, as well as structures of different sizes, it is important to choose an appropriate value of the step size s. Whereas Nagao and Matsuyama worked with a single step size, the following heuristic criterion is proposed to infer a step size adaptively. First, variable Straightness_s is computed for all values of $s \in \{1, 2, 4, 8, 16, 32\}$. Next, the final straightness measure is taken as the one with the maximum value:

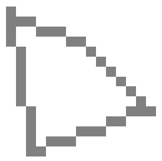
$$\text{Straightness} = \max_s \text{Straightness}_s. \tag{4.3}$$

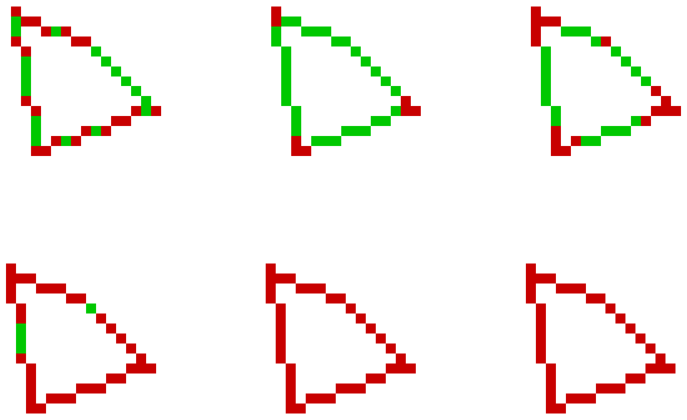
This adaptive learning process is illustrated in Figure 4.2.

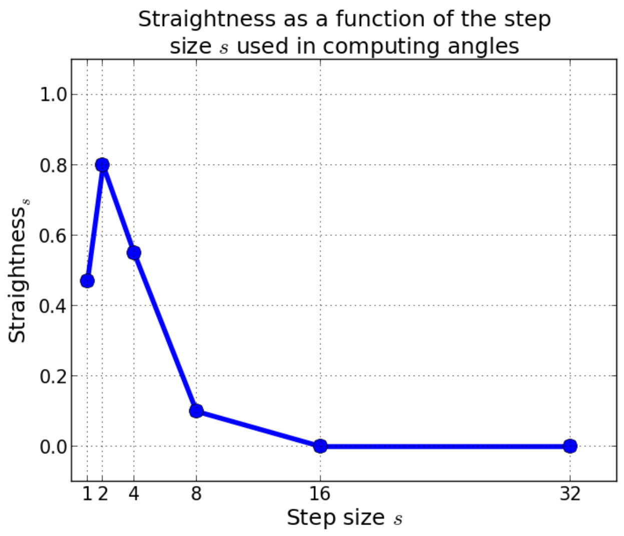
4.4.6 Morphological Multiscale Characteristic

Pesaresi and Benediktsson compute a per-pixel differential morphological profile (DMP) as an intermediate step to accomplish image segmentation [62]. The DMP is used in order to compute a per-pixel morphological multiscale characteristic, defined as the morphological scale where each pixel's DMP scores its maximum. In our view, the morphological multiscale characteristic is a viable alternative to existing estimators of local autocorrelation values such as the local indicators of spatial association (LISA), i.e., it allows an effective and efficient per-pixel estimate of the


(a) Original region.

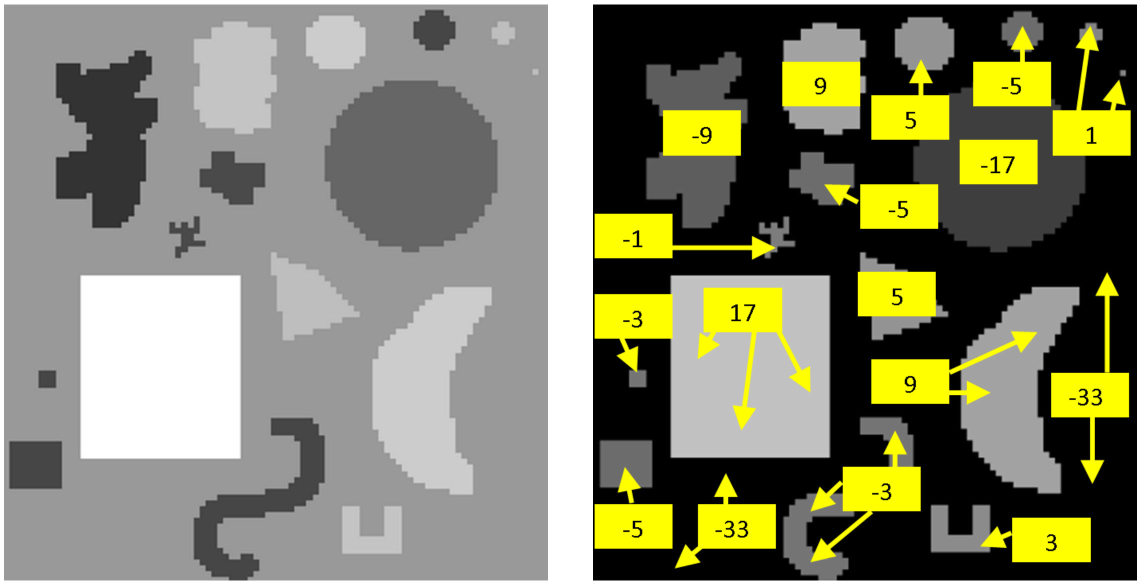

(b) Region boundary.


(c) Straight and non-straight boundary pixels as a function of step size s. Pixels in green were considered straight and those in red were considered non-straight. From left to right, top to bottom, step sizes are $1, 2, 4, 8, 16, 32$. Respective straightness values are show in (c).



(d) Straightness of boundary for the region in (a) as a function of step size s.

Figure 4.2: An example region illustrating the process of computing the straightness of boundary measure over multiple step sizes.



(a) (b)

Figure 4.3: This figure presents a synthetic example to illustrate the behavior of the morphological multiscale characteristic. In (a), a synthetic grayscale image, featuring several shapes of different sizes, is shown. In (b), values of the morphological multiscale characteristic at each pixel are shown in the yellow highlights, where a negative value indicates that the closing DMP contains the largest DMP response (segment was originally darker than background) and, as a consequence, it provides the characteristic scale, while a positive value indicates that the opening DMP contains the largest DMP response (segment was lighter than background), so that it provides the characteristic scale (see Equation 4.12). Note how the values indicate roughly the size of the image-objects and also that, within each object, the values are constant. For this image, the sampled dyadic scales were $\lambda_i = 0, 1, 3, 5, 9, 17, 33$ (see text), resulting in the displayed scale values.

(1-D) size (in pixel units) of the image-object each pixel belongs to, while at the same time respecting segment boundaries, in line with statements by Pesaresi and Benediktsson in their original paper [62]. An illustrative example of the behavior of the morphological multiscale characteristic is presented in Figure 4.3. For each image-segment, the average of the characteristic scale over its pixels is adopted as an attribute describing the segment's (1-D) size.

In this Section, the per-pixel DMP is defined and discussed before moving on to the definition of the morphological multiscale characteristic. Next, an efficient procedure to quickly compute morphological openings and closings by reconstruction, which are required for generating the DMP, is proposed. Finally, in Section 4.4.7, popular global and local spatial autocorrelation estimators are reviewed and related to the multiscale morphological characteristic.

The definition of a per-pixel DMP requires definitions of opening and closing by reconstruction of a grayscale image I. An *opening by reconstruction* is defined as

$$\gamma_\lambda^* I = \mathrm{Rec}(\varepsilon_\lambda I, I), \tag{4.4}$$

where $\varepsilon_\lambda I$ denotes an erosion of I with a structuring element (SE) of size λ and $\mathrm{Rec}(\varepsilon_\lambda I, I)$ denotes the reconstruction by dilation of I from $\varepsilon_\lambda I$. For a detailed formal definition of the erosion and reconstruction operations, the reader is referred to [62,102]. The traditional opening of an image by a SE of size λ is used to filter out bright structures that are smaller than λ. The opening by reconstruction operator also filters out these small bright structures, but without affecting the fine-scale details of larger structures, since these are recovered in the reconstruction step. On the other hand, a potential disadvantage of opening by reconstruction is a high computational cost. This will be discussed in more detail further in this section. Analogously to above, a *closing by reconstruction* is defined as

$$\phi_\lambda^* I = \overline{\mathrm{Rec}}(\delta_\lambda I, I), \tag{4.5}$$

where $\delta_\lambda I$ denotes a dilation of I with an SE of size λ and $\overline{\text{Rec}}(\delta_\lambda I, I)$ denotes the reconstruction by erosion of I from $\delta_\lambda I$. Analogously to the opening by reconstruction, a closing by reconstruction filters out dark structures that are smaller than λ.

Starting from these definitions, the *opening profile* of an image is composed of a series of openings by reconstruction. We use a a dyadic sequence of structuring element (SE) sizes λ_i for $i \in [0, n]$ such that $\lambda_0 = 0$, $\lambda_1 = 1$, and $\lambda_i = 2^{i-1} + 1$ for $i \in [2, n]$. As an example of the sequence of sizes, if $n = 4$ the resulting sequence is $\lambda_i = 0, 1, 3, 5, 9$. Given this sequence, the *opening profile* at pixel x is defined as the vector

$$\Pi\gamma(x) = \{\Pi\gamma_{\lambda_i} : \Pi\gamma_{\lambda_i} = \gamma^*_{\lambda_i}(x), \forall i \in [0, n]\}. \tag{4.6}$$

Correspondingly, the *closing profile* at x is

$$\Pi\phi(x) = \{\Pi\phi_{\lambda_i} : \Pi\phi_{\lambda_i} = \phi^*_{\lambda_i}(x), \forall i \in [0, n]\}. \tag{4.7}$$

Differently from Pesaresi and Benediktsson [62], we have chosen to take dyadic SE sizes λ_i, considered to be more practical (and biologically more plausible), so that the resulting profile may be computed in a reasonable amount of time while being able to handle image structures of varying sizes. Other than that difference, our definitions throughout are the same as those given by the previous authors.

The per-pixel DMP records the rate of change in the opening and closing profiles. For each pixel, its DMP provides an estimate of the importance of structures of size λ to which the pixel might belong. The *derivative of the opening profile* $\Delta\gamma(x)$

and the *derivative of the closing profile* $\Delta\phi(x)$ are defined respectively as

$$\Delta\gamma(x) = \left\{ \Delta\gamma_{\lambda_i} : \Delta\gamma_{\lambda_i} = \frac{|\Pi\gamma_{\lambda_i} - \Pi\gamma_{\lambda_{i-1}}|}{(\lambda_i - \lambda_{i-1})}, \forall i \in [1, n] \right\}, \tag{4.8}$$

$$\Delta\phi(x) = \left\{ \Delta\phi_{\lambda_i} : \Delta\phi_{\lambda_i} = \frac{|\Pi\phi_{\lambda_i} - \Pi\phi_{\lambda_{i-1}}|}{(\lambda_i - \lambda_{i-1})}, \forall i \in [1, n] \right\}. \tag{4.9}$$

The complete DMP is obtained by concatenating the two DMP components above.

Pesaresi and Benediktsson [62] used the DMP to develop two versions of a segmentation method. The first operated on a single scale, requiring that the scale of the structures of interest be known beforehand. The second segmentation method was multiscale, capable of simultaneously dealing with structures of different sizes. The multiscale segmentation algorithm made use of the *morphological multiscale characteristic*, whose definition we recall below. A DMP-driven image segmentation approach is also adopted by Shackelford and Davis for the detection of buildings in spaceborne imagery [55, 103]. Benediktsson et al. [104] later used the DMP to compose feature vectors for pixel-based supervised classification.

In order to define the *morphological multiscale characteristic* [62], the *multiscale-opening characteristic* and *multiscale-closing characteristic* must be first defined. The *multiscale-opening characteristic* $\Phi\gamma(x)$ of an image at pixel x is the SE size at which the opening DMP takes on the largest value,

$$\Phi\gamma(x) = \{\lambda : \Delta\gamma_\lambda(x) = \vee\Delta\gamma(x)\}, \tag{4.10}$$

where $\vee$ denotes the supremum. Analogously, the *multiscale-closing characteristic* is the SE size for which the closing DMP has its maximum value,

$$\Phi\phi(x) = \{\lambda : \Delta\phi_\lambda(x) = \vee\Delta\phi(x)\}. \tag{4.11}$$

The *morphological multiscale characteristic* $\Phi(x)$ is taken to be the scale at which the DMP is maximum, whether taken from the opening or closing profile. It can be defined as

$$\Phi(x) = \begin{cases} \Phi\gamma(x) : \vee\Delta\gamma(x) > \vee\Delta\phi(x) \\ \Phi\phi(x) : \vee\Delta\gamma(x) < \vee\Delta\phi(x) \\ 0 : \vee\Delta\gamma(x) = \vee\Delta\phi(x) \end{cases} . \tag{4.12}$$

In the case of ties – for which the largest DMP response is the same in the derivative of the opening and closing profiles – the definition sets $\Phi(x) = 0$. An illustrative example of applying this definition to all pixels of an image was presented in Figure 4.3.

A practical challenge in using the pixel-based morphological multiscale characteristic value is that computing openings and closings by reconstruction for large images can be prohibitively slow. In order to speed up this process, two techniques are selected from the existing literature: decomposable filters for dilation and erosion [63], and the downhill filter for reconstruction [64]. The way in which these techniques are used is briefly discussed below. Hereafter, the sole case of opening by reconstruction is discussed, because its dual problem is closing by reconstruction for which the same observations hold.

The initial erosion operation $\varepsilon_\lambda I$ from Equation 4.4, that occurs before performing the reconstruction $\text{Rec}(\varepsilon_\lambda I, I)$, can be very fast depending on the structuring element that is used. Instead of using a disk, which seems like the natural choice, it is much faster to approximate the result that would be obtained from a disk by using a regular polygon [105]. In particular, erosion with a square structuring element is extremely fast, since it can be decomposed into horizontal and vertical operations [63]. The purpose of the initial erosion operation is to eliminate bright structures that are smaller than the structuring element size. The reconstruction step that follows will guarantee that finer scale details of the remaining structures are recovered. Thus, the final results obtained from opening by reconstruction are found to be similar whether using a square or a disk for the initial erosion operation. Hence, the use of a square structuring element is preferred here due to its lower computation time.

Due to time considerations, special attention must be given to the reconstruction by dilation operation $\text{Rec}(\varepsilon_\lambda I, I)$ from Equation 4.4. Vincent defined the notion of morphological reconstruction for grayscale images [106], noting that the naive implementation using the mathematical definition of reconstruction showed to be extremely slow. Thus, three successively faster algorithms were proposed, speeding up execution times considerably [102, 106]. Later, Robinson and Whelan presented the downhill filter for image reconstruction [64], with a precondition to be satisfied by input images in order to guarantee correctness. In the weak form, which is used here, the precondition requires that, in the case of reconstruction by dilation, the marker image be everywhere less than or equal to the mask image. Fortunately, in the reconstruction $\text{Rec}(\varepsilon_\lambda I, I)$, this will always be true, since $\varepsilon_\lambda I \leq I$. (An analogous, dual, argument applies to the case of reconstruction by erosion $\overline{\text{Rec}}(\delta_\lambda I, I)$, presented in Equation 4.5.) The downhill filter computes the reconstruction in a single pass, guaranteeing a fast and consistent execution time. In timed comparisons to the algorithms proposed by Vincent [102] over a range of input images, the downhill filter showed a consistent and oftentimes large speedup [64]. We thus adopt the downhill filter to compute the required reconstructions.

4.4.7 Conceptual Links Between the Morphological Multiscale Characteristic and Measurements of Spatial Autocorrelation

Hereafter, conceptual links between the pixel-based characteristic scale and image-wide spatial autocorrelation functions are investigated. Autocorrelation is the cross-correlation of a signal with itself, ranging from -1 (high negative autocorrelation) across 0 (no autocorrelation) to $+1$ (high positive autocorrelation). Informally, it is the similarity between pairs of observations as a function of the time or spatial lag between them. It is even-symmetric and maximum at lag equals zero. Autocorrelation never goes to zero for a periodic function. It is a mathematical tool for finding repeating patterns, such as the presence of a periodic signal obscured by noise. Beyond a lag value called autocorrelation value, the autocorrelation function reduces to (approximately) zero, which means that pairs of observations beyond that lag are independent.

Whereas temporal autocorrelation is about proximity in time, spatial autocorrelation is about proximity in (multidimensional) space, i.e., spatial autocorrelation is characterized by a correlation in a signal among nearby locations in a 2- or 3-D spatial domain. Hence, spatial autocorrelation can be more complex than temporal (one-dimensional) autocorrelation because spatial correlation is either 2- or 3-dimensional.

In statistics, the well-known Moran's I spatial autocorrelation index is a measure of global spatial autocorrelation. It ranges from -1 (indicating perfect dispersion) to $+1$ (perfect correlation). A zero value indicates a random spatial pattern. Moran's I is inversely related to another well-known global spatial autocorrelation index, Geary's C, but it is not identical. Moran's I is a measure of global spatial autocorrelation, while Geary's C is also a global statistic, but it is more sensitive to local spatial autocorrelation. The value of Geary's C lies between 0 and 2. Value 1 means no spatial autocorrelation. Values lower than 1 demonstrate increasing positive spatial autocorrelation, whilst values higher than 1 illustrate increasing negative spatial autocorrelation.

Alternative to Moran's I and Geary's C, the semivariogram of an image is another popular strategy for spatial autocorrelation estimation [70, 107–110]. An experimental semivariogram γ characterizes an image I as a function of the separation distance (or lag) h. It can be computed as

$$\gamma(h) = \frac{1}{2N(h)} \sum_{i=1}^{N(h)} \left[I(x_i) - I(x_i + h) \right]^2 , \tag{4.13}$$

where x_i are 2-D image locations, h provides the separation between pixels, and $N(h)$

is the number of data pairs considered at separation h. From the equation we note that the semivariogram is measuring the overall dissimilarity of image structures that are separated by a distance h. The semivariogram can be used for texture description and characterization of land use, detection of periodicity and anisotropy and estimation of the density and size of objects [107, 110]

A common technique when dealing with images is to compute experimental semivariograms along several different directions and average them out to obtain the omnidirectional semivariogram [107] as a one-dimensional summary signature. The analysis of the omnidirectional semivariogram can then provide us with information about the underlying image structures.

Woodcock et al. [109, 110] investigated the behavior of the semivariogram in relation to the sizes of objects in images. They noted that distinct changes in the variogram happen most often at distances corresponding to object sizes. In particular, for an image containing objects of roughly the same size, the *range of influence* (or simply *range*) of the semivariogram indicates their size. The range of influence of a semivariogram is the distance at which it approaches its highest value. From this distance onwards, the correlation between image pixels is small, indicating that they are statistically independent. In practice, depending on image content, some variograms might reach this highest value very slowly, though it may still be possible to detect breaks in the slopes of the variograms that indicate the underlying size of objects [110].

The space discretization unit or unit of geospatial information we are looking for is the so-called (discrete and, at least on theory, symbolic) geo-object, according to the nomenclature of Goodchild *et al.* [51]. For example, the so-called local indicator of spatial association (LISA) is a local Moran's I_i value computed for each i-th spatial unit (image-object, e.g., an image tile) provided beforehand [111].

To cope with geo-objects as units of geo-spatial information, in the OBIA literature, where an image segmentation algorithm adopts a so-called spatial scale as input parameter to be user-defined based on heuristics, Kim *et al.* estimate the "best" scale for optimally derived segments in an image by means of a trial-and-error procedure where they estimate: (i) an image-wide average per-object variance graphed against segmentation scale and (ii) an image-wide average inter-object Moran's I spatial autocorrelation, graphed against segmentation scale. The former was found to increase with the magnitude of the segmentation scale, leveling off at the optimal spatial scale. The latter was lowest, and indeed negative, at the optimal scale. The two criteria provided nearly the same optimal characteristic scale value [112].

In this work, the morphological multiscale characteristic is used as a viable alternative to existing estimators of local autocorrelation values, such as LISA. It allows an effective and efficient per-pixel estimate of the size of the image-object each pixel belongs to, while having the advantage of being boundary-preserving, i.e., it provides reliable estimates for every pixel, irrespective of whether or not it is located in the vicinity of the segment boundary.

4.4.8 Elongatedness

A new measure of elongatedness is formulated in the current work to overcome operational limitations of previous measures. Nagao and Matsuyama present three different ways of measuring a region's elongatedness, each at an increasing level of sophistication [56], as summarized below.

1. The simplest elongatedness measure is defined as the ratio between the length and width of the minimum enclosing rectangle of the region. Though this measure is easy to compute, it does not correctly capture the elongatedness of curved regions. For example, a highly elongated curved region that has a square as its minimum enclosing rectangle would produce an elongatedness value of 1 according to this definition.

2. Another elongatedness measure, which works better for curved regions, is defined using the region's maximum width. First, eventual holes that the region might have are filled in. The region is then successively shrunk by a pixel at a time until it disappears. If we denote by d the number of shrinking steps required to eliminate the region, and A its original area, then the measure of elongatedness is defined as $A/(2d)^2$. Here, $2d$ corresponds approximately to the maximum width of the region. This elongatedness measure has also been used by others, such as Rosenfeld and Kak [113].

3. Instead of using the maximum width of the region, a more precise alternative is to use an average width. Again, the holes of the region are filled in, after

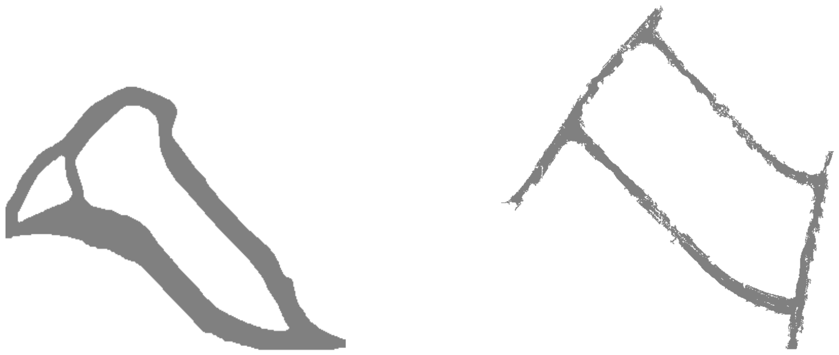


Figure 4.4: The river segment (shown on left) and road segment (shown on right) contain large holes. If their holes were to be filled in, they would present low or intermediate elongatedness measures, whereas expert photointepreters would expect these segments to score very high in elongatedness.

which the region's skeleton is computed via thinning. The longest path along the skeleton is then found. The length of the longest path will be used to define the elongatedness. For each point along the path, the local width of the region is computed, and the average of these is taken. If we denote the length of the longest path by L_{NM}, and the average width along the path as W_{NM}, then the final elongatedness measure defined by Nagao and Matsuyama is

$$\mathrm{Elongatedness}_{\mathrm{NM}} = L_{\mathrm{NM}}/W_{\mathrm{NM}}. \tag{4.14}$$

A practical limitation of the $\mathrm{Elongatedness}_{\mathrm{NM}}$ operator is that filling in the holes of a region before computing its elongatedness may not always be appropriate. Figure 4.4 shows examples of segments of roads and rivers, extracted from a satellite image, that a human domain expert would probably think of as highly elongated, but which would not produce high $\mathrm{Elongatedness}_{\mathrm{NM}}$ values since within-segment holes would get filled in during the procedure.

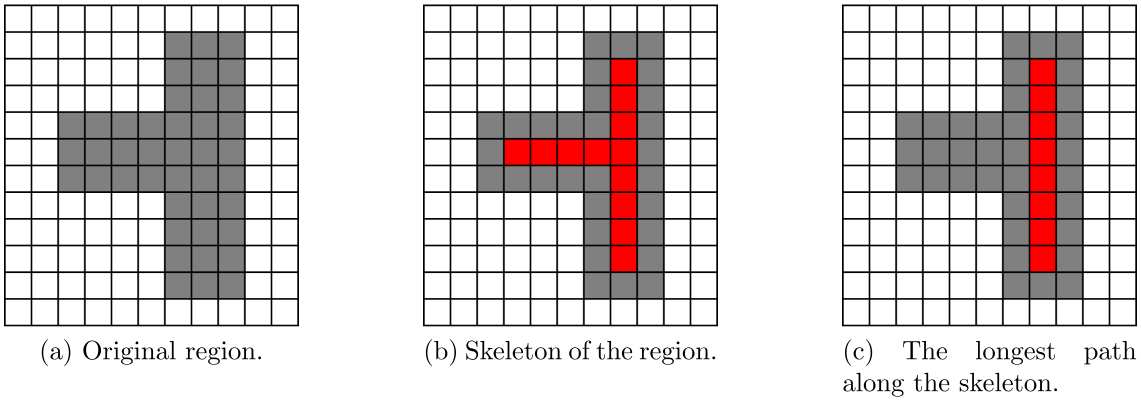
The aforementioned limitation justifies the introduction of an improved measure of elongatedness, featuring several differences from Nagao and Matsuyama's. Within-segment holes, if any, are *not* filled in. After computing the original region's skeleton, our novel measure of elongatedness is defined as

$$\text{Elongatedness} = L/W, \tag{4.15}$$

where L is a measure of the length of the complete skeleton (without removing holes), being defined simply as the total number of skeleton pixels, and W is the average width computed over the skeleton pixels. Below, more details are provided on the meaning and computation of this novel elongatedness measure, including how the average width measure W is calculated.

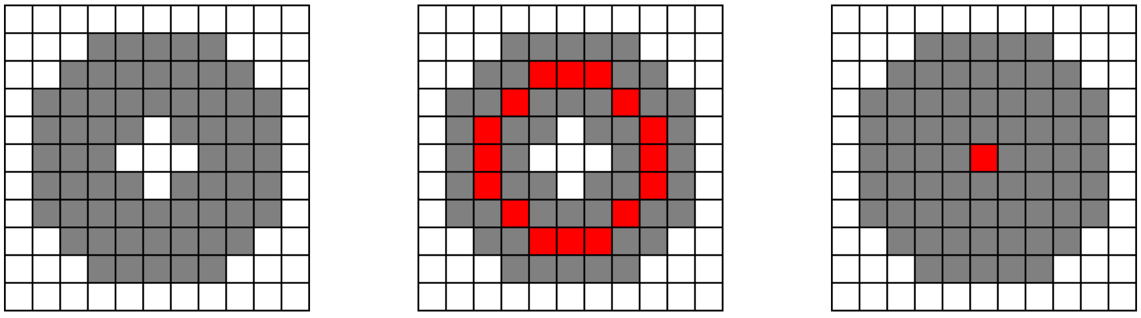
The novel measure of Elongatedness captures a different notion of elongatedness than that proposed by Nagao and Matsuyama. First, our length measurement L is taken to be the number of pixels in the whole skeleton, as opposed to just its longest path. In this manner, it always takes into consideration every part of a region. This difference is illustrated in the example in Figure 4.5. Second, by not filling in eventual holes, the novel measure of Elongatedness treats holes as intrinsic characteristics of image-objects, as opposed to ignoring them. Figure 4.6 shows how different elongatedness measures behave on a toy region with a hole.

Another important difference of the proposed Elongatedness variable in comparison with Nagao and Matsuyama's is in the algorithm used to compute the region's skeleton. Rather than a pure thinning procedure, as suggested by Na-



(a) Original region. (b) Skeleton of the region. (c) The longest path along the skeleton.

Figure 4.5: The region illustrates the difference in elongatedness measures due to using the whole skeleton as opposed to using just its longest path. Though different methods compute local widths differently, for the purposes of illustration, let us assume that the width computed at every point on the skeleton is of 4 pixels, so that the resulting average widths are $W = W_{\mathrm{NM}} = 4$ pixels (see Equations 4.14 and 4.15). When we define the elongatedness using either the whole skeleton or just its longest path, we obtain different results. Here, we have Elongatedness $= L/W = 12/4 = 3$, whereas Elongatedness$_{\mathrm{NM}} = L_{\mathrm{NM}}/W_{\mathrm{NM}} = 8/4 = 2$.



(a) Original region. (b) Skeleton of the region. (c) Skeleton of the region without holes.

Figure 4.6: The region illustrates the difference in elongatedness measures due to not filling in holes. When the hole is not filled in, at the same time that the length of the skeleton is larger, the average width will be smaller. In this example, we have Elongatedness $= L/W = 16.0/4.1 = 3.9$, whereas Elongatedness$_{\mathrm{NM}} = L_{\mathrm{NM}}/W_{\mathrm{NM}} = 1/10 = 0.1$.

gao and Matsuyama, the region's filtered Euclidean skeleton is computed. Many thinning-based skeletonization methods have been proposed and analyzed over the years [114]. However, more recently, developments have been made in techniques for fast computation of Euclidean skeletons. These new techniques ensure a correct topology and allow for filtering out eventual noise [61]. An important advantage of using Euclidean skeletons is that they are better defined: in principle, an Euclidean skeleton can be defined simply as the set of points that are centered in the shape with respect to the Euclidean distance[3]. On the other hand, direct thinning skeletonization procedures, though formulated with certain desirable properties in mind, present results that are not always predictable. Further, in many important practical cases, Euclidean skeletonization algorithms can compute faster than region thinning algorithms.

Couprie et al.'s skeletonization procedure begins by computing the squared Euclidean distance transform of the region, for which there are linear-time algorithms available. The distance transform will also allow the skeleton to be filtered later on: potential skeleton pixels that are too close to the region's border can be discarded to obtain a less detailed or less noisy skeleton. A constrained distance-guided thinning procedure is then applied to the region to obtain a skeleton that contains the medial axis transform of the region as a subset. The thinning procedure has two very important properties. First, it guarantees a homotopic skeleton, which is not trivial [115]. Second, it is such that even skeleton points that do not

[3]In spite of the simplicity of its mathematical definition, in practice, computing Euclidean skeletons is quite complicated due to the discrete nature of images.

belong to the medial axis transform are centered with respect to the region. Finally, the approach also allows filtering the skeleton with respect to its bisector angles. The bisector angle of a given skeleton point is the maximal angle that it forms with its two closest points that belong to the region's border. Skeleton points with small bisector angles may oftentimes be attributed to discretization noise and can be eliminated by setting an appropriate threshold. Couprie et al.'s overall approach is carefully designed to be very fast.

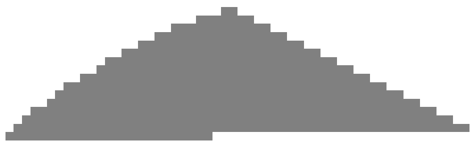
The skeletonization method possesses two parameters that control how the skeleton is filtered. The first is the threshold on the medial axis transform that removes skeleton points that are too close to the region border. With the purpose of obtaining a robust elongatedness measure, we set this parameter to be $d/2$, where d is the maximum value found in the distance transform, roughly corresponding to half of the region's maximum width. We found that in practice this preserves relevant structures in the skeleton, while discarding much of the noise. The second parameter is the threshold on the bisector angles. Skeleton pixels with bisector angles smaller than the threshold are rejected as being due to noise or irrelevant. We set this parameter to $\pi/5$, as this has worked well in practice on our image segments.

A filtered Euclidean skeleton is obtained via Couprie et al.'s method [61]. A problem with the resulting skeleton is it may be 4-adjacency connected, when it would be preferable for our purposes that all skeletons be 8-adjacency connected. This becomes important in order to obtain more consistent measurements when computing the length of the skeleton as its number of pixels. We apply a cycle of Cychosz's [116] fast implementation of Rosenfeld's parallel thinning algorithm [113,117] in order to obtain the final 8-adjacency connected skeleton. As shown in proposition 6 of [117], this guarantees the final result will be 8-adjacency connected. Figure 4.7 illustrates step by step the complete skeletonization procedure.

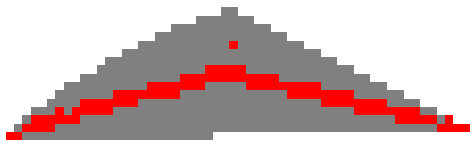
Finally, in order to compute the average width W in Equation 4.15, the width of each pixel along the skeleton must be estimated. Fortunately, the Euclidean distance transform of the region is found as an intermediate product of computing the Euclidean skeleton. To find W, the required distances are looked up in the stored distance transform, incurring little additional computation.

4.4.9 Simple-Connectivity

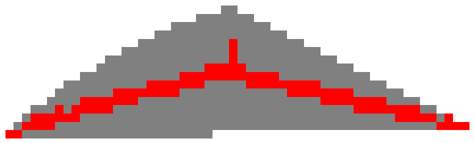
Within-segment holes can be very important image features suitable for identification of different classes of real-world objects depicted in images. Bertamini [53] reviewed a series of works that strongly indicate that the boundary of a hole is perceived by people as belonging to the object that encloses it. While some lines of work on shape analysis tend to ignore the presence of holes, the work by Bertamini showed that, from a perceptual standpoint, it is important to consider holes as constituent features of their enclosing objects. Besides being important image-object features, holes affect several of the presented attribute measures. In particular, holes increase a segment's measure of elongatedness and decrease the segment's measure of convexity and roundness. Quantifying the presence of holes in a segment allows for better reasoning about these effects during image analysis. In this section, an



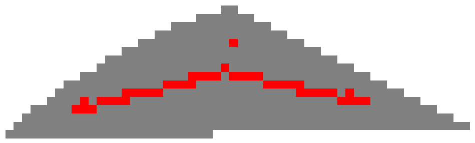
(a) Original region.



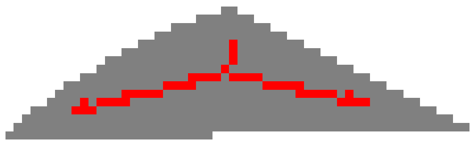
(b) Medial axis transform.



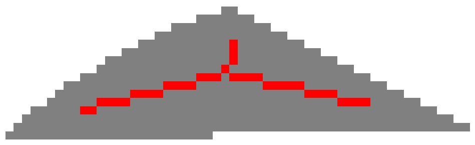
(c) Euclidean skeleton.



(d) Filtered medial axis transform.



(e) Filtered Euclidean skeleton.



(f) Final 8-connected skeleton.

Figure 4.7: The skeletonization procedure is applied to the example region in (a). First, the medial axis transform (MAT) of the region is computed, shown here in (b). The MAT consists of the centers of maximal balls inside the region. Note that, in this example, it is disconnected. The MAT is then used to help obtain the Euclidean skeleton of the region (the Euclidean skeleton must contain the MAT), shown in (c). The MAT is then filtered to remove skeleton points which are either too close to the region's border or whose bisector angle is too small. The filtered version is shown in (d). The filtered MAT is used to constrain the Euclidean skeleton from (c) to produce the filtered Euclidean skeleton shown in (e). Finally, we apply a cycle of the parallel thinning algorithm to the filtered Euclidean skeleton in order to obtain the 8-connected result in (f).

original *simple-connectivity* measure is proposed to quantify the extent to which a segment is simply-connected, i.e., the extent to which the segment is free of holes. A more in-depth discussion of this new measure, with several detailed examples, was presented in a technical report [118].

There has been some limited amount of previous work on measures that assess the presence of holes. Perhaps the simplest measure is the absolute number of holes of a region, or otherwise its Euler number [34, 78]. While the number of holes is an informative measure, it gives no clue about the hole size or extent. Soffer and Samet [60] and Wentz [69] have defined measures based on the area of the holes relative to that of the region. A disadvantage of area-based measures is that they are insensitive to the presence of multiple small holes, since they only take into account the holes' total area. This justifies the presentation of a novel *simple-connectivity* measure. Our measure is estimated as a fuzzy-and (minimum) combination of two simpler measure components. The first component quantifies the presence of holes by relating the length of the boundaries of the holes to the total length of all boundaries of the region. A limitation of this measure is that it is not sensitive to large holes that may have small boundary lengths. In order to address this issue, the final simple-connectivity measure is defined as a combination of the boundary-based measure with a more traditional area-based one.

First, the measure of simple-connectivity based only on boundary lengths, which we call SimpleConnectivity4Adjacency, is defined:

$$\text{SimpleConnectivity4Adjacency} = \frac{\text{4-adjacency cross-aura measure of the } \textit{external} \text{ boundary}}{\text{4-adjacency cross-aura measure of the } \textit{total} \text{ boundary}}. \tag{4.16}$$

Above, the "4-adjacency cross-aura measure of the *external* boundary" does not take into account the parts of the boundary that the region forms with its holes. It is computed in the following manner. For each pixel of the boundary, we count its 4-neighbors that: (i) do not belong to the region, and (ii) do not belong to the region's holes. These counts are tallied up, resulting in the desired cross-aura measure. The counting procedure is such that a pixel that does not belong to the region (nor its holes) and is a 4-neighbor to multiple boundary pixels will be counted multiple times (once for each of its 4-neighboring boundary pixels).

The "4-adjacency cross-aura measure of the *total* boundary" is defined similarly, but also takes into account contributions from holes. For each pixel in the boundary, we count its 4-neighbors that do not belong to the region, irrespective of whether they belong to a hole or not. These counts are added up over all boundary pixels. Again, a pixel that does not belong to the region and is a 4-neighbor to multiple boundary pixels will be counted multiple times. Examples showing how these lengths are computed can be found in a previous technical report [118].

The disadvantage of SimpleConnectivity4Adjacency is that holes with large area, but small boundary lengths, present only intermediate values of SimpleConnectivity4Adjacency. An example of this behavior is given in Figures 4.8 and 4.10.

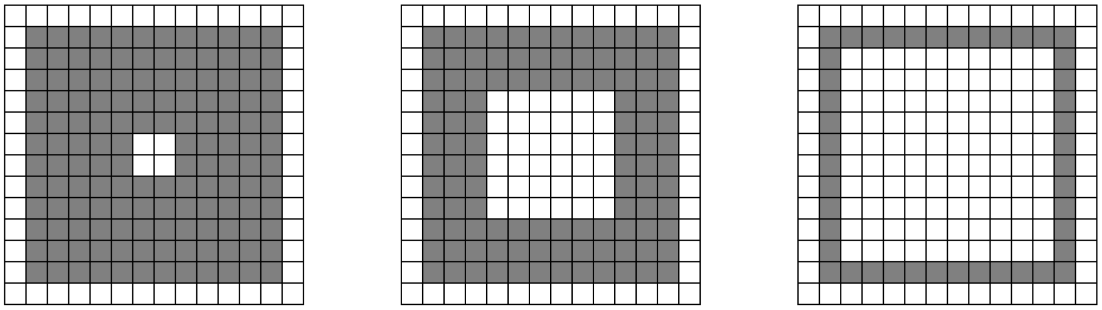


Figure 4.8: Example regions used to illustrate the different sensitivities of the simple-connectivity measures to a hole of increasing size. The respective measures are: (a) SimpleConnectivity4Adjacency $\approx$ 0.86, and FilledAreaRatio $\approx$ 0.97; (b) SimpleConnectivity4Adjacency $\approx$ 0.67, and FilledAreaRatio $\approx$ 0.75; and (c) SimpleConnectivity4Adjacency $\approx$ 0.55, and FilledAreaRatio $\approx$ 0.31. Note that SimpleConnectivity4Adjacency is not very sensitive to the hole in (c), which has large area, but a simple boundary.

This result is undesirable, since it is somewhat counter-intuitive: perceptually, regions containing holes with large area should have low simple-connectivity values. This problem motivated combining SimpleConnectivity4Adjacency with an area-based measure.

We define the FilledAreaRatio as

$$\text{FilledAreaRatio} = \frac{\text{Area}}{\text{FilledArea}}, \tag{4.17}$$

where FilledArea is the area of the region with its holes filled in. As noted previously, the disadvantage of area-based measures is that they are not sensitive to the presence of multiple small holes, or holes that have large boundaries while maintaining small area. This is illustrated in Figures 4.9 and 4.10.

The proposed final measure of simple-connectivity is the fuzzy-AND (mini-

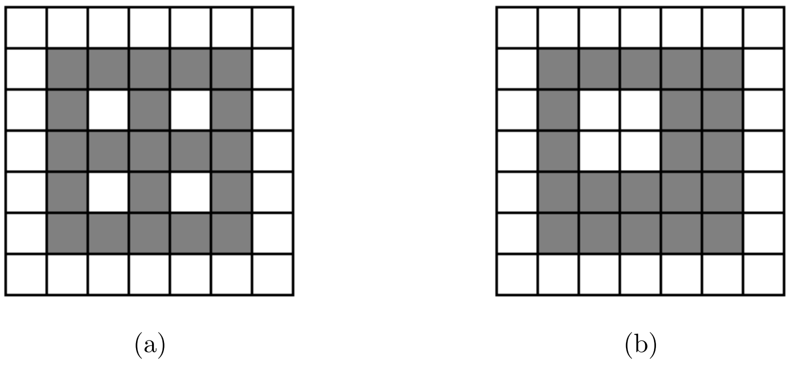


Figure 4.9: Two regions, used to illustrate the different sensitivities of the simple-connectivity measures to the presence of multiple holes. The total area of the holes is the same in (a) and (b), though in (a) the region contains 4 holes, while in (b) there is only one. The respective measures are: (a) SimpleConnectivity4Adjacency ≈ 0.56, and FilledAreaRatio $= 0.84$; (b) SimpleConnectivity4Adjacency ≈ 0.71, and FilledAreaRatio $= 0.84$

mum) between the two previously defined fuzzy membership values,

$$\text{SimpleConnectivity} = \min\{\text{SimpleConnectivity4Adjacency}, \text{FilledAreaRatio}\}.$$

Figure 4.10 illustrates how this final measure performs in better agreement with our intuition about holes, by means of a conservative combination of the two input fuzzy membership values. Noteworthy, all the simple-connectivity measures belong to the range $[0, 1]$, with a value of 1 only when the region has no holes. The reader is referred to [118] for a more detailed discussion of these measures with several examples.

4.4.10 Average Contrast Along the Boundary

Matsuyama and Hwang [119] suggest a measure of contrast along a segment's boundary, which is included in our attribute set. This measure is defined simply

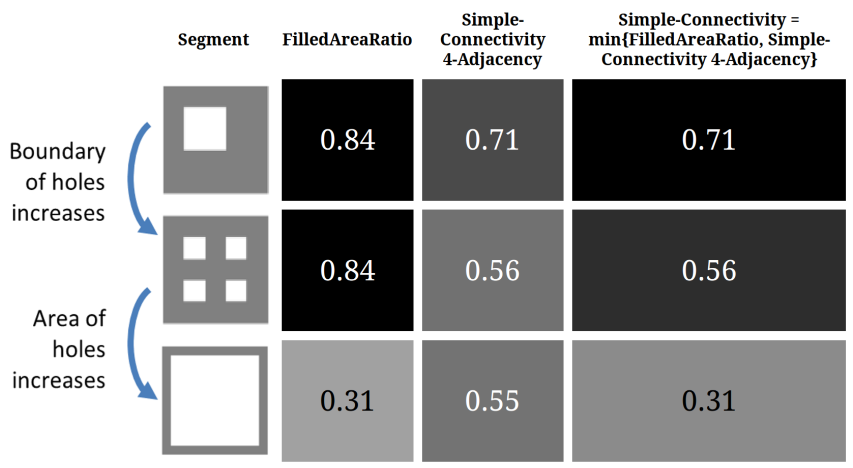

Segment
FilledAreaRatio
Simple-Connectivity 4-Adjacency
Simple-Connectivity = min{FilledAreaRatio, Simple-Connectivity 4-Adjacency}
Boundary of holes increases
Area of holes increases
0.84
0.71
0.71
0.84
0.56
0.56
0.31
0.55
0.31

Figure 4.10: The figure illustrates how the different measures of simple-connectivity behave. The total area of the holes in the top and middle segments is the same, so that FilledAreaRatio does not change between them. However, since the boundary lengths do change, SimpleConnectivity4Adjacency is able to capture their difference. In the middle and bottom examples, the boundary lengths are similar, even though the total area of the holes has changed. In this case, FilledAreaRatio is able to capture the change, while SimpleConnectivity4Adjacency is not. The final measure, SimpleConnectivity, behaves in agreement with human perception of both types of changes in hole properties by combining with a fuzzy-and (minimum) operator the previous two measures.

as the difference between the average value of the pixels in the region that are adjacent to the background and the average value of the pixels in background that are adjacent to the region.

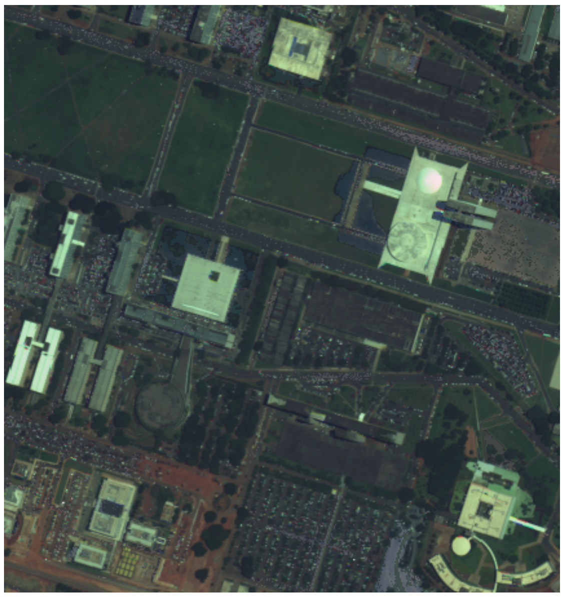
4.5 OBIA Systems Employing a Segment-Specific Convergence-of-Evidence Approach

According to the OBIA paradigm, where second-stage statistical (inductive data learning) or prior knowledge-based inference systems are input with image segments detected at the first stage, two combinations of per-segment attributes are tested with two different types of input images. A spaceborne multi-spectral image of an urban area is automatically pre-classified and segmented by the first-stage SIAM expert system. Color images of leaves acquired with consumer-level digital cameras are segmented by the leaf image segmentation algorithm proposed in [1,46].

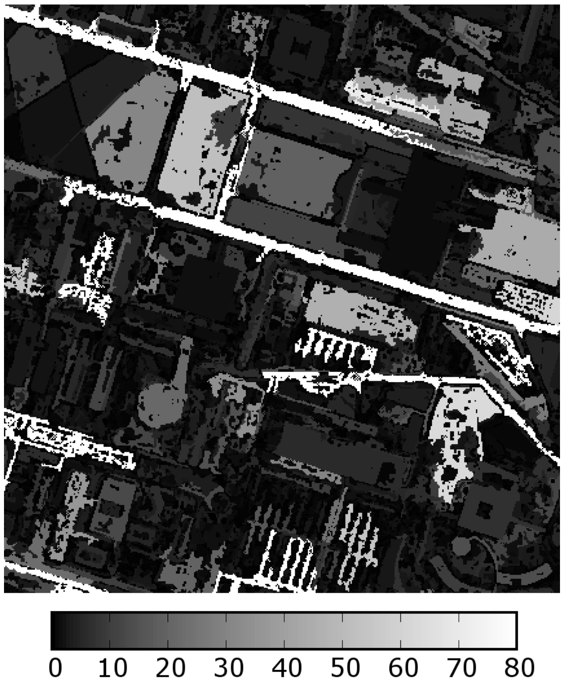
Figure 4.11 presents images of segment-specific attribute values generated from the spaceborne test image, namely, elongatedness, simple-connectivity and straightness of boundary. Figure 4.11b demonstrates the effectiveness of elongatedness in highlighting roads and parking lots. Segments that contain holes also present somewhat high values of elongatedness even though they might be relatively compact. These types of segments can be identified by their high simple-connectivity values, as show in Figure 4.11c. Finally, the straightness of boundary measure, depicted in 4.11d, appears suitable for capturing man-made structures, which usually present straight boundaries independently of whether their overall shapes are simple or complex.

Figure 4.12 presents a sample of the segments detected by SIAM in the spaceborne test image of an urban area. Segments numbered 1 through 6 are buildings, or parts of buildings, while segments 7 through 9 are pieces of roads. With respect to distinguishing buildings from roads, some general trends can be observed. Probably the most discriminative attribute is elongatedness. Though buildings can be somewhat elongated, roads usually present very high values of elongatedness. Additionally, roads are typically non-compact, resulting in lower measures of roundness. Regarding appearance, bright segments (provided with high values of "mean panchromatic intensity") are usually indicative of buildings. Though buildings can appear as either bright or dark structures, asphalt roads are always dark, so that bright segments are highly indicative of non-road structures. Another interesting observation is that not all rectangular segments are buildings, as exemplified by the mostly rectangular road segment 9.

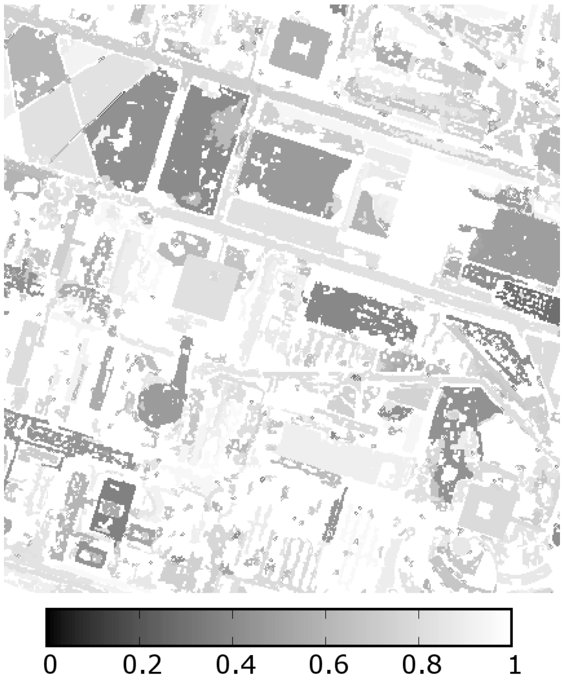
Some more specific observations from Figure 4.12 regarding the different shape attributes are as follows. Segments 4 and 5 represent buildings whose change in brightness generate holes in their surface area, lowering their simple-connectivity values. Nonetheless, both segments have high values of the polygon-based approximate rectangularity measure, as well as of straightness of boundary. Segment 6 is particularly interesting, since it represents a building, yet it has an overall concave shape. This results in low measures of convexity and roundness, though the segment still has a high straightness of boundary value. Segment 7 depicts a piece of a road network, which results in a very low value of convexity. Noteworthy, this is not the



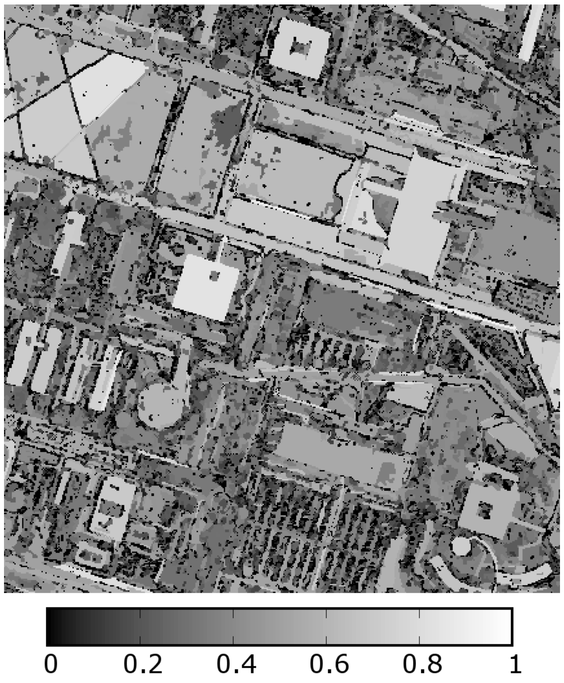
(a) Original



(b) Elongatedness



(c) Simple-Connectivity

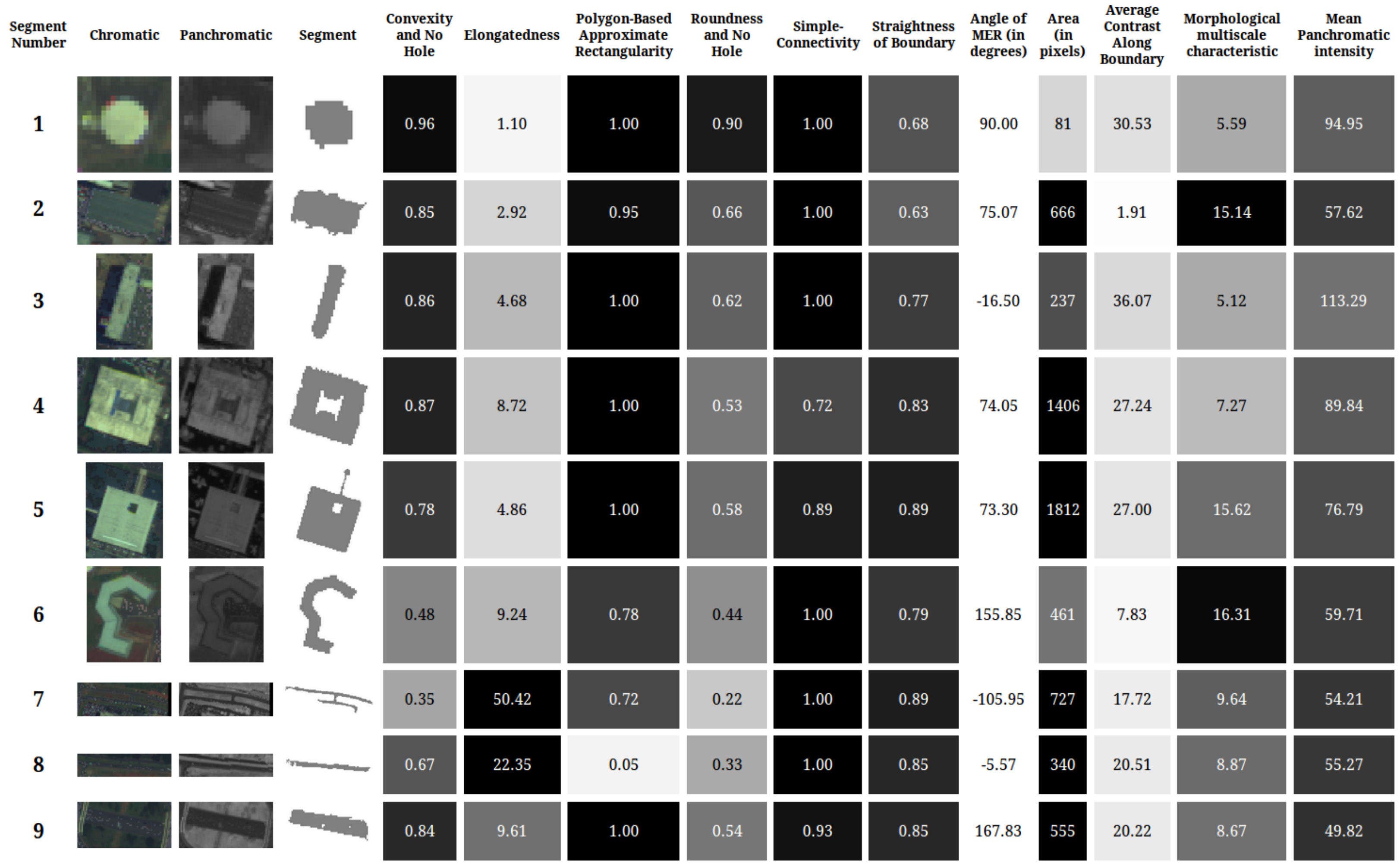


(d) Straightness of boundary

Figure 4.11: Attribute maps showing values of: (b) elongatedness, (c) simple-connectivity and (d) straightness of boundary. Each segment's attribute value is encoded as its gray scale. In (b), the range of values represented is from 0 (black) to 80 (white), while in (c) and (d) the range goes from 0 (black) to 1 (white). In (a), a false color depiction of the urban area being analyzed is shown.

case in segment 8, which represents a single straight piece of road. Nonetheless, both segments present very high values of elongatedness, distinguishing them from other structures. Finally, segment 9, though originating from a piece of road, is very hard to distinguish from a building in general. However, it becomes clear that the segment is a piece of road once its neighborhood (not shown in the Figure) is observed: it consists of road-like segments that possess the same overall orientation and cast no shadow, unlike buildings.

Figure 4.13 presents a series of leaves, each from a different tree species. These examples illustrate how metrological/statistically-based shape and size attributes, featuring a physical meaning, may provide a photointerpreter or an expert system with a vocabulary suitable for leaf description and discrimination. For example, in segment number 1, the cluster of pine leaves stands out for having a very small area, high elongatedness, and low roundness. Segments 2 through 4, which depict compound leaves, all have outstanding values of elongatedness. In particular, segment 2, which has very thin leaflets, presents the highest elongatedness value. Compound leaves also tend to present holes formed from the overlap of separate individual leaflets, as exemplified by segments 2 and 3. This results in a decrease of their simple-connectivity measures. Segments 6 and 7 represent two different species of oak trees, though the marked protrusions in segment 7 give rise to lower convexity, roundness, and straightness of boundary. The sycamore leaf in segment 9 can be distinguished from the maple in segment 8 mainly by its low straightness of boundary. Segment number 5 is a simple leaf with a smooth boundary, presented here to provide a reference set of attribute values.



Segment Number	Chromatic	Panchromatic	Segment	Convexity and No Hole	Elongatedness	Polygon-Based Approximate Rectangularity	Roundness and No Hole	Simple-Connectivity	Straightness of Boundary	Angle of MER (in degrees)	Area (in pixels)	Average Contrast Along Boundary	Morphological multiscale characteristic	Mean Panchromatic intensity
1				0.96	1.10	1.00	0.90	1.00	0.68	90.00	81	30.53	5.59	94.95
2				0.85	2.92	0.95	0.66	1.00	0.63	75.07	666	1.91	15.14	57.62
3				0.86	4.68	1.00	0.62	1.00	0.77	-16.50	237	36.07	5.12	113.29
4				0.87	8.72	1.00	0.53	0.72	0.83	74.05	1406	27.24	7.27	89.84
5				0.78	4.86	1.00	0.58	0.89	0.89	73.30	1812	27.00	15.62	76.79
6				0.48	9.24	0.78	0.44	1.00	0.79	155.85	461	7.83	16.31	59.71
7				0.35	50.42	0.72	0.22	1.00	0.89	-105.95	727	17.72	9.64	54.21
8				0.67	22.35	0.05	0.33	1.00	0.85	-5.57	340	20.51	8.87	55.27
9				0.84	9.61	1.00	0.54	0.93	0.85	167.83	555	20.22	8.67	49.82

Figure 4.12: The table illustrates how the combination of attributes allows for discrimination of buildings and roads in spaceborne images of urban areas. The gray levels in each cell represent the numerical values, such that darker cells correspond to higher values (except for the "mean panchromatic intensity" column, where the gray level is the mean intensity itself). Segments 1 through 6 are buildings or part-of-buildings, while segments 7 through 9 are pieces of roads.

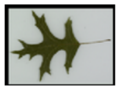
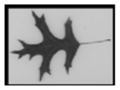
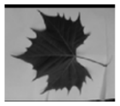
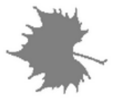
Segment Number	Chromatic	Panchromatic	Segment	Convexity and No Hole	Elongatedness	Polygon-Based Approximate Rectangularity	Roundness and No Hole	Simple-Connectivity	Straightness of Boundary	Angle of MER (in degrees)	Area (in pixels)	Average Contrast Along Boundary	Morphological multiscale characteristic	Mean Panchromatic intensity
1				0.34	46.14	0.00	0.24	0.91	0.93	-3.37	356	39.65	1.39	58.85
2				0.68	125.64	0.00	0.16	0.70	0.70	-63.00	4502	32.39	3.05	109.80
3				0.51	29.78	0.00	0.23	0.88	0.83	91.15	3444	42.87	7.49	40.39
4				0.59	14.52	0.00	0.36	0.94	0.88	177.61	6004	59.83	15.34	68.36
5				0.95	1.31	0.82	0.75	1.00	0.95	116.03	5112	43.14	30.41	43.30
6				0.85	3.47	0.09	0.65	1.00	0.82	157.46	2765	32.60	29.69	32.74
7				0.39	4.21	0.00	0.30	1.00	0.80	-118.86	2462	45.73	7.61	74.82
8				0.63	2.11	0.32	0.48	1.00	0.80	122.59	2692	45.92	14.87	25.25
9				0.61	3.43	0.00	0.35	1.00	0.69	37.57	3927	42.51	28.30	49.09

Figure 4.13: The table illustrates how the combination of attributes helps to differentiate between leaves of different tree species. The gray levels in each cell represent the numerical values, such that darker cells correspond to higher values (except for the "mean panchromatic intensity" column, where the gray level is the mean intensity itself). Each segment represents a leaf of a different tree species. See text for a discussion.

4.6 Conclusions and Future Directions

A comprehensive battery of complementary segment-specific feature estimators is proposed for operational use in OBIA systems encompassing an image segmentation first stage. Estimated per-segment shape, size and photometric attributes are: (i) provided with a physical (intuitive) meaning, (ii) scale invariant, (iii) computationally efficient and (iv) complementary. In practice, selected and implemented estimators are novel in terms of: (a) individual formulation and/or implementation and (b) overall combination of individuals, whose aim is comprehensiveness, which implies application-domain independence, and non-redundancy of different indexes.

The potential utility of the proposed set of complementary attributes is investigated in two test cases, where a segment-based categorization approach, based on convergence-of-evidence, is sketched to cope with a satellite image of urban areas and with images of leaves of different tree species

The proposed set of shape features comprises both contour- and region-based variables in the spatial domain, which are intuitive to understand[4]. This quality is important to allow an expert to model his/her prior knowledge of target objects into the classification stage of an OBIA system. In addition, as suggested by Peura and Iivarinen [57], the availability of semantically simple basic shape descriptors guarantees their scalability and transferability to different application domains.

On the other hand, it is well known by the computer vision community that, in general, contour-based and region-based shape descriptors in the spatial domain are

[4]*Spatial* is mentioned here in opposition to a transform domain, such as the frequency domain.

sensitive to noise [59]. To compensate for this inherent lack of robustness to image noise, our proposed set of contour-based and region-based shape descriptors in the spatial domain adopts several strategies. For example, first, simple-connectivity is explicitly measured and used as an indicator of noise intensity. Second, in the straightness of boundary variable estimation, an optimal scale factor is selected based on multi-scale analysis, so that noise occurring at other scales may be ignored. Third, polygon-based approximate rectangularity adopts a fuzzy rule set, to cope with fuzzy information, and so forth.

In the existing literature, different strategies are adopted to mitigate the effect of image noise in the estimation of shape attributes. In two-stage OBIA system development, a second-stage segment-specific prior knowledge-based (non-adaptive to input data) fuzzy decision tree can be adopted to cope with fuzzy information sources [120], like in the hybrid-inference OBIA system proposed in [54,55]. A fuzzy decision tree provides an intuitive (linguistic) structure to distinguish between a discrete and finite set of mutually exclusive classes, according to a convergence-of-evidence approach. At the same time, the fuzzy decision tree can be totally exhaustive, to provide a complete partition of the measurement space. Noteworthy, though noise may affect individual attributes, the convergence of multiple independent sources of evidence allows strong conjectures. For example, in the presence of image noise, segments are typically affected by holes, whose presence affects shape measures, such as elongatedness, roundness, and convexity. Hence, the simple-connectivity measure should be investigated at the first hierarchical stage of a fuzzy decision tree, as a first estimate of the degree of image noise affecting the reliability of further shape descriptors.

According to Zhang and Lu [59], a viable alternative to shape descriptors in the spatial domain is the generic Fourier descriptor (GFD), claimed to be robust to noise, simple to compute, compact, and capable of high retrieval performance. Unfortunately, the GFD has no intuitive meaning and its use typically requires training examples. As a future development of the present study, the combination of the GFD, which is non intuitive, but robust to noise, with intuitive, but sensitive to noise shape attributes in the spatial domain will be investigated to develop a novel generation of hybrid-inference OBIA systems, capable of taking advantage of the unique features of each family of geometric descriptors in the spatial or frequency domain and overcome their shortcomings [66].

As future work, the proposed attribute set can be used for the implementation of a knowledge-based object recognition system for analysis of urban area satellite images. To begin with, we can segment satellite images using the Satellite Image Automatic Mapper (SIAM) proposed by Baraldi et al. [65–67]. SIAM associates to each image pixel a preliminary category label (PCL). A segmentation is then obtained by forming a segment from each connected component with the same PCL. The PCLs correspond to semiconcepts [66], which carry some amount of semantic meaning, though incomplete. For example, we might say the *vegetation* class corresponds to a semiconcept, while the *deciduous forest* class corresponds to a concept, since it carries more specific meaning. The meaning provided by semiconcepts, however, can be very useful during the analysis of segments in a knowledge-based system, as we exemplify later below. Another potential benefit of the PCLs is that they allow for a better management of computational resources during image analysis. For example, if we wanted to focus on analyzing urban areas, we could avoid computing complex high-level features from large areas of vegetation or bodies of water.

The attributes proposed in this chapter may be used as a starting point to develop a fuzzy rule set for detection of buildings, similar to the one proposed by Shackelford and Davis [54]. However, here we are able to provide intuitive detection rules by using attributes, facilitating the incorporation of prior world knowledge. For example, a rule to compute the membership of segment s in class *Rectangular building* could look like the following.

> IF PCL(s) compatible with building AND s has high *Approximate Rectangularity* AND s has appropriate Area THEN
>
> s membership in *Rectangular building* is high.

Above, PCL(s) denotes the preliminary category label of segment s, provided by SIAM, as summarized previously.

It is also important to give special attention to relationships between neighboring segments when developing a rule set. Of particular importance for building detection is the fact that a shadow in an urban area is very indicative of a neighboring building. Additionally, two neighboring segments that are somewhat building-like may reinforce each other, or be joined for further analysis. Similar to Shackelford and Davis's implementation (but using attributes), a rule that uses the presence of shadows to indicate the presence of a building could look like the following.

IF r membership in *Shadow* is high AND s is a neighbor of r THEN

s membership in *Building* is high.

Above, r and s are two distinct segments. In order to detect that some segment r is a potential shadow, again, a series of rules would be used. For example, we could define rules describing the fact that shadows of buildings usually appear as thin rectangles or L-shapes. Attributes such as the segment's approximate rectangularity, area, the straightness of its boundary, and its PCL would help narrow narrow down segments that are potentially shadows of buildings.

Chapter 5: Conclusion

This thesis presented work on object classification, following a traditional approach of image segmentation followed by segment-based analysis. This framework allows for computing shape-based attributes, which was one focus of the thesis. The contributions presented were targeted at two real-world applications: identification of tree species from leaf photos and categorization of objects in satellite imagery of urban areas. In common, both the applications make use of object shape as a distinctive feature and are bound by time constraints.

Leafsnap was the first mobile system to employ automatic recognition of tree species. A series of practical challenges were faced during its development, in order to achieve a useful real-world functioning system. Leafsnap's mobile apps require a user-friendly interface, which prompted the development of a classifier to detect whether a leaf photo was correctly taken by the user: so that the leaves may be accurately segmented, users are required to take photos of a single leaf against a plain light-colored background. For leaf classification, the segmentation and shape recognition methods were developed with the constraint that users receive final results in a matter of seconds.

The large amount of data collected by Leafsnap users could be of great value.

Leafsnap has received much public interest, counting with over 1.5 million app downloads. In the US, users uploaded 1.7 million valid images, 1.3 million of which are GPS-tagged. This large collection of images shows potential for tracking tree species across geographic location and time. Additionally, interaction with the dataset has the potential to aid in the discovery of new tree species by allowing better targeted field studies. An ongoing challenge is to label these uploaded images with their corresponding tree species. Over 100 thousand of the GPS-tagged images have user-supplied labels. This represents a small fraction relative to the 1.3 million GPS-tagged images, though it should be sufficient for interesting analyzes. A study by botanists from the Smithsonian Institution is currently underway to assess the quality of user-provided labels. As future work, it would be interesting to explore new ways of collecting and verifying the correctness of species labels, whether via volunteer work of experts or crowd-sourcing. The data analysis could be expanded later on with the collection of more user-submitted images. Leafsnap has also been launched in the UK, and thus could be used to map that country's species. Additional users could be attracted with the launch of an Android app, whose development is currently underway.

A study on segmentation of leaves in semi-controlled conditions was presented. The datasets experimented with consisted of leaf photos taken against plain light-colored backgrounds. These are the kinds of images used in practice in Leafsnap for species identification. Popular segmentation algorithms were tested on the segmentation task, and their weaknesses noted. Though the segmentation problem is somewhat controlled, several practical challenges were observed, the main ones being: presence of cast shadows and specularities, the variety of possible leaf shapes and sizes, and the constraint on time. A custom-tailored algorithm was presented to overcome these difficulties, returning results in one or two seconds and at the same time showing state-of-the-art results. Nonetheless, there is still room for improved accuracy. It is possible that better segmentation results could be obtained by making use of additional image features, such as texture, color, and shape. Perhaps a challenge in this direction is to maintain time efficiency while taking advantage of additional features.

After image segmentation, segment-based features are usually extracted for further analysis. We have presented a set of image-object attributes suitable for knowledge-based systems. As such, the attributes are easily describable and intuitive. Attributes were selected based on their potential relevance to multiple problem domains, and such that they could be computed efficiently. The presented set of attributes improves on the previous literature in two respects. First, it is more comprehensive than those presented in previous works, containing a larger number of attributes. Second, individual attributes collected from the literature are improved upon via new formulations and computational methods. The benefits of the proposed set were demonstrated on toy cases and on two real-world applications: discrimination of man-made objects in satellite images of urban areas, and identification of tree species from images of leaves.

A line of future work is to develop rule-based recognition systems on top of the proposed set of attributes. In the case of discrimination of man-made objects from satellite images, Shackelford and Davis presented a framework to achieve this [54].

However, here we are able to provide intuitive detection rules by using attributes, facilitating the incorporation of prior world knowledge. An implemented rule-based recognition system would allow for an objective evaluation of the attribute set in terms of classification accuracy.

In general, the use of attributes in computer vision systems lessens the dependence on training examples. Attributes have been used for image search via keywords, as well as object recognition via textual descriptions [74–77]. The use of attributes also affords modularity: attributes can be shared across different categories, and new object categories can be learned from very few examples, or even solely from textual descriptions. Along this line, the use of attributes also allows data from varying sources to be more easily combined, by decoupling prior domain knowledge from specific low-level features or parameters. In particular, knowledge from domain experts can be easily integrated by, for example, defining rules based on attribute values [71, 72]. A common limitation of attribute-based approaches is that they usually do not provide a complete representation of objects, eventually missing properties that are difficult to be described in words. This can be mitigated in practice by combining the use of attributes with more traditional sets of features, which in general would also require a set of training examples.

Bibliography

[1] N. Kumar, P.N. Belhumeur, A. Biswas, D.W. Jacobs, W.J. Kress, I.C. Lopez, and J.V.B. Soares. Leafsnap: A computer vision system for automatic plant species identification. In *European Conference on Computer Vision (ECCV)*, pages 502–516, 2012.

[2] Steve Branson, Catherine Wah, Boris Babenko, Florian Schroff, Peter Welinder, Pietro Perona, and Serge Belongie. Visual recognition with humans in the loop. In *European Conference on Computer Vision (ECCV)*, pages 438–451, 2010.

[3] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, pages 722–729, 2008.

[4] Hervé Goëau, Pierre Bonnet, Alexis Joly, Itheri Yahiaoui, Daniel Barthélémy, Nozha Boujemaa, and Jean-François Molino. The ImageCLEF 2012 plant identification task. In *Working notes of CLEF 2012 conference*, 2012.

[5] Gaurav Agarwal, Peter Belhumeur, Steven Feiner, David Jacobs, W. John Kress, Ravi Ramamoorthi, Norman A. Bourg, Nandan Dixit, Haibin Ling, Dhruv Mahajan, Rusty Russell, Sameer Shirdhonkar, Kalyan Sunkavalli, and Sean White. First steps toward an electronic field guide for plants. *Taxon*, 55(3):pp. 597–610, 2006.

[6] Peter Belhumeur, Daozheng Chen, Steven Feiner, David Jacobs, W. Kress, Haibin Ling, Ida Lopez, Ravi Ramamoorthi, Sameer Sheorey, Sean White, and Ling Zhang. Searching the world's herbaria: A system for visual identification of plant species. In *European Conference on Computer Vision (ECCV)*, pages 116–129, 2008.

[7] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42:145–175, 2001.

[8] Jacob Whitehill, Ting-Fan Wu, Jacob Bergsma, Javier R. Movellan, and Paul L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in neural information processing systems (NIPS)*, pages 2035–2043, 2009.

[9] U.S. Geological Survey. Digital representations of tree species range maps from "Atlas of United States Trees" by Elbert L. Little, Jr., 1999. `http://esp.cr.usgs.gov/data/little/`.

[10] Robert S. Thompson, Katherine H. Anderson, and Patrick J. Bartlein. Atlas of relations between climatic parameters and distributions of important trees and shrubs in North America. U.S. Geological Survey Professional Paper 1650 A&B, December 1999. `pubs.usgs.gov/pp/p1650-a/`.

[11] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

[12] Pablo Arbelaez, Charless Fowlkes, David Martin, and Jitendra Malik. Berkeley segmentation and boundary detection benchmark and dataset, Accessed 2011. `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html`.

[13] Sharon Alpert, Meirav Galun, Ronen Basri, and Achi Brandt. Image segmentation by probabilistic bottom-up aggregation and cue integration. In *Computer Vision and Pattern Recognition (CVPR)*, June 2007.

[14] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, January 1979.

[15] Guillaume Cerutti, Violaine Antoine, Laure Tougne, Julien Mille, Didier Coquin, and Antoine Vacavant. ReVeS Participation - Tree species classification using random forests and botanical features. In *Working notes of CLEF 2012 conference*, 2012.

[16] A.-G. Manh, G. Rabatel, L. Assemat, and M.-J. Aldon. Weed leaf image segmentation by deformable templates. *Journal of Agricultural Engineering Research*, 80(2):139–146, 2001.

[17] M-E. Nilsback and A. Zisserman. Delving into the whorl of flower segmentation. In *British Machine Vision Conference (BMVC)*, 2007.

[18] M.P. Kumar, P.H.S. Torr, and A. Zisserman. Obj Cut. In *Computer Vision and Pattern Recognition (CVPR)*, pages 18–25, 2005.

[19] Dalcimar Casanova, João Batista Florindo, Wesley Nunes Gonçalves, and Odemir Martinez Bruno. IFSC/USP at ImageCLEF 2012: Plant identification task. In *Working notes of CLEF 2012 conference*, 2012.

[20] Berrin Yanikoglu, Erchan Aptoula, and Caglar Tirkaz. Sabanci-Okan System at ImageClef 2012: Combining features and classifiers for plant identification. In *Working notes of CLEF 2012 conference*, 2012.

[21] João Camargo Neto, George E. Meyer, and David D. Jones. Individual leaf extractions from young canopy images using Gustafson–Kessel clustering and a genetic algorithm. *Computers and electronics in agriculture*, 51(1):66–85, 2006.

[22] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[23] Akhil Arora, Ankit Gupta, Nitesh Bagma, Shashwat Mishra, and Arnab Bhattacharya. A plant identification system using shape and morphological features on segmented leaflets. In *Working notes of CLEF 2012 conference*, 2012.

[24] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "GrabCut": Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, pages 309–314, 2004.

[25] Vera Bakić, Itheri Yahiaoui, Sofiene Mouine, Saloua Ouertani-Litayem, Wajih Ouertani, Anne Verroust-Blondet, Hervé Goëau, and Alexis Joly. Inria IMEDIA2's participation at ImageCLEF 2012 plant identification task. In *Working notes of CLEF 2012 conference*, 2012.

[26] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[27] Y.Y. Boykov and M-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *International Conference on Computer Vision*, pages 105–112, 2001.

[28] Sanjiv Kumar and Martial Hebert. Discriminative random fields. *International Journal of Computer Vision*, 68(2):179–201, 2006.

[29] P. Kohli, L. Ladický, and P.H.S. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

[30] Yuning Chai, Victor Lempitsky, and Andrew Zisserman. BiCos: A bi-level co-segmentation method for image classification. In *International Conference on Computer Vision (ICCV)*, pages 2579–2586, 2011.

[31] Christopher M. Bishop. *Pattern recognition and machine learning.* Information science and statistics. Springer, 2006.

[32] Bradley Efron. The efficiency of logistic regression compared to normal discriminant analysis. *Journal of the American Statistical Association*, 70(352):892–898, 1975.

[33] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[34] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Pearson/Prentice Hall, 3rd edition, 2008.

[35] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Academic press, second edition, 1990.

[36] Pedro Felzenszwalb and Daniel Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59:167–181, 2004.

[37] C. Christoudias, B. Georgescu, and P. Meer. Synergism in low-level vision. In *International Conference on Pattern Recognition*, volume 4, pages 150–155, August 2002.

[38] Open source computer vision library (OpenCV). `http://opencv.org/`.

[39] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[40] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1124–1131, 2005.

[41] Thomas Leung and Jitendra Malik. Contour continuity in region based image segmentation. In *European Conference on Computer Vision (ECCV)*, pages 544–559. Springer Berlin / Heidelberg, 1998.

[42] Eitan Sharon, Meirav Galun, Dahlia Sharon, Ronen Basri, and Achi Brandt. Hierarchy and adaptivity in segmenting visual scenes. *Nature*, 442(7104):719–846, June 2006.

[43] Meirav Galun, Eitan Sharon, Ronen Basri, and Achi Brandt. Texture segmentation by multiscale aggregation of filter responses and shape elements. In *Computer Vision and Pattern Recognition (CVPR)*, pages 716–723, 2003.

[44] D.R. Martin, C.C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, 2004.

[45] W.J. Dixon and A.M. Mood. The statistical sign test. *Journal of the American Statistical Association*, 41(236):pp. 557–566, 1946.

[46] João V.B. Soares and David W. Jacobs. Efficient segmentation of leaves in semi-controlled conditions. *Machine Vision and Applications*, 24(8):1623–1643, November 2013.

[47] Rolf A. Heckemann, Joseph V. Hajnal, Paul Aljabar, Daniel Rueckert, and Alexander Hammers. Automatic anatomical brain MRI segmentation combining label propagation and decision fusion. *NeuroImage*, 33(1):115–126, 2006.

[48] Thomas Blaschke, Geoffrey J. Hay, Maggi Kelly, Stefan Lang, Peter Hofmann, Elisabeth Addink, Raul Queiroz Feitosa, Freek van der Meer, Harald van der Werff, Frieke van Coillie, and Dirk Tiede. Geographic object-based image analysis – towards a new paradigm. *ISPRS Journal of Photogrammetry and Remote Sensing*, 87:180–191, January 2014.

[49] M. Bertero, T.A. Poggio, and V. Torre. Ill-posed problems in early vision. *Proceedings of the IEEE*, 76(8):869–889, August 1988.

[50] Andrea Baraldi and Luigi Boschetti. Operational automatic remote sensing image understanding systems: Beyond geographic object-based and object-oriented image analysis (GEOBIA/GEOOIA). Part 1: Introduction. *Remote Sensing*, 4(9):2694–2735, September 2012.

[51] Michael F. Goodchild, May Yuan, and Thomas J. Cova. Towards a general theory of geographic representation in GIS. *International Journal of Geographical Information Science*, 21(3):239–260, 2007.

[52] Open Geospatial Consortium Inc. OpenGIS implementation standard for geographic information - simple feature access - Part 1: Common architecture, May 2011.

[53] Marco Bertamini. Who owns the contour of a visual hole? *Perception*, 35:883–894, 2006.

[54] Aaron K. Shackelford and Curt H. Davis. A combined fuzzy pixel-based and object-based approach for classification of high-resolution multispectral data over urban areas. *IEEE Transactions on Geoscience and Remote Sensing*, 41(10):2354–2363, 2003.

[55] Aaron K. Shackelford. *Development of urban area geospatial information products from high resolution satellite imagery using advanced image analysis techniques.* PhD thesis, University of Missouri-Columbia, 2004.

[56] Makoto Nagao and Takashi Matsuyama. *A structural analysis of complex aerial photographs.* Plenum Press, New York, 1980.

[57] Markus Peura and Jukka Iivarinen. Efficiency of simple shape descriptors. In *Proceedings of the third international workshop on visual form*, pages 443–451, 1997.

[58] Paul L. Rosin. Measuring shape: ellipticity, rectangularity, and triangularity. *Machine Vision and Applications*, 14(3):172–184, July 2003.

[59] Dengsheng Zhang and Guojun Lu. Review of shape representation and description techniques. *Pattern recognition*, 37(1):1–19, 2004.

[60] Aya Soffer and Hanan Samet. Negative shape features for image databases consisting of geographic symbols. In *Proc. 3rd International Workshop on Visual Form*, 1997.

[61] Michel Couprie, David Coeurjolly, and Rita Zrour. Discrete bisector function and Euclidean skeleton in 2D and 3D. *Image and Vision Computing*, 25(10):1543–1556, 2007.

[62] Martino Pesaresi and Jon Atli Benediktsson. A new approach for the morphological segmentation of high-resolution satellite imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 39(2):309–320, 2001.

[63] J. Gil and M. Werman. Computing 2-D min, median, and max filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):504–507, 1993.

[64] Kevin Robinson and Paul F. Whelan. Efficient morphological reconstruction: a downhill filter. *Pattern Recognition Letters*, 25(15):1759–1767, 2004.

[65] Andrea Baraldi, Virginia Puzzolo, Palma Blonda, Lorenzo Bruzzone, and Cristina Tarantino. Automatic spectral rule-based preliminary mapping of calibrated landsat TM and ETM+ images. *IEEE Transactions on Geoscience and Remote Sensing*, 44(9):2563–2586, 2006.

[66] Andrea Baraldi, Laurent Durieux, Dario Simonetti, Giulia Conchedda, Francesco Holecz, and Palma Blonda. Automatic spectral-rule-based preliminary classification of radiometrically calibrated SPOT-4/-5/IRS, AVHRR/MSG, AATSR, IKONOS/QuickBird/OrbView/GeoEye, and DMC/SPOT-1/-2 imagery – Part I: System design and implementation. *IEEE Transactions on Geoscience and Remote Sensing*, 48(3):1299–1325, 2010.

[67] Andrea Baraldi, Laurent Durieux, Dario Simonetti, Giulia Conchedda, Francesco Holecz, and Palma Blonda. Automatic spectral rule-based preliminary classification of radiometrically calibrated SPOT-4/-5/IRS, AVHRR/MSG, AATSR, IKONOS/QuickBird/OrbView/GeoEye, and DMC/SPOT-1/-2 imagery – Part II: Classification accuracy assessment. *IEEE Transactions on Geoscience and Remote Sensing*, 48(3):1326–1354, 2010.

[68] Kesheng Wu, Ekow Otoo, and Kenji Suzuki. Optimizing two-pass connected-component labeling algorithms. *Pattern Analysis and Applications*, 12(2):117–135, 2009.

[69] Elizabeth A. Wentz. A shape definition for geographic applications based on edge, elongation, and perforation. *Geographical Analysis*, 32(2):95–112, 2000.

[70] L.A. Ruiz, J.A. Recio, A. Fernández-Sarría, and T. Hermosilla. A feature extraction software tool for agricultural object-based image analysis. *Computers and Electronics in Agriculture*, 76(2):284–296, 2011.

[71] C.E. Liedtke, J. Bückner, O. Grau, S. Growe, and R. Tönjes. AIDA: A system for the knowledge based interpretation of remote sensing data. In *3rd International Airborne Remote Sensing Conference*, 1997.

[72] J. Bückner, M. Pahl, O. Stahlhut, and C.E. Liedtke. geoAIDA – A knowledge based automatic image data analyser for remote sensing data. In *ICSC Congress on Computational Intelligence Methods and Applications (CIMA)*, 2001.

[73] A. Baraldi, L. Bruzzone, and P. Blonda. Quality assessment of classification and cluster maps without ground truth knowledge. *IEEE Transactions on Geoscience and Remote Sensing*, 43(4):857–873, 2005.

[74] Neeraj Kumar, Alexander Berg, Peter N. Belhumeur, and Shree Nayar. Describable visual attributes for face verification and image search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):1962–1977, 2011.

[75] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition (CVPR)*, pages 951–958, 2009.

[76] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1778–1785, 2009.

[77] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (CSUR)*, 40(2):5, 2008.

[78] Babu M. Mehtre, Mohan S. Kankanhalli, and Wing Foon Lee. Shape measures for content based image retrieval: a comparison. *Information Processing & Management*, 33(3):319–337, 1997.

[79] Luciano da Fontoura Costa and Roberto Marcondes Cesar Jr. *Shape analysis and classification: theory and practice*. CRC Press, 2000.

[80] Farzin Mokhtarian and Alan K. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):789–805, 1992.

[81] Haili Chui and Anand Rangarajan. A new point matching algorithm for non-rigid registration. *Computer Vision and Image Understanding*, 89(2):114–141, 2003.

[82] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):509–522, 2002.

[83] Serge Belongie, Greg Mori, and Jitendra Malik. Matching with shape contexts. In *Statistics and Analysis of Shapes*, pages 81–105. Springer, 2006.

[84] Thomas B. Sebastian, Philip N. Klein, and Benjamin B. Kimia. Recognition of shapes by editing their shock graphs. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(5):550–571, 2004.

[85] Pedro F. Felzenszwalb. Representation and detection of deformable shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(2):208–220, 2005.

[86] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, 2005.

[87] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[88] Krystian Mikolajczyk, Andrew Zisserman, and Cordelia Schmid. Shape recognition with edge-based features. In *British Machine Vision Conference (BMVC)*, volume 2, pages 779–788, 2003.

[89] Owen Carmichael and Martial Hebert. Shape-based recognition of wiry objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(12):1537–1552, 2004.

[90] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 401–408. ACM, 2007.

[91] Jamie Shotton, Andrew Blake, and Roberto Cipolla. Contour-based learning for object detection. In *International Conference on Computer Vision (ICCV)*, volume 1, pages 503–510, 2005.

[92] Jamie Shotton, Andrew Blake, and Roberto Cipolla. Multiscale categorical object recognition using contour fragments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(7):1270–1281, 2008.

[93] Andreas Opelt, Axel Pinz, and Andrew Zisserman. A boundary-fragment-model for object detection. In *European Conference on Computer Vision (ECCV)*, pages 575–588. Springer, 2006.

[94] Vittorio Ferrari, Loic Fevrier, Frederic Jurie, and Cordelia Schmid. Groups of adjacent contour segments for object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(1):36–51, 2008.

[95] Dennis S. Arnon and John P. Gieselmann. A linear time algorithm for the minimum area rectangle enclosing a convex polygon. Technical report, Purdue University, 1983.

[96] George A Moore. Automatic scanning and computer processes for the quantitative analysis of micrographs and equivalent subjects. In *Pictorial Pattern Recognition*, pages 275–326, Washington, DC, 1968.

[97] Avraham A. Melkman. On-line construction of the convex hull of a simple polyline. *Information Processing Letters*, 25:11–12, 1987.

[98] S.B. Tor and A.E. Middleditch. Convex decomposition of simple polygons. *ACM Transactions on Graphics (TOG)*, 3(4):244–265, 1984.

[99] Bart Braden. The surveyor's area formula. *The College Mathematics Journal*, 17(4):326–337, 1986.

[100] Paul S. Heckbert and Michael Garland. Survey of polygonal surface simplification algorithms. In *Multiresolution Surface Modeling Course Notes*. ACM SIGGRAPH, 1997.

[101] Azriel Rosenfeld. Compact figures in digital pictures. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-4(2):221–223, 1974.

[102] Luc Vincent. Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms. *IEEE Transactions on Image Processing*, 2(2):176–201, 1993.

[103] Aaron K. Shackelford, Curt H. Davis, and Xiangyun Wang. Automated 2-D building footprint extraction from high-resolution satellite multispectral imagery. In *International Geoscience and Remote Sensing Symposium (IGARSS)*, volume 3, pages 1996–1999, 2004.

[104] Jon Atli Benediktsson, Martino Pesaresi, and Kolbeinn Amason. Classification and feature extraction for remote sensing images from urban areas based on morphological transformations. *IEEE Transactions on Geoscience and Remote Sensing*, 41(9):1940–1949, 2003.

[105] R. Adams. Radial decomposition of disks and spheres. *CVGIP: Graphical Models and Image Processing*, 55(5):325–332, 1993.

[106] Luc Vincent. Morphological grayscale reconstruction: definition, efficient algorithm and applications in image analysis. In *Computer Vision and Pattern Recognition (CVPR)*, pages 633–635, 1992.

[107] A. Balaguer, L.A. Ruiz, T. Hermosilla, and J.A. Recio. Definition of a comprehensive set of texture semivariogram features and their evaluation for object-oriented image classification. *Computers & Geosciences*, 36(2):231–240, February 2010.

[108] M. Armstrong. *Basic Linear Geostatistics*. Springer, 1998.

[109] Curtis E. Woodcock, Alan H. Strahler, and David L. B. Jupp. The use of variograms in remote sensing: I. Scene models and simulated images. *Remote Sensing of Environment*, 25(3):323–348, August 1988.

[110] Curtis E. Woodcock, Alan H. Strahler, and David L. B. Jupp. The use of variograms in remote sensing: II. Real digital images. *Remote Sensing of Environment*, 25(3):349–379, August 1988.

[111] Yanguang Chen. New approaches for calculating Moran's index of spatial autocorrelation. *PLoS ONE*, 8(7):e68336, July 2013.

[112] M. Kim, M. Madden, and T. Warner. Estimation of optimal image object size for the segmentation of forest stands with multispectral IKONOS imagery. In Thomas Blaschke, Stefan Lang, and Geoffrey J. Hay, editors, *Object-Based Image Analysis*, Lecture Notes in Geoinformation and Cartography, pages 291–307. Springer Berlin Heidelberg, January 2008.

[113] Azriel Rosenfeld and Avinash C. Kak. *Digital picture processing*. Academic Press, Inc., 1982.

[114] Louisa Lam, Seong-Whan Lee, and Ching Y. Suen. Thinning methodologies – a comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 14(9):869–885, 1992.

[115] Luc Vincent. Efficient computation of various types of skeletons. In *Medical Imaging V: Image Processing*, pages 297–311. International Society for Optics and Photonics, 1991.

[116] Joseph M. Cychosz. Efficient binary image thinning using neighborhood maps. In Paul S. Heckbert, editor, *Graphics gems IV*, pages 465–473. Academic Press Professional, Inc., 1994.

[117] Azriel Rosenfeld. A characterization of parallel thinning algorithms. *Information and control*, 29(3):286–291, 1975.

[118] João V.B. Soares, Andrea Baraldi, and David W. Jacobs. Segment-based simple-connectivity measure design and implementation. Technical report, University of Maryland, College Park, 2014.

[119] Takashi Matsuyama and Vincent Shang-Shouq Hwang. *SIGMA: A knowledge-based aerial image understanding system*. Plenum Press, New York, 1990.

[120] C.Z. Janikow. Fuzzy decision trees: issues and methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 28(1):1–14, February 1998.