DGK Deutsche Geodätische Kommission
der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 704

Dejan Šeatović

Methods for Real-Time Plant Detection in 3-D Point Clouds

München 2013

Verlag der Bayerischen Akademie der Wissenschaften
in Kommission beim Verlag C. H. Beck

ISSN 0065-5325 ISBN 978-3-7696-5116-4

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 310, Hannover 2013

DGK Deutsche Geodätische Kommission
der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 704

Methods for Real-Time Plant Detection in 3-D Point Clouds

Von der Fakultät für Bauingenieurwesen und Geodäsie
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation

von

Dipl.-Ing. Dejan Šeatović

München 2013

Verlag der Bayerischen Akademie der Wissenschaften
in Kommission bei der C. H. Beck'schen Verlagsbuchhandlung München

ISSN 0065-5325 ISBN 978-3-7696-5116-4

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover
ISSN 0174-1454, Nr. 310, Hannover 2013

Adresse der Deutschen Geodätischen Kommission:

Deutsche Geodätische Kommission

Alfons-Goppel-Straße 11 ● D – 80 539 München

Telefon +49 – 89 – 23 031 1113 ● Telefax +49 – 89 – 23 031 - 1283 / - 1100

e-mail hornik@dgfi.badw.de ● http://www.dgk.badw.de

Prüfungskommission

Vorsitzender:	Prof. Dr.-Ing. Steffen Schön
Referent:	Prof. Dr.-Ing. habil. Hansjörg Kutterer
Korreferenten:	Prof. Dr.-Ing. Uwe Sörgel
	Prof. Dr.-Ing. Willfried Schwarz
Gutachter:	Prof. (FH) Dr.-Ing. Hans Wernher van de Venn

Tag der Einreichung der Arbeit: 02.01.2013

Tag der mündlichen Prüfung: 17.05.2013

© 2013 Deutsche Geodätische Kommission, München

Alle Rechte vorbehalten. Ohne Genehmigung der Herausgeber ist es auch nicht gestattet,
die Veröffentlichung oder Teile daraus auf photomechanischem Wege (Photokopie, Mikrokopie) zu vervielfältigen

ISSN 0065-5325 ISBN 978-3-7696-5116-4

Contents

I Introduction **1**

1 General Thoughts and Motivation **3**

2 Problem Description **7**
2.1 Sensor And Resolution - the sensor selection problem . . . 8
2.1.1 The Pose . . . 8
2.2 Feature Extraction - the segmentation problem . . . 9
2.3 Workforce Replacement - the feature selection and classification problem . . . 10
2.4 Real Time Performance . . . 10

II Preliminary Research **15**

3 The Leaf and its Boundary **17**
3.1 General Remarks About Plant Detection and Automation . . . 17
3.2 Shape Descriptors and their Requirements . . . 17
3.2.1 Regional Descriptors . . . 18
3.3 Leaf Morphology . . . 19
3.4 The Boundary and Boundary Description . . . 20
3.4.1 Freeman Chain Code . . . 21
3.4.2 Polygonal approximation . . . 22
3.4.3 Elliptic Fourier Descriptors . . . 23

4 Processing Necessities for 3-D Point Clouds **29**
4.1 Curve and Surface . . . 29
4.2 Point Clouds . . . 30
4.3 Grid Processing . . . 31
4.3.1 Remarks on Processing Principles Correlation and Convolution . . . 31
4.3.2 Remarks on Morphological Operations . . . 32
4.4 Surface Representation and Modeling . . . 33
4.5 Unfolding . . . 34
4.5.1 The Cylinder Projection . . . 35
4.5.2 The Cone Projection . . . 36
4.5.3 The Tangent Surface Projection . . . 37

5 Recursive Data Filtering and Modeling **41**

6 Classification and Machine Learning **45**
6.1 Why Machine Learning? . . . 45
6.2 General Remarks . . . 46
6.3 Machine Learning: Learning by Examples . . . 46
6.4 Support Vector Machine . . . 48
6.5 k-Means Clustering . . . 50

7 Research Projects **51**
7.1 Computer Graphics . . . 51
7.2 Vision-Based Plant Detection and Weed Control . . . 52
7.3 Segmentation of Point Clouds . . . 52
7.4 Weed Treatment . . . 53
7.5 Summary . . . 54

III Conceptual Model **55**

8 Sensor Discussion **57**
8.1 The Focus of the Chapter . . . 57
8.2 The Minimum Resolution . . . 57
8.2.1 The Results of k-Means Clustering . . . 59
8.3 Conclusion . . . 62
8.4 Data Acquisition and Sensor Discussion . . . 62

9 Data Processing **67**
9.1 The Data Model . . . 67
9.2 Data Processing Principles . . . 69
9.2.1 Preprocessing . . . 69
9.2.2 Low-level Feature Extraction . . . 69
9.3 Region-of-Interest Analysis . . . 71
9.4 Filter Features Summary . . . 73
9.4.1 Edge detection . . . 74
9.5 Parametrization . . . 75
9.6 Filtering Strategy . . . 75
9.7 Leaf Flattening Using Filter Results . . . 79
9.7.1 Flattening by Parameter Line Fitting . . . 81

IV Evaluation and Experiments **87**

10 Experimental System **89**
10.1 General System Information . . . 89
10.2 Sensor System And Its Carrier . . . 89

11 The Data Processing Realization **91**
11.1 Data Flow and Real Time Processing . . . 91
11.2 Software Design and Key Components Description . . . 91

12 Experiments **97**
12.1 General Remarks About Experiments . . . 97
12.2 Data Description . . . 97
12.3 Synthetic Data and Results . . . 98
12.4 Field Data and Results . . . 100
12.4.1 Field Data Example 1 . . . 100
12.4.2 Field Data Example 2 . . . 101
12.5 Remarks About GPU Acceleration Potential. . . . 104
12.6 Conclusion . . . 104

V Conclusion and Outlook 105

13 Summary and Review 107
13.1 Review and Achievements . . . 107
13.2 Critics . . . 108
13.3 Further Work . . . 108
13.4 Final Conclusion . . . 109

Bibliography 122

VI Appendices 129

A Results of the Analysis of the Predefined Shapes 131
A.1 General . . . 131
A.2 Effect of Altering Shape Orientation . . . 132
A.2.1 Differences Between Original Feature Vector and Vectors from Rotated Shapes . . . 137
A.3 Effect of Decreasing Sensor Resolution . . . 140
A.3.1 Differences Between the Original Feature Vector and Vectors Obtained from Lower Resolution Images . . . 140
A.4 Shape Analysis Data, Facts and Tables . . . 143
A.4.1 Separability Analysis and k-Means Clustering . . . 143
A.4.2 Euclidean Method . . . 144
A.4.3 City Block Method . . . 145
A.4.4 Cosine Method . . . 145
A.4.5 Correlation Method . . . 146
A.5 Synthetic Data Images . . . 147

B Algorithms And Sources 153
B.1 Low Level Feature Extraction Algorithm . . . 153
B.2 Recursive Filter Algorithms . . . 154
B.3 Matlab script for the resolution and orientation analysis . . . 156
B.4 GPU Performance Analysis Source . . . 160

Abstract

Automatic shape detection, localization and classification in an unordered and cluttered environment has been a challenge since early days of computer vision and production automation. Since the start of research in this area, a variety of solutions and systems have been designed, built and evaluated. A computational performance increase and improvements in 3-D sensor technology, reveal new possibilities for the automatic plant detection problem: 3-D data-based automatic single plant detection in real-time.

Under the assumption that each plant can be identified by the shape of its leaf, the main problem elaborated in this thesis is formulated: How to provide reliable plant detection systems/algorithms based on leaf shapes; algorithms/systems which use a database created from leaves that were scanned with a flatbed scanner?

Considering a leaf is a two dimensional manifold, located in three dimensional space, the major part of the solution is to close the gap between 3-D grassland data and the 2-D lab data obtained by scanning samples on flatbed scanner. An additional requirement is: data acquisition, segmentation and classification must succeed within the small time interval of 1 *second*. Meeting these two requirements allows the development of a system that has the potential to solve many problems of modern agriculture. A main feature source for the solution presented in this thesis is the plant leaf, particularly its boundary. The boundary does not only exactly describe leaf *shape*, but also contains the leaf *margin*: it fuses two of three descriptors of a leaf. The boundary provides information about a leaf requiring only a small fraction of data, compared to the amount of data that whole leaf surface provides. In a cluttered environment such as grassland, reliable segmentation of objects still represents an unsolved problem, but with help of the third dimension, more reliable and more accurate algorithms can be developed. In this thesis such an algorithm, based on recursive least squares, shows that the increased computation time is justified in a complex environment such as a grassland. Once extracted, a 3-D object has to be compared with the samples stored in a database. The projection of the 3-D leaf to the plane is described here as "flattening", since it emulates the process of pressing a leaf on the flatbed scanner. The projection of a 3-D irregular surface to the plane is solved by polynomial approximation of data in a depth image and the unfolding of each polynomial to the plane. The resulting surface distortions are similar (although *not* equal) to the distortions of leaf samples caused by a flatbed scanner. Extracted boundaries of flattened 3-D leaves appear to be more similar to the species samples in the 2-D database, than the same (and other) shapes axonometrically projected to the plane.

Although the solution derived in this thesis does not represent a silver bullet, it provides significant improvement in plant detection in grasslands. The experimental system, equipped with laser triangulation sensor, has successfully localized 80% of weed specimens on grasslands, which had been cut between two and five weeks prior to the experiment. The aim to achieve maximally 15% false classifications was not achieved; the number of false classifications rose to 40% for single leaves. The major cause of false positive classifications is low spatial resolution of the sensor (e.g. 39 *dpi* instead 400 *dpi*) and the corresponding lack of accuracy in derived height. To solve the problem of single plant detection, it is necessary to invest in further research in area of sensor technology and processing algorithms. An autonomous, reliable and robust single plant detection system will, when realized, revolutionize plant detection and treatment paradigms in future agriculture.

Keywords: Real-Time Single Plant Detection, Depth Image Processing, Surface Projection, Sensor Analysis, Laser Triangulation, Real-Time Data Processing, Precision Farming, 3-D Point Cloud Processing, 3-D Shape Analysis, Elliptic Fourier Descriptors.

Zusammenfassung

Die automatisierte Erkennung, Ortsbestimmung und Klassifizierung von Objekten in einer ungeordneten Umgebung mit Überlappungen stellen seit den Anfängen der digitalen Bildverarbeitung und der automatisierten Produktion eine große Herausforderung dar. Eine Vielzahl von Lösungen und Systemen wurde entworfen, realisiert und geprüft. Die stetige Steigerung der Rechnerleistungen und Fortschritte bei der 3-D-Sensorik eröffnen heute Möglichkeiten bei der Detektion von Einzelpflanzen in Echtzeit.

Die Hauptfragestellung der vorliegenden Arbeit beruht auf der Annahme, dass jede Pflanze anhand ihrer Blattform zu identifizieren ist: Wie kann ein zuverlässiger Pflanzenerkennungsalgorithmus entwickelt werden, der auf einer Datenbank gescannter Blätter basiert?

Ein Blatt ist eine zweidimensionale Mannigfaltigkeit, die sich im dreidimensionalen Raum befindet. Der Lösungsweg führt folglich über eine Transformation der auf einer Wiese aufgenommenen 3-D-Daten in die 2-D-Referenz der gescannten Blätter der Formendatenbank. Dabei müssen die Datenakquisition, die Segmentierung, die Transformation und die Klassifizierung innerhalb eines Sekundenbruchteils erfolgen. Entscheidend für ein System, das die diversen Anforderungen der modernen Landwirtschaft erfüllen will, sind die Arbeitsschritte der Segmentierung und Transformation.

Der aus den digitalen Daten extrahierte Rand liefert zwei der drei wesentlichen Beschreibungsmerkmale eines Blattes: die Gestalt der Blattspreite und dessen Spreitenrand. Außerdem ist der Rand als Merkmal sehr gut zur Speicherung und Auswertung geeignet: Er belegt nur einen Bruchteil der gesamten Blattfläche, somit fällt nur eine minimale Datenmenge an. Das Problem einer zuverlässigen Segmentierung von Objekten in einer ungeordneten Umgebung wie einer Wiese ist dabei weiterhin nicht gelöst. Mit Hilfe der dritten Dimension ist es jedoch möglich, robustere und zuverlässigere Algorithmen zu entwickeln.

In der vorliegenden Arbeit wird gezeigt, dass ein Algorithmus, der auf der rekursiven Methode der kleinsten Quadrate basiert, eine höhere Berechnungszeit in einer komplexen Umgebung wie einer Wiese rechtfertigt. Das so extrahierte 3-D-Objekt wird mit den 2-D-Referenzbeispielen verglichen. Hierfür wird es in die Ebene transformiert und dabei abgeflacht, als ob es auf einen Flachbettscanner gepresst würde. Die Projektion der unregelmäßigen 3-D-Oberfläche auf eine Ebene wird mit einer Annäherung der Daten des Tiefenbilds mit Hilfe von Polynomen und der anschließenden Entfaltung jedes einzelnen Polynoms in die Ebene gelöst. Die dabei entstehenden Verzerrungen ähneln den Verzerrungen beim Aufpressen eines Blattes auf einen Flachbettscanner in hohem Maße. Die extrahierten Randbereiche der abgeflachten 3-D-Blätter sind den Referenzformen der 2-D-Daten ähnlicher als bei einer axonometrischen Projektion.

Die hier präsentierte Lösung ist noch nicht produktreif, doch stellt sie einen signifikanten Fortschritt im Bereich der Einzelpflanzenerkennung in einer Wiese dar. Das Testsystem, ausgerüstet mit einem Laser–Triangulationssensor, ortet 80 Prozent der Unkrautpflanzen auf einer Wiese. Das Ziel maximal 15 Prozent falscher Detektionen wurde jedoch verfehlt; bis zu 40 Prozent der Einzelblätter wurden falsch positiv klassifiziert. Diese falsch positive Erkennung ist vor allem auf eine zu geringe Auflösung des Sensors und die daraus resultierende niedrigere Höhengenauigkeit zurückzuführen. Weitere Forschung im Bereich der Sensortechnik und der unterstützenden Algorithmen steht somit aus. Ein autonomes, zuverlässiges, robustes Einzelpflanzenerkennungssystem wird die Unkrauterkennung und -behandlung in der zukünftigen Landwirtschaft revolutionieren.

Schlagworte: Echtzeit Einzelpflanzenerkennung, Tiefenbild Analyse, Flächen Projektion, Sensor Analyse, Laser Triangulation, Echtzeit Datenanalyse, Precision Farming, 3-D Punktewolke Analyse, 3-D Formanalyse, Elliptische Fourier Deskriptoren.

Sažetak

Automatsko prepoznavanje oblika, njihovog položaja i klasifikacija u prirodnoj okolini je izazov od prvih dana razvoja računala i računalno-podržane obrade slike. Od početka proučavanja te teme dizajniran je veliki broj različitih sustava, realiziranih i provjerenih u praksi. Stalni napredak u računalnoj tehnologiji i trodimenzionalnim senzorima otvara nove mogućnosti za prepoznavanje različitih biljaka u realnom vremenu.

Glavno pitanje u ovoj disertaciji proizlazi iz činjenice da se svaka biljka može prepoznati na temelju oblika njenih listova i glasi: Kako se može kreirati pouzdan algoritam za prepoznavanje biljaka, koji se oslanja na bazu podataka čiji su osnovni elementi skenirani listovi?

List je dvodimenzionalna površina koja se nalazi u trodimenzionalnom prostoru. Rješenje je u transformaciji trodimenzionalnih podataka dobivenih snimanjem vegetacije na livadi u domenu dvodimenzionalne baze podataka koja služi kao referentna. Sljedeći koraci obrade podataka: (prikupljanje, segmentiranje, transformacija i klasificiranje) moraju biti realizirani u par desetinki sekunde. Pri razmatranju sustava koji ispunjava potrebe automatizirane poljoprivrede, kao posebno važni koraci obrade podataka kristaliziraju se segmentiranje i transformacija podataka.

Glavni izvor podataka za klasifikaciju biljaka je pripadajući list, preciznije rečeno njegov rub. Rub lista objedinjuje dva bitna obilježja lista: njegov oblik i njegovu marginu. Rub se tretira kao osnovni izvor informacije koja se vrlo efikasno može procesirati i pohraniti u memoriji, jer predstavlja minimalnu količinu podataka koja jedinstveno predstavlja taj list. Problem pouzdane i robusne segmentacije objekata u njihovoj prirodnoj okolini kao što je livada nije definitivno riješen. Uz pomoć treće dimenzije moguće je razviti algoritme koji su robusniji i pouzdaniji.

U ovoj disertaciji pokazano je da algoritam koji se bazira na rekurzivnoj metodi najmanjih kvadrata primjenljiv, da je duže vrijeme obrade opravdano, osobito u kompleksnoj okolini kao što je livada. S tom metodom ekstrahirani 3-D objekt uspoređen je s 2-D referencom iz baze podataka. Za tu usporedbu on je transformiran u ravninu i u tom procesu je tako spljošten kao da je bio pritisnut na skener. Projekcija neregularne trodimenzionalne površine na ravninu realizirana je aproksimacijom površine uz pomoć polinoma i "odvijanjem" pojedinačnog polinoma u pravac. U tom procesu nastale deformacije vrlo su slične deformacijama nastalima prilikom skeniranja listova. Rubovi spljoštenih oblika su znatno sličniji dvodimenzionalnim referentnim oblicima u bazi podataka nego aksonometrijske projekcije istih oblika u ravninu.

U ovom tekstu prikazano rješenje nije spremno za široku upotrebu, ali predstavlja značajan korak naprijed u klasifikaciji biljaka u njihovoj prirodnoj okolini. Eksperimentalni sustav opremljen s triangulacijskim senzorom u stanju je prepoznati 80% korova na jednoj livadi. Ambiciozni cilj od maksimalno 15% pogrešnih detekcija nije ostvaren; do 40% pojedinačnih listova pogrešno je detektirano kao korov. Uzrok je niska rezolucija senzora, koja uvjetuje netočne vrijednosti u trećoj dimenziji: visini. Sljedeći korak u razvoju je jasan: poboljšana tehnika senzora i algoritmi za njihovu podršku. Samostalan, pouzdan i robustan sustav za prepoznavanje korova i njegovo odstranjivanje značit će revoluciju u poljoprivredi u budućnosti.

Ključne riječi: Automatsko prepoznavanje vrsta biljki u realnom vremenu, obrada dubinske matrice, projekcija površina, analiza senzora, laserska triangulacija, analiza podataka u realnom vremenu, precision farming, obrada 3-D oblaka točaka, 3-D analiza oblika, eliptični Fourier odrednici

Beware of programmers who carry screwdrivers.
-Leonard Brandwein-

Part I

Introduction

Chapter 1

General Thoughts and Motivation

(a) (b)

Figure 1.1: Strawberry: (a) Whole Plant, (b) Represents Fruit Detail.

Within the last decade the performance of spatial data acquisition devices has increased to a level where, in a split second, several gigabytes of observation data can be acquired at once. With the increased performance of computer vision and 3-D measurement devices, processing time and complexity of large amounts of data have increased as well. The employment of the human work force for data processing and analysis is expensive, and has limited capabilities. To transform *data* to *information,* automation procedures for segmentation, labeling and interpretation of acquired data are necessary to support human decision makers in generalization and information classification. In computer vision there is a distinction between *range images* and *intensity images*; see Besl and Jain (1988). Range images contain explicit information whereas *picture elements* (*pixels*)[1] are distances from one defined point on the sensor: the pixels are spatial information. In contrast to the range images, intensity images contain implicit information; in this case image elements are the projection of radiation reflections onto the image plane or sensor. The pixels are distinguished by their intensity differences in parameters such as colors or gray values. Humans can interpret both image types without effort. The human brain is a well-trained, neural network pattern recognition machine that can extract the contextual information needed from partially and completely visible objects. This capability is the result of sustained training during the evolution process.

For example, Figure 1.1 shows a strawberry plant, (a) and a detail of the image which has been extracted and distorted by a filter, (b). A human brain will associate the distorted image with a strawberry fruit. It can recall all context information of the experience "database". A human being is able to recall and cross-connect memories and events and derive new information out of it, such as a conclusion. Currently there is no computer-aided system that performs as a brain, especially not as well as the human brain. In certain situations however, specialized systems outperform humans: for example in ordered environments such as high-speed production lines and money sorting machines. These systems are designed for a particular task and cannot perform any other: they are not adaptive.

To design an adaptive system, several problems have to be solved: identification, localization, and manipulation problems, to mention the most challenging ones. Recognizing strawberry fruits in a strawberry field is an example of *the identification* problem, whereas obtaining the position of the fruit on a field is a typical localization example; harvesting fruits belongs to the area of manipulation.

The identification problem is the main research focus of Artificial Intelligence (AI) studies. AI is one of the more recent disciplines of science and engineering. A large number of scientific fields are fused in the subject

[1] "In digital imaging, a pixel (or picture element) is a single point in a raster image. The pixel is the smallest addressable screen element; it is the smallest unit of a picture that can be controlled", Dummer et al. (1999). Pixels, if used as an unit, will be denoted as [px].

of AI; currently almost every engineering or scientific discipline intersects with it. Russell and Norvig (2009) summarized the definition of AI in the following four groups:

Thinking Humanly	**Thinking Rationally**
Haugeland, Bellman	Charniak and McDermott, Winston
Acting Humanly	**Acting Rationally**
Kurzweil, Richt and Knight	Poole et al., Nilsson

Although AI as a science is not the focus of this thesis, it touches on most of the topics and methods elaborated here: data acquisition, segmentation, training, classification and recognition. No topic of study is the exclusive preserve of a particular discipline; they invariably overlap two or more disciplines. The subject matter of AI thus found in other science disciplines as well and the interaction of such subject matter is bidirectional between disciplines. Similarly, geodesy and remote sensing also gather selected subject matter from many other disciplines to their research area.

The mission statement for the thesis is stated:

> Describe methods for real-time (RT) classification of plant species in grasslands based on 3-D point clouds and software building blocks, for a system that is able to assist or replace the human workforce in the plant cultivation process

The processing of intensity images has one major disadvantage: the scenery which is actually three-dimensional, is projected to the sensor plane and becomes a two-dimensional, color-coded image. The relationship between pixels, or measurement points, lose one important feature as a result of projection: *depth*. Especially in cluttered and/or low contrast environments like grasslands, where the dominating color is green, the clarity of the resulting images suffer because of low contrast. Accordingly, object extraction is computationally intensive because several iterations are necessary to group pixels and label them to become one single object; such a solution is shown in Gebhardt (2007).

When processing range images, the data obviates the need for iterative searching and component labeling: the "pixel" is three dimensional, which enables computation of direct relationships between the points, by means of angles and distances. So, in 3-D data the edges are not modeled as intensity transitions; they are explicitly enclosed in the point cloud itself. The points which are organized in triangulated networks, allow robust surface analysis: see Wilke (2002); Borkowski (2004). Merging the points to a surface such as a triangulated network or polygonal meshes, is almost trivial if the boundaries of disjoint objects are determined. Considering this fact more closely, the range images might represent more suitable data sources of information for object recognition within complex and changing environments; see Šeatović (2008).

Modern measurement devices are able to acquire high-resolution, high quality, low noise and multi-spectral data at a high data rate. Smart cameras can process high-resolution frames at rates up to 35 kHz Sick (SICK Ranger E: see 2013), and 2-D and 3-D laser scanners can acquire 508 000 points in one second (Z+F, 2010); the performance of acquisition devices is expected to double within the next five years. Consider a smart camera device with a frame rate of 30 kHz that will collect approximately 44 GiB[2] of data within one second. Such an extreme data rate also requires extreme processing power for the transition from *data* to *information*.

When observing the shape of an object, which was acquired in the 3-D point cloud using a high resolution 3-D sensor, the observed point size is $0.1\,mm^2$. At this resolution a leaf of size $200\,cm^2$ is represented by 2 000 000 points, which results in a 45.8 MiB memory footprint of the leaf. Each point in the cloud is considered as a triplet $\{x, y, z\}$ of IEEE 754 single precision, floating point values. The analysis of a $1\,m^2$ patch with variable number of objects is a challenge, so the requirement for data reduction is obvious. In computer vision such a problem is a part of shape analysis, where a shape is described by its boundary[3]; see Nixon and Aguado (2008, pages 281,282). By analyzing a shape boundary, the data size reduction is better than 90%:

> A "smooth" or simple leaf boundary is a subset which includes only 2% of all points acquired. This subset increases up to 7% for lobated or similar shapes. Taking account of the worst case where a boundary consists of 10% of object points, the data volume is reduced by 90% without information loss. The memory footprint of $200\,cm^2$ decreases from 45.8 MiB to 4.6 MiB.

The *reduction of complexity* is indirectly the major issue and focus of this research. On one hand, the analysis of three-dimensional point clouds allows straightforward localization and computation of edges, but on the other, it contains much more information than is necessary to classify the shape. A solution has to be found that allows for very fast and robust object extraction out of the three-dimensional data. The aim is to find a robust and fast procedure to transform the 3-D objects into plain shapes, using reliable and fast transformation and then to classify the shapes using performance-proven procedures; procedures like kernel-based methods or neural

[2] $1\,KiB = 2^{10}\,byte$, $1\,MiB = 2^{20}\,byte$, $1\,GiB = 2^{30}\,bytes\cdots$

[3] *Shape* analysis should not be confused with the *area* analysis where all points of one object will be analyzed.

networks, just to mention two. The *second goal* is to show that the extraction and classification of objects out of point clouds within real-time constraints, is more reliable and flexible than when *vision (2-D) systems* are used. A system for recognition and treatment of plants in grassland has been implemented as part of the *SmartWeeder* project; see Šeatović and Grüninger (2007); Šeatović (2008); Šeatović et al. (2010). The system developed in the SmartWeeder project was used as the experimental platform for evaluation purposes. The following features are of interest and can be extracted out of a 3-D point cloud:

Shape: The *leaf's boundary*; the geometrical description of the leaf.

Surface: Parametrized on regular grid (u, v) out of point cloud data: $\mathcal{S}(x, y, z) \subset \mathbb{R}^3$ $x = x(u, v)$, $y = y(u, v)$, $z = z(u, v)$, $u, v \in \mathbb{R}$, $u_{n+1} - u_n = v_{n+1} - v_n = const.$

Size: Metrics of the surface: area.

Orientation: Orientation of the leaf, considering its origin like plant root or stipe.

Location: Position of the leaf in the given coordinate system. The mass centroid of the surface is defined as leaf's position.

These features can be used for the automatic plant classification when they are unique to a particular plant species. The *shape* and *size* of the leaves and their *orientation* are the result of many influences. The plants change their shape, color or both during the development process; also, the environment causes changes to plants. Following this train of thought for a single leaf: intense rain, disease or other physical influences can change the structure of the leaf.

The leaf's location and form can be altered to a similar, but significantly different object in a new spatial position, due many influences. In addition, the spatial distribution of the plants in grassland does not follow any pattern that can be described by a simple algorithm; this prevents the use of comprehensive prior knowledge. Even if prior information about any attributes were available, such attributes may have changed over the intervening time. Reliable, automatic classification of the plants must be "resistant" to alterations; this "resistance" is defined as an *invariant.* Hence, the stable features should be extracted and analyzed.

The science of botany is focused on the development of plants and their influence on the environment and fauna. Major shape change of a plant is caused by its growth (development). For a comprehensive introduction to, and in-depth information on flora, see "Strassburger: Lehrbuch der Botanik" by Bresinsky et al. (2008).

Three main factors influence plant development: *light*, *temperature* and *water* (Bresinsky et al., 2008, pages 378, 467 and 482); minerals in the ground can also significantly affect plant development. Furthermore, plant growth varies through the seasons and is characteristic for each plant species, defined by their genetic code (Bresinsky et al., 2008, 970 ff.). Table 1.1 lists the most significant environmental factors and their corresponding modifying influences on a plant's features. In addition, the last column of the table gives the expected duration of the influencing factor. Besides the influencing factors listed in Table 1.1, grasslands are exposed to a multiplicity of other secondary influencing factors.

Factor	Shape	Size	Orientation	Position	Duration
Growth	×	×	×	×	Continuous
Sun	×	×	×	×	Continuous
Humidity	×	×			Short Term
Wind	×		×	×	Temporary
Fauna, non-destructive	×	×	×	×	Temporary
Fauna, destructive	×	×	×	×	Final
Disease	×	×			Final
Season	×	×	×	×	Continuous

Table 1.1: Influences on Features Selected for Analysis and Automatic Classification.

Chapter 2

Problem Description

The human ability to learn and recognize an object visually within a very short period of time is a heritage of evolution. No machine can perform as well as a human in object recognition. The procedure of recognizing objects visually is denoted as *vision-based recognition*. How can a machine successfully complete a vision-based recognition task that can match the performance of an *untrained human*?

Note that there is a size limit to the visual recognition capability of humans: some plants or objects differ from each other in microscopic features that are only visible when using an electronic microscope. In these cases even well-educated and experienced experts can only partially classify such plants or objects visually; for example to determine the plant family. Botanists collect, classify and preserve dried specimens in plant herbaria. One grassland can contain up to 1 500 different species. A trained botanist is able to recognize the plants at a glance. Also, not so highly qualified personnel can distinguish a small set of plants after only a brief training session.

For example: seasonal workers can be trained in weed treatment. The plant species can be determined through the observation of the plant as a whole or its parts: stem, roots, flowers and leaves. Not every plant species possesses all the parts just mentioned; plant detection is therefore limited to the subset of plants that posses discernible leaves. The initial question is: what are suitable methods to acquire the data that describes the leaf or leaf part, so that a unique classification of the plant is possible?

Computer vision solutions, based on the digital camera images (intensity images), require an ideal pose[1] of the sensor relative to the object, to function reliably. Since leaf orientation in space is random, it is necessary to have a sensor that can cover at least a range of orientation angles. For optimal results the surface normals should not exceed an elevation angle of 75°; the leaf azimuth should be covered completely. A 3-D sensor meets these requirements: it delivers the necessary data that resolves the pose problem of the vision system to certain extent, but in the process a few other problems are created, notably:

- The *sensor selection* problem. Which sensor suits for 3-D data acquisition in a cluttered environment?
- The *segmentation* problem. How to extract a single leaf surface out of 3-D point clouds?
- The *feature selection* problem. Which features of plants are characteristic and can be acquired by a sensor?
- The *classification* problem. Is it possible to classify a wide range of plants based just on the selected features?

The problems listed above consider data in a local or sensor coordinate system. The question arises whether such a system that operates locally, is a suitable replacement for the human workforce; the answer is obviously not. Plant recognition and the localization issue can considered in isolation for following two reasons:

1. A solution proposal in this thesis aims for immediate plant treatment. A local coordinate system fuses a sensor and the treatment device coordinate systems. Hence, plant locations are only valid temporarily, until the treatment device reaches the plant location. After a treatment, a plant position neglected.
2. Coordinate transformation from a local to the global coordinate system is a research issue elaborated on in many publications. Determining the optimal transformation parameters is the research focus of many institutions which work in fields such as precision farming, autonomous navigation and mobile mapping devices. In this thesis is assumed that existing sensors and algorithms can, on demand, be included to extend the systems developed in the derivation of the solution proposed in this thesis, so as to provide global coordinates of classified plants.

[1] The camera pose is the relative position and orientation of the camera to the object: The extrinsic camera parameters.

For convenience, a plant position in a conventional geodetic coordinate system is thus considered as derived information, and not as a major problem that has to be systematically analyzed in this thesis. The issue is also touched on in Chapter 7.

2.1 Sensor And Resolution - the sensor selection problem

Feature	Value / Constrain
Acquisition Speed	$1.4 \geqslant v_a \geqslant 2.8\, m^2 s^{-1}$
Resolution	$1\, mm^2 \leqq$
Distance Accuracy	$\sigma \leqslant 0.7\, mm$
Area: width, height	$\geqslant 1.0\, m$, $1\, m$
NIR	Required
Band Pass Filters Exist	Required
Intensity Image Acquisition Possible	Optional

Table 2.1: Sensor Requirements.

The paper by Šeatović (2008) shows that in order to achieve reliable segmentation results at a moderate forward vehicle speed, the minimum resolution a sensor must deliver is a 3-D data point size no larger than $2\, mm^3$. In addition, the height resolution must be less than $1\, mm$. The sensor must be able to produce such high-resolution and reliable data.

The resolution requirements a sensor has to meet are therefore fairly strict, and currently there are very few sensors on the market that meet these requirements. Until recently, suitable sensors were limited to those that were designed for indoor use; of late however, suitable outdoor sensors have appeared on the market, but they are bulky, heavy and expensive at this stage; the Zoeller (2012) is an example.

One has to choose between highly reliable, mostly very expensive, robust, high-performance scanners as used in geodetic applications, or one has to modify or improve indoor sensing devices for one, dedicated purpose. Table 2.1 shows the requirements that such a sensor must fulfill. Considering real-time constraints, the raw sensor data should be:

low-noise: see the accuracy requirement.

unique: there should be no overlapping or redundancy in the location of measurements, that is, each position is measured only once.

sequential: observations should arrive sequentially in time. Measurements must arrive with the same time relation between consecutive samples. No changes in the order of arrival may take place until the target system has received the data.

complete: there should be no areas in the sequence that have been missed.

Omitted redundancy decreases the reliability of the sensor and the system, but it increases its acquisition speed. There is a choice between high performance and lower reliability or lower performance and higher reliability; in this context the word 'performance' is used to describe the *measurement speed* of the system. The importance of the speed of processing and the time frame within which it has to fit is discussed in Section 8.4.

2.1.1 The Pose

The intensity image data generally suffers under conditions where the positioning of the sensor relative to the object, is not optimal; the resulting shape of the object in image space, will be detrimentally affected. There is a simple, but partial solution to this problem: installation of multiple camera systems that observe the common area simultaneously. It solves the problem only partly; although the hemispheric setup of the cameras could compensate for the pose problem, there would be a significant increase in system complexity.

From the color image in Figure 2.1 one can see that the green-on-green situation actually represents a major challenge for segmentation algorithms; the low contrast area is indicated by a magenta color in Figure 2.1b. The edges were extracted exactly in the areas where the contrast is the highest, but in low contrast areas no edges are detected. The weak contrast areas are indicated as magenta shadows.

In Figure 2.1 the major error sources of a vision system are shown. The error letters given below correspond to letters shown on the figure:

(a) (b) (c)

Figure 2.1: The intensity image of Rumex Obstusifolius L and edge detection image. The error sources caused by the camera: Ill-pose of the sensor (A), the partly visible leaves (B), the bending of the leaf surface (C) and the leaf separation through the blade of grass (D).

A Ill-pose of the sensor is a major cause of shape aberration. This effect can be partially modeled by machine learning algorithms; if however, the false classifications are substantial, application of the algorithm applied to data acquired with poor positioning can quite easily result in an elliptical leaf being recorded as circular.

B If a partially visible shape should be analyzed, the situation becomes even more complex if the shape boundary is the only data source. In Figure 2.1c, the resulting edge detection, based on a color image, is shown; even a human eye will fail to recognize single leaves in Figure 2.1c.

C Leaves are not plane objects, so the surface of a leaf can be arbitrarily deformed in the space. The central projection of such a surface on a sensor plane does not correspond to the shape retrieved from the herbaria or a physically flattened leaf. Classification training might not succeed, since there is no separation between the classes.

D From an observer's point of view, a single grass blade can separate one leaf into two disjoint surfaces. Such a separation is best seen in Figure 2.1c where the edges are already extracted. Using only the intensity image, there is no reliable algorithm to determine which is the blade that causes the split and which are the leaf's parts, separated by the grass blade.

This leads to the next problem elaborated: the segmentation problem.

2.2 Feature Extraction - the segmentation problem

When comparing the intensity image processing containing 2-D data with range or depth image processing containing 3-D data, the additional dimension increases the number of computational operations by at least an order of magnitude. The increase in processing complexity results in a processing-power hunger on one side and requires additional effort from the segmentation algorithms on the other. The additional effort consists of a one-ring neighbor analysis on 3-D surfaces, a 3-D space curve analysis, and connected component labeling. The leaf is two-manifold in $\mathbb{R}^3$, hence the complexity of the problem can be reduced to $\mathbb{R}^2$. Leaf extraction out of the 3-D point clouds is covered in this thesis, since reliable segmentation is the major issue to be solved: Section 9.3 covers recursive filtering and compares its results with standard image processing filters. Extracted 3-D shapes are flattened and projected onto a plane: this is the final processing step. Feature extraction is the final step in complexity reduction: the 1-D border represents the 3-D surface as its unique feature. Analysis of the problem and its solution is proposed in Section 3.3.

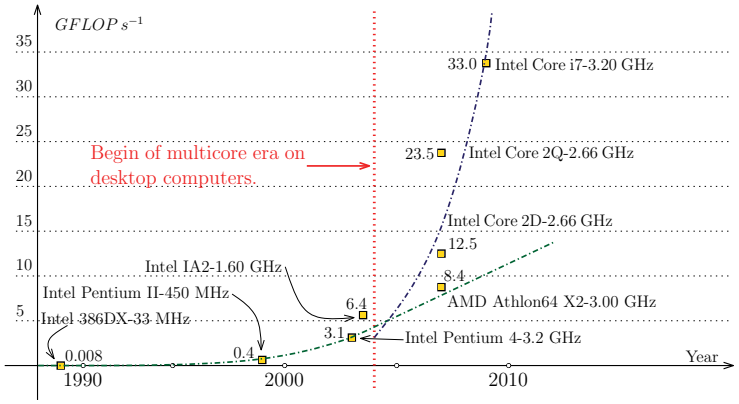


Figure 2.2: Desktop Computers: Theoretical CPU Performance Since 1989.

2.3 Workforce Replacement - the feature selection and classification problem

Goal	MER	Recognition
Recognize 100% visible leaf.	$\leqslant$15%	$\geqslant$90%
Recognize 80% visible leaf.	$\leqslant$25%	$\geqslant$80%
Recognize 50% visible leaf.	$\leqslant$30%	$\geqslant$65%

Table 2.2: Recognition Goal.

Schuster et al. (2007); Gerhards and Christensen (2003) show the performance of a system in a cultivated environment: a sugar root field. Certain types of weed are successfully recognized using an automatic procedure that outperforms humans. Prior information plays a key role in recognition, but the success of classification is based on the image processing support.

See Table 2.2 for an outline of the goals of a treatment system. A vision-based algorithm that achieves a 90% detection rate, needs approximately 45 seconds to process an image with a resolution of 0.36 mm^2, see Gebhardt (2007). A human can walk comfortably at about 4 km/h, (~1 ms^{-1}); it is therefore possible to walk at this pace and simultaneously detect and treat plants. The next logical improvement to recent achievements is a system that can keep pace with humans and is as reliable or better than an intensity sensor based solution.

The maximum error rate *(MER)* represents the maximum percentage of false positive classifications. The column *Recognition* denotes the percentage of correctly classified shapes. Natural grassland is a cluttered, non-uniform environment, and the probability that the acquired plant leaf is only partially visible, increases with vegetation growth. Therefore it is necessary to determine the time point in the season when automated plant recognition has the largest chance of succeeding. The analysis of the natural meadow data provides such information; the success rates are shown in Part IV. The vehicle prototype used for the experiments shows one possible application for the weed treatment.

2.4 Real Time Performance

The demanding requirements of the sensor technology and the amount of data that must be processed creates a challenging problem for real-time (RT) systems. To efficiently complete visual recognition based on three dimensional data in real-time, the processing must be completed in a predefined time slot. It must be *deterministic*[2] and consistent. There are two real-time system types:

1. a *hard* real-time limit where the processor must be able to fulfill its task in a strict time slot.
2. a *soft* real-time limit where the time constraints are less rigorous.

Relevant examples of hard RT systems are the control of a rocket engine or nuclear power plant. MP3-encoding of music stream or DVD movies are soft real-time tasks. Flora classification and treatment are certainly less critical than an emergency nuclear reactor shutdown, but it also demands several hard-RT tasks and a minor

[2] "A system is deterministic if, for each possible state and each set of inputs, a unique set of outputs and next state of the system can be determined.", Laplante (2004).

count of soft RT tasks to be performed. Sensor control and navigation system control are hard-RT tasks. In data processing: feature extraction, and localization belong to soft RT tasks. The problem description for the algorithms part is summarized in following questions:

- Which filters must be applied to remove or minimize noise in the data?
- Due to RT constraints, which filters are deterministic and applicable?
- Is it possible to complete a data-processing chain at the required speed?

Strict demands on hard-RT systems and their deterministic behavior, force real-time operating systems to switch off many modern processor features such as hyper threading and out-of-order execution, just to mention two. These features though, *are* the computing power of modern processors. Computer architecture provides high-performance computing using only parallel and optimized data processing approaches. Figure 2.2 depicts the performance of personal computer processors since the late 1980s; from the graph it is clear a major breakthrough occurred around 2004 when the first multi-core processors came onto the market. The performance increase is obvious. The estimation of desktop processor performance is based on synthetic benchmarks using Lapack (2012) (yellow squares). The figure shows a coarse overview of current CPUs on the market. Although this list is not complete or absolutely accurate, it does show the advance of the computing power on desktop CPUs. The current improvement rate of processing abilities of multi-core processors supports the decision to implement algorithms on general purpose hardware and neglect complex and demanding hard-RT systems. Keeping the performance trend line in mind, there is no justification for the design of specific hardware components for the following two reasons:

1. Specialized hardware is generally designed for a single dedicated purpose and the development and testing of such hardware requires facilities that are rare and only available to specialists. The usage of field programmable gate arrays (FPGA) or digital signal processors (DSP) closes the gap between general purpose CPUs and specialized hardware, but because their flexibility is very limited, they are not considered.
2. The algorithms for data analysis are implemented for general purpose multi-processor personal computers: symmetric multi-processor (SMP) architectures. As opposed to specialized hardware these algorithms implemented for general purpose CPUs are, as far as possible, portable and generic.

Summary

The chapters following show a possibility to acquire, segment and analyze 3-D data to fulfill the task of single plant detection and treatment in RT. The novel approach described in the thesis, unifies multiple techniques, algorithms and algorithm modifications to enable seamless comparison between the 3-D shapes and the 2-D shapes database. Although there is an increase in processing complexity by at least one order of magnitude due to the additional dimension, the processing time does not increase significantly and, in some cases, even decreases compared to its two dimensional counterpart. The algorithms are designed and implemented as *parallel* algorithms. The immediate or real-time processing of complex scenery is fundamental to automatic and autonomous systems. The extent to which it is possible to replace human workforce in this particular task is still unknown; the problem is to increase versatility of RT processing, so that it exceeds the context of one scientific project. Considering specialized systems such as mobile mapping vehicles, production line robots and airplane autopilots, it is obvious that there are possibilities to replace humans to a certain extent. Still, in only very few cases is successful replacement complete. The methods described in this thesis might be a step towards a workforce replacement system for tasks that require adaptive machine behavior. Such a system must implement flexible and reliable machine learning (ML) procedures. The system must be simple and intuitive to operate; a user understanding the process only vaguely, should know how to interact with the system. Such intuitive interaction with the system is only possible with sophisticated encapsulation of the procedures and a reduction of interaction complexity. Hence, the machine implements behavioral patterns that model the process *and* hides the complexity of it from the end user. The approach of complexity reduction for human machine interaction has already revolutionized the interaction between operators and machines. A common example is the domination of “smart-phones” on the mobile phone market. Such successes can only be achieved with excessive use of the most recent technology and a massive reduction of the interaction steps between user and device. Such a system could consist of a simple flatbed scanner or digitizer board and an average user who has been instructed to use the system in this straight forward way:

> The user picks some samples of the plant which requires to be treated and puts it on a flatbed scanner. The system scans the plant species and trains the classifier; the blue path in Figure 2.3. The training data is transferred to the system in the field, the machine starts autonomously to scan preselected area of the grassland, recognizes the species and treats the identified weeds according to the user’s settings; the red path in Figure 2.3.

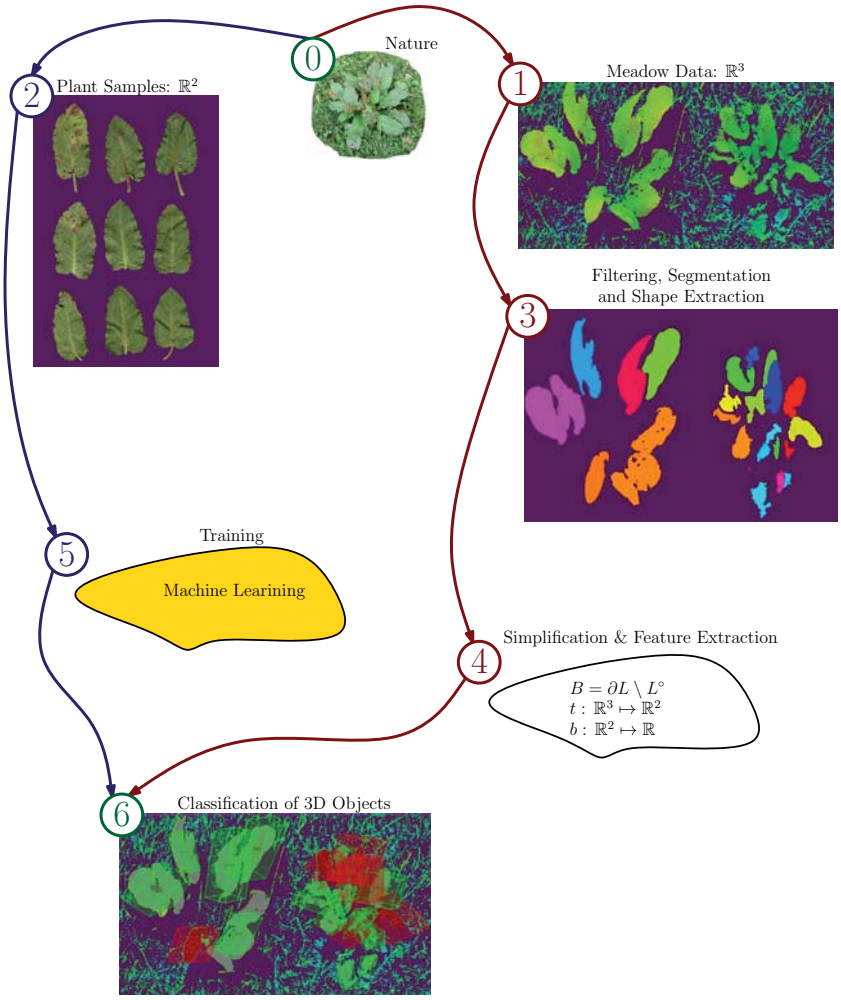


Figure 2.3: The Processing Steps of Proposed System for Automatic Plant Detection.

Considering the difficulties of the vision system shown in Figure 2.1, the 3-D approach should significantly outperform 2-D systems, because for 3-D systems the information necessary to discover object boundaries, is enclosed in the third dimension and does not need to be derived. Hence the object boundaries need not be computed implicitly. If the additional complexity in the data is compensated for by sophisticated algorithm design and implementation, such as recursive filtering methods for robust data screening, then the solution can potentially be used for solving the problem of single plant detection in natural environments.

To achieve this goal, the processing sequence of the range image has to be developed so that the unique identifiers[3] for the each plant species are extracted out of 3-D data and compared with the features of those extracted from leaves scanned on a flatbed. Chapter 3 will elaborate on the basic concepts covered here: the requirements of the feature extraction algorithm, leaf morphology, the principles of boundary extraction, and elliptic Fourier descriptors. This is followed by an explanation of 3-D point cloud processing necessities, and recursive filtering methods. The final section is an excerpt of machine learning algorithms. In Chapter 7: Research Projects similar research projects and their goals are described. Part III: Conceptual Model describes the proposed solution containing sensor selection, minimum resolution analysis, data model, data acquisition and data processing. Part IV: Evaluation and Experiments describes the experiments and experimental platform and evaluates the performance of the algorithms; this chapter also contains a critical review of the proposed solution and a proposal for further work, envisaged to build on the output of this research. The last chapter contains acknowledgments to people who, and institutions that, made this research possible. Figure 2.4 maps the main themes of each chapter to assist with navigating this multi-disciplinary thesis.

[3] *Unique identifiers* are also denoted as *features*, so in further text both terms will be used.

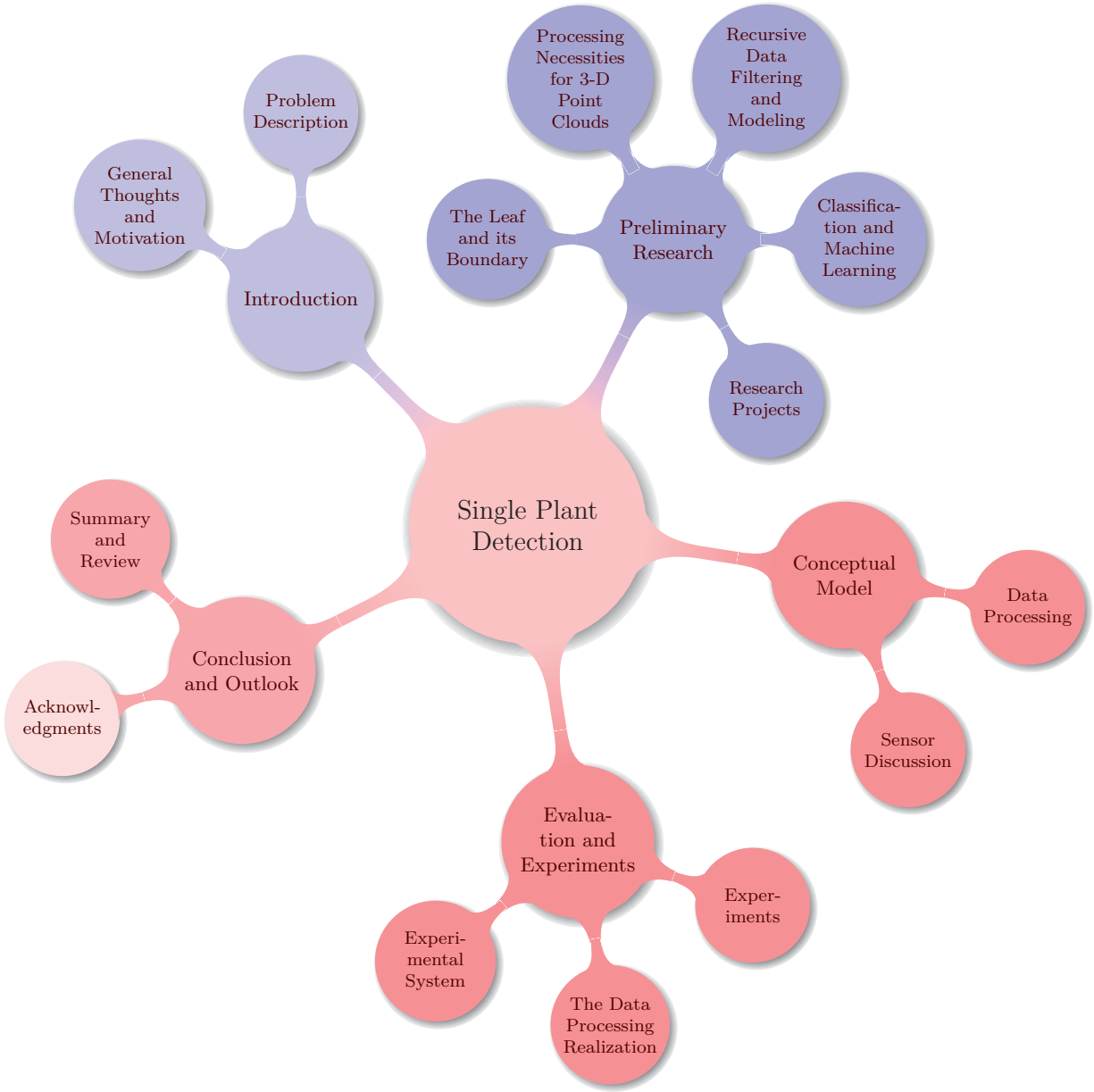


Figure 2.4: The Thesis Outline And Navigation.

Part II

Preliminary Research

Chapter 3

The Leaf and its Boundary

3.1 General Remarks About Plant Detection and Automation

To achieve industrial level automation in unordered and continuously changing environments like forests or fields, robust detection procedures need to be developed. They are key for a completely automatic treatment of diseases, weed treatment, landscape care, etc. For a wide range of applications, detection algorithms must perform similar to, or better than humans. If the algorithms perform well, they can, by means of leaf classification, be used to replace manual plant identification. Major plant detection solutions are based on:

- Vision systems/methods that analyze objects by means of intensity images acquired with a camera. These systems analyze the shape and texture of 2-D projections. Most of the systems of this type are well established, not only because of the high availability and low cost of the sensors, but also because of the long tradition in digital image analysis. Recent publications from this field are Steffen (2006); Neto et al. (2006); Gebhardt and Kuehbauch (2007); Gebhardt (2007).
- Spectral analysis systems/methods of fluorescence and induced fluorescence measurements. For induced fluorescence measurements, chlorophyll is induced with an artificial light source. After induction the radiation is measured at a preset wavelength or a wavelength interval; for example ultraviolet UV or infrared IR bands. This is still a research issue. A recent publication is by Panneton et al. (2010b,a); older work can be found in a paper by Hilton (2000).
- Hybrid systems/methods are based on the use of multi-spectral images and spatial information (2-D, 3-D). The most prominent examples of such systems are the LANDSAT and SPOT satellite systems. These remote sensing systems are state-of-the-art, multi-purpose, data acquisition devices. Bruce et al. (2002) show the typical processing procedure for multi-spectral data, using digital image processing algorithms. The airborne systems are state-of-the-art systems that acquire and provide data for mapping, surveying and monitoring purposes. An analysis example on fused LIDAR and multi-spectral data to estimate orchard health and its current growth state, is described in Viau et al. (2005). In addition, similar recent publications (Reum and Zhang, 2007; Ge et al., 2007) cover the problem of weed detection and chlorophyll analysis of multi-spectral data. This is a dense research area with many publications.
- Pure GNSS-based systems are state-of-the-art solutions in precision farming. They are actually "blind"; during the seeding session, the germ or seed position is recorded to a $\sim 3\,[cm]$ accuracy. Plant treatment in successive sessions is based on these recorded positions. In industrial crop management, protection and soil management, these methods are widely used. The performance of these methods is verified in many publications: Gan-Mor et al. (2007) analyzes the exact application just described. Major surveying vendors have off-the-shelf solutions specifically for this application (e.g. Leica (2010)).

3.2 Shape Descriptors and their Requirements

A *leaf* is a two-manifold *shape* in $\mathbb{R}^3$; the shapes (leaves) are acquired and extracted out of 3-D data. As source for shape analysis, 3-D is chosen for several reasons; the most pertinent emanate from recent research in plant identification using intensity images. They report low computing performance (long processing time) when using high-resolution images (see Steffen (2006); Gebhardt (2007); Gebhardt and Kuehbauch (2007)). Severe challenges arise from the nature of the intensity images of grasslands. One major challenge is the low contrast of the images ("green-on-green"); see Šeatović and Grüninger (2007); Šeatović (2008); Šeatović et al. (2009, 2010). Furthermore the camera pose relative to the observed object, causes additional errors due to sub-optimal

perspective. This pose issue is presumed to be negated when using a 3-D sensor since it collects observations from all three dimensions and therefore the data does not suffer loss of information due the sub-optimal pose. Although a 3-D sensor has its limitations when considering the position and orientation relative to an observed object(s), it allows a wider range of sensor positions and orientations in space. Advantages of 3-D data are elaborated in Chapter 9.

This chapter analyzes methods of distinguishing between plants through automatic leaf shape analysis. Each leaf's *description* is extracted from a 3-D point cloud. The results and conclusions of this chapter form the basis for the concepts elaborated on in Section 8.3. A definition of the subject of the research undertaken in this thesis, namely a leaf, follows:

> *"A usually green, flattened, lateral structure attached to a stem and functioning as a principal organ of photosynthesis and transpiration in most plants."*[1]

Although the leaf, as an elastic flat body has all six degrees of freedom in $\mathbb{R}^3$ and can be arbitrary deformed, once flattened, it is a near-plane surface. As water evaporates out of cells of a fresh leaf, the cell structure changes. The change of the cell structure causes shape changes which are different for each specie and depends on a variety of factors; see Bresinsky et al. (2008). If the leaf is pressed between two plane elements, it becomes flattened to a nearly completely plane shape, and maintains this state for a long period of time. Repetition of this process with different leaves creates a catalog of plant species, a herbaria, which is then later used for plant classification performed by humans. Given a herbaria as a digital database that is integrated into a computer algorithm or program, an optimal *automated* classification solution can be realized.

The shape of a 2-D object can be analyzed as a *boundary* and a *region*. It is common practice to use both analysis methods for object description and/or classification. In order to realize a robust plant detection system, it is initially necessary to determine which method solves the problem best. Not just shape, but a unique description must be found; furthermore, the extraction of the shape descriptors represents a major challenge in a natural and changing environment. For shape descriptors, the following minimal requirements are listed:

1. *Translation* invariance in $\mathbb{R}^3$.
2. *Rotation* invariance in $\mathbb{R}^3$.
3. *Scale* change invariance in $\mathbb{R}^3$.

At the very least the descriptors must be invariant to rigid transformations in $\mathbb{R}^3$. In Figure 3.1 the three transformations are shown as examples in $\mathbb{R}^2$.

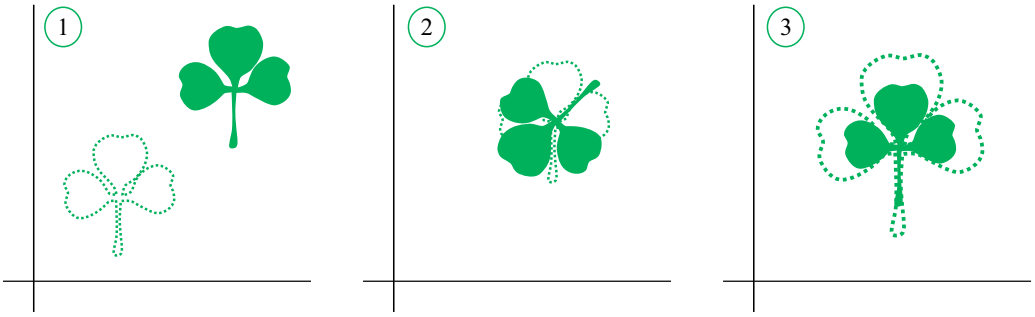


Figure 3.1: Translation (1), Rotation (2) and Scale Change (3) of a Shape.

3.2.1 Regional Descriptors

A region is a set of connected points. The nature of the connection between points is not relevant to understanding. If the region contains only one point, the analysis of such a region is trivial. The subject of an analysis are regions that contain more than one point. These regions are defined as: $R = \{p_0, \ldots, p_n\}$, $n \gg 1$, where p_n are points in one region. Regional analysis encloses several different descriptors. They can be as simple as an area, a perimeter, a circularity or a compactness, to mention just a few. Alternatively they can be as complex as statistical moments, topological descriptors or textural measures (for intensity images only); for details see Gonzalez and Woods (2006, pages 823-842). The focus of this research is on the boundary descriptors, since surface texture is not acquired by 3-D sensors.

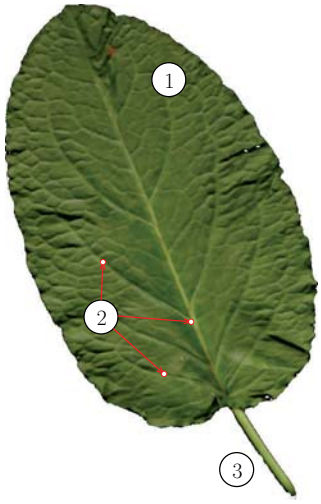


Figure 3.2: Simple Leaf Structure.

3.3 Leaf Morphology

The form diversity of natural objects like leaves is difficult to quantify. For the purposes of automatic classification it is necessary to determine which species can be classified by computer algorithms and what conditions the data must meet. With the help of boundary descriptors such as elliptic Fourier descriptors (see Section 3.4.3), the complex structure of a the leaf is reduced to a vector with features that are unique to the leaf itself. It is expected that similar leaves' boundaries produce similar feature vectors (see Neto et al. (2006)). The focus of the research here is on grassland plants. Below the characteristics of a simple leaf are described; if more detailed information is required, see Bresinsky et al. (2008).

The leaf shown in Figure 3.2 consists of:

1. Lamina (leaf blade).
2. Veins.
3. Petiole (leaf stalk).
4. Stem (not shown in the Figure).

The leaf morphology is divided into three sections: shape & arrangement, margin and venation. Figure 3.3 provides comprehensive illustrations of generalized leaf morphology[2]. The collection of leaf shapes shown is not an exhaustive representation of known flora. The set of shapes shown in Figure 3.3a does however cover the most frequently encountered plant leaves, so it is useful for establishing a preliminary proof of the concept. Classification based on the leaf shape is determined by the lamina and the petiole. With the leaf boundary in mind, the following information is contained in the leaf blade:

1. Leaf shape type (see Figure 3.3a), which belongs to the lower frequency coefficients.
2. Leaf margin type (see Figure 3.3b), which belongs to the higher frequency coefficients.

Leaf veins also contain valuable features. Veins are typical candidates for texture analysis. The focus is however on the leaf boundary, since it contains 2/3 of the information that leaf provides: leaf shape and margin. These two characteristics and their resulting descriptors are expected to be unique for *each* leaf species. The leaf shape can be described by its boundary, which is why the characteristics of the different boundary descriptors are analyzed more rigorously in the next section.

[1]The American Heritage® Dictionary of the English Language, Fourth Edition copyright ©2000 by Houghton Mifflin Company. Updated in 2009. Published by Houghton Mifflin Company. All rights reserved.

[2]Copyright Note: The image is retrieved and altered from its original source: de Bivort (2013). The original image is published under the GNU Free Documentation License. The file itself is published under the Creative Commons Attribution Share Alike 3.0 Unported. Therefore, the altered parts of the image used in this thesis are published under the same licenses as the original images.

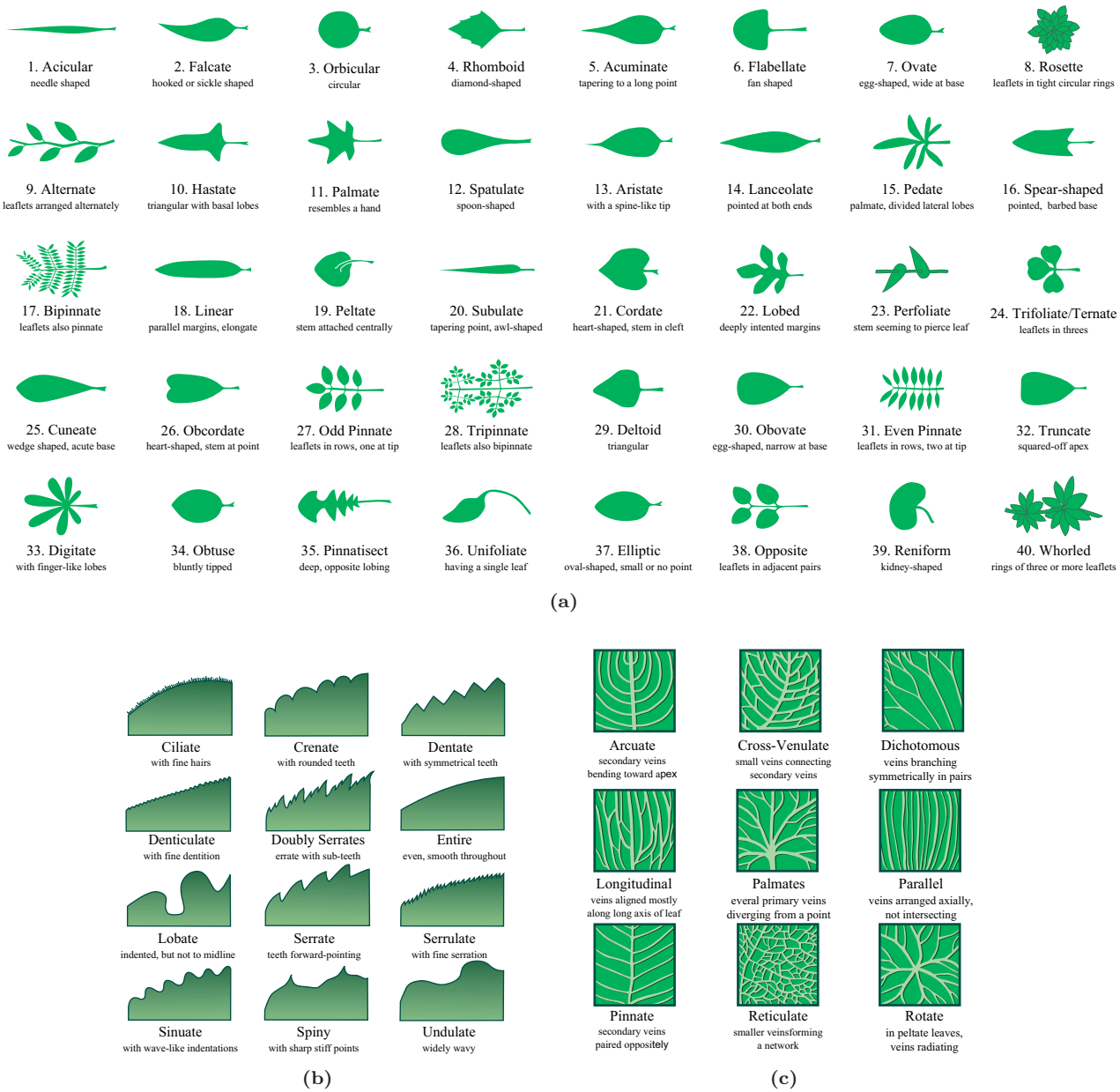


Figure 3.3: Leaf Morphology[2]: (a) Leaf Shapes and Arrangement, (b) Margin and (c) Venation. From (de Bivort, 2013).

3.4 The Boundary and Boundary Description

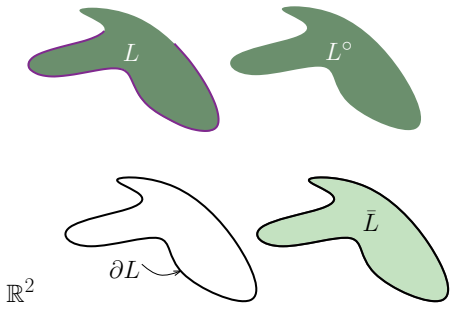


Figure 3.4: Boundary Extraction.

A leaf is a compact, connected set of points, L, in $\mathbb{R}^2$, enfolded by a simple closed curve containing no double or multiple points; see Figure 3.4, Eq. (3.2). The curve ∂L is the *boundary* and by definition it is the closure of L, subtracted from interior of L:

$$L \subseteq \mathbb{R}^2 \tag{3.1}$$

$$\partial L = \bar{L} \backslash L^{\circ} \tag{3.2}$$

A variety of boundary descriptors and their computation methods can be found in computer vision literature. With increasing computational performance, more complex boundary descriptors and analysis methods can be computed in a very short time period ($10\,ms$ <). The computations of boundary descriptors mentioned above, range from straightforward to complex computational methods. The methods require that the input data contains a closed curve without double or multiple points. For a digital binary image a boundary *pixels* extraction consists of two steps; erosion of the shape and consecutive subtraction of the eroded shape from the original one; see Eq. (3.3), Gonzalez and Woods (2006, pages 642-644). The definition in Eq. (3.2) thus becomes:

$$\partial L = \bar{L} - (\bar{L} \ominus e) \tag{3.3}$$

where e is a structure element that is used for the erosion operation. Subtraction of the eroded shape from its original is the straightforward realization of the mathematical definition of the boundary, given by Eq. (3.2). Extraction of the boundary of a shape as a digital image using Eq. (3.3) produces suboptimal results if structure element, e is not optimally chosen. For more reliable boundary extraction, edge detection methods are used. The topic of edge detection is discussed in Chapter 9.

Having extracted the boundary, it is ready to be described. The boundary description methods given in the following sections represent the basis for consecutive boundary analysis.

3.4.1 Freeman Chain Code

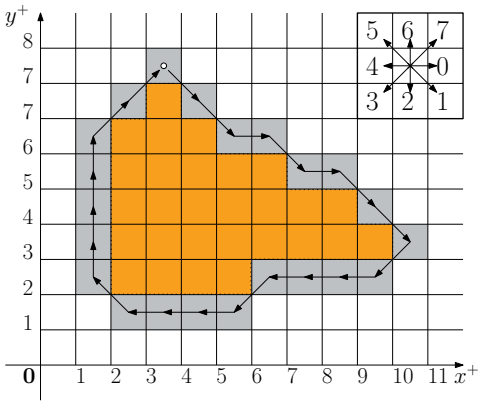


Figure 3.5: Shape and its Description Using Chain Codes.

An elementary description of discrete boundaries is given by a *chain code* known as the *Freeman chain code* (Nixon and Aguado, 2008, page 283). Figure 3.5 shows the digital shape and the boundary, encoded as chain, c. The most upper left point is chosen as the start. Traveling clockwise along the gray boundary, the chain code is extracted using the direction table shown in upper right corner in Figure 3.5. The code is determined by a clockwise search for the next neighbor. The boundary code, c is then:

c={1 2 0 0 1 0 1 1 3 4 4 4 2 4 4 4 4 5 6 6 6 6 7 7}

The Freeman code depends on the selection of the starting point and shape's orientation. Furthermore the scale of the shape has a direct influence on the chain code. The following invariants are computed:

1. Starting point invariance is achieved if the resulting chain code is shifted until the starting code reaches a minimum:
   c'={0 1 0 1 1 3 4 4 4 3 4 4 4 5 6 6 6 6 7 7 1 2}

2. Rotation invariance against rotations of $n\,\pi/2$, $n \in \mathbb{Z}$ is achieved if the code is expressed as the difference between two consecutive codes:

$$\Delta c_i' = (c_{i+1}' - c_i') \mod 7 \Rightarrow \tag{3.4}$$

$\Delta c'$={x 1 6 1 0 2 1 0 0 6 1 0 0 1 1 0 0 0 1 0 1 1}, where 'x' denotes starting point

3.4.2 Polygonal approximation

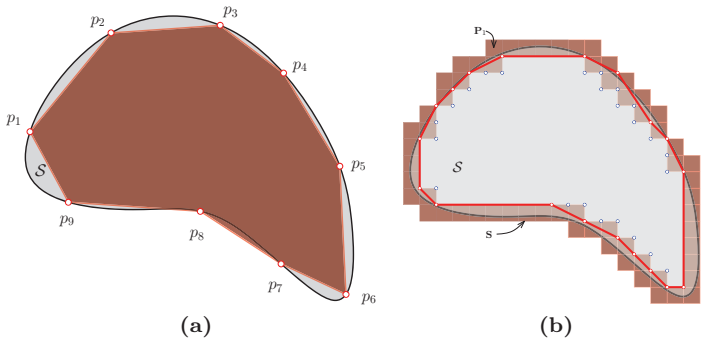


Figure 3.6: Polygonal Representation of the Boundary of a shape $\mathcal{S}$ and its discrete pendant $\mathbf{S}$.

Another boundary description method is polygonal approximation. The method is applied on intensity images where each pixel is not considered as point; it is considered as square. The polygons approximate the shape through "key points" located on the corner of the pixel square; see blue points in Figure 3.6b.

To find an optimal set of polygons involves a non-deterministic and a complex iterative search. The goal of this search is to find an optimum polygonal representation of the shape, $\mathcal{S}$ as a set of discrete points, $p_1, p_2, \ldots, p_9$ that approximates the shape optimally; see Figure 3.6a. The boundary of $\mathcal{S}$ is a continuous curve, and hence the polygonal approximation can be parametrized using fixed segment lengths, or the maximal difference between the segment length and the original curve segment. A discrete pendant of the shape, $\mathcal{S}$ denoted as $\mathbf{S}$, requires additional effort if the number of polygon points is to be reduced; see Figure 3.6b. A polygon approximates the boundary of the shape, $\mathcal{S}$ by using different criteria; one of most powerful is the minimal perimeter, or shortest perpendicular distance of the polygon segment to the original curve. The balance between the number of points and the discrepancy to the original discrete curve is a problem which is only partially solved. For more detailed insight into polygonal approximation methods consult Gonzalez and Woods (2006, page 801 ff.) and Klette and Rosenfeld (2004); Sloboda et al. (1998); Coeurjolly and Klette (2004).

The most powerful algorithm that approximates the discrete boundary of an object with the help of polygons, is the *minimum-perimeter polygon (MPP)* approximation. Besides the MPP, conceptually simpler approaches for polygonal approximation are discussed in detail on page 807 of Gonzalez and Woods (2006). The polygonal approximation can also be computed as a discrete boundary using these two methods below:

Merging techniques add points along the contour as long as the error computed by the least squares adjustment does not exceed a previously set threshold.

Splitting approach divides the boundary into segments until a specific criteria is not fulfilled; these can be length, error threshold, etc.

For shape analysis in this thesis all of the discrete points of a boundary are used; no approximation takes place before the elliptic Fourier descriptors are computed. For the analysis of the curve, which aims for reliable classification of the shape, point count reduction is contra-productive. A detailed description of the approach is given in next section. Boundaries of shapes analyzed in this thesis are considered as closed, simple curves. A simple curve has an arbitrary form: its underlying function is unknown and noise changes its characteristics; thus without prior knowledge of the underlying function, Fourier-based analysis is a safe choice for retrieving the characteristics of the curve. Neto and his group have analyzed elliptic Fourier descriptors as boundary descriptors; see Neto et al. (2006). They have achieved a high accuracy in detecting different species, using manually scanned and preprocessed leaves.

3.4.3 Elliptic Fourier Descriptors

The following explanations are based on descriptions and derivations from the book by Nixon and Aguado (Nixon and Aguado, 2008, pages: 285-311) the notation used, follows that of the book as closely as possible. The original paper on elliptic Fourier descriptors was published by Granlund (1972); the notation used in this publication is based on complex numbers. Trigonometric notation is more intuitive and is used for derivations in this thesis; see also Kuhl and Giardina (1982). Elliptic Fourier descriptors are derived from the Fourier expansion; for details see Nixon and Aguado (2008, pages: 287 ff.).

The continuous plane shape, $\mathcal{S}$ is analyzed. A Cartesian coordinate system is defined by axes x and y. The boundary of the continuous shape $\mathcal{S}$, $c(t)$ is characterized using an elliptic phasor; see Figure 3.7. There are no double or multiple points in $c(t)$; it is a simple closed curve.

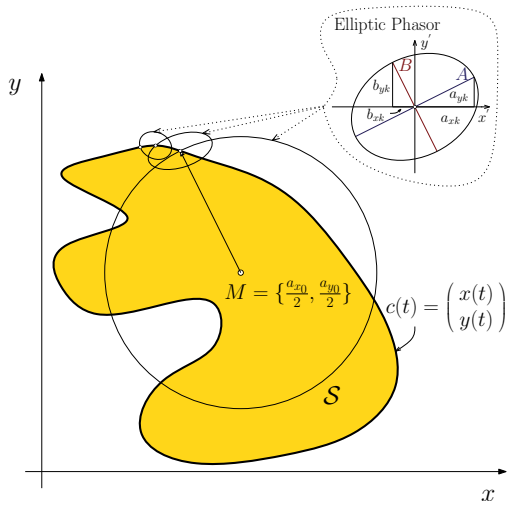


Figure 3.7: A Shape in the Cartesian Coordinate System.

Let:

$$\mathbb{R} \supset I = [a,\, b] \rightarrow \mathbb{R}^2 \tag{3.5}$$

Let period T be defined as:

$$T \in \mathbb{R}^+ \tag{3.6}$$

The closed plane curve, $c(t)$ is then:

$$c(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} \tag{3.7}$$

$$t \mapsto c(t) \tag{3.8}$$

Trigonometric coefficients are defined as:

$$a_{xk} = \frac{2}{T} \int_0^T x(t)\, \cos(k\omega t)\, dt \qquad b_{xk} = \frac{2}{T} \int_0^T x(t)\, \sin(k\omega t)\, dt \tag{3.9}$$

$$a_{yk} = \frac{2}{T} \int_0^T y(t)\, \cos(k\omega t)\, dt \qquad b_{yk} = \frac{2}{T} \int_0^T y(t)\, \sin(k\omega t)\, dt \tag{3.10}$$

The curve, $c(t)$ is expressed in trigonometric form as (see Nixon and Aguado, 2008, page 290-292 ff.):

$$c(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \underbrace{\frac{1}{2} \begin{bmatrix} a_{x0} \\ a_{y0} \end{bmatrix}}_{C} + \underbrace{\sum_{k=1}^{\infty} \begin{bmatrix} a_{xk} & b_{xk} \\ a_{yk} & b_{yk} \end{bmatrix} \begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix}}_{H} \tag{3.11}$$

$$\mathbf{A}_k = \begin{bmatrix} a_{xk} \\ a_{yk} \end{bmatrix} \tag{3.12}$$

$$\mathbf{B}_k = \begin{bmatrix} b_{xk} \\ b_{yk} \end{bmatrix} \tag{3.13}$$

The term C represents the centroid, M of the shape; see Figure 3.7. The term H represents the plain curve around M.

Invariants of Elliptic Fourier Descriptors

The coefficients of elliptic Fourier descriptors, a_{xk}, a_{yk}, b_{xk} and b_{yk} are combined in the elliptic Fourier descriptor invariants using Eq. (3.14); see Granlund (1972); Kuhl and Giardina (1982):

$$I_{EFD_k} = \frac{|\mathbf{A}_k|}{|\mathbf{A}_1|} + \frac{|\mathbf{B}_k|}{|\mathbf{B}_1|} = e_k \tag{3.14}$$

$$e_k = \frac{|\mathbf{A}_k|}{|\mathbf{A}_1|} + \frac{|\mathbf{B}_k|}{|\mathbf{B}_1|} = \frac{\sqrt{a_{xk}^2 + a_{yk}^2}}{\sqrt{a_{x1}^2 + a_{y1}^2}} + \frac{\sqrt{b_{xk}^2 + b_{yk}^2}}{\sqrt{b_{x1}^2 + b_{y1}^2}} = e_k,\ k > 1 \tag{3.15}$$

Explanation: the coefficients, a_{x0} and a_{y0} represent the curve centroid. Further, one also has to consider the coefficients, a_{x1}, a_{y1} and b_{x1}, b_{y1} which represent the first harmonic radii: $A_1 = \sqrt{a_{x1}^2 + a_{y1}^2}$ and $B_1 = \sqrt{b_{x1}^2 + b_{y1}^2}$. With help of the first harmonics radii, Granlund defines elliptic Fourier descriptor invariants by means of Eq. (3.14); see Granlund (1972).

The invariance is required for the descriptors, e_k, since the 3-D shapes are arbitrarily rotated, scaled and translated in $\mathbb{R}^3$, hence the projection of the shapes onto the plane are also affected by these three operations. Since the analyzed boundaries are plane curves, the proof is accomplished for $\mathbb{R}^2$. The invariance is proven by applying a translation, $\mathbf{v}$, rotation, R and scale, s to the plain curve, $c(t)$. It has to be proven that $|\mathbf{A}_k|$ and $|\mathbf{B}_k|$ are invariants to the operations. The proof is valid for the continuous and the discrete case.

Eq. (3.11) rewritten with all three transformations, yields:

$$\tilde{c}(t) = s\,R\,c(t) + \mathbf{v} = s\,R\,\frac{1}{2}\begin{pmatrix} a_{x0} \\ a_{y0} \end{pmatrix} + s\,R\sum_{k=1}^{\infty}\begin{bmatrix} a_{xk} & b_{xk} \\ a_{yk} & b_{yk} \end{bmatrix}\begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix} + \mathbf{v} \tag{3.16}$$

The scale and rotation alter the orientation and length of the curve. The relationship in Eq. (3.15) however, contains neither scale, $\mathbf{s}$ nor rotation, ϕ. If the continuous curves, $c(t)$ or $\tilde{c}(t)$ are considered, the number of coefficients that result from the EFD computation is infinite; practically, the coefficients are computed from a finite set of points acquired by a digital sensor, after a digital-to-analog conversion. The next section elaborates on the discrete case.

Translation Invariance

The curve representation, $\tilde{c}$ is a translation invariant if the shift of the curve by $\mathbf{v}$, does not affect the distance between two arbitrarily chosen points of $\tilde{c}(t)$. Let:

$$\begin{aligned} d :&\mathbb{R} \mapsto \mathbb{R}^2 \\ &t \mapsto c(t) + \mathbf{v},\ \mathbf{v} \in \mathbb{R}^2 (\mathbf{v} = const.) \\ &\mathbf{v} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \tilde{c}(t) = c(t) + \mathbf{v} &= \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \frac{1}{2}\begin{bmatrix} a_{x0} \\ a_{y0} \end{bmatrix} + \sum_{k=1}^{\infty}\begin{bmatrix} a_{xk} & b_{xk} \\ a_{yk} & b_{yk} \end{bmatrix}\begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \\ &= \frac{1}{2}\begin{bmatrix} a_{x0} + t_x \\ a_{y0} + t_y \end{bmatrix} + \cdots \end{aligned} \tag{3.17}$$

Conclusion: Translation affects only the centroid of the curve $c(t)$ which is the point M, it does not affect curve characteristics. The invariants, $|\mathbf{A}_k|$ and $|\mathbf{B}_k|$ depend on the coefficients for $k \geqslant 1$; hence the invariants, $e_1, e_2, \ldots, e_k$ remain unchanged by the translation, which fulfills the requirement #1.

Rotation Invariance

Using same methodology, rotation of the shape through the rotation matrix, R is observed:

$$\tilde{c}: I \to \mathbb{R}^2 \tag{3.18}$$
$$t \mapsto R \cdot c(t) \tag{3.19}$$
$$\phi \in [0, 2\pi) \tag{3.20}$$

with

$$R = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} = \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{3.21}$$

Which yields to:

$$\tilde{c}(t) = R\,c(t) = R \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = R\,\frac{1}{2} \begin{bmatrix} a_{x0} \\ a_{y0} \end{bmatrix} + R \sum_{k=1}^{\infty} \begin{bmatrix} a_{xk} & b_{xk} \\ a_{yk} & b_{yk} \end{bmatrix} \begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix} \tag{3.22}$$

The rotation of the centroid M results in curve displacement, hence translation. Translation invariance is already proven. Translating the curve at its origin, then rotating it, and then translating it back, slightly modifies the previous equation. The curve translated in its origin is $\tilde{c}_0(t)$:

$$\tilde{c}_0(t) = c(t) - M \tag{3.23}$$

$$R\,\tilde{c}_0(t) = R\,\frac{1}{2} \begin{bmatrix} 0 \\ 0 \end{bmatrix} + R \sum_{k=1}^{\infty} \underbrace{\begin{bmatrix} a_{xk} & b_{xk} \\ a_{yk} & b_{yk} \end{bmatrix}}_{E} \begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix} = \cdots \tag{3.24}$$

$$\cdots = R \sum_{k=1}^{\infty} E \begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix} = \sum_{k=1}^{\infty} R\,E \begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix} \tag{3.25}$$

Now rotation of the harmonics radii matrix, E around the origin gives:

$$\begin{aligned} R\,E &= \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} a_{xk} & b_{xk} \\ a_{yk} & b_{yk} \end{bmatrix} \\ &= \begin{bmatrix} \cos(\phi)\,a_{xk} + \sin(\phi)\,a_{yk} & \cos(\phi)\,b_{xk} + \sin(\phi)\,b_{yk} \\ -\sin(\phi)\,a_{xk} + \cos(\phi)\,a_{yk} & -\sin(\phi)\,b_{xk} + \cos(\phi)\,b_{yk} \end{bmatrix} \end{aligned} \tag{3.26}$$

Invariance is proven for $|\mathbf{A}_k|$:

$$|\mathbf{A}_k| = \sqrt{a_{xk}^2 + a_{yk}^2} \tag{3.27}$$

Rotated invariance, $\mathbf{A}_k$ is denoted as $\tilde{\mathbf{A}}_k$:

$$\tilde{a}_{xk} = \cos(\phi)\,a_{xk} + \sin(\phi)\,a_{yk} \tag{3.28}$$
$$\tilde{a}_{yk} = -\sin(\phi)\,a_{xk} + \cos(\phi)\,a_{yk} \tag{3.29}$$
$$\left|\tilde{\mathbf{A}}_k\right| = \sqrt{\tilde{a}_{xk}^2 + \tilde{a}_{yk}^2} \tag{3.30}$$

$$\left|\tilde{\mathbf{A}}_k\right| = \sqrt{\underbrace{\left(\cos(\phi)\,a_{xk} + \sin(\phi)\,a_{yk}\right)^2 + \left(-\sin(\phi)\,a_{xk} + \cos(\phi)\,a_{yk}\right)^2}_{*}} = \tag{3.31}$$

The term $*$ within the square root developed, is then:

$$\begin{aligned} * = \cos^2(\phi)\,a_{xk}^2 &+ 2\cos(\phi)\sin(\phi)\,a_{xk}\,a_{yk} + \sin^2(\phi)\,a_{yk}^2 + \cdots \\ \cdots + \sin^2(\phi)\,a_{xk}^2 &- 2\cos(\phi)\sin(\phi)\,a_{xk}\,a_{yk} + \cos^2(\phi)\,a_{yk}^2 = \cdots \end{aligned} \tag{3.32}$$

$$\cdots = \left(\cos^2(\phi) + \sin^2(\phi)\right) a_{xk}^2 + \left(\cos^2(\phi) + \sin^2(\phi)\right) a_{yk}^2 = \cdots \tag{3.33}$$

$$\cdots = a_{xk}^2 + a_{yk}^2 \Rightarrow \sqrt{a_{xk}^2 + a_{yk}^2} = |\mathbf{A}_k| \tag{3.34}$$

The proof is identical for $|\mathbf{B}_k|$. Considering elliptic Fourier coefficients, a_{xk}, a_{yk}, b_{xk} and b_{yk}, Eq. (3.24) shows that the original coefficients are altered by rotation. Eq. (3.25)-Eq. (3.34) prove that invariants, $|\mathbf{A}_k|$ and $|\mathbf{B}_k|$ remain unaffected by the rotation operation.

Conclusion: The elliptic Fourier descriptor invariants hence fulfill requirement #2.

Scale Invariance

As last step, scale invariance for elliptic Fourier descriptors is to be proven.

$$t \mapsto s\, c(t),\ s \in \mathbb{R}^+ \tag{3.35}$$

Similar to rotation invariance, where the coefficients, $\begin{bmatrix} a_{xk} & b_{xk} \\ a_{yk} & b_{yk} \end{bmatrix}$ are multiplied by rotation matrix, R, the harmonics radii are multiplied by the scale, s. Again the case where the curve, $c(t)$ is translated in its origin is considered:

$$\tilde{c}_0(t) = c(t) - M \tag{3.36}$$

$$s\,\tilde{c}_0(t) = s\,\frac{1}{2}\begin{bmatrix} 0 \\ 0 \end{bmatrix} + s\sum_{k=1}^{\infty}\begin{bmatrix} a_{xk} & b_{xk} \\ a_{yk} & b_{yk} \end{bmatrix}\begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix} \tag{3.37}$$

then:

$$s\,\tilde{c}_0(t) = \sum_{k=1}^{\infty}\begin{bmatrix} s\,a_{xk} & s\,b_{xk} \\ s\,a_{yk} & s\,b_{yk} \end{bmatrix}\begin{bmatrix} \cos(k\omega t) \\ \sin(k\omega t) \end{bmatrix} \tag{3.38}$$

The shape curve remains unchanged due the scale operation, analogous to rotation and translation. However, the *spatial* size of $\tilde{c}_0(t)$ is altered by the transformation. The resulting curve, $\tilde{c}_0(t)$ dilates/shrinks symmetrically around its center, dependent on the value of s. Dilation occurs if $s > 1$ and shrinking if $0 < s < 1$. For $s = 1$ the curve remains unchanged.

Considering the scale change influence on elliptic Fourier descriptor invariants, Eq. (3.15) becomes:

$$\frac{\sqrt{s^2a_{xk}^2 + s^2a_{yk}^2}}{\sqrt{s^2a_{x1}^2 + s^2a_{y1}^2}} + \frac{\sqrt{s^2b_{xk}^2 + s^2b_{yk}^2}}{\sqrt{s^2b_{x1}^2 + s^2b_{y1}^2}} \Rightarrow \frac{\sqrt{a_{xk}^2 + a_{yk}^2}}{\sqrt{a_{x1}^2 + a_{y1}^2}} + \frac{\sqrt{b_{xk}^2 + b_{yk}^2}}{\sqrt{b_{x1}^2 + b_{y1}^2}} = e_k \tag{3.39}$$

Conclusion: The elliptic Fourier descriptors are not affected by scale change, and hence requirement #3 is fulfilled.

Discrete Case

In signal processing theory, the process of *digitization* of an analog signal is defined as sampling and quantization. The Nyquist–Shannon sampling theorem (Shannon, 1949), states that the sampling frequency, f_s must be greater than double the bandwidth: $f_s > 2(f_{max} - f_{min})$ of sampled signal, to be able to reconstruct the signal out of the sampled points. The sampling rate determines the resolution of the *digitized* sequence of points, C. The coefficients computed by Eq. (3.15): $I_{EFDs} = \{e_2, e_3, \ldots, e_n\}$ for the shape $\mathcal{S}$ (in Figure 3.8), are shown in Figure 3.9.

For the discrete case the continuous curve, $c(t)$ is digitized by a sensor of finite resolution; Figure 3.8 shows result of such a digitization. The result of digitization is the sequence of points, C containing the coordinates of the centroids of the sensor elements (depicted as transparent red squares in the Figure 3.8). The boundary of the shape is extracted using Eq. (3.3). The number elements in the sequence C is finite, and therefore the number of elliptic Fourier descriptors is limited to that count as well. Let sequence of points C be:

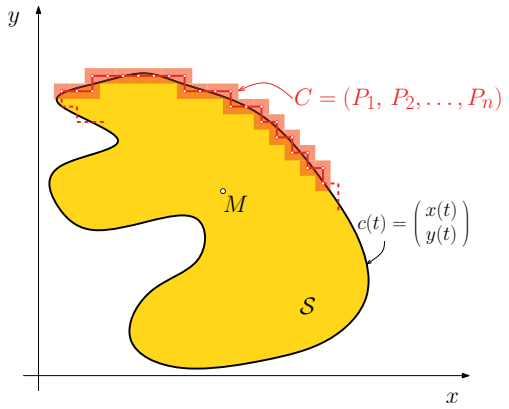


Figure 3.8: Curve $c(t)$ digitized by sensor of finite resolution.

$$C = (P_1,\, P_2, \ldots, P_i),\; i \in \mathbb{N} \tag{3.40}$$
$$P_i = (x_i,\, y_i) \tag{3.41}$$

Hence, the discrete boundary is also a closed, simple, periodic, plane curve with:

$$\begin{aligned}
\mathbb{R} \supset \quad I &= \quad [a,\, b] \to \mathbb{R}^2 && (3.42)\\
t &\mapsto \quad c(t) && (3.43)\\
c(t) &= \quad c(t+\tau),\; \forall\, t \in \mathbb{R} && (3.44)\\
c(a) &= \quad c(b) && (3.45)\\
T &\in \quad \mathbb{R}\, const. && (3.46)\\
\omega &= \quad \frac{2\pi}{T} && (3.47)\\
m &= \quad n/2 && (3.48)\\
\tau &= \quad \frac{T}{m} && (3.49)
\end{aligned}$$

Discrete coefficients are approximations of Eq. (3.9):

$$\mathrm{a}_{xk} = \frac{2}{m}\sum_{i=1}^{m} x_i \cos(k\omega i\tau) \qquad \mathrm{b}_{xk} = \frac{2}{m}\sum_{i=1}^{m} x_i \sin(k\omega i\tau) \tag{3.50}$$
$$\mathrm{a}_{yk} = \frac{2}{m}\sum_{i=1}^{m} y_i \cos(k\omega i\tau) \qquad \mathrm{b}_{yk} = \frac{2}{m}\sum_{i=1}^{m} y_i \sin(k\omega i\tau) \tag{3.51}$$

The sequence of points, C expressed by elliptic Fourier descriptors, is denoted by $\hat{\mathrm{c}}$:

$$\hat{\mathrm{c}} = \frac{1}{2}\begin{pmatrix} \mathrm{a}_{x0} \\ \mathrm{a}_{y0} \end{pmatrix} + \sum_{k=1}^{m} \begin{bmatrix} \mathrm{a}_{xk} & \mathrm{b}_{xk} \\ \mathrm{a}_{yk} & \mathrm{b}_{yk} \end{bmatrix} \begin{bmatrix} \cos(k\omega i\tau) \\ \sin(k\omega i\tau) \end{bmatrix} \tag{3.52}$$

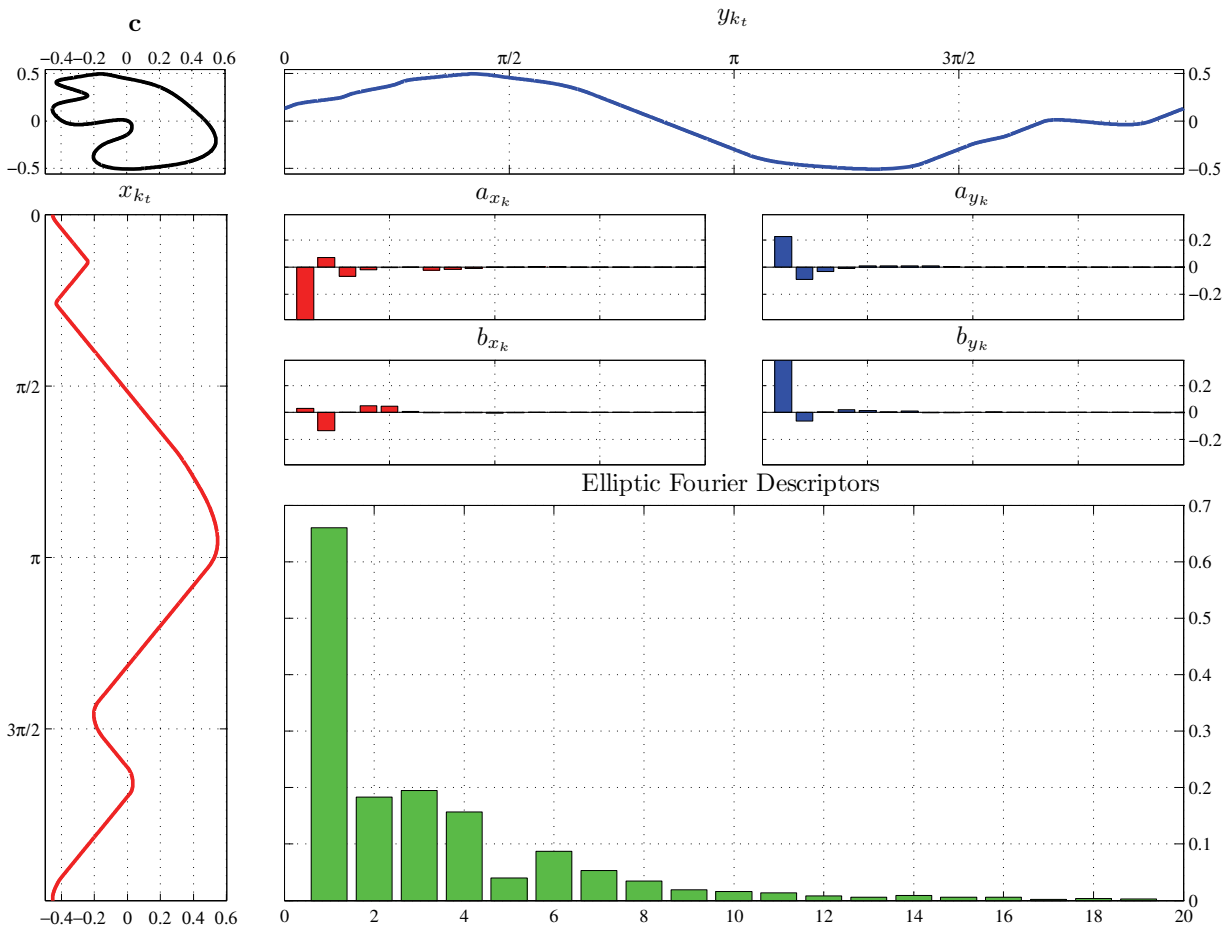


Figure 3.9: Computed Elliptic Fourier Descriptors for Shape S.

Conclusion: The digitized curve, C is a *finite* sequence of points, consisting of coordinate pairs:

$$C = \{(x_1, y_1), \ldots, (x_k, y_k)\}, \ \forall\, k \in \mathbb{N} \wedge 0 < k \ll \infty \tag{3.53}$$

The coefficients, $e_1, \ldots, e_n$ are used uniquely to describe the shape, $\mathcal{S}$. The Fourier descriptors represent unique shape descriptors; from Nixon and Aguado (2008, page 309):

> *"Fourier descriptors are one of the most popular boundary descriptions. As such, they have attracted considerable attention and there are many further aspects. We can use the descriptors for the shape recognition (Aguado et al., 1998). It is important to mention that some work has suggested that there is some ambiguity in the Fourier characterization. Thus, an alternative set of descriptors has been designed specifically to reduce ambiguities (Crimmins, 1982). However, it is well known that Fourier expansions are unique. Thus, Fourier characterization should uniquely represent a curve."*

When considering a unique boundary description method, the following questions arise:

1. How many descriptors are necessary to distinguish a set of different leaf shapes?
2. What is the minimum resolution at which a digitizer still provides sufficient data to compute all the descriptors?

These crucial questions are part of the sensor analysis and are elaborated on in Section 8.2. The next section considers the raw data: point clouds.

Chapter 4

Processing Necessities for 3-D Point Clouds

4.1 Curve and Surface

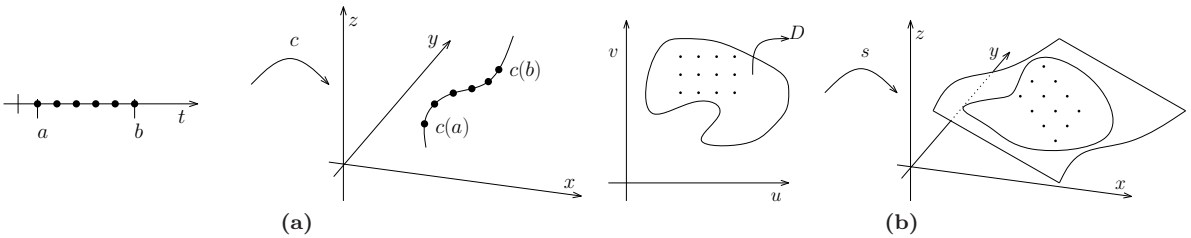


Figure 4.1: Curve and Surface Definitions.

The definitions stated here are necessary for the analysis of 3-D point clouds. Relations between the surfaces and parameter lines are elaborated in Section 4.4.

A curve (see Figure 4.1a) is defined as:

$$c : \mathbb{R} \supset I = [a,\, b] \rightarrow \mathbb{R}^n,\; n = 2,\, 3 \tag{4.1}$$

$$t \mapsto c(t) = \begin{pmatrix} x(t) \\ y(t) \\ \cdots \\ z(t) \end{pmatrix} \tag{4.2}$$

The curve sampled at a rate, Δt results in a sequence of points, depicted by black dots in Figure 4.1a. A surface patch in $\mathbb{R}^3$(see Figure 4.1b), is defined as:

$$s : \mathbb{R}^2 \supset D \rightarrow \mathbb{R}^3 \tag{4.3}$$

$$(u, v) \mapsto s(u,\, v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} \tag{4.4}$$

$$s : \mathbb{R}^2 \rightarrow \mathbb{R}^3 \tag{4.5}$$

$$(u, v) \mapsto \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} = s(u, v) \tag{4.6}$$

The surface patch is a map of an atlas. A continuous surface patch, discretizied, results in sequence of points, s_{ik}:

$$u_i = u_0 + i\Delta u,\ i = 1, \ldots, m \tag{4.7}$$

$$v_k = v_0 + k\Delta v,\ k = 1, \ldots, n \tag{4.8}$$

$$s_{ik} = \begin{pmatrix} x_{ik} \\ y_{ik} \\ z_{ik} \end{pmatrix} \tag{4.9}$$

Organization of point sequences is elaborated in the next section.

4.2 Point Clouds

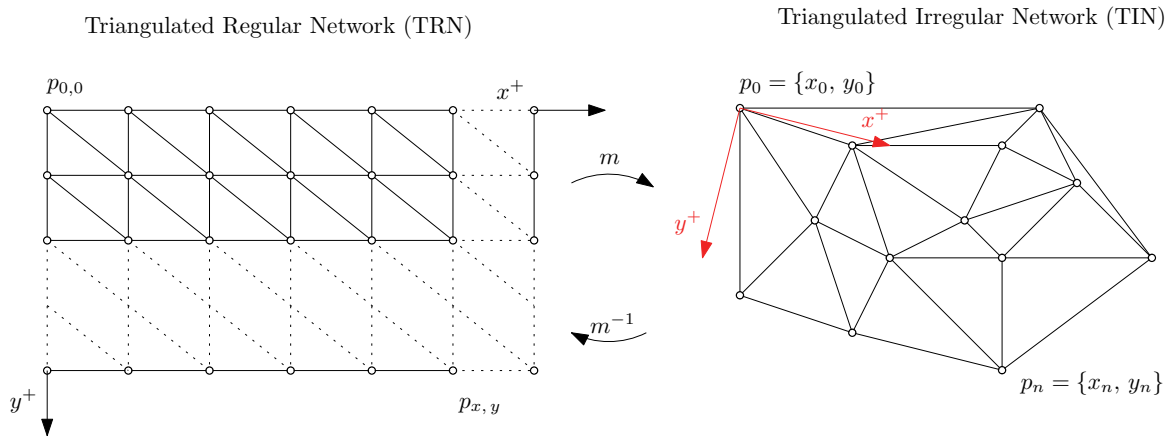


Figure 4.2: Network types. Given a TRN, the function m converts TRN to a triangulated irregular network TIN. The inverse function m^{-1} restores the TRN from the TIN data.

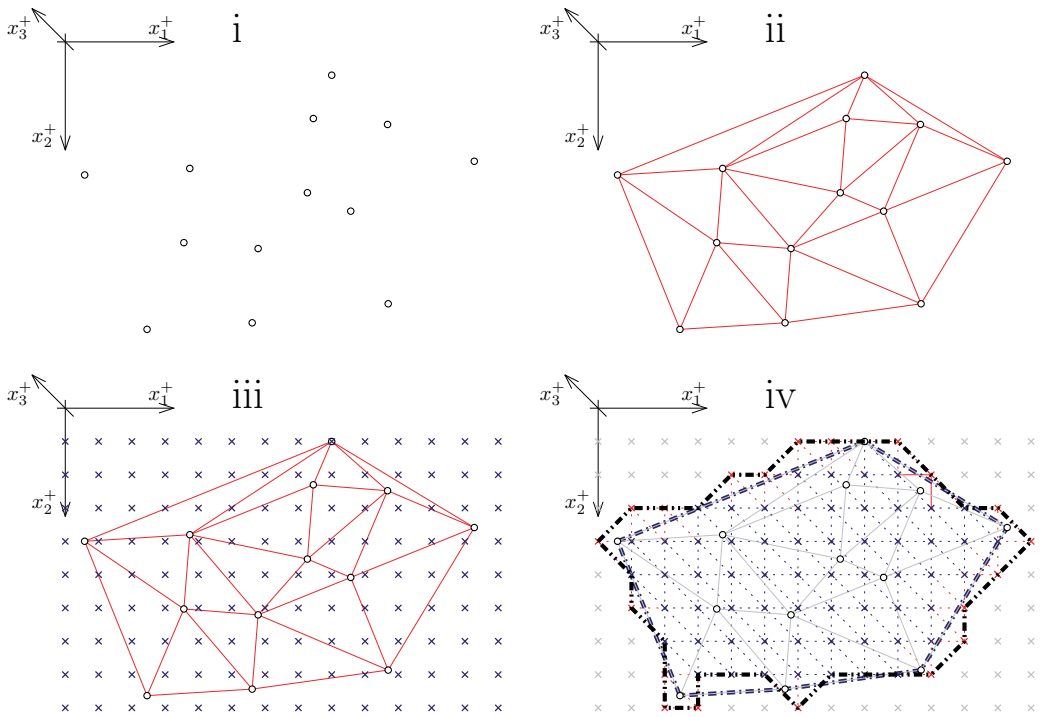


Figure 4.3: Transformation from TIN to TRN. Showing irregular triangulated network TRN (i: black circles) through the Delaunay triangulation (ii: red lines), definition of the regular grid in arbitrary density (iii: blue crosses), the subsequent interpolation using the grid results in triangulated regular network (iv: blue and red crosses, dotted lines in same colors).

A point cloud is a set of vertices, with each vertex being a vector with coordinates, $\mathbf{x}$. Besides coordinates, each vertex can additionally include information about its attributes, $\mathbf{v}$. Each vertex, $\mathbf{p}$ thus consists of its coordinate vector, $\mathbf{x}$ and its attributes, $\mathbf{v}$: $\mathbf{p} = \{\mathbf{x}, \mathbf{v}\}$. Each coordinate vector can be n-dimensional: $\mathbf{x} \in \mathbb{R}^n$, $n \in N$. The point cloud, $\mathcal{P}$ containing more than one vertex $\mathbf{p}$ is given by:

$$\mathcal{P} = \{\mathbf{p}_k \in \mathbb{R}^3 | \, k = 1, 2, \ldots, n \wedge n \geqslant 3\} \tag{4.10}$$

The attributes are arbitrary and optional additional information, which differs by data type and dimension. The attributes model, or contain physical information, which can be surface reflection, color, spectral information, etc. They are modeled as *complex* data types and are explained in Section 9.1.

More important here, is the structure of the point cloud, in particular the relation between the vertices. Network organization is distinguished by the basic geometric form that is considered as "atomic": triangulated or squared for two-dimensional networks, and cube or dodecahedron for three-dimensional networks.

Considering the distance and direction between the neighboring vertices, the organization of the network can be regular (isotropic) or irregular (anisotropic). Regular networks are also called *grids*; see Section 4.1. Examples of triangulated networks are shown in Figure 4.2.

If the connections (edges) between the vertices are regular, then the surface, $\mathbf{S}$ is a 3-D *mesh*. These meshes are distinguished as being: irregular, semiregular, highly regular or regular; see Botsch, Kobbelt, Pauly, Alliez, and Levy (2010, Chapter Mesh Data Structures and Chapter Remeshing). A basic example of the geometric interpretation of the relationship between neighboring vertices is shown in Figure 4.2. An ordered (regular) network is represented by a *triangulated regular network* (TRN). The edges are regular in each dimension of the manifold; the orientation and magnitude are constant in each dimension. The resulting mesh contains equal triangles; see Figure 4.2.

Vertices in an irregular network have disordered neighboring relationships; the distances and orientation of the edges are not equal, and the resulting mesh contains triangles of an arbitrary size. Such a network is called a Triangulated Irregular Network (TIN). If a bijective map $m : TRN \rightarrow ITN$ exists, and hence $m^{-1} : ITN \rightarrow TRN$, then a neighbor relationship analysis of the structure can be executed on both the network types TIN and TRN. Regular network analysis is favored above irregular network analysis, because of the simplicity of the algorithms, as well as their availability and efficiency.

The processing of point clouds by the analysis of TIN, is the theme of a variety of publications and text books; those that overlap with problems described in this text, are briefly mentioned. The most recent "publication" is an application programming interface (API), Point Cloud Library (PCL); see PCL (2011); Rusu (2009). This library contains the framework for a full processing pipeline for point cloud storage and processing. It is an open source project, gathering input from a diverse group of contributors.

Also, other research institutions and groups provide their own libraries for the processing and visualization of point clouds and triangle meshes:

> *"OpenMesh is a generic and efficient data structure for representing and manipulating polygonal meshes." (Kobbelt, 2012)*
>
> *"MeshLab is an open source, portable, and extensible system for the processing and editing of unstructured 3D triangular meshes." (Scopigno, 2012)*

The basic principles of processing point clouds are elaborated in Luhmann (2010, Pages 74 ff.); the book provides a comprehensive overview of the theme for the reader with a geodetic and photogrammetic background. Computer graphics is the major application field of concern when considering mesh and point cloud processing; Lengyel (2002) provides the basics of processing data from a visualization point of view. The large majority of the publications and problems have their origin in the automation of production lines and quality assurance; see Bi and Wang (2010).

4.3 Grid Processing

Grid processing is advantageous since the data can be processed as a 2-D scalar field or matrix. The majority of basic and advanced (intensity) image processing algorithms operate on grid or matrix data. In the subsections that follow, brief outlines of the basic principles pertaining to correlation, convolution and morphological operations, are given.

4.3.1 Remarks on Processing Principles Correlation and Convolution

Basic image processing operations are mainly:

Convolution: continuous case: $(f \star g)(t) \equiv \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$; discrete case $(f \star g)[n] = \sum_{m=-\infty}^{\infty} f[m]\, g[n-m]$, see Mathworld (2013a).

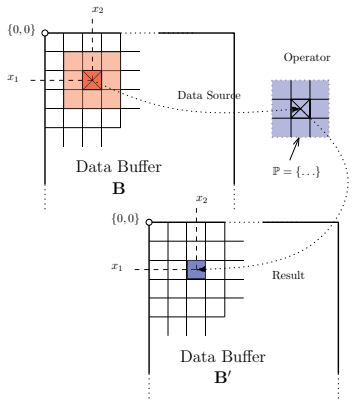


Figure 4.4: Basic Convolution Principle.

Correlation: continuous case: $(f \star g)(t) \equiv \int_{-\infty}^{\infty} \bar{f}(\tau)\, g(t+\tau)\, d\tau$; discrete case: $(f \star g)[n] = \sum_{m=-\infty}^{\infty} \bar{f}[m]\, g[n+m]$, see Mathworld (2013b).

Where f and g are continuous or discrete functions. $\bar{f}$ denotes the complex conjugate of f; for further details see Gonzalez and Woods (2006, Chapters 3 and 4). These operations are applied in the spatial or frequency domain of a digital image:

$$\mathbf{B}' = \mathbf{B} * \mathbf{O} = \sum_{(\mathbf{x}) \in \mathbf{B}} \langle \mathbf{O_x} \mid \mathbf{B} \rangle\,,\ n\mathbf{O_x} \subset \mathbf{B} \tag{4.11}$$

where $\mathbf{B}$ is the data buffer, and $\mathbf{O}$ the operator matrix. The operator matrix size is smaller than the buffer itself. $\mathbf{O_x}$ denotes the translated operator matrix at position $\mathbf{x}$. The operation can be in-place; in which case $\mathbf{B} = \mathbf{B}'$ and the content of $\mathbf{B}$ is changed. If the result of convolution is stored into a separate buffer, then $\mathbf{B} \neq \mathbf{B}'$, and the content of $\mathbf{B}$ remains unchanged. This kind of operation is illustrated in Figure 4.4. Basic image processing operators, convolution principles, frequency domain fundamentals and space domain processing fundamentals, are described in Gonzalez and Woods (2006) and Nixon and Aguado (2008).

In general object and/or pattern recognition procedures of intensity images, the following steps are included: acquisition, storing, filtering, segmentation and classification. For every step, one or more algorithms are applied.

4.3.2 Remarks on Morphological Operations

Gonzalez and Woods (2006, Page 628) states:

> *"The language of mathematical morphology is set theory. As such, morphology offers a unified and powerful approach to numerous image processing problems..."*

A digital binary image is considered as set in $\mathbb{Z}^2$, hence the TRN as shown in Figure 4.2, can also be considered as such a set. In this case the morphological operations applied to digital binary images, can be applied to TRNs without modification. The following definitions are extracted from Gonzalez and Woods (2006, Page 628 ff.):

$$TRN^2 \subseteq \mathbb{Z}^2,\ TRN^3 \subseteq \mathbb{Z}^3 \tag{4.12}$$

Let discrete sensor data deliver a point cloud in an arbitrary dimension n, then:

$$\mathcal{P}^n \subseteq \mathbb{Z}^n \,\forall\, \mathcal{P}^n = (\mathbf{p}_1, \ldots, \mathbf{p}_k),\ n,\ k \in \mathbb{N} \vee k > 3 \tag{4.13}$$

$$\mathbf{p}_k^n = (x_1, \ldots, x_n),\ x_{1\ldots n} \in \mathbb{Z} \tag{4.14}$$

Small subsets of points that are used in morphological operations are called '*Structured Elements*' or SEs; see Gonzalez and Woods (2006, page 629) for more details. The basic morphological operations such as erosion, $A \ominus B$ and dilation, $A \oplus B$ are used with a variety of different SEs to achieve optimal results. The SEs can be combined and modified according to the stated problem; this process is state-of-the-art in digital image processing. Since morphological operations are set operations and set structures, as are point clouds, morphological operators can be applied to them.

Segmentation performance and spatial relation queries, are significantly improved if they are based on morphology operations applied to triangulated networks:

> *"High-level spatial relation and configuration modeling issues are gaining momentum in the image analysis and pattern recognition fields. In particular, it is deemed important whenever one needs to mine high-content images or large scale image databases in a more expressive way than a purely statistically one [...]*
>
> *[...] provide an original mesh lattice framework which is more convenient for structural representations of large image data by the means of interest point sets and their morphological analysis. The set of designed numerical operators is based on a specific dilation operator that makes it possible to handle concepts like "between" or "left of" over sparse representations of image data such as graphs [...]*
>
> *[...] basically proposing a new theoretical framework to reason about images [...]" Lomenie et al. (2000); Loménie and Stamon (2008)*

A 3-D ranger produces large amounts of data which requires efficient processing algorithms: one such process is certainly morphological operations. Grid data and depth images represent a suitable domain for most morphological operations, and therefore optimal processing conditions for real-time processing are present.

4.4 Surface Representation and Modeling

In Strubecker (1964, 1969a,b) and do Carmo (1976) the basics of surface and curve theory are elaborated. Schneider and Eberly (2002), Agoston (2005) and Erickson (2011) provide methods for the analysis of discrete geometry elements, such as polygons and point clouds. The authors focus on continuous differential geometry and explain the basics of surface theory, curvature and projection. Surface representation and surface processing as polygonal meshes are elaborated in-depth in Botsch et al. (2010); this book shows recent achievements in computational geometry and its applications. A comprehensive text book on the analysis of computational geometry has been written by Goodman and O'Rourke (2004); the book gives extensive insight and addresses most of the current problems and their solutions, as well as recent research (until 2004).

This section of the thesis describes the relationship between a point cloud, $\mathcal{P}$ and a surface patch, $\oplus$. $\mathcal{P}$ is generally a discrete, incomplete and biased momentary recording of a complete, continuous and non-regular, reflecting[1] surface. Thus, each point cloud can by analyzed as:

- a discrete set of points.
- a continuous surface whose parameters has been derived from a point cloud, $\mathcal{P}$. A regular surface can be efficiently derived from discrete data.

Each representation contains advantages and disadvantages, and therefore simultaneous processing (analysis) of both representations is necessary to achieve optimal results. The optimal result differs according to the problem stated. For the purposes of shape analysis, a regular surface is beneficial because it instantly allows the application of differential geometry algorithms. Furthermore, if polynomial functions are used to approximate the surface, computations such as derivation and integration especially, are efficiently executed by a CPU. To analyze an arbitrary point cloud, $\mathcal{P}$ in both discrete and continuous representations, the mapping between the domains must fulfill the following conditions:

Let Φ be a two-manifold, regular surface patch in $\mathbb{R}^3$, $\Phi \subseteq \mathbb{R}^3$, and $\mathcal{F}$ a set of discrete points on Φ, $\mathcal{F} \in \Phi$ and $\mathcal{F} \subseteq \mathcal{P}$. Let Γ be on surface with zero curvature in $\mathbb{R}^3$ (Gaussian curvature is zero: $\kappa_1\kappa_2 = 0$). To be able to analyze a surface patch Φ, when it is projected onto the surface patch Γ, the map (m) must be diffeomorphism:

$$m : \Phi \mapsto \Gamma \wedge \exists\, m^{-1} : \Gamma \mapsto \Phi, \wedge \ddot{m} \neq 0 \tag{4.15}$$

The surface patch Φ in $\mathbb{R}^3$ is defined as

$$\mathcal{U} \subseteq \mathbb{R}^2 \mapsto \Phi \in \mathbb{R}^3 \tag{4.16}$$

$$(u,\, v) \mapsto \begin{pmatrix} x(u,\, v) \\ y(u,\, v) \\ z(u,\, v) \end{pmatrix} \tag{4.17}$$

To analyze the boundary in $\mathbb{R}^3$ of the surface patch, Φ, one possibility is to project the shape onto a plane. Without deformations, such a projection is only possible for very few surfaces; see Section 4.5. Still, it is possible to project moderately curved surfaces onto a plane so that the surface shape does not suffer extreme deformation. The visualization of this situation is shown in Figure 4.5: The surface patch Φ should be projected onto the analytic surface, Γ in order to analyze the boundary of Φ. Both surfaces exist in same coordinate system

[1] REFLECTING in this case describes the property of the surface to reflect a electromagnetic wave of an arbitrary wavelength and the reflected wave is captured by the recording sensor.

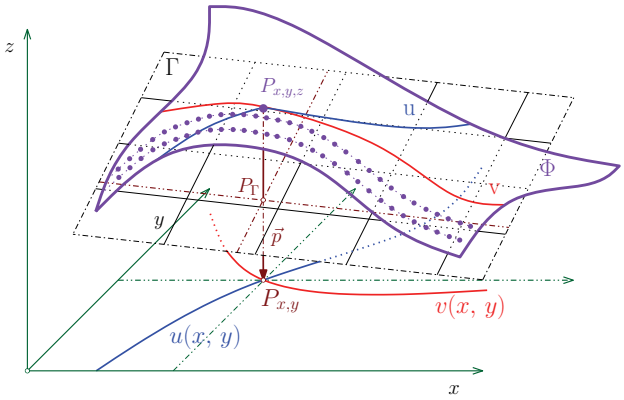


Figure 4.5: Surface Patch. Purple dots depict the discrete points ($\mathcal{F}$) on surface Φ. Red and blue lines are parameter lines. Point $P \in \Phi$ is projected on Γ surface and then on the x, y plane (dark red arrow). The projection sketch represents an ideal case.

and share common parameter lines, u and v (Eq. (4.17)). The aim is to determine the projection function that describes the orientation and magnitude of the projection vector, $\vec{p}$.

The process proposed here consists of two steps. The first projection involves the transformation of an arbitrary discrete point cloud, $\mathcal{P}$ (or its subset $\mathcal{F}$), onto the regular surface patch, Φ. The resulting regular surface is obtained using approximation or estimation algorithms. The results of the analysis provide a best guess for the parameters of the second analytic surface, whose Gaussian curvature must be equal to zero - a detailed explanation is presented in next section. The point cloud points that are projected onto Γ, are members of the set, $\mathcal{F}$. Thus, either representation of $\mathcal{P}$ can be used for the subsequent plane shape analysis. Both surfaces, Φ and Γ, as well as point cloud, $\mathcal{P}$, share the same parameter domain, $\mathcal{U}$.

4.5 Unfolding

This section explains the transformation of the original discrete data: from point cloud, $\mathcal{P}$ to surface Φ, and the subsequent projection of Φ to the analytic surface, Γ. Such a projection of a surface, Φ to Γ, is an atlas, $\mathcal{A}$; it contains at least one map $\mathcal{M}_n$:

$$\mathcal{A} = \{\mathcal{M}_1, \ldots, \mathcal{M}_n\},\ n \in \mathbb{N},\ \mathcal{M}_n : \mathbb{R}^k \mapsto \mathbb{R}^l,\ k, l \in \mathbb{N} \tag{4.18}$$

A typical example of an unfolding application is a geographic map. For the Gauß-Krüger projection it consists of two steps; the first step is selection of the datum or reference surface. In the second step the atlas is created; the ellipsoid is intersected with cylinders starting at the 0° meridian. Each cylinder is a map. Consecutive maps overlap, hence there are points that are common for each map. Further, the deformation of projected points increases with distance from the origin meridian. Thus, projections differ from map to map within the atlas and coordinates have to be homogenized; see Kahmen (2005). The projection is not holomorphic, hence it is definite that it cannot be biholomorphic.

In Figure 4.6 regular surfaces are shown. There are three regular surfaces besides the plane, that allow biholomorphic projection onto the plane (see Strubecker (1969b, Page 14) and Botsch et al. (2010, Page 31 ff.)):

1. Cylinder.
2. Cone.
3. Tangent surface.

All surfaces fulfill the following criteria: Gaussian curvature equals zero, thus $K = 0$, where κ_1 and κ_2 are the principal curvatures:

$$K = \frac{LN - M^2}{EG - F^2} = \kappa_1 \kappa_2 \tag{4.19}$$

Where E, F and G result from the first Gaussian fundamental form (Eq. (4.21)) and L, M and N result from the second Gaussian fundamental form Eq. (4.23):

$$s(u,v) = \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}, \; s_u = \frac{\partial s}{\partial u} s_v = \frac{\partial s}{\partial v} \tag{4.20}$$

$$E = s_u \cdot s_u \qquad F = s_u \cdot s_v \qquad G = s_v \cdot s_v \tag{4.21}$$

$$\mathbf{n} = s_u \times s_v \qquad \mathbf{n}_u = \frac{\partial \mathbf{n}}{\partial u} \qquad \mathbf{n}_v = \frac{\partial \mathbf{n}}{\partial v} \tag{4.22}$$

$$L = -\mathbf{n}_u \cdot s_u \qquad M = -\frac{1}{2}(\mathbf{n}_u s_u + \mathbf{n}_v s_v) \qquad N = -\mathbf{n}_v s_v \tag{4.23}$$

For proofs and explanations consider consulting Strubecker (1969b) or do Carmo (1976). Modern nomenclature and a comprehensive overview can be found in Botsch et al. (2010, Chapter 3, Differential Geometry). An explanation of plane-to-plane projection is omitted here since it is a basic algorithm and an approximate plane shape in nature. The sections that follow, provide the basic formulae to project a point cloud onto a regular surface. The coordinate axes of a point cloud and the axes of the projection surface are collinear. The projection equations are derived in vector notation.

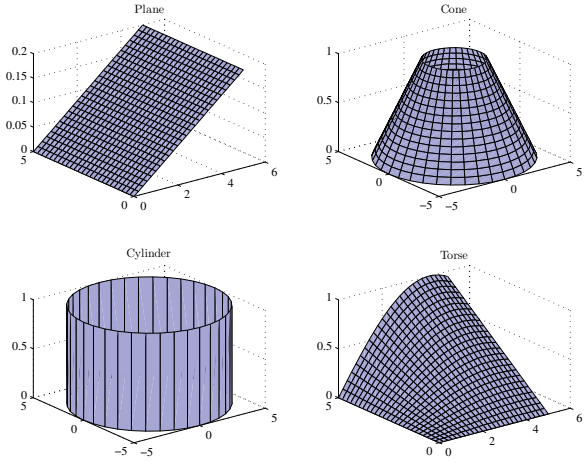


Figure 4.6: Regular Surfaces.

4.5.1 The Cylinder Projection

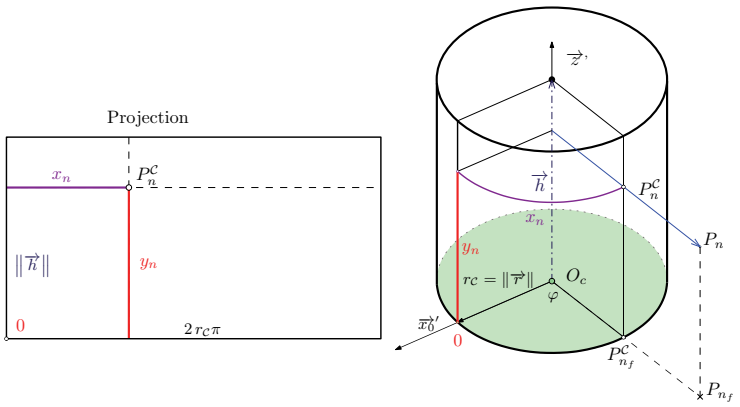


Figure 4.7: Cylinder Projection.

The cylinder $\mathcal{C}$ shown in Figure 4.7 yields the projection surface Γ, where:

- O_c is the origin point of the cylinder.
- $\vec{h} = \vec{z}'$ is the cylinder rotation axis. The height of cylinder is $h = \left\|\vec{h}\right\|$.

- $r_{\mathcal{C}} = \|\vec{r}\|$ is the cylinder radius.
- $P_n^{\mathcal{C}}$ is a point on the cylinder surface in the Cartesian coordinate system.
- P_{n_f} is the projected point, P_n onto the $xy-$plane of the cylinder in the Cartesian coordinate system.
- x_n, y_n are rectangular cylinder nappe coordinates of the point, $P_n^{\mathcal{C}}$.
- $\vec{x}' = \vec{r}$, $\vec{z}' = \vec{h}$, $\vec{y}' = -(\vec{x} \times \vec{z})$ are the axes of cylinder local in the Cartesian coordinate system.

Then an arbitrary point cloud, $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ with $\mathbf{p}_n = \{\xi_n, \eta_n, \upsilon_n\}$ expressed in cylinder coordinates, is projected through the map, p to the cylinder, $\mathcal{C}$:

$$p : \mathcal{P} \mapsto \mathcal{C} \tag{4.24}$$

$$P_{n_f} = \{\xi_n, \eta_n, 0\} \tag{4.25}$$

$$\vec{p}_n = O_c \vec{P}_{n_f} \tag{4.26}$$

$$\varphi_n = \begin{cases} \arccos\left(\frac{\vec{r}\cdot\vec{p}_n}{\|\vec{r}\|\|\vec{p}_n\|}\right) & \det(\vec{r}, \vec{p_n}, \vec{h}) > 0 \\ -\arccos\left(\frac{\vec{r}\cdot\vec{p}_n}{\|\vec{r}\|\|\vec{p}_n\|}\right) & \det(\vec{r}, \vec{p}_n, \vec{h}) < 0 \end{cases} \tag{4.27}$$

$$P_n^{\mathcal{C}} = \begin{bmatrix} \xi_n^{\mathcal{C}} \\ \eta_n^{\mathcal{C}} \\ \nu_n^{\mathcal{C}} \end{bmatrix} = \begin{bmatrix} r_{\mathcal{C}} \cos\varphi \\ r_{\mathcal{C}} \sin\varphi \\ \nu_n \end{bmatrix} \tag{4.28}$$

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} r_c\varphi \\ \nu_n \end{bmatrix} \tag{4.29}$$

4.5.2 The Cone Projection

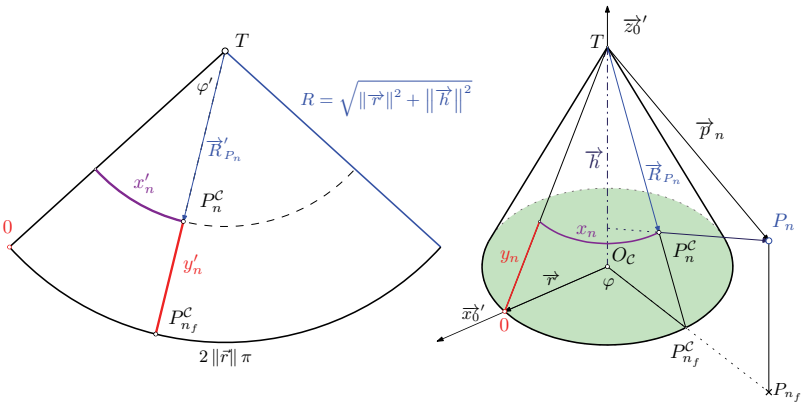


Figure 4.8: Cone Projection Fundamentals.

Let a cone, $\mathcal{C}$ in $\mathbb{R}^3$ (see Figure 4.8) be defined by:

- $O_{\mathcal{C}}$, the origin of the point of the cone,
- T, the peak cone point,
- $\vec{h}$, the vector of cone axis and $z-$axis of the cone coordinate system: $\overrightarrow{O_{\mathcal{C}}T}$,
- $\overrightarrow{r}$, the cone radius vector and $x-$axis of the cone coordinate system.

Then an arbitrary point cloud, $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ with $\mathbf{p}_n = \{\xi_n, \eta_n, \upsilon_n\}$ expressed by the cone coordinates, is projected through the map p onto the cylinder $\mathcal{C}$:

$$p : \mathcal{P} \mapsto \mathcal{C} \tag{4.30}$$

A cone coordinate system is defined by $O_{\mathcal{C}}$, $\overrightarrow{x_0} = \{\xi_x, \eta_x, \upsilon_x\} = \{1, 0, 0\}$, $\overrightarrow{z_0}' = \{\xi_z, \eta_z, \upsilon_z\} = \{0, 0, 1\}$ and $\overrightarrow{y_0} = -(\overrightarrow{x_0} \times \overrightarrow{z_0})$; an arbitrary point from $\mathcal{P}$ is mapped onto the cone nappe. An illustration of the projection is shown in Figure 4.8, and the actual projection formulae are derived below:

$$P_{n_f} = \{\xi_n, \eta_n, 0\} \qquad \vec{p}_n = \overrightarrow{TP_n} \tag{4.31}$$

$$P^{\mathcal{C}}_{n_f} = \|\vec{r}\| \frac{\overrightarrow{O^{\mathcal{C}} P_{n_f}}}{\left\|\overrightarrow{O^{\mathcal{C}} P_{n_f}}\right\|} \qquad = \vec{p}_{n_f} \tag{4.32}$$

$$\vec{R}_{P_n} = \overrightarrow{TP^{\mathcal{C}}_{n_f}} \qquad \vec{R}_{0_{P_n}} = \frac{\vec{R}_{P_n}}{\left\|\vec{R}_{P_n}\right\|} \tag{4.33}$$

$$P^{\mathcal{C}}_n = \|\vec{p}_n\| \, \vec{R}^{\mathcal{C}}_{0_{P_n}} \tag{4.34}$$

The angle, φ_n between the $\vec{r}$ and $\vec{p}_{n_f}$, is computed using the scalar product rule:

$$\varphi_n = \begin{cases} \arccos\left(\frac{\vec{r}\cdot\vec{p}_n}{\|\vec{r}\|\|\vec{p}_n\|}\right) & \det(\vec{r}, \vec{p}_{n_f}, \vec{h}) > 0 \\ -\arccos\left(\frac{\vec{r}\cdot\vec{p}_n}{\|\vec{r}\|\|\vec{p}_n\|}\right) & \det(\vec{r}, \vec{p}_{n_f}, \vec{h}) < 0 \end{cases} \tag{4.35}$$

The angle projection, $\varphi_n^{'}$ is the scale factor between the angles in the cone coordinate system, and the angles on the cone nappe, which are derived from the perimeter of the cone base:

$$s_\varphi = \frac{\frac{2\|\vec{r}\|\pi}{R}}{\frac{2\pi}{1}} = \frac{\|\vec{r}\|}{R} \tag{4.36}$$

$$\varphi_n^{'} = s_\varphi \varphi_n \tag{4.37}$$

The cone nappe coordinates are:

$$P^{\mathcal{C}}_n = \begin{bmatrix} x_n^{'} \\ y_n^{'} \end{bmatrix} = \begin{bmatrix} \left\|\vec{R}_{P_n}\right\| \varphi_n^{'} \\ R - \left\|\vec{R}_{P_n}\right\| \end{bmatrix} \tag{4.38}$$

Since the angles between the parameter lines on the cone are perpendicular, the resulting coordinates can be considered as a plane rectangular grid, and they are suitable for use in the shape analysis as explained in Section 3.4.3.

4.5.3 The Tangent Surface Projection

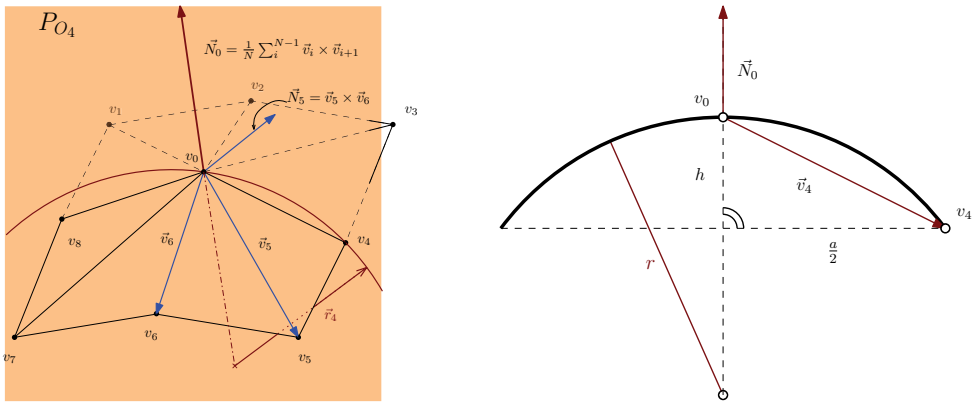
Lastly, the regular parametrized surface for projection onto a plane, is a tangent surface; see do Carmo (1976, page 78 ff.) or Strubecker (1969a, page 76 ff.). After do Carmo (1976, page 78 ff.):

> *"[. . .] A parametrized surface* $\mathbf{x} : U \subset \mathbb{R}^2 \mapsto \mathbb{R}^2$ *is a differentiable map* $\mathbf{x}$ *from an open set* $U \subset \mathbb{R}^2$ *into* $\mathbb{R}^3$*. The set* $\mathbf{x}(U) \subset \mathbb{R}^3$ *is called the trace of* $\mathbf{x}$ *[. . .]"*
>
> *"[. . .] Let* $\alpha : I \mapsto \mathbb{R}^3$ *be regular parametrized curve.*
>
> *Define:*
>
> $$\mathbf{x}(t, v) = \alpha(t) + v\alpha'(t), \, (t, v) \in I \times \mathbb{R}$$
>
> *where* $\mathbf{x}$ *is parametrized surface called the tangent surface of* α *. . .*
>
> *Assume now that the curvature* $k(t)$*,* $t \in I$*, of* α *is nonzero for all* $t \in I$*, and restrict the domain of* $\mathbf{x}$ *to* $U = \{(t, v) \in I \times \mathbb{R}; \, v \neq 0\}$*.*
>
> *Then:*
>
> $$\frac{\partial \mathbf{x}}{\partial t} = \alpha'(t) + v\alpha''(t), \, \frac{\partial \mathbf{x}}{\partial v} = \alpha'(t)$$
>
> *and*
>
> $$\frac{\partial \mathbf{x}}{\partial t} \wedge \frac{\partial \mathbf{x}}{\partial t} = v\alpha''(t) \wedge \alpha'(t) \neq 0, \, (t, v) \in U$$
>
> *[. . .]" (do Carmo, 1976, page 78 ff.)*

In other words the tangent surface can be considered as a hull along the space curve, $\bar{c}_k$; for the computation of $\bar{c}_k$ see Section 9.7. The tangent surface principle is substantially modified to obtain a shape projection onto the plane Cartesian coordinate system that is as realistic as possible. To achieve this aim, polynomial approximation is applied on the parameter lines, u and v. A detailed explanation is given in Section 9.7. The polynomial degree is determined through previous curvature analysis of the surface, Φ. Since the average, maximum and minimum curvature are necessary parameters to determine the polynomial degree, the curvature parameters are computed on the point cloud using approximate discrete curvature computation.

Discrete Curvature

Consider all curves on a smooth surface, S in $\mathbb{R}^3$ to run through the point, $P \in S$. P is either an umbilical point, in which case the principal curvatures are equal and $\kappa_1 = \kappa_2$, or there are two extreme values in each point; the principal curvatures, in which case κ_1 and κ_2 are in the direction of two tangent vectors $\mathbf{t}_1$ and $\mathbf{t}_2$. At each point of the surface where the Gaussian fundamental forms are defined, there exist an infinite number of normal planes, spanned by tangent vectors ($\hat{\mathbf{t}}_n$), and a normal vector. The relationship between the normal curvatures and principal curvatures is defined by the Euler theorem: $\kappa_n(\hat{\mathbf{t}}_n) = \kappa_1 \cos^2\psi + \kappa_2 \sin^2\psi$, $\psi = \angle(\mathbf{t}_1, \mathbf{t}_2)$. The relationship allows the omission of the curvature of the tangential vectors if only the the principal curvatures are known.



(a) Eight Vertex Neighborhood (One Ring Neighborhood).

(b) Curvature Radius Situation.

Figure 4.9: Discrete Curvature.

The computation of the discrete curvature at an arbitrary vertex, v is an approximation. The curvature is computed within the normal planes, P_{O_i} that each contain the normal unit vector, $\vec{N}_0$ at a central vertex, and a vertex vector $\vec{v}_i$. The approximate curvature radius computation is straightforward; from Figure 4.9b it is obvious that:

$$\left(\frac{a}{2}\right)^2 = \|\vec{v}_i\|^2 - h^2 \Rightarrow \tag{4.39}$$

$$2\sqrt{2hr_i - h^2} = 2\sqrt{\|\vec{v}_i\|^2 - h^2} \Rightarrow \tag{4.40}$$

$$r_i = \frac{\|\vec{v}_i\|^2}{2\left\|\vec{v}_i \cdot \vec{N}_0\right\|} \Rightarrow \kappa_i = \frac{1}{r_i} \tag{4.41}$$

with the mean normal unit vector in the 8-connected neighborhood (see Figure 4.9a):

$$\vec{N}_0 = \frac{1}{7}\sum_{i=1}^{7} \vec{v}_i^0 \times \vec{v}_{i+1}^0 \tag{4.42}$$

The computation of the mean unit normal vector is not limited to a one ring neighborhood, nor do the vertices have to be equidistant; see Botsch et al. (2010).

The computation of an osculating circle is reduced to the computation of the circumcircle of a triangle. Consider an 8-vertex neighborhood given by:

- $\vec{N}_0$, the mean unit normal vector for the vertex neighborhood.
- P_{O_4}, an osculating circle plane, defined by:
  - a vertex, v_0,
  - a mean normal vector, $\vec{N}_0$,
  - a vertex vector, $\vec{v}_i$, $i = \{1, \ldots, 8\}$.
- $\vec{r}_i$, an osculating circle radius vector.
- κ_i, an osculating circle curvature.

Conclusion

The major problem for the reliable classification of the leaves based on their shape, is solved if the projection of the two-manifold surface of a leaf onto a plane is realized. Considering the cone and cylinder projection, and relating their suitability to the problem solution, it is obvious that they can help in very few cases. The resulting distortion of the leaf shapes projected on a cone or cylinder, might even be worse when compared to results achieved with the raw intensity images analyzed in Gebhardt (2007); Gebhardt and Kuehbauch (2007); Steffen (2006). Shape distortion is inevitable for majority of the cases, though by applying tangential surface projection, local distortion will not occur or will be minimal. The expected distortion will be very similar to the distortion that occurs during the leaf flattening for the herbaria. An approach that might be suitable is described in Section 9.7; it shows that the camera pose problem and inhomogeneous surface curvatures can be minimized by using the tangential surface projection method.

Chapter 5

Recursive Data Filtering and Modeling

In the preceding chapters the basics necessary for 3-D shape analysis, and the projection of the boundary of a 3-D shape onto a plane were set out. The methods did not consider noise in the data; noise can significantly affect the result. Basic filters such as the Gaussian filter, and the median filter are limited by the kernel they use, so that only the direct neighborhood affects the filtered result. Also, morphological operations are limited to structured elements, so both kernel-based filtering methods are suitable when the *size* of the objects in the raw data is known. The size of the objects in raw data is however unknown, therefore a filtering method is necessary which is independent of the object size; a *recursive filter* would be suitable.

A 3-D point cloud, $\mathcal{P}$ is the data input on which the filter operates; the domain of $\mathcal{P}$, $\mathcal{U}$, is a regular grid. The point cloud, $\mathcal{P}$ is then a *range image* where u and v are parameter lines (see definition in Section 4.4). Separation between neighbored objects in the range image can be detected only if there is a detectable edge in the range data. So separation is possible if the difference in the range[1] is larger than the sensor resolution. The filters' task is to use a mathematical model to compute:

1. The approximation function parameters for each of: $u = const.$ and $v = const.$
2. The separation positions of the edge points: $u = const.$ and $v = const.$

Each range value is sequentially fed into a filter, so that a filter updates its parameters with each observation; the influence of the noise has to be minimized and outliers must be detected during the filtering process. A filter must detect the edges between distinct objects. A filter estimates the function parameters or *states*. The Kalman Filter (KF) is one of the most widely used estimators for linear, or linearized and non-linear problems with a large range of real-time applications. Available processing power allows real-time use of the Particle Filter (PF), which is a particular solver of non-linear problems. A particle filter finds its application in the real-time algorithm: simultaneous localization and mapping (SLAM) (Sachniss et al., 2012). For both filter types a large number of applications and publications are available. At this point an excerpt from the literature list yields the following publications in which these topics are considered: Jazwinski (1970); Gelb et al. (1974); Durrant-Whyte (2001); Grewal and Andrews (2001); Arulampalam et al. (2002); Ristic et al. (2004); Brown et al. (2005); Simon (2006); Welch and Bishop (2010).

The idea behind the choice of recursive filtering is that each *curve* on a surface, Φ, which is the result of the intersection of a plane with the surface, Φ, must be as continuous and smooth as the surface itself. Given these conditions it will be sufficient to analyze the intersection curves and deduce the characteristics of the surface from that analysis. The "filtered" surface, Φ will then be a set of surface curves.

Below are the basic formulae for recursive filtering. The recursive filter consists of a state estimation part (Eq. (5.1)) and a measurement model part (Eq. (5.2)):

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{w}_{k-1}) \tag{5.1}$$

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{v}_k) \tag{5.2}$$

where $\mathbf{w}_{k-1}$ denotes system or process noise, and $\mathbf{v}_{k-1}$ measurement noise[2].
After Simon (2006):

[1] The z-axis represents the distance from the sensor plane to points of the point cloud.

[2] The notation of system and measurement noise is taken from Simon (2006), other authors use different notations: for example Ristic et al. (2004) denotes measurement noise with w and system noise with v. Such inconsistency might confuse or limit recognition of the formulae if the respective cited publications are consulted.

"[...] The noise processes $\{w_k\}$ and $\{v_k\}$ are white, zero-mean, uncorrelated, and have known covariance matrices Q_k and R_k, respectively:

$$\mathbf{w}_k \sim \mathcal{N}(0,\, Q_k) \tag{5.3}$$

$$\mathbf{v}_k \sim \mathcal{N}(0,\, R_k) \tag{5.4}$$

[...]"

After Simon (2006, page 84. ff), the recursive least squares estimator is written in the form given below (the derivation is neglected here). The index, k represents t_k on the time scale: "[...]

$$\mathbf{y}_k = H_k\mathbf{x}_k + \mathbf{v}_k \tag{5.5}$$
$$K_k = P_{k-1}H_k^T(H_kP_{k-1}H_k^T + R_k)^{-1} \tag{5.6}$$
$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + K_k(\mathbf{y}_k - H_k\hat{\mathbf{x}}_{k-1}) \tag{5.7}$$
$$P_k = (I - K_kH_k)P_{k-1} \tag{5.8}$$

[...]"

For the purpose of being consistent and to achieve the obvious time relation, the index k is substituted by t, and Equations (5.5-5.8) become:

$$\mathbf{y}_t = H_t\mathbf{x}_t + \mathbf{v}_t \tag{5.9}$$
$$K_t = P_{t-1}H_t^T(H_tP_{t-1}H_t^T + R_t)^{-1} \tag{5.10}$$
$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} + K_t(\mathbf{y}_t - H_t\hat{\mathbf{x}}_{t-1}) \tag{5.11}$$
$$P_t = (I - K_tH_t)P_{t-1} \tag{5.12}$$

The process of recursive filtering comprises the following three major tasks:

1. Filtering of the data.
2. Approximation of the data according to the theoretical model.
3. Robust edge detection that allows reliable object separation.

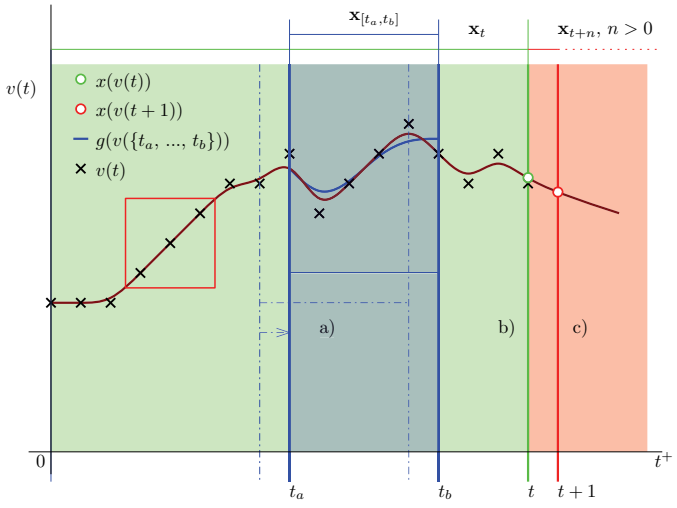


Figure 5.1: Recursive Filter Types.

According to Grewal and Andrews (2001, page 184. ff) the recursive state estimation is distinguished by the time point of the system state. The system state vector is denoted as $\mathbf{x}_{t_i}$; this is shown in Figure 5.1 and discussed below:

- The *filter* computes the state of the system based on measurements completed in the past with $t \leqslant i$ and the current measurement $t = i$, where i denotes discrete time. The region covered by the filter is colored green in Figure 5.1. The green line marked as 'b)' in the figure, indicates the limit of the filter.

- The *predictor* computes the state of the system in the future with $t > i$, based on an estimate from the filter. The region occupied by the predictor is colored red in Figure 5.1. The red line marked as 'c)' indicates the predicted value based on the last valid state, $\mathbf{x}_{t_i}$.

- The *fixed-interval smoother* estimates $\hat{\mathbf{x}}_{[t_a,t_b]}$ in the given interval, $[t_a, t_b]$. The region occupied by the fixed-interval smoother is colored blue in Figure 5.1. The estimated state, $\hat{\mathbf{x}}_{[t_a,t_b]}$ generally differs from the estimated state computed by the filter in the interval $[t_a, t_b]$: $\hat{\mathbf{x}}_{[t_a,t_b]} \neq \mathbf{x}_{[t_a,t_b]}$. The reason is obvious: consider $X = \{v(t_1), v(t_2), \ldots, v(t_i)\}$, as a set of discrete observations in time; these observations are shown as $\times$'s in Figure 5.1. Then the interval $[t_a, t_b] \rightarrow I_{a,b} = \{v(t_a), \ldots, v(t_b)\}$ with $b > a \Rightarrow I_{a,b} \subset X$. The estimation of the system state is computed from a subset of observations $I_{a,b}$. Identical values for the system state are generally contingent, but not improbable; see the red rectangle in Figure 5.1.

For the sake of completeness the following two smoothers are mentioned but not illustrated in Figure 5.1:

- The *fixed-lag smoother* delays the estimation of the state by a constant, Δt to improve the signal quality. It uses data until time t, but the state of the system $\hat{\mathbf{x}}_{t-\Delta t}$ is estimated in the past.

- The *fixed-point smoother* uses the whole data set for the estimation of the system state, $\mathbf{x}$. It fixes the system state for time, t_{fixed}:
  Fixed-point smoothers function as predictors when the current time, $t > t_{fixed}$, as filters when $t = t_{fixed}$ and as smoothers when $t < t_{fixed}$; see Grewal and Andrews (2001, page 186, section 5.1.3.3). If it contains an error, the predictor computes the state of the system in the future, hence $t > t_{fixed}$, and smooths the data for $t < t_{fixed}$.

The contribution of recursive filter methods for edge detection and robust segmentation is explained in detail in Section 9.3. The result of the parametrization of 3-D curves is a polynomial representation of surface curves. These are used to project the 3-D surface onto a plane, retaining the surface's boundary properties. The boundary points, now projected onto a plane, are used to compute elliptic Fourier descriptors; see Section 3.4.3. These descriptors are classified at the end process, using a support vector machine (SVM), which is described in the following chapter.

Chapter 6

Classification and Machine Learning

6.1 Why Machine Learning?

Figure 6.1: Russell and Norvig (2009, 693 ff.) Artificial Intelligence Disciplines.

The previous chapter describes a subset of the methods for noise reduction, feature extraction and transformation from a two-manifold shape in $\mathbb{R}^3$ to a plane shape in $\mathbb{R}^2$. The shape of extracted objects underlies a variety of effects caused by sensor noise, environmental influences, biological diversity, seasonal influences, cluttering, etc. Since these effects and their interaction are not yet incorporated in mathematical and/or physical models, the next best classification approach is to allow the computer to "learn" from samples. The learning process postulates a hypothesis and each new feature can be classified using a statistical test based on that hypothesis; this general statement considers a small subset of the methods for automatic classification. The subset encompasses "statistical learning" methods; see Bishop (2006). A short introduction to the machine learning (ML) process follows.

6.2 General Remarks

Artificial intelligence as a science examines the learning *process* and *knowledge* management from a variety of perspectives. Russell and Norvig (2009, 693 ff.) consider the discipline of AI as being comprised of four different areas: Figure 6.1 shows the differentiation of these areas graphically. For the purpose of plant detection, the required classification algorithms belong to the area *"Learning"*, sub area *"From Examples"*. These algorithms provide sufficient functionality to learn and automatically classify samples. State-of-the-art AI science (2012) is still limited: machines (computers) cannot learn as humans do. However, cutting-edge research projects are attempting to create a computer program or set of programs that can learn as humans do:

> *"Blue Brain is a resounding success. In five years of work, Henry Markram's team has perfected a facility that can create realistic models of one of the brain's essential building blocks. This process is entirely data driven and essentially automatically executed on the supercomputer. Meanwhile the generated models show a behavior already observed in years of neuroscientific experiments. These models will be basic building blocks for larger scale models leading towards a complete virtual brain." (EPFL, 2012a)*

In the near future, if the project succeeds, which is highly likely, a model of the human brain with its learning capabilities will be available for practical applications. Other scientists are pursuing the path of designing electronic circuits that behave as human nerve cells do; see Krzysteczko et al. (2012). Once such hardware having the human brain's learning capabilities becomes a reality, the biggest barrier towards creating an autonomous and intelligent machine will have been surmounted. The classification algorithm applied in this thesis has to fulfill far simpler but still challenging requirements.

6.3 Machine Learning: Learning by Examples

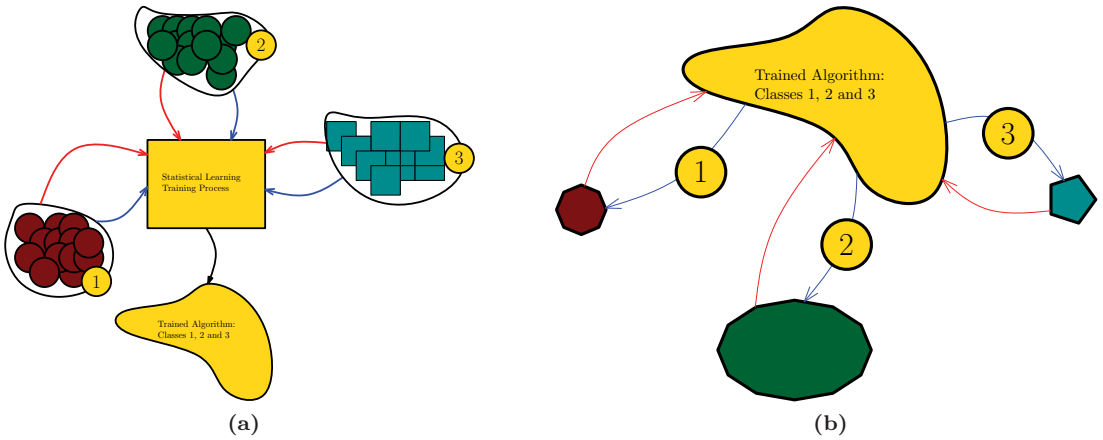


Figure 6.2: (a) Learning from Samples (Statistical Learning Process) Scheme; the red lines represent the feature vector input into the ML algorithm. During the learning process the class of the feature vector is also required as an input. (b) Classification Process Scheme; the classification requires the feature vector only - the output of the ML algorithm is the feature class.

Machine learning (ML) is a subset of algorithms within the artificial intelligence (AI) discipline that provides functionality to the computer in order to model the data and derive a conclusion from the results. The objective of this thesis (see mission statement Chapter 1) requires a machine learning algorithm of high reliability with minimal learning effort. The requirements for the classification algorithm can be summarized by following *use-cases*:

1. A user requires a single specific shape, $\mathcal{S}_0$ to be extracted from an entity, $\mathcal{E}$; $\mathcal{S}_0$ is known while other shapes are unknown: $\mathcal{S}_0 \subset \mathcal{E}$. This is the *one* class problem.

2. A user requires the entity, $\mathcal{E}$ to be separated into specific, unique classes, with full a-priori knowledge: $\mathcal{S}_n$, $n = \{1, 2, \ldots\}$: $\mathcal{E} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_n$. This is the *multiple* class problem.

3. A user requires the entity, $\mathcal{E}$ to be separated into unique classes, with partial a-priori knowledge: $\mathcal{S}_n$ and $\tilde{\mathcal{S}}_k$, k, $n, \in \mathbb{N}$. The $\tilde{\mathcal{S}}_k$ denotes that also untrained species have to be distinguished as such and not classified into the most similar class: $\mathcal{E} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_n \cup \tilde{\mathcal{S}}_1 \cup \cdots \cup \tilde{\mathcal{S}}_k$. This is an *extension* of the *multiple* class problem.

Given the features that describe shapes in a unique manner, the classification algorithm can separate them into their classes. One possible solution could be when two important elements for classification are known: the number of shapes in an entity, $\mathcal{E}$ and the centroid of each shape feature, $\bar{\mathcal{S}}_n$. Each $\mathcal{S}_n$ is thus represented by its centroid, $\bar{\mathcal{S}}_n$, so that during the training process the algorithm computes the mean centroid for a class, as well as its range. The classifier "learns" the shape from the samples, so that new shapes can be classified into a unique class after analysis; see Figure 6.2. How well the classifier performs, can be verified against a known data base. If the algorithm fails to provide the results required, the process can be repeated with altered learning parameters. Verification and training processes are divided into three types:

1. Supervised Learning.
2. Unsupervised Learning.
3. Reinforcement Learning.

Bishop (2006, page 3 ff.) defines learning methods as:

> "*[. . .]* Supervised Learning*: "Applications in which the training data comprises examples of the input vectors along with their corresponding target vectors are known as* SUPERVISED LEARNING *problems."*
> Unsupervised Learning*: ". . . In other pattern recognition problems, the training data consists of a set of input vectors x without any corresponding target values. The goal in such unsupervised learning problems may be to discover groups of similar examples within the data,. . . "*
> Reinforcement Learning*: "Finally, the technique of reinforcement learning (Sutton and Barto, 1998)*[1] *is concerned with the problem of finding suitable actions to take in a given situation to maximize a reward. [. . .] "*

According to Russell and Norvig (2009, page 695), the differentiation between these learning methods is not that distinct in practice:

> *"[. . .] In practice, these distinctions are not always crisp. In semi-supervised learning we are given a few labeled examples and must make what we can of a large collection of unlabeled examples. [. . .]"*

The multiple class approach represents a standard requirement for automatic classification. Figure 6.2a illustrates an example of the learning process where two different geometrical shapes (circles and squares) with three different colors are learned by an imaginary ML algorithm. The classification of new shapes in Figure 6.2b shows that the classifier sorts each shape into its nearest class, even though the shapes are not exactly the same as the training shapes. This classification routine shows good generalization behavior. Bad generalization behavior is the result of insufficient training and an evaluation set with the following limitations: data that is too homogenous, too few samples or too many samples. The ML methods that provide functionality for the illustrated problem (use-case 2) are: k-Means clustering, Artificial Neural Networks (Russell and Norvig (2009, pages 727-744), Marsland (2009)) and Support Vector Machines (SVM) (Russell and Norvig (2009, pages 744-748), Bishop (2006, pages 325 ff.), Schölkopf and Smola (2002), Press et al. (2007, pages 883 ff.) Chang and Lin (2001)). These three methods certainly do not cover the whole range of machine learning approaches, but due to their wide range of applications and analysis, they are serious contenders for use as classifiers in the plant detection process. Upon closer analysis however, Artificial Neural Networks (ANNs) can be disregarded as a suitable classifier for following reasons:

1. Training an ANN requires a large number of samples; publications give numbers which vary from a few thousand to 10,000 or more. An overview of the software aspects of the problem are given in Blais and Mertz (2001); StatSoft (2012)
2. An ANN is a "black-box" process. The output is dependent on numerous parameters: the hidden layers count, triggering functions and weights. ANNs are powerful, non-linear ML and regression algorithms, but opaque to the operator.

The Support Vector Machine (SVM) is a transparent classifier, which requires significantly fewer training samples; $n < 100$. Russell and Norvig (2009, page 744 ff.) state that:

> *"The support vector machine or SVM framework is currently the most popular approach for "off-the-shelf" supervised learning [. . .] "*

Bishop (2006) on page 325 in the chapter entitled *"Sparse Kernel Machines"* provides the reasoning for the selection of an SVM:

> *"[. . .] We begin by looking in some detail at the support vector machine (SVM), which became popular in some years ago for solving problems in classification, regression and novelty detection. An important property of support vector machines is that the determination of the model parameters corresponds to a convex optimization problem, and so any local solution is also a global optimum. [. . .]"*

[1] Publication reference: Barto and Sutton (1998).

Wang (V. Kecman in 2005) summarizes the motivation even better:

> *"[...] SVMs have been developed in the reverse order to the development of neural networks (NNs). SVMs evolved from the sound theory to the implementation and experiments, while the NNs followed a more heuristic path, from applications and extensive experimentation to the theory. [...]"*

The LIBSVM supports basic SVM kernels, requires moderate CPU time for training and has a straightforward API; see Chang and Lin (2001). The same library is integrated with the OpenCV library; the OpenCV library is applied to a wide range of engineering and scientific problems, hence it is intensively tested; see Bradski and Kaehler (2008).

6.4 Support Vector Machine

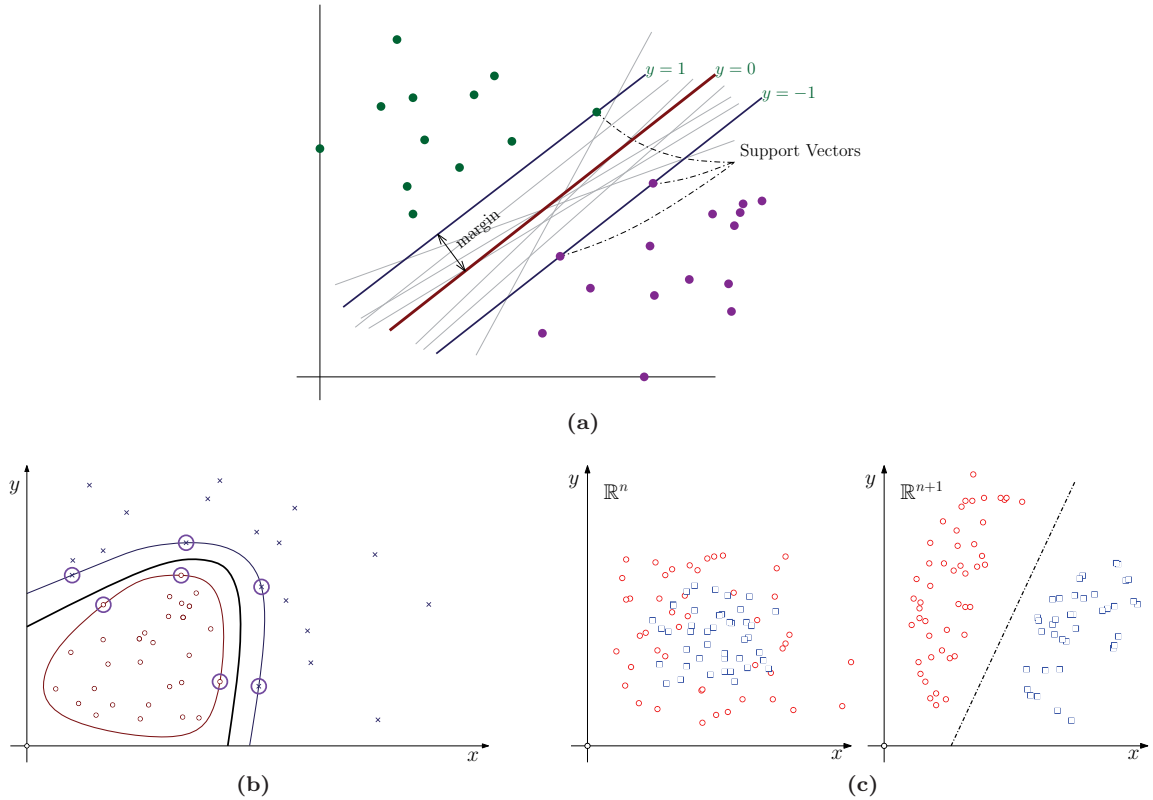


Figure 6.3: (a) Maximum Margin Classifier Principle. Examples of Support Vector Machine: (b) Scheme of two-class problem where red circles and blue crosses represent classes. the margin boundaries are drawn in the class color. Decision boundary is drawn as thick black line between two margins. The purple circles emphasize support vectors. (c) The left image shows linearly non-separable data, if the problem is mapped to the higher dimensional space (feature space) the data might be more easily linearly separable into two classes (the right image).

Support vector machine principles should be explained here to provide a fundamental understanding of the classification process during the experiments. The SVM method is widely used for a variety of classification problems; the IEEE database reports 1,110,000 papers, webinars, journal papers, etc. that contain the phrase "support vector machine".

A comprehensive, structured overview of the state-of-the-art research and applications of SVM is given on the web page by Sewell (2011). It is an excellent entry point for the full spectra of interest and knowledge levels. SVMs are *"maximum margin classifiers"* which can be explained as a two-class problem: $y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$ where $\phi(\mathbf{x})$ is a fixed feature space transformation and b is a bias parameter; see Bishop (2006, page 326 ff.). Two linearly separable classes of features, $\mathbf{x}$ with margin and support vectors are shown in Figure 6.3a. The gray lines show the possibility that more than one solution for separating the data set into two classes, exists. The training data contains input vectors, $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and their classes, $o_1, \ldots, o_n$, where $o_n \in \{-1, 1\}$. Each new feature, $\mathbf{x}_{n+j}$ is classified by the function, $\text{sig}(y(\mathbf{x}_{n+j}))$. The maximal margin can be expressed as the distance between two convex hulls:

$$\left\langle \mathbf{w},\, \mathbf{x}^{+} \right\rangle = +1 \qquad \left\langle \mathbf{w},\, \mathbf{x}^{-} \right\rangle = -1 \qquad \left\langle \mathbf{w},\, (\mathbf{x}^{+} - \mathbf{x}^{-}) \right\rangle = 2 \tag{6.1}$$

$$\left\langle \tfrac{\mathbf{w}}{\|\mathbf{w}\|},\, (\mathbf{x}^{+} - \mathbf{x}^{-}) \right\rangle = \frac{2}{\|\mathbf{w}\|} \tag{6.2}$$

This yields the primal problem of minimization of $\langle \mathbf{w},\, \mathbf{x} \rangle$ subject to $y_i(\langle w, x_i \rangle + b) \geqslant 1$. The real data contaminated with noise, yields overlapping distributions and classification errors; hence the margin hyperplane should be "softened" to allow for data points located on the wrong side of hyperplane. Those points are also penalized proportionally to their distance from the hyperplane. This principle is introduced by Bennett and Mangasarian (1992); Cortes and Vapnik (1995) and it uses the *slack variable* $\xi_i \geqslant 0$ for each training data point, Bishop (2006, page 331 ff.):

> *"[. . .] These are defined by $\xi_n = 0$ for data points that are on or inside the correct margin boundary and $\xi_n |t_n - y(\mathbf{x}_n)|$ for other points. Thus, a data point on the decision boundary $y(\mathbf{x}_n) = 0$ will have $\xi_n = 1$, and points with $\xi_n > 1$ will be misclassified. The exact classification constraints ($t_n \left(\mathbf{w}^T \phi(\mathbf{x}_n) + b\right) \geqslant 1$ $n = 1, \ldots, N$)[2] are then replaced with*
>
> $$t_n y(\mathbf{x}_n) \geqslant 1 - \xi_n, \quad n = 1, \ldots, N$$
>
> *in which the slack variables are constrained to satisfy $\xi_n \geqslant 0$. [. . .]"*

The aim of the process is to

> *"[. . .] maximize the margin while softly penalizing points that lie on the wrong side of the margin boundary. We therefore minimize:*
>
> $$\min \left(C \sum_{n=1}^{N} \xi_n + \frac{1}{2} \|\mathbf{w}\|^2 \right)$$
>
> *[. . .]"*

where C controls the relation between the penalty and the margin. For the full derivation and explanation of the maximum margin classifier, including the Karush-Kuhn-Tucker (KKT) condition, see Bishop (2006); Wang (2005).

Consider the situation in Figure 6.3. Using a clustering method like k-Means in feature space (in this case $\mathbb{R}^2$) would hardly be successful. Such non-linear separation problems are modeled by a variety of distinctive *non-linear kernels*. The "core" of the SVM is its kernel. There are five basic (popular) kernels after Wang (2005) with the conditions (C)PD = Conditionally Positive Definite:

- linear, dot product, Eq. (6.3) CPD,
- complete polynomial of degree d Eq. (6.4) PD,
- radial basis function (RBF) or Gaussian 6.5 PD,
- sigmoid Eq. (6.6) CPD and
- inverse multi-quadric function, Eq. (6.7) PD.

The kernel parameters are γ, r, β and d (see Chang and Lin (2001); Schölkopf and Smola (2002); Wang (2005)):

$$K_l(\mathbf{x}_i,\, \mathbf{x}_j) = \langle \mathbf{x}_i,\, \mathbf{x}_j \rangle \tag{6.3}$$

$$K_p(\mathbf{x}_i,\, \mathbf{x}_j) = (\gamma\, \langle \mathbf{x}_i,\, \mathbf{x}_j \rangle + r)^d,\ \gamma > 0 \tag{6.4}$$

$$K_R(\mathbf{x}_i,\, \mathbf{x}_j) = e^{-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2},\ \gamma > 0 \tag{6.5}$$

$$K_s(\mathbf{x}_i,\, \mathbf{x}_j) = \tanh(\gamma\, \langle \mathbf{x}_i,\, \mathbf{x}_j \rangle + r) \tag{6.6}$$

$$K_s(\mathbf{x}_i,\, \mathbf{x}_j) = \frac{1}{\sqrt{\|\mathbf{x}_i - \mathbf{x}j\|^2 + \beta}} \tag{6.7}$$

Preliminary analysis of the data has shown how narrow the boundaries between similar leaf shape classes are; see Section 3.4. For classification purposes a shape boundary is described by elliptic Fourier descriptors. Each

[2] The constraint formula is inserted by the author. In the original text there is cross-reference. Original forumulae enumeration is neglected.

leaf has its own unique feature vector, $\mathbf{x}_i$ of finite length (see Section 3.4.3). Considering the variance-covariance matrix in Figure 8.2b, the problem of separability of the features arises, since reliable classification is primarily possible for the first ten features where correlation coefficients have significantly smaller magnitude. Declaring a feature to be significant (truth) or insignificant (trash) is elaborated in Thornton and Qh (1999). Analyzing the EFDs with k-Means clustering, an acceptable separability was achieved; see Figure 8.5. But considering the results shown in Figure 8.2, only one conclusion is possible: only a few vector elements do not correlate to each other and strong correlation between the features hinders their separability. Hypothetically any machine learning method fails with data that contains only similar features; this would apply even in the case of a non-existent "highly sophisticated machine learning method". SVM can separate such data mapping to a "higher" dimension than the feature vectors actually are. Since SVM is a transparent and straightforward method, it is chosen, even though it requires a supervised learning process and the kernel selection requires a rough knowledge about data characteristics. With help of a suitable computer program these disadvantages are problems which are relatively easily solved.

6.5 *k*-Means Clustering

The high correlation between particular feature elements affects their separation. Furthermore such features unnecessarily extend the training process, and therefore careful feature or part-feature selection is recommended; the aim is to avoid unnecessary feeding of ML algorithms with useless data: see Thornton and Qh (1999); Mthembu and Marwala (2008). Feature correlation can be analyzed by standard statistical methods. For the initial estimate as to how the system will perform, k-Means clustering is the method of choice; see Bishop (2006, page 424 ff.). In Appendix A the preliminary analysis of the shape descriptors is accomplished in this way.

Chapter 7

Research Projects

The analysis of the 3-D solids has become increasingly important with the higher level of automation in the industrial production process. With automation pervading every aspect of production and living, the complexity of the tasks and the need for robust and efficient solutions rises daily. In addition to requirements in the manufacturing industry for production, monitoring or other automation processes, the entertainment industry requires 3-D data processing in real-time for the smooth, realistic terrain and animation behavior. The rendering of point clouds, wire models, polygonal meshes, RTN, ITN, etc. is required by countless applications such as film production, computer games, and computer aided design (CAD). State-of-the-art research projects by four research groups in the field of plant detection and surface modeling are briefly discussed in this section.

Research projects in the area of sensor technology are not included in this list of projects to be discussed. Sensor discussion (Chapter 8) focuses on a hypothetical device which delivers optimal data for plant detection, so it starts at the point where practical sensor analysis is concluded: sensor characteristics are theoretically and experimentally determined. Therefore, data acquisition and accuracy are considered as pre-existing, and not as an issue to be analyzed. The following three sections present the research institutions and groups that investigate information content in the data, rather than data or sensors themselves.The fourth section gives insight into the aimed application of this thesis: organic plant treatment, which is also a state-of-the-art research issue.

7.1 Computer Graphics

The following four research groups have been selected as being representative of cutting-edge computer graphics research:

AGG & CGGL Applied Geometry Group ETH Zurich and Computer Graphics and Geometry Laboratory at EPFL in Switzerland:

> *"Research at the EPFL Computer Graphics and Geometry Laboratory focuses on efficient representations and algorithms for digital 3D models. The goal of our research is to understand the principles of geometric computing and modeling, and to develop new algorithms and tools for efficient 3D model representation, shape analysis, simulation, design, and interaction. [...]", EPFL (2012b)*
>
> *"Research at the Applied Geometry Group focuses on efficient representations and algorithms for digital 3D models with a particular emphasis on dynamic shapes, i.e., 3D objects that undergo frequent changes of shape or appearance. Dynamic shapes play an important role in many fields, such as protein folding simulation, 3D shape design and analysis for mechanical engineering, computer animation for games and feature films, or medical applications for diagnosis and surgery simulation. The goal of our research is to understand the principles of geometric computing and modeling, and to develop new algorithms and tools for efficient 3D model representation, shape analysis, simulation, and interaction. [...]", AGG (2012)*

Microsoft Research The Computer Graphics Group, led by Dr. Hugues Hoppe, operates at Microsoft (2012), Redmond, USA. They are developing high-performance computer graphics algorithms for variety of problems. The group has addressed almost all issues concerning computer graphics. Currently their focus is on GPU accelerated processing and massive parallelization of different aspects of computer graphics, graphics processing, image rendering, etc.

RWTH Computer Graphics & Multimedia group led by Prof. Dr. Leif Kobbelt, University of Aachen, Germany, Kobbelt (2012). Special attention is focused on Sattler et al. (2011); Sattler (2012) discusses the problem of recognition of three dimensional objects and pose estimation of the camera which enables localization of the observer and the observed object.

> *"Research of the computer graphics group at RWTH Aachen focuses on geometry acquisition and processing, on interactive visualization, and on related areas such as computer vision, photo-realistic image synthesis, and ultra high speed multimedia data transmission. [...]"*

The fundamentals of the 3-D solids modeling is given by Taubin (1991); one can state that Prof. Dr. Taubin has made major contributions in the area of 3-D data processing and surface analysis (Taubin (2012)). The generalized eigenvector fit represents the key technique for the comparison of two curves in space. Recently Botsch et al. (2010) provided a comprehensive overview of both state-of-the art, and cutting-edge research on the analysis of polygon meshes for modeling and rendering purposes. Computer models and geometry processing provide basic methods for 3-D to 2-D point cloud transformations, segmentation and feature extraction. The actual comparison of solids constructed in CAD takes place in controlled environments, and though the problems are similar, the comparison procedure differs considerably to the problem statement elaborated in this thesis. Most of the approaches are based on vision sensor and intensity image analysis. The initial paper that examines the performance of elliptic Fourier descriptors for plant detection is by Neto et al. (2006): The shapes are scanned manually on a flatbed scanner and afterwards the outline of the leaf is extracted and analyzed. For the training of a classifier, high resolution boundaries are extracted using a similar procedure; see Chapter 6.

7.2 Vision-Based Plant Detection and Weed Control

The process of automatic weed control has its origins in commercially available computer vision sensors. The problem solutions vary significantly. Crop protection on a field is continually becoming more complex through the restrictions in the type and amount of toxins that can be applied. Environmental protection is a higher good in the long term sense; therefore intelligent systems for real-time treatment have been the subject of research for decades. The majority of systems and solutions are applicable to small areas, monoculture crops and mechanically processed soils. The University of Hohenheim's Department of Weed Science represents one of highest competency centers in Europe; see UniHohenheim (2012). Weed-control research and digital image processing for weed identification are fields of special interest: Weis and Sökefeld (2010); Weis and Gerhards (2007a); Weis (2010); Weis and Gerhards (2009, 2007b). Another intensity image-based approach for the detection of *Rumex obtusifolius L* has already been mentioned in the work of Gebhardt: Gebhardt (2007); Gebhardt and Kuehbauch (2007); Steffen (2006). The problem statement touches the kernel of this thesis, however it focuses on the search and extraction of homogenous areas in an intensity image, assuming that the objects in the meadow are broad-leaved.

In Weis (2010, Chapter 2: Publications) a detailed and comprehensive list of the research projects and publications concerning intensity image analysis for the different plant detection problems is given.

7.3 Segmentation of Point Clouds

Extending the basic algorithms and methods encountered in Section 7.1, 3-D point cloud segmentation and analysis is a research area that overlaps with many disciplines such as:

Geodesy This is probably the first scientific discipline that made use of the analysis of stereoscopic images, airborne radar and LIDAR sensors. The core of state-of-the-art data processing is formed by: remote sensing using multispectral image analysis, terrain models creation, airborne photogrammetry and terrestrial laser scanners; together these processes represent a subset of data-acquisition, processing and interpretation. Considering the problems described, Steinle (2005) analyzes the segmentation of artificial and natural objects obtained from airborne laser scanner data. Large scale GIS projects and maps are based on terrestrial mobile mapping devices, such as Google Maps Google (2012). Mobile mapping systems collect large amounts of 3-D data and require extensive automatic segmentation; see Petrie (2010).

Medicine Computed Tomography (CT), Magneto Resonance Imaging (MRI), and ultrasound are state-of-the-art medical imaging systems used for diagnostic purposes. The models of the human body or body parts in 3-D are created from slices of intensity images. Segmentation's main goal in the process is the extraction of an organ or a body part in real-time and post-processing. There are plenty of publications that cover this area; they include scientific publications and patents. Two of them are chosen because they cover a wide range of medical problems and applications: the well-established approach of 3-D sensing in medical applications as given by Shahidi et al. (1998) and Bayesian approach for segmentation of prior-known anatomical regions as set out by Bowsher et al. (1996) - both approaches make use of off-line processing. Real-time segmentation and fiber tracking is the theme of a more recent publication by Ho et al. (2012).

CAD For CAD/CAM[1] purposes, 3-D surface analysis and surface parametrization is described by Besl and Jain (1988). A later publication by Wilke (2002) describes the segmentation of large point clouds. A large part of this thesis describes the measurement principles for 3-D point cloud data acquisition; see Wilke (2002, Chapter 2: "Optische Messung von 3D-Objekten").

PCL 3-D point cloud processing is still a research problem, however recent achievements contribute to the stepwise solution of the problem. Furthermore, developer communities and university joint-ventures are contributing together to the software library (API) for the high and low-level 3-D point cloud processing. One such library is PCL, which has its supporters within the industry and academia.

> *"The Point Cloud Library (PCL) is a standalone, large scale, open project for 3D point cloud processing. [...]", PCL (2011).*

Robot behavior in an unordered and changing environment represents unsolved research problems; in Rusu (2009) automatic segmentation and semantic map creation are analyzed. The problem solution of Rusu's thesis elaborates on the core of the 3-D segmentation and the classification problem. Reliable segmentation is the basis of the problem's solution; when noisy data threatens to alter the shape of the object, the noise must be handled in a robust and reliable way. In Rusu's thesis, Chapter 8 "Surface and Object Class Learning", the recognition accuracy of geometric primitive objects is elaborated. The classification reaches an accuracy of up to 97% using standard machine learning methods. His thesis foci are indoor environments and human made objects.

7.4 Weed Treatment

Notwithstanding that plant treatment does not form part of the subject matter addressed in this thesis, one example of weed treatment applied to grassland using single plant detection, is illustrated. Organic food production especially, requires novel technologies such as steam or microwave technology, which do not contaminate the crop or the environment; see Latsch and Sauter (2011). Climate change has became an issue in the last five years, which is why microwave technology has been neglected; it results in a suboptimal CO_2 balance. Within the SmartWeeder Project (see Šeatović et al. (2010)), weed treatment was analyzed as a parallel task to plant recognition. The analysis focused on three different methods:

1. Mechanical
2. Microwave
3. Steam

Each of the methods is challenging and research is still on-going, although remarkable results were achieved; see the publication excerpts below:

Microwave treatment:

> *"Experiments with microwave technology were carried out in order to provide an alternative method of controlling dock weed on grassland. Two microwave devices of 4.8 kW and 18 kW were therefore tested under different site and weather conditions. The microwaves were applied to heat the extremely regenerative dock roots to their physiological collapse. Time series were used to obtain optimum treatment times for a minimum plant dying rate of 80%; therefore, four different trial varieties were chosen. In general, microwave technology is suitable for the treatment of dock plants and to prevent re-sprouting. The optimum heating time with an 18 kW microwave device is 28 s, which means 0.091 litre diesel per plant. The required length of all treatment variants, and hence the amount of energy to be applied, were shown to be high. [...]", Latsch and Sauter (2010); further publications by the author(s): Latsch et al. (2009, 2001); Latsch and Sauter (2009a,b).*

Water-Steam treatment:

> *"A control method for broad-leaved dock (Rumex obtusifolius) with hot steam was tested in grasslands in Switzerland. More than 700 plants were treated with a mixture of hot water and steam at a temperature of 120°C and a pressure of 30 bar. The aim was to identify an optimum heating time, so that the plant mortality rate was more than 80 %. The necessary diesel and water consumption for the heating were measured. The success of the method is strongly related to the soil moisture. The results show, that the intended mortality rate of 80 % at a soil moisture of 30 % can be realized at a calculated heating time of 30 seconds. For this treatment 2.4 litre of water and 0.046 litre of diesel were needed. [...]", Latsch et al. (2011b,a).*

[1] Computer Aided Manufacturing

7.5 Summary

Single plant detection, if robust and applicable in a cost-effective manner, will change food production paradigms and decrease the current pollution footprint of agricultural production. There is a strong motivation to develop such systems to be cost-effective, while decreasing the dependency on human labor. In this chapter it was shown that research in the area of 3-D data analysis is widespread in a variety of scientific disciplines and that the results achieved have almost instantaneous application in a variety of commercial products.

Part III

Conceptual Model

Chapter 8

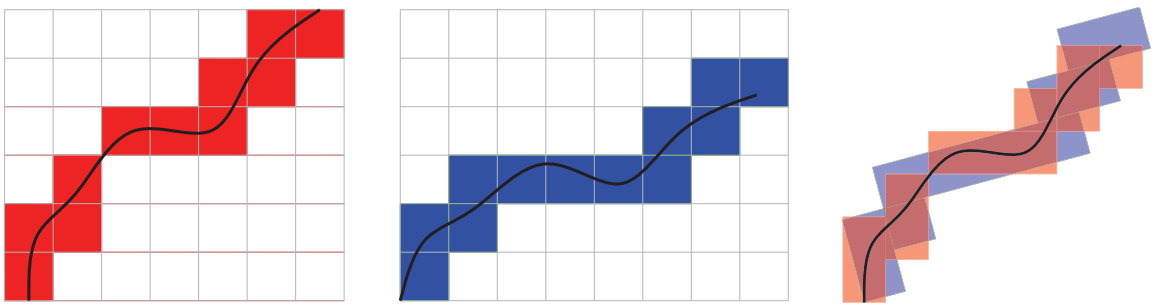
Sensor Discussion

8.1 The Focus of the Chapter

In this Chapter the focus is on research relating to the three major problems stated in Part I: Introduction. To analyze leaf shapes reliably, a minimum sensor resolution has to be determined. Then, after the resolution discussion, data acquisition is considered, and this is followed by the data model and data processing in a software engineering sense. The algorithms for segmentation and filtering the 3-D data are explained in detail.

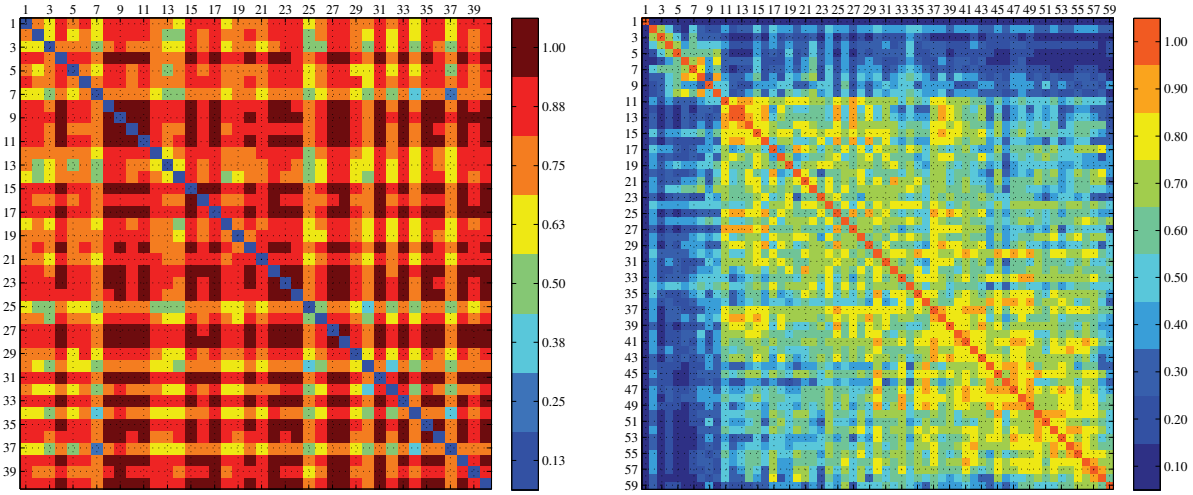
8.2 The Minimum Resolution

The first step towards a reliable plant detection algorithm is the determination of the required sensor resolution. For this purpose a representative set of different leaf shapes is collected; see Figure 3.3a. The shapes are artificial and there are no variations for the different shapes shown. To simulate data acquisition under real world conditions each of the shapes are rotated in steps of 15° in the interval [0°, 90°]. This rotation covers the digitization caused by the rotational pose change errors of the sensor, keeping the sensor resolution constant. The effect on the curve digitization due such rotation of 15° is illustrated in Figure 8.1. To soften the effect, bi-cubical interpolation is used to compute the gray values of the rotated shape. The aim of the analysis is to determine which forms differ significantly from each other; this is done by using elliptic Fourier descriptor invariants EFDs; see Eq. (3.14). Appendix A shows the set of major leaf shapes that are analyzed. Shapes are created with a vector drawing application; size and orientation alterations do not affect the shape form. For comparison, and to simulate the digitization process of the sensor, vector data is converted to the digital image format. The proportions of the shapes are kept constant. Table A.1 shows the shape sizes in mm. The largest shape width is about 60 mm, and the smallest about 25 mm. In Section 3.4.3 it is shown that the elliptic Fourier descriptors are translation, rotation and scale invariant. With rotation, shape pose changes relative to the sensor grid, so that different sensor elements will observe the signal and as a result, the pixel coordinates will differ; see Figure 8.1. Besides the digitization errors caused by the orientation of a shape to the sensor, the scale influence also has to be quantified. The scale of a shape on a sensor grid changes due to distance variations between the object and the sensor. To quantify the scale influence data separability must determined firstly. The borders of shapes from Figure 3.3a are analyzed by the EFD method. Each feature vector, $\mathbf{x}$ contains



(a) (b) (c)

Figure 8.1: Continuous Line Digitization: (a) Original (black) continuous line and its sensor values (red squares). (b) For 15° rotated continuous and corresponding sensor values (blue squares). (c) Superimposed sensor values with the original continuous line.



(a) **(b)**

Figure 8.2: Simplified Analysis of Feature Vectors: (a) Normalized Mahalanobis Distance between the Feature Vector (Classes). (b) Graphical Illustration of Correlation Coefficients Between Single Features.

invariants, e_k; see Eq. (3.15). Thus:

$$\mathbf{x} = \{x_0, \ldots, x_i\},\ i \in \mathbb{N} \tag{8.1}$$

is the set of coefficients to be analyzed. Thornton (**?**) says that feature separability can be precomputed and preselected to achieve optimum performance by a machine learning algorithm. The Mahalanobis distance (see Mahalanobis, 1936) is a straightforward computation of the weighted distance between feature sets. Although elementary, it gives a separability measurement between random vectors of the same size. In an ideal case the distances between features are equally large. To compute the Mahalanobis distance a variance-covariance matrix, C of the feature vectors is needed. C is a variance-covariance matrix of versatile Gaussian random variables:

$$C_{i,j} = E((\mathbf{x}^i - \overline{x}^i) \cdot (\mathbf{x}^j - \overline{x}^j)^t) \tag{8.2}$$

$$\bar{x} = E(\mathbf{x}) \tag{8.3}$$

where $E\{\cdot\}$ is the expectation operator. Then the Mahalanobis distance between the feature vectors, d is (see Mahalanobis (1936)):

$$d_{i,j} = \sqrt{(\mathbf{x^i} - \mathbf{x^j})^t\, C^{-1}\, (\mathbf{x^i} - \mathbf{x^j})},\ i, j \in \mathbb{N} \tag{8.4}$$

where $d_{M_{i,j}}$ becomes the Euclidean distance for $C = I$.

For every feature vector of the respective shape illustrated in Figure 3.3a, one-to-all Mahalonobis distances are computed. Figure 8.2a is a graphical illustration of the resulting distances, normalized to the largest one. The distance itself allows a coarse estimation of the separation possibilities between the feature vectors. Additionally it is important to see the correlation between each feature in the feature vectors set. Figure 8.2b shows that the higher order features (EFD coefficients) correlate more strongly than the lower order features. Thus, the difference between the shapes is the largest when considering their basic geometric shape (circle, ellipse, etc.): the information is stored in lower frequency terms. The higher frequencies are more sensitive to digitization errors and are directly affected by the sensor resolution, so they have to be approached with a higher degree of caution. Besides the Mahalonobis distance between the feature vectors, the angle between each feature vector also gives the separation property of the set:

$$\phi_i = \arccos\left(\frac{(\mathbf{x^i})^\mathbf{t} \cdot \mathbf{x^j}}{|\mathbf{x^i}| \cdot |\mathbf{x^j}|}\right),\ i, j \in \{1, \ldots, 40\} \tag{8.5}$$

Remark: The dimension of the feature vector is $\dim(\mathbf{x}) \geq 2$; the drawback of the reduction of multidimensional data onto a plane is that there is significant information loss. For illustration purposes however, the distance and angle are optimal quantities.

The feature vectors are shown in Figure 8.3 and the scaled angle and distance graph in Figure 8.4. The angles ϕ_i are stretched to the interval $[0,\ 2\pi)$ to provide a slightly tidier graph:

$$\phi_i^{'} = \phi_i \frac{2\pi}{\max(\phi_i) - \min(\phi_i)} - \min(\phi_i) \tag{8.6}$$

For clarity, the graph in Figure 8.3 has been reduced to the first 12 features[1]. A basic clustering approach is used to obtain separability between feature vectors. Thus, k-Means clustering with precomputed class centroids is applied: the class means are computed from the feature vectors and thereafter k-Means clustering is applied to the feature set. The prior computation of approximate class means is considered as a "training" process. This kind of training process is unfair since the same data is used for training and classification. The influence however, is minimal due to the nature of the k-Means method; for details about the k-Means method see MacQueen (1967).

8.2.1 The Results of k-Means Clustering

The shapes analyzed are shown in Figure 3.3a. Further, Figure 8.4 shows how small the distance between similar shape classes are; the situation worsens with the presence of the noise, which blurs already tight separation boundaries. So feature selection is crucial for reliable classification; hence care must be taken when analyzing the data. The figure also shows the distribution of particular feature vectors around the estimated centroid of the set.

The cluster centroids in Figure 8.4 are indicated as $\otimes$ and their respective classes are uniformly color-coded. Besides not being a particularly robust method, k-Means clustering is considered as an elementary unsupervised classification method. The quality of the results depends on the initial centroid values which can differ with each run. Figure 8.4 displays a result of multiple runs, where final centroids displacements do not exceed 1% of initial values.

It is expected that the error rate (false classified feature vectors) will be equal to or better than 20%. The results obtained however, are significantly better than expected, with an error rate of 2.5%; as a real world example this can be considered a major breakthrough. Full-scale experiment results are listed in Section A.4.1.

Sensor resolution plays a key role when it comes to the digitization of shapes. In-plane shape rotation and subsequent EFD computation, deliver different results for each new angle. The cause for the variation is the relative orientation of the object to the sensor raster that digitizes it.

The optimum sensor resolution for the shapes under consideration is $508\,dpi = 20\,px/mm$. Every pixel has diagonal of $0.071\,mm$. Complex leaf types like pinnate, bipinate and tripinate contain tiny connections to the petiole. They can "lose" this connection during the interpolation procedure and this can change their shape significantly. These leaf types have the largest variance in this experimental set and cannot be distinguished from each other by use of the k-Means method; see Figure 8.6.

The next task is to determine how invariants of the same shape change if the sensor resolution decreases. To illustrate this, sample shapes from Figure 3.3a are re-sized in steps of 10% to simulate a decrease in sensor resolution. The resulting elliptic Fourier descriptors are then compared to the original descriptors. The method of k-Means clustering is repeated for each resolution. The results of all distance computation methods are illustrated in Figure 8.5.

[1] Total number of features used for analysis is 59.

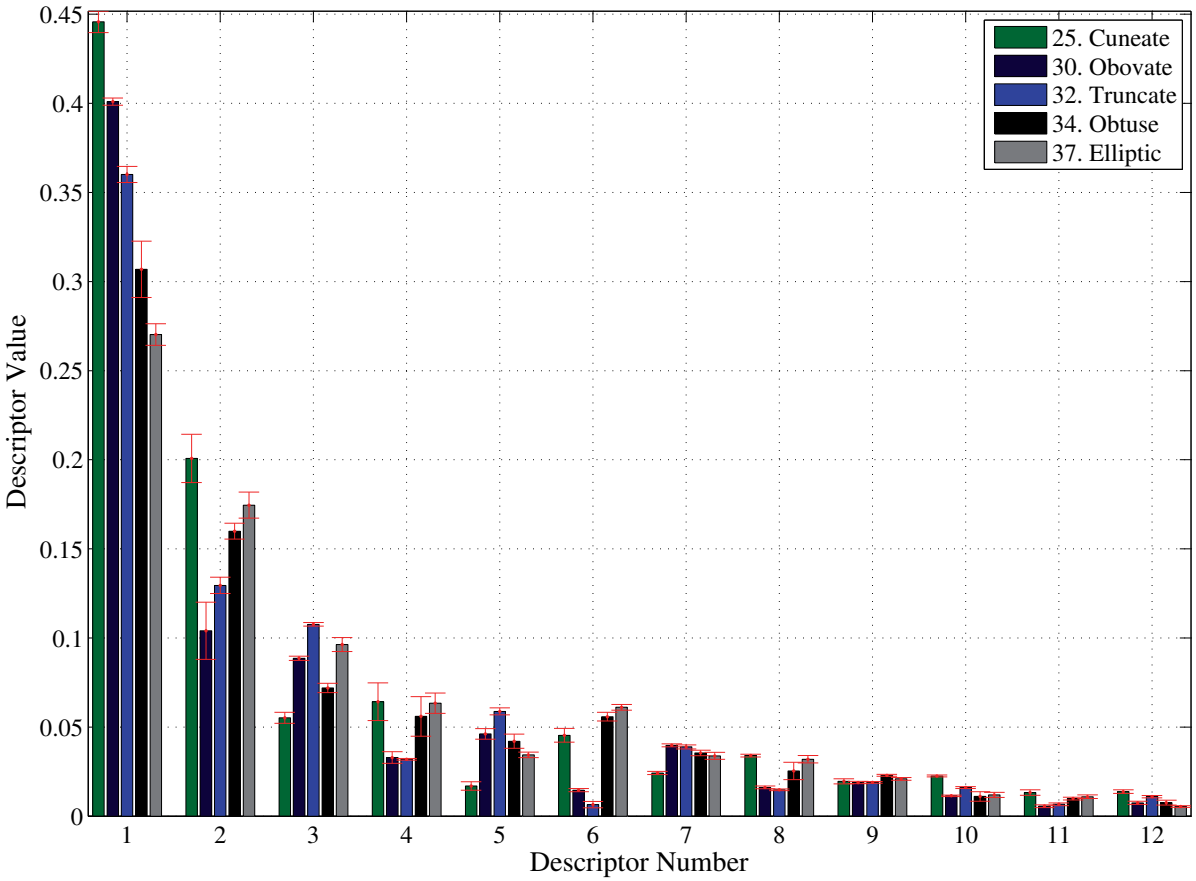


Figure 8.3: Elliptic Fourier Descriptors of Similar Simple Leaf Shapes.

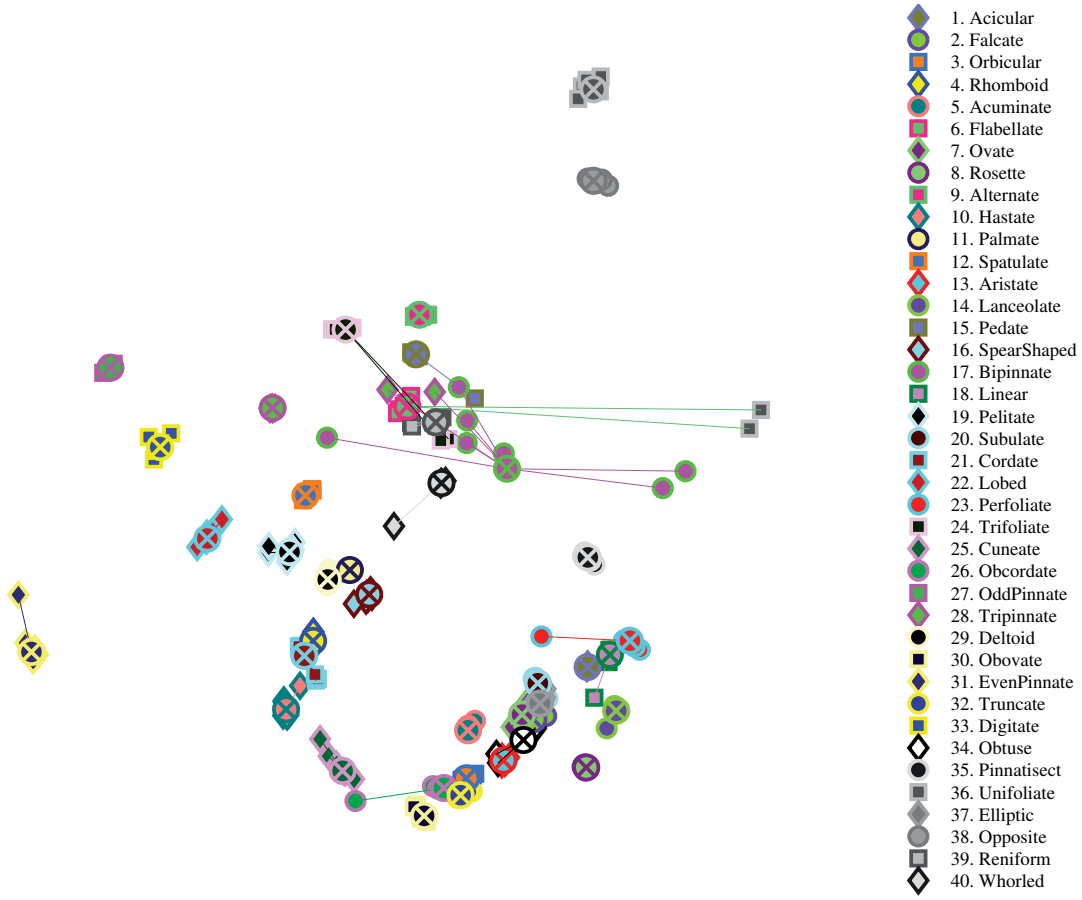


Figure 8.4: Distance and Angle Graph with k-Means Clusters.

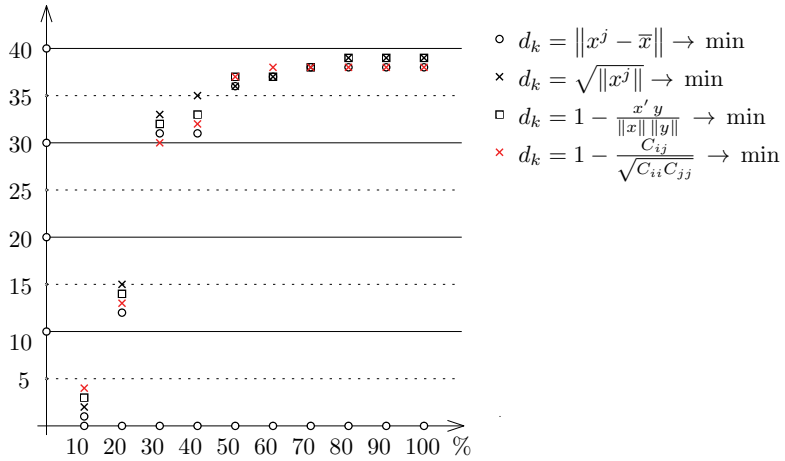


Figure 8.5: k-Means Results; $x-$axis: simulated relative sensor resolution in %; $y-$axis: number of correctly distinguished leaf shapes. For each distance computation method different marker and color are shown.

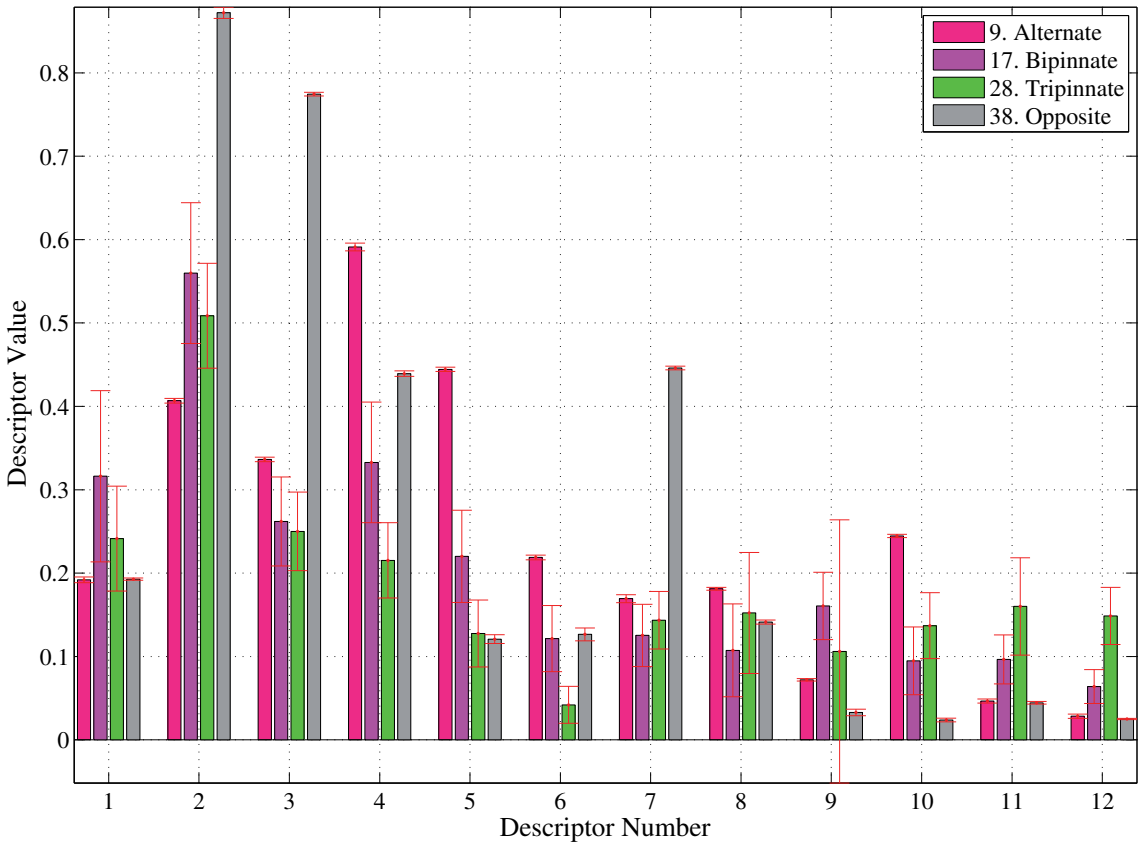
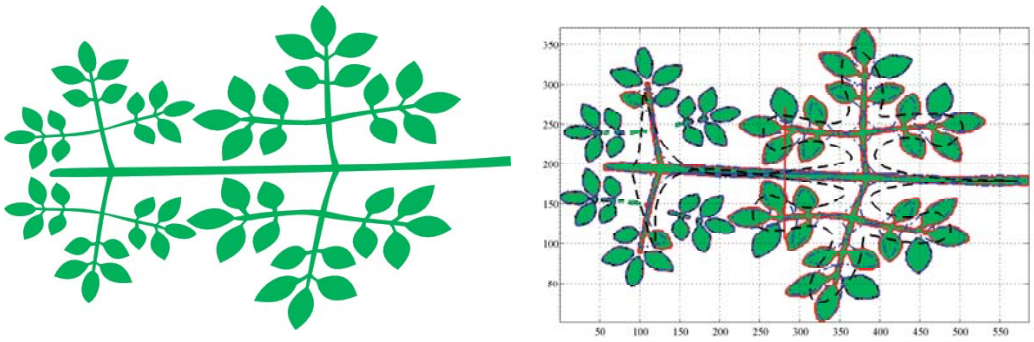


Figure 8.6: Elliptic Fourier Descriptors for Complex Leaves. Large standard deviation of the features is caused by digitization errors: the tiny connections to petiole disappear in the digitized image. The shapes *9.Alternate* and *38.Opposite* are not affected: the connection is sufficiently thick to not to be affected by the sensor resolution.



(a) (b)

Figure 8.7: Tripinate Shape and Boundary Approximation: (a) Original Shape. (b) Boundary approximation: red solid - longest boundary extracted; black dashed - classification boundary (20 harmonics); blue dashed - reconstructed boundary using 59 harmonics.

Feature	Value / Constrain
Acquisition Speed	$1.4 \geqslant v_a \geqslant 2.8\, m^2 s^{-1}$
Resolution	$\sim 3 \cdot 10^{-2}\, mm^2 \leqq$
Accuracy	$\sigma = 0.01\, mm$
Range: width, height	$1.0\, m \geqq$, $1\, m$
...	...

Table 8.1: Excerpt of Sensor Requirements from Table 2.1 with Reviewed Values.

8.3 Conclusion

In previous text it was shown that it is possible to classify a leaf by analyzing its boundary. Furthermore, with the use of a basic classification approach, in this case k-Means clustering, acceptable results were achieved: a 2.5% error at the highest resolution. However, the methods of boundary analysis fail when applied to complex leave shapes: i.e. 28.-Tripinate $\Leftrightarrow$ 17.-Bipinate - see Figure 3.3a, Table A.2. The cause of the error is not in the method itself; insufficient sensor resolution causes the filigrane structures to disappear and hence the shape "loses" its key features because of a significant change in its data: see Figure 8.7.

The lowest sensor resolution necessary to keep classification error at 2.5%, is just below $400\, dpi \approx 15.748\, [px\, mm^{-1}]$; thus the diagonal length is $0.09\, mm$. These values are relative to the size of the objects that are analyzed. When determining the sensor resolution for field use, it is necessary to know or estimate the size of the smallest shape that will be observed. Using the classification method just presented, a resolution of $\sim 16\, pt/mm$ is necessary to achieve a distinction rate as was achieved in this experiment. For a coarse estimate of sensor resolution, R, one can use the following rule of thumb:

$$R = \frac{25}{O} \cdot 15.748 \sim \frac{400}{O}\, [px/mm] \tag{8.7}$$
$$O = \text{minimum object size in } [mm]$$

In addition, it expected that natural plants or samples will have a larger variance and that the separation of plant species will be more difficult; see in Part IV. Hence, a high reliability classification process cannot exist without high-quality data. To acquire such data a high-resolution and high-performance sensor is necessary.

8.4 Data Acquisition and Sensor Discussion

With the knowledge obtained and the rule of thumb given above (Eq. (8.7)), it is obvious that the required sensor resolution is just below $400\, dpi$ for objects of a size $\geqslant 2.5\, cm$. At this resolution species can be distinguished by their boundary description; see Table A.2. Which sensors are able to achieve such a high resolution? Starting with a common sense consideration: an off-the-shelf flatbed scanner achieves $9600\, dpi$ and costs about 110 USD. Cameras and flatbed scanners provide intensity images with known limitations; see Taubin (1991); Viau et al. (2005). Therefore a 3-D sensor with a resolution of $400\, dpi$ ($\sim 16\ pt/mm$) or better, is required. This is a significant change from the requirements initially formulated in Section 2.1; see Table 2.1.

Keeping in mind the sensor requirements from Table 8.1, consider the measurement principles of such a sensor. Figure 8.8 illustrates a sensor and its measurement area. The aim is to acquire the surface, Φ in 3-D within a

Acquisition Performance $[m^2 s^{-1}]$	Points $[Mpx]$	$T_{pt}[ns]$	$T_{row}[\mu s]$
1.4	358.4	2.79	52.82
2.0	512.0	1.95	44.94
2.8	716.8	1.40	37.35

Table 8.2: Required Sensor Performance.

set time slot and at the required resolution. Temporarily ignoring the accuracy requirement, the focus lies on the sensor resolution and how to achieve the it. The sensor can be one of these three types (A, B and C):

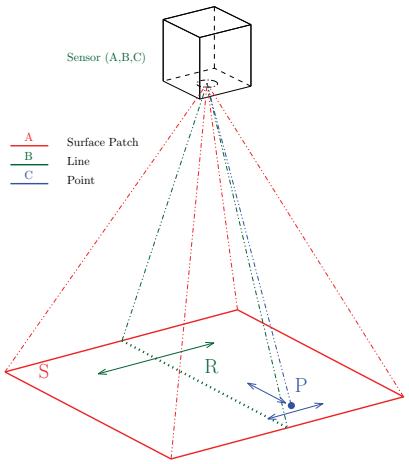


Figure 8.8: Sensor Scheme.

A The sensor acquires the surface patch and produces a 3-D point cloud of the measurement area. This is a typical range image sensor. This kind of sensor does not need any kind of translation or rotation to acquire the measurement area.

B The sensor acquires one line of the surface at a time and produces a profile of part of the surface patch. To acquire the whole surface, the sensor must move across the patch. Alternatively, the sensor must have an optical and/or a mechanical device that replaces this translation (in Figure 8.8 the green line $\longleftrightarrow$). This kind of sensor needs at least one translation or one rotation during data acquisition.

C The sensor acquires one point of the surface at a time. To acquire whole surface, the sensor must move in two directions across the patch. Alternatively, the sensor must have an optical and/or a mechanical device that replaces these translations (in Figure 8.8 the blue lines $\longleftrightarrow$).

In order to provide 3-D point clouds, all three sensors need to measure *distances* and *angles.*

The right approach for sensor selection is to setup a morphological box containing the measurement methods and then to choose the optimal combination from it. In the first step the methods selection must be narrowed down in accordance with the requirements for data acquisition as given in Table 8.1. The first aspect that must be considered is the amount of data that must be acquired per second: $1.4 \geqslant v_a \geqslant 2.8\, m^2 s^{-1}$ at a resolution of $16\, px\, mm^{-1}$. Table 8.2 shows the implications of the sensor requirements from Table 8.1. The first column in Table 8.2, *Acquisition Performance* quantifies rate of data acquisition as a area unit per second. In the second column, *Points,* the number of points (mega pixels) for the each area is quantified. Currently the highest portable camera resolution is 80 Mpx which is 4.5 times less than necessary to capture an area of $1.4\, m^2$ at the required resolution.

The last two columns show the single point cycle (T_{pt}) and row cycle (T_{row}): given a sensor that acquires the data through single point measurement, then the sensor must acquire each point within a time interval that is less or equal to $2.79\, ns$. In this time period *light* travels[2] $0.84\, m$. A typical high-end laser ranger such as the Z+F PROFILER 9012 acquires data at $\sim 41\, kpx\, s^{-1}$. A suitable sensor for the plant recognition purposes is derived from the sensor requirements and Table 8.2. Furthermore, the sensor platform must be able to acquire data during movement, so the sensor must be suitable for a mobile platform. Considering the layout presented in Figure 8.8 and the measurement principles described above, the sensor properties are obtained using the following formulae:

[2]In this case the speed of light in air is by index of refraction1.00029 , $299\,795\,543.4\, m\, s^{-1}$. The speed of light in vacuum is $299\,792\,458\, m\, s^1$, see SI (2012).

$$d_{max} \cdot m_s + m_c \leqslant r \tag{8.8}$$
$$d_{max} \cdot \Delta\alpha_{rad} \leqslant r \tag{8.9}$$

d_{max}: The maximum distance between the sensor and an observed point $[m]$.

m_s: A scale factor of one distance observation accuracy.

m_c: A constant distance observation accuracy; an additive constant $[m]$.

r: The required resolution $[m]$.

$\Delta\alpha_{rad}$: The angle resolution $[rad]$.

The technical realization of a sensor is not considered here. Though, the optimum resolution is $16\,px/mm$ and the minimum acquisition performance at this resolution is $1.4\,m^2s^{-1}$.

Optomechanical Time of Flight (TOF)/ Phase Difference (PD) 3D Sensor (Sensor Type C)

Considering Table 8.2, the realization of a sensor that measures polar coordinates: angles, α and ϕ and distance, $d_{\alpha,\phi}$, is not feasible with current state-of-the-art technology. The measurement frequency is $\sim 358\,MHz$. Without elaborating on the potential device in detail (such device does not yet exist) its maximum range would have to be $0.42\,m$.

Optical 3-D TOF/PD & Geometric Sensor (Sensor Type A)

The angle measurement for this sensor type is based on central projection geometry: pin hole camera. Therefore angle measurement, strictly considered, is a look-up table derived from the camera principal point, a camera constant and imaging sensor geometry; see Kraus (2007); Luhmann (2010).

TOF/PD The sensor acquires angles, α and ϕ relative to the optical axis and distance, $d_{\alpha,\phi}$ relative to projection center of the system. Angles are determined by the sensor aperture and its geometric properties (pinhole camera). Distance measurement can be feasible with the TOF and/or the PD method. The resolution is limited by the optics and the resolution of the CCD/CMOS sensor used, as well as the distance to the object. Such a sensor exists and is called a "Time-Of-Flight Camera" or a "Ranger Camera". Current realizations of such a sensor provides low resolution devices of $176 \times 144\,px$; see Mesa (2011).

Geometric The measurement principle is based on straight intersection by known sensor object geometry. This is a typical photogrammetric problem: stereo camera or structured light systems. The resolution is directly proportional to the distance between the sensor and the object. The sensor is modeled as a pin-hole camera.

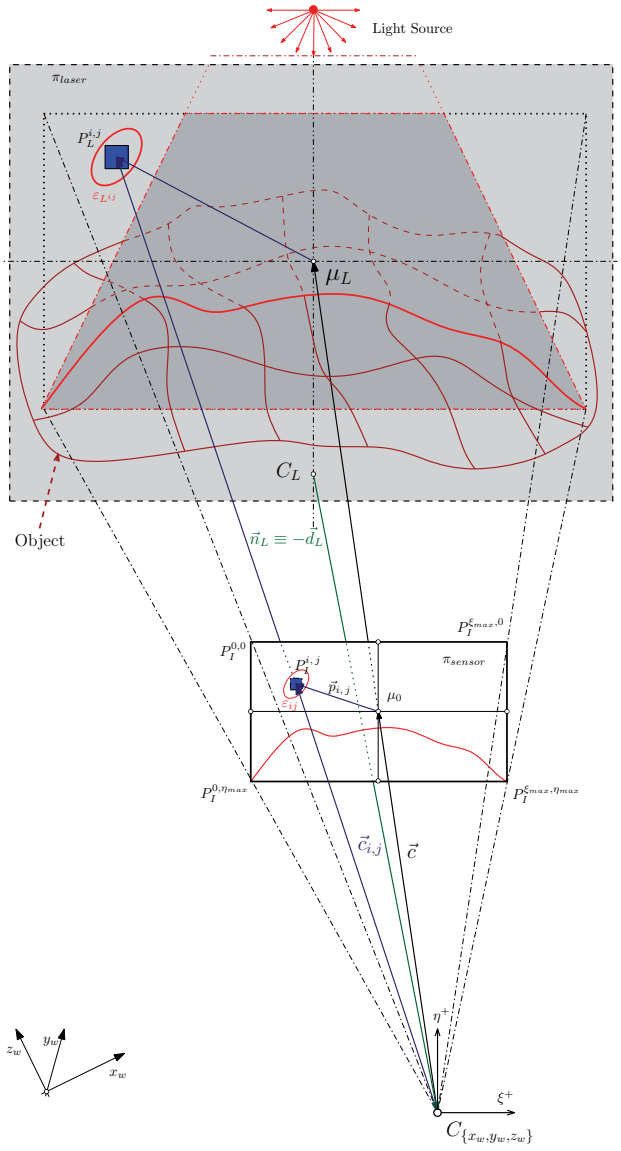
Plenoptic Camera A plenoptic camera is a regular camera with an array of microlenses between the objective and sensor. The first commercial plenoptic camera is provided by tha Raytrix GmbH http://www.raytrix.de/. In Perwass and Wietzke (2012) a general overview of the technology and results are presented.

Optomechanical TOF/PD 2-D Sensor (Sensor Type B)

At current state-of-the-art technology it is not feasible to physically manufacture a sensor, that fulfills the requirements listed in Table 8.2; see "Optomechanical Time of Flight (TOF)/ Phase Difference (PD) 3D Sensor (Sensor Type C)" above for explanation.

Geometric: Laser Triangulation

Of all the geometric methods for the profile measurement, laser triangulation is applied in most cases. It neither requires moving parts, nor is it limited in its accuracy or speed. The sensor system consists of two distinct parts: a light source and the light sensor in a wider sense (sensor array or matrix, such as a CCD or CMOS sensors). The laser triangulation measurement method uses a basic measurement principle to acquire the profiles of a 3-D object at almost any resolution. Figure 8.9 illustrates the situation. Vector notation is selected for the description because it is more intuitive than trigonometric notation. Furthermore vector notation is used in the majority of computer vision and computational geometry publications; see Hartley and Zisserman (2004). A comprehensive introduction to the area of computational geometry area is provided by Erickson (2011). There are also two specialized compendiums on the topic: Schneider and Eberly (2002); Agoston (2005).



π_{sensor}	Sensor plane in world coordinate system (WCS).
π_{laser}	Laser plane in WCS.
μ_0	Projection of cameras principal point to the sensor plane. Projection vector of the point $C_{\{\}}$ is $\vec{c}$.
μ_L	Intersection point between the straight defined by $\overrightarrow{c}$ and π_{laser}.
$\overrightarrow{c}$	Projection vector that is a normal vector to the sensor plane: $\overrightarrow{c} \perp \pi_{sensor}$. Further, the direction of the $\overrightarrow{c}$ in WCS determines the orientation of the sensor in that space.
P_I^{ij}	Points in the sensor plane.
P_L^{ij}	Points in the laser plane.
$\vec{n}_L$	Normal vector to the laser plane.
$\vec{d}_L$	Projection vector of the camera principal point onto the laser plane.
$C_{\{\}}$	Pin-hole camera principal point in WCS.
C_L	Projection of the sensor origin to a point projection on the laser plane.
ε_{ij}	Uncertainty of the projection vector $\vec{c}_{ij}$.
$\varepsilon_{L^{ij}}$	Uncertainty of ϵ_{ij} projected onto the laser plane.

Figure 8.9: Laser Triangulation.

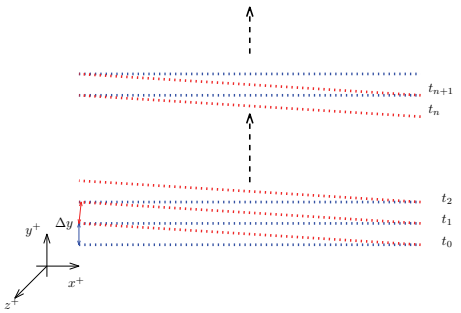


Figure 8.10: Profile Observation.

Conclusion

The optimal sensor for data acquisition is a surface-acquiring sensor such as a TOF camera. The simultaneous acquisition of an underlying object has the advantage that the resulting point clouds represent independent measurements, and the neighbor-relation between points belongs to an identical measurement epoch. The data acquisition process does not differ from airborne photogrammetry; the areas have to overlap by at least 30%. At present there is no available sensor on the market that delivers a 400 dpi depth image for a 1 m^2 area and a rate of $5-10\,Hz$. TOF cameras deliver resolutions that fall far short of the required resolution and acquisition rate; the acquisition frequency however is fair for a static system. The required resolution-improvement factor with respect to current TOF cameras (for example Mesa (2011)) is >13 000×. It is thus evident that the required 3-D sensor is currently not available off-the-shelf.

The 3-D LIDAR sensor marketed by Z+F (2010) almost fulfills the required angle resolution, but fails the acquisition performance requirement by far; the required resolution is 0.0017° and the sensor provides 0.0018° which is ∼ 6% less. According to the data sheet the observation time for one square meter is 13′ 28”. This long observation time rules out the use of dynamic carrier used for real-time, continuous observation and treatment, though its specifications do satisfy requirements of an eventual stop-go system.

Obviously current performance is not the last word in laser scanner technology and there may be suitable sensors in the near future. At present sensor type B probably represents the optimal selection for real-time plant recognition from 3-D point clouds. To acquire 3-D data, translation of the sensor is necessary. The sensor principle point position and sensor orientation must be known for every observation. In Figure 8.9 these parameters are gathered in $C_{\{\}}$ and $\vec{c}$.

Figure 8.10 illustrates the difference between a laser triangulation sensor and a 2-D LIDAR sensor. The red dots represent points observed by a LIDAR sensor, translated with constant velocity and orientation. The blue dots represent points observed by a laser triangulation sensor. The spatial distance between the measurement epochs is constant: $\Delta y = const.$ While a triangulation sensor delivers profiles with a constant angle to the translation vector, the angle of the LIDAR profile changes with the velocity; the higher velocity, the steeper the angle. To achieve a result equal to a triangulation sensor, the rotation speed of the sensor unit must adapt to the translation speed of the sensor itself.

To obtain an efficient, affordable and flexible experimental platform, the laser triangulation sensor is selected; see Section 10.2. For data acquisition a laser-line light source and a high resolution CCD or CMOS are suggested for following reasons:

1. The resolution of a sensor is almost arbitrary. The spatial resolution can be altered by varying the following parameters:
   (a) The distance to the object.
   (b) The CCD or CMOS sensor resolution.
   (c) The lens system.
2. Each point in a profile is acquired simultaneously. Position and orientation of the sensor is identical for all points within the profile.
3. The data acquisition speed depends on the camera system used, $20\,kHz$ is already realized in several systems; one of them is the Sick IVP camera Sick (2013).

The system and the functionality of its core components are briefly described in Part IV. The proof of the data acquisition concept with the experimental system is explained in Section 10.2. In the remaining chapters the data model and data processing of a depth image acquired on a row sensor, will be elaborated; processing is completely independent of sensor type. Each depth image of the required resolution and accuracy (see Section 8.3) is suitable for processing by application of the algorithms explained subsequent to this section.

Chapter 9

Data Processing

9.1 The Data Model

The data model is an abstraction of the real data, such as observations or results. It is designed for efficient storage and processing algorithms. Data model design is the first step in the application design process. The computer algorithms described in this thesis are subroutines of the application that processes the data. The data is organized into a structure defined by the data model. The explanations that follow cover only those parts of the data model, that are required for an understanding of the algorithms.

Data *structures,* defined by the data model, store the data into a data *container*. Containers' internal organization differs, dependent on their purpose; commonly encountered containers are sequence containers, and associative containers and their adaptors. Associative containers require a relation between the data elements, to store the data in an ordered way. Some containers, such as a `std::map`, require the ordering value separately; a unique identifier called a *key*, allows algorithms to quickly access the elements in the container. The key is ordered: $std :: map < T_1, T_2 >$ T_1 is a key, and T_2 is a value. T_1 is ordered for the case when there is an operator $< (T_1, T_1)$ with $t_1 < t_2$ or $t_2 < t_1$ or $t_1 = t_2$. Most container implementations in C++ are template-based and are derived from one of the basic containers of the standard template library (STL); see Meyers (1995); Josuttis (1999); Stroustrup (2000); Alexandrescu (2001); Meyers (2001); Sutter and Alexandrescu (2004); Dewhurst (2005); Meyers (2005).

The data model is designed to aid efficient access to each data structure, which is necessary for the algorithms to work optimally. The container is designed to store any type of range image that is organized as a regular grid; see Figure 9.1. The unique key for each data structure is *time*. A ranger sensor delivers its specific *raw data*, and hence the container for storing particular raw data, differs from sensor to sensor. The procedures shown in this thesis are designed to process a *range* or a *depth* image. The range image geometry is defined as follows: x and y build a regular grid; z is a scalar that contains the distance of the point to the $xy-$ plane. The container that stores the depth image values, has a matrix form and is denoted later in the text as: $\mathbf{B}_{m,n}$, $m \wedge n > 0$, where m and n are matrix dimensions that map the x and y coordinate axes. From a software engineer's point of view, the container provides two main features:

1. Efficient and well-established storing capability in matrix form.
2. Fast and intuitive data access, again in matrix form.

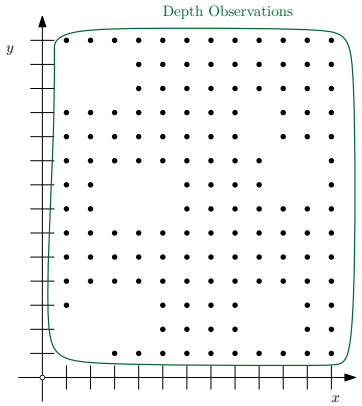


Figure 9.1: Range Image.

The segmentation, feature extraction and shape analysis algorithms here, are also designed and implemented to work on the *range* image container mentioned above. Only the very last step of processing requires real spatial information, which is then derived from the range image; see Section 9.2. The known relationship between the range and spatial data types allows for bidirectional updating of results in both representations; they are treated as equals. The synthesis of a range image is sensor independent; the resulting range images do not differ even if they originate from different sensor types; a row-based depth image synthesis is described here. The sensor acquires the area by collecting the profiles denoted as $\mathbf{r}_t$, where i is a *unique index*:

$$\mathbf{r}_i = \{i, o_0, \ldots, o_k\},\ i,\ k \in \mathbb{N},\ o \in \{0, \ldots, c_{max}\} \tag{9.1}$$

where:

i Row *unique index*. Two rows cannot have the the same value as an observation time; the *index* axis is collinear with the $y-$axis in the range image.

o_k Depth value. o_k is collinear with $z-$axis of the range image; hence the k index is collinear with $x-$axis of the range image, where $k \in \{0, \ldots, k_{max}\}$. k_{max} is the sensor's horizontal resolution. The value range of $o = \{0, \ldots, o_{max}\}$, $\forall o_{n+1} - o_n = const.$, is determined by sensor's vertical resolution; see Figure 8.9.

The *point cloud* is derived from the depth image, and its container is now considered: the model defines storage for a *set* of discrete point observations, denoted here as $\mathcal{P}$, where $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_i\}$, $i \in \mathbb{N}^+$, so that the chronological order of the observations is stored as well; each observation time is represented by the index i. The *time stamp* t is derived from the unique index i. Thus, for each point, $\mathbf{p}$:

$$\mathbf{p}_t = \{\mathbf{x}, t\},\ t \in \mathbb{R}^+ \tag{9.2}$$

$$\mathbf{x} \in \mathbb{R}^n \tag{9.3}$$

$$\mathbf{x} = \{x_1, \ldots, x_n\},\ n \in \mathbb{N}^+ \tag{9.4}$$

The index of vector, t, represents the *epoch* of the measurement in milliseconds, $\mathbf{p}$; the index n denotes the point dimension, and in the case elaborated here, $n = 3$. Each $\mathbf{p}_t$ stores *spatial* information $\{x, y, z\}$ together with its *time stamp* t.

Obviously, each sensor type provides differently organized data, and therefore the raw data container and a raw-to-depth data converter must be implemented for each sensor type, if the algorithms provided here are to be reused. The range image container is: $\mathbf{B}^{\mathcal{R}} = \mathbf{r}_1 \cup \mathbf{r}_2 \cup \cdots \cup \mathbf{r}_t = [\mathbf{t}\,\mathbf{O}]$ where $\mathbf{t}$ is a time vector and $\mathbf{O}$ is a depth matrix; the matrix form of the container is:

$$\mathbf{B}^{\mathcal{R}}_{(m,k)} = \begin{pmatrix} \mathbf{r_0} \\ \mathbf{r_1} \\ \mathbf{r_2} \\ \vdots \\ \mathbf{r_m} \end{pmatrix} = \begin{pmatrix} t_0 & o_0 & o_1 & \cdots & o_k \\ t_1 & o_0 & o_1 & \cdots & o_k \\ t_2 & o_0 & o_1 & \cdots & o_k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_m & o_0 & o_1 & \cdots & o_k \end{pmatrix},\ m,\ k \in \mathbb{N} \tag{9.5}$$

The container, $\mathbf{B}^{\mathcal{R}}_{(m,k)}$ is able to store a set of one-dimensional vectors, $\mathbf{r}$. The range data provides raw information delivered by the sensor; the algorithms however need additional information space to store results and apply logical operations, just to mention two requirements. Such information is stored in a *complex data* type[1] or structure that also contains the depth information. Thus, one can write: $\mathbf{B}^{\mathcal{R}}_{(m,k)} \subset \mathbf{B}^{\mathcal{E}}_{(m,k)}$, where $\mathbf{B}^{\mathcal{E}}_{(m,k)}$ is the complex data container (see Stroustrup, 2000, pages 87 ff.), holding the following information:

$$\mathbf{B}^{\mathcal{E}}_{(m,k)} = \begin{bmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,k} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m,1} & d_{m,2} & \cdots & d_{m,k} \end{bmatrix} \tag{9.6}$$

The main advantage of this solution is that the structure of the container does not change if the data type, $d_{m,k}$ is altered. The container has a geometry that is obviously a matrix, so that algorithm access to each

[1] Complex data types are defined differently for each programming language. Although the differences are small, they might cause porting problems. Here all the data structures imply the C++ programming language. The complex data type described in text is implemented as a structure (keyword *struct*) in the C++ programming language.

Type	Name	Comment	Range
integer	z_n	Observed height value.	$z_{min} \leqslant z_n \leqslant z_{max}$
double	$\mathbf{x}$	Vector of corrected Cartesian coordinates.	$\mathbf{x}_{min} \leqslant \mathbf{x} \leqslant \mathbf{x}_{max}$
unsigned integer	C	Label of the observation.	$1, \ldots, 2^{32} - 1$
integer	E	Edge type.	0: no edge, 1: edge, 2: discontinuity edge, ...
bit-field	f	Different flags for logical operations.	-

Table 9.2: Complex Data Type: $d_{m,k}$.

element is via two indices, m and k. The method of accessing the elements gives all the advantages of matrix operations, and additionally the complex data structure provides the additional storage for the specific needs of the algorithms; an excerpt of such a structure is shown in Table 9.2. The algorithms that operate on the container are implemented as *template functors* (see Josuttis, 1999, Chapter 5.9 Function Objects for detailed explanation), so operations on the data and data storage are fully separated; *generic programing* is the key-phrase describing this programming paradigm. Having the data model separated from the processing model, it is easier to modify or extend each part of the application separately with minimal errors. Consider two containers: the source, $\mathbf{B}^S_{(m,k)}$ and the destination, $\mathbf{B}^D_{(r,j)}$, and a functor, f that processes the data it reads from the source, and then stores the results in the destination container:

$$f : \mathbf{B}^S_{(m,k)} \rightarrow \mathbf{B}^D_{(r,j)} \tag{9.7}$$

This kind of operation does not change the content of the source. As opposed to Eq. (9.7), the in-place operation does change the source, because the source and destination container are the same; so $f : \mathbf{B}^S_{(m,k)} \rightarrow \mathbf{B}^S_{(m,k)}$.
The arbitrary computer algorithm, A is a set containing at least one function f. Ideally the processing algorithm is implemented as a template function or a template functor. Both the function and the functor involve generic implementation of processing steps that do not require previous knowledge of the underlying data being processed. In practice such implementation for complex problems such as the one described in the text, require longer development and even longer testing. Therefore the template functions and functors in most cases are "primitives" that realize computations such as basic trigonometric functions, matrix and vector operations, etc.

9.2 Data Processing Principles

9.2.1 Preprocessing

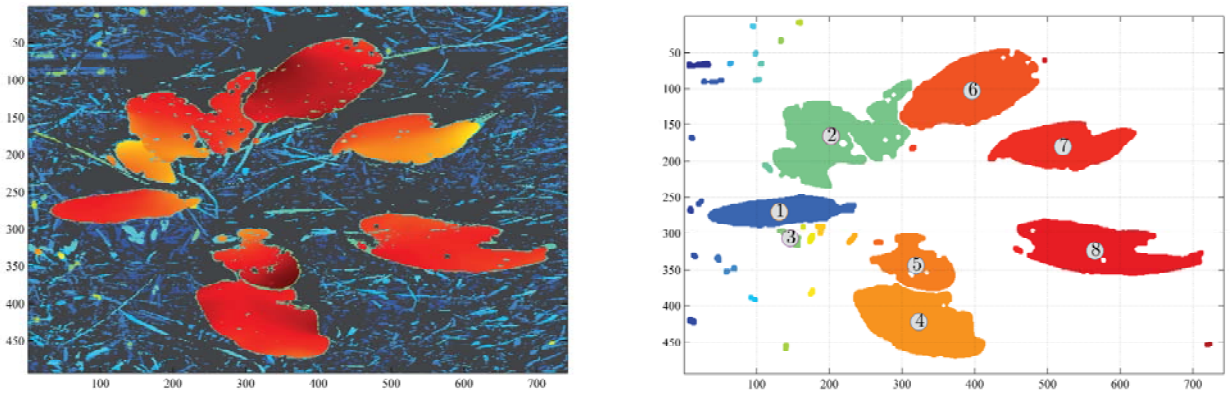
Preprocessing encloses a set of standard smoothing and segmentation algorithms that are mainly used to separate regions of interest (ROIs) from the rest of the data. Regions of interest contain a high density of observations compared to the very sparse observations of areas which are not of interest. Each extracted ROI is an object of more intensive analysis in the main processing. The preprocessing process is briefly described here to provide a seamless description of the processing flow.

9.2.2 Low-level Feature Extraction

The basic segmentation algorithms require noise removal to achieve reliable results. In the first step, noise removal is based on standard noise removal algorithms; this involves a relatively low number of simple operations and therefore a high execution speed. Since data preprocessing is a preliminary step to more sophisticated processing, a major goal is to keep the data as close to original as possible. The second goal is the computation of *coarse* disjoint sets from observed data: low level segmentation. These two goals are met by the following:

1. Low-pass filtering of the data through a median filter. The window size and shape is computed during the calibration of the sensor and depends on the surface patch threshold. Larger windows are used if broad-leaved plants are the subject of recognition, and smaller windows are used for tiny structures. For performance purposes, the M-*th* largest element selection algorithm is used; see Press et al. (2007, Page 431 ff.).

2. Low-level segmentation, and morphological edge detection.



(a) (b)

Figure 9.2: Standard Segmentation Results.

3. Low-level segmentation, and edge detection based on surface normals, explained in next section.
4. Connected components labeling. Extraction of the ROIs from the rest of the data; each region that covers a larger area than a threshold, is queued for the further analysis.

In this case the aim of the pre-processing is to remove noise from the data and prepare it for the filtering steps and low-level feature extraction (edges and connected components). Generally only a fraction of the acquired data contains relevant information. To increase the performance of the processing procedure, areas of interest are localized.

Edge Detection Using Triangle Normals

Triangle normals give sufficient information to achieve fast edge detection, which is more sensitive than that obtained from standard methods. Especially when analyzing regular networks, triangle normals provide better results; see Šeatović (2008); Šeatović et al. (2010). Below are the main characteristics of the algorithm:
Let:

$$P_i,\, P_i^j \in \mathbb{R}^3 \tag{9.8}$$

Let $N_i = \{P_i^0, \ldots, P_i^7\}$ be the point neighborhood of P_i:

$$N_i = \begin{array}{ccccc} P_i^0 & & P_i^1 & & P_i^2 \\ & \nwarrow & \uparrow & \nearrow & \\ P_i^7 & \longleftarrow & P_i & \longrightarrow & P_i^3 \\ & \swarrow & \downarrow & \searrow & \\ P_i^6 & & P_i^5 & & P_i^4 \end{array} \tag{9.9}$$

The point, P_i builds a surface patch with its neighbors:

$$\begin{aligned} j &= \{0,\, 1, \ldots,\, 7\} \\ k &= \{0,\, 2,\, 4\} \\ i &= const. \end{aligned}$$

$$\mathbf{n}_j = (P_i^j - P_i) \times (P_i^{(j+1) \mod (8)} - P_i) \tag{9.10}$$

$$\mathbf{n}_k = (P_i^k - P_i) \times (P_i^{(k+2) \mod (8)} - P_i) \tag{9.11}$$

The point, P_i is an edge point when the following conditions are satisfied:

$$\cos(\varphi_j) = \frac{\mathbf{n}_j \cdot \mathbf{n}_{(j+1) \mod (8)}}{\|\mathbf{n}_j\| \left\|\mathbf{n}_{(j+1) \mod (8)}\right\|} \tag{9.12}$$

$$\cos(\varphi_k) = \frac{\mathbf{n}_k \cdot \mathbf{n}_{(k+2) \mod (8)}}{\|\mathbf{n}_k\| \left\|\mathbf{n}_{(k+2) \mod (8)}\right\|} \tag{9.13}$$

$$s_j = \frac{\|\mathbf{n}_j\|}{\sum_{t=0, t\neq j}^{7} \|\mathbf{n}_t\|} \tag{9.14}$$

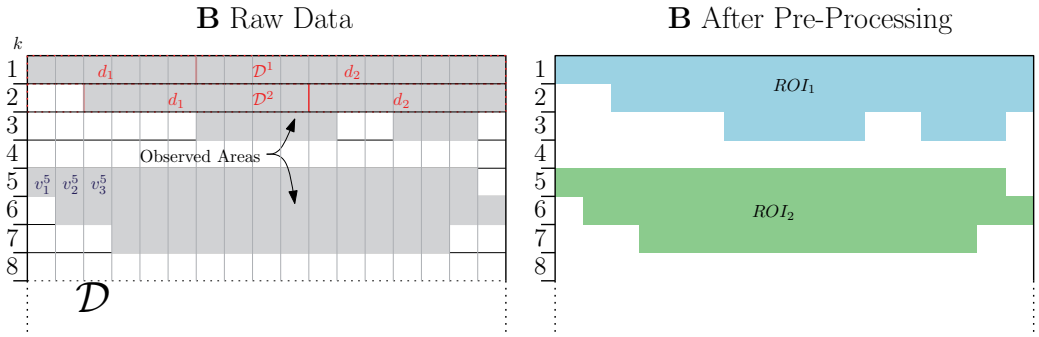


Figure 9.3: Raw Data and Region Of Interest Extraction Result.

$$\text{edge}(P_i) = \begin{cases} \exists\, P_i^j = 0 \\ \cos(\varphi_j) > T_\alpha \\ \cos(\varphi_k) > T_\alpha \\ s_j > T_s \end{cases} \tag{9.15}$$

where:

$\mathbf{n}_j$ Normal vectors on subsequent $\triangle$ (P_i^j, P_i, $P_i^{(j+1) \mod (8)}$); see Eq. (9.10).

$\mathbf{n}_j$ Normal vectors on subsequent $\triangle$ (P_i^k, P_i, $P_i^{(k+2) \mod (8)}$); see Eq. (9.10).

N_i Vector of neighborhood points described in Eq. (9.9)

T_α Angle threshold.

T_S Surface threshold.

The final step in preprocessing is the labeling of objects surrounded by a closed edge-path, creating disjoint sets each having a unique label: connected component labeling. The result is a bit-mask overlay of the buffer, **B** where each extracted feature has its unique label and it can be processed independently from others: "divide-and-conquer"; see Boxer and Miller (2005, page 200 ff.). The local one-ring neighborhood analysis fails if the data around the edge point is noisy. Also, for the situation where two shapes are in contact, the analysis area must be extended. Each object extracted in preprocessing is considered as a region of interest and requires deeper analysis.

9.3 Region-of-Interest Analysis

The goal of the filter explained here, is to verify extracted objects from the preprocessing step. The analysis is focused on potential additional edges within the region of interest *and* simultaneous parametrization of the data through variable degree polynomials. The additional edges help to separate the objects in cluttered areas, and computed polynomials deliver the smoothed surface curves. An ideal filter enhances the data in one step, which means no iterations are necessary; it is fast and deterministic, utilizing its execution time and scales extremely well. Furthermore, it is able to separate data, based on the constraints as determined. The basics of recursive filtering have already been described in Part II, Chapter 5. The filter explained here is both a smoothing and a segmenting filter; it parametrizes the observations and computes the boundaries between the differently parametrized sections. The entity of the data is enclosed in the set, $\mathcal{D} = \mathcal{D}^1 \cup \mathcal{D}^2 \cup \cdots \cup \mathcal{D}^k$, $k \in \mathbb{N}$ and it is the *domain* for all functions that parametrize the data. The index, k denotes the index of the subset, derived from the observation time, which is unique for the whole processing procedure; so $k = k(t)$ where t corresponds to the time of observation as explained in Section 9.1.

The purpose of the filter is to find the areas within one sub-domain, $\mathcal{D}^k$, where the modeling function, a is *analytic*. In the text following the "analytic function a" is denoted "function a". The result is the union of domains $\mathcal{D}^k = d_1 \cup d_2 \cup \cdots \cup d_n$. Each domain, d corresponds to one function a, and function a is valid only for the corresponding domain d. The function evaluation on each element of d is stored in s; hence $a : d \mapsto s$ - the indices are ignored for the moment.

Now follows the explanation in detail: domain properties are given in Eq. (9.16):

$$\mathcal{D} \supseteq \mathcal{D}^k = \{d_1, \ldots, d_n\},\ \exists\, d_i \cap d_j = \emptyset\, \forall\, i \neq j,\ n \in \mathbb{N} \tag{9.16}$$

Figure 9.3 shows an example of the data observed, with domains $\mathcal{D}^1$ and $\mathcal{D}^2$. Each of the domains also shows the intervals within each domain denoted as d_1 and d_2. The set of analytic functions computed by the filter on the domain $\mathcal{D}^k$, is denoted as $\mathcal{A}^k$. The index n corresponds to the equally named index in Eq. (9.16):

$$\mathcal{A}^k = \{a_1, \ldots, a_n\} \tag{9.17}$$

The precise notation of each function is:

$$a_n^k : d_n^k \mapsto s_n^k,\ n,\ k \in \mathbb{N} \tag{9.18}$$

Each function, a is the *state* of the filter after the measurement and time update. Finally, the set of observations in the domain, $\mathcal{D}$, is denoted as $\mathcal{V}$. The observations are discrete and unique; so each element of $\mathcal{D}$ contains one and only one observation that is an element of $\mathcal{V}$; see Figure 9.3.

$$\mathcal{V} \supseteq \mathcal{V}^k = \{v_1, \ldots, v_j\},\ j \in \mathbb{N} \tag{9.19}$$

A function a, models observations v, on a domain d, and again, ignore the indices. Then the difference between the model and real observations is: $\delta = s - v$, or exactly with indices:

$$\delta_j^k = \left| s_j^k - v_j^k \right| \tag{9.20}$$

The difference δ_j^k is the criteria for classification of the observation v_j^k in one of four sets:

$\mathcal{V}$ Set of valid observations.

$\mathcal{O}$ Set of indices of the outliers. An outlier can be *temporary* or *confirmed*; see the red- and blue-outlined squares in Figure 9.4.

$\mathcal{R}$ Set of indices of invalid observations and data gaps.

$\mathcal{E}$ Set of indices of discontinuities in the data.

The sets fulfill the following conditions:

1. $\mathcal{V} \cap \mathcal{O} \vee \mathcal{V} \cap \mathcal{E} \neq \emptyset$: A valid observation can be an outlier and a discontinuity point.
2. $\mathcal{V} \cap \mathcal{R} = \emptyset$: Invalid observations and valid observations are disjoint sets.

The classification of an observation v_j^k is executed by the decision function, $D(\cdot)$. For reasons of efficiency, not the observation, but its index is classified. To simplify the algorithm description, index and observation classification are treated as equals. The decision function criteria for the observation classification, is based on the following parameters, which are given as: j^- for the last valid observation index or "Last Valid Observation (LVO)", and j^0 for the first valid observation.

The following parameters need to be defined:

$\Delta j_{g_{max}}$ Maximal index difference without measurements. If the gap observation index, j_g exceeds the length: $\Delta j_g = j - j^- > \Delta j_{g_{max}},\ j > j^-$; then the interval $d_*^k = [j^0, j^-]$ is a set.

$\Delta j_{o_{max}}$ Maximal index difference that outliers can continuously occur in the measurements. If the outlier observation index, j exceeds the interval: $\Delta j_o = j - j^- > \Delta j_{o_{max}},\ j > j^-$; then the interval $d_*^k = [j^0, j^-]$ is a set.

Δ_{max} Maximal index difference for the cases where the outliers and gaps alternate. If $\Delta = \Delta j_g + \Delta j_o > \Delta_{max} = \min(\Delta j_{g_{max}}, \Delta j_{o_{max}})$; then the interval $d_*^k = [j^0, j^-]$ is a set.

By "closing" the upper interval border, the modeling function, a_*^k is also a set; obviously the state vector of the filter and its variance-covariance matrix is stored for the computed interval. The filter itself is reset and fed with observations starting at j^-, which is then the new j^0 or first valid observation. The filter state vector and variance-covariance matrix are gathered in the set, F:

$$F = \{a,\ P\} \tag{9.21}$$

where P is the variance-covariance matrix of the state vector a. The main filter feature is outlier detection, based on the measurements history. Each *new* observation is compared to the *predicted* value computed from

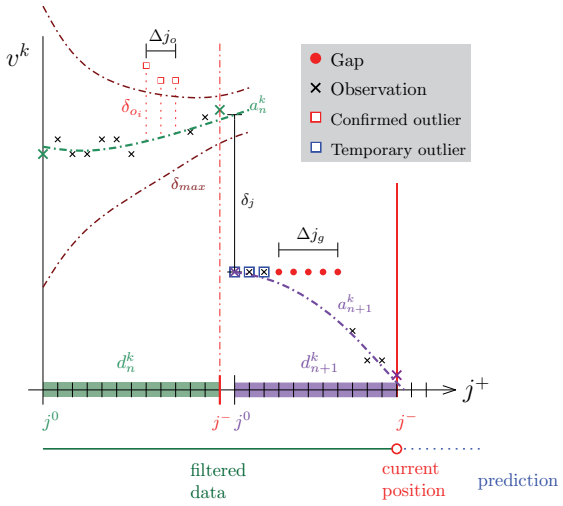


Figure 9.4: Filter Functionality and Events.

posterior parameters, F^+. The raw observations are considered to be Gaussian random values with mean $\bar{x}$ and variance σ^2: $x \sim \mathcal{N}(\bar{x}, \sigma^2)$. The following values are computed for the subsequent student $t-$test:

$$
\begin{aligned}
s_{j+1} &= a_j(j+1) && (9.22)\\
\delta_{j+1} &= |s_{j+1} - v_{j+1}| && (9.23)\\
\Delta j &= j^- - j^0 && (9.24)\\
\tilde{\sigma}^2_{s_{j+1}} &= HP^-H^T && (9.25)\\
\delta_{max} &= \mathbf{t}_{\frac{\alpha}{2},\Delta j}\frac{\tilde{\sigma}_{s_{j+1}}}{\Delta j} && (9.26)
\end{aligned}
$$

The $t-$test is part of the decision function, $D(\cdot)$:

$$
\delta_{j+1} = \begin{cases} \geqslant \delta_{max} & update\,\Delta j_o \\ < \delta_{max} & measurement\ update \end{cases} \tag{9.27}
$$

The decision function $D(\cdot)$ uses parameters already described above: j, Δj_o, Δj_g, δ_{max}, $\Delta j_{o_{max}}$, $\Delta j_{g_{max}}$. It decides in case of an outlier or data gap what happens in the next step:

Case	Action
$\Delta j_o + \Delta j_g < \Delta_{max}$	Continue processing; observation will not contribute to the state estimation. See Algorithm Algorithm B.3.
$\Delta j_o + \Delta j_g \geqslant \Delta_{max}$	Reset the filter; Restart at last valid position. See Algorithm B.2.

The processing sequence, of which formulae are presented above is, shown in Algorithm B.5.

9.4 Filter Features Summary

The filter implements the following features:

1. Computes a set of functions $\mathcal{A}^k$ (from Eq. (9.17)), with a set of disjoint domains within each $\mathcal{D}^k$ (from Eq. (9.16)), so that by filter terminology, each $\mathcal{A}^k$ contains a set of states obtained through the recursive parameter estimation based on data from $\mathcal{V}^k$. If necessary for function $a_n^k(d_n^k)$, its final variance-covariance matrix, P_n^k is stored too. The variance-covariance matrix provides the convergence information of the smoothing process.

2. Detects data outliers, $\mathcal{O}$ reliably, so that outliers are not included in the state estimation and hence do not affect the state of the system. During the filtering process outliers must be confirmed. An outlier is confirmed when a subsequent observation with index $j < j^- + \arg\min(\Delta j_{g_{max}}, \Delta j_{o_{max}})$ is a valid observation; see red-outlined squares in the Figure 9.4. Observations are qualified as *temporary* outliers,

until they are not included in the computation of subsequent the function a^k_{n+1}; see blue-outlined squares in Figure 9.4. A data gaps are considered a special case of an outlier and they are also not included in the state estimation. A maximum number of consecutive gaps, $\Delta j_{g_{max}}$, will cause the filter reset. The outliers and gaps are called discontinuity *events*. The separation of two domains of adjacent functions, a^k_n and a^k_{n+1} occurs if one of the following conditions is fulfilled:

(a) the data gap exceeds a maximal value: $\Delta j_g \geqslant \Delta j_{g_{max}}$, or

(b) the count of successive outliers exceeds a maximal value: $\Delta j_o \geqslant \Delta_{o_{max}}$, or

(c) the sum of discontinuity events exceeds the minimums of the constraints: $\Delta j_g + \Delta j_o \geqslant \arg\min(\Delta j_{g_{max}}, \Delta j_{o_{max}})$.

3. Provides bookkeeping of events (outliers and gaps) for extended filter logic.

Figure 9.4 shows the filter behavior for all the cases described in text above. The $\delta_{max}(j)$ function graph shows the boundaries of the confidence interval. The expected behavior of the filter is that with each processed observation, δ_{max} decreases: $\delta_{max}(j+1) < \delta_{max}(j)$. In Figure 9.4 dark red dash-dot lines illustrate the expected behavior. Implementing the features just mentioned, the requirements for simultaneous smoothing and segmentation are realized. The algorithms B.2-B.4 implement crucial elements of recursive filter operations. In the next sub-section the filter's edge detection logic is considered in more detail.

9.4.1 Edge detection

In contrast to texture-based classification approaches, shape-based classification requires even more reliable object distinction within the data. The solution proposed here relies on a robust estimate of the space curve and the assumption that space curves differ significantly at the edge points. The edge is defined by two successive points, the first point determines the end of one domain, d_n and the second point the beginning of the successive domain d_{n+1}. The spatial sequence of the points is defined by the filtering direction. A single point represents the edge if there is an invalid observation, $v_{j\pm1} = 0$. The "edge logic" consists of four cases, shown by way of explanation in Figure 9.5:

Figure 9.5a Dense environment: The dense environment contains either no data gaps or the number of successive data gaps is always smaller than the predefined maximum: $\Delta j_g < \Delta j_{g_{max}}$. The edge is detected in the case that two successive observations, $v_j$and v_{j+1} belong to two different space curves modeled by a^k_n and a^k_{n+1}. The evaluation of the functions at position j and $j+1$ differ by an amount that exceeds the test value δ_{max} for both positions. The edge points are emphasized with red circles on the left side.

Figure 9.5b The simplest case of an edge point is at the last point observed.

Figure 9.5c In the case that the two successive points are divided with a gap, $\Delta j_g \geqslant \Delta j_{g_{max}}$, they are considered as edge points of two space curves.

Figure 9.5d In a cluttered environment such as a meadow, a thin grass blade (2) might separate a larger leaf (1 and 3) into two separate surfaces because the sensor is not able to obtain observation through the leaf structure. Although curves 1 and 3 belong to the same surface, they remain separated by curve 2, as shown in the figure. To join two separated surfaces, the following test must be repeated for each pair of separated curves:
Let j_1^- be the last index in the domain, d_1, and j_3^0 the first index in the domain, d_3, of two functions a_1 and a_3. Let $s_1^- = a_1(j_1^-)$, $s_1^3 = a_1(j_3^0)$, $s_3^0 = a_3(j_3^0)$ and $s_3^1 = a_3(j_1^-)$ be evaluations of the functions on the respective positions, $j_1^-$and j_3^0. The variances, $\tilde{\sigma}^2_{s_1^3}$ and $\tilde{\sigma}^2_{s_2^3}$ are computed using Eq. (9.25). The hypothesis $\mathbf{H}_0 : s_1^- = s_1^3 \vee s_3^0 = s_3^1$ is subjected to a t-test. If $\mathbf{H}_0$ is not rejected, the two curves, a_1 and a_3 are considered to be identical: $a_1 \equiv a_3$ and their parameters can be merged. Missing observations, like the area covered by the grass blade, can be interpolated if necessary, so that shape analysis does not fail.

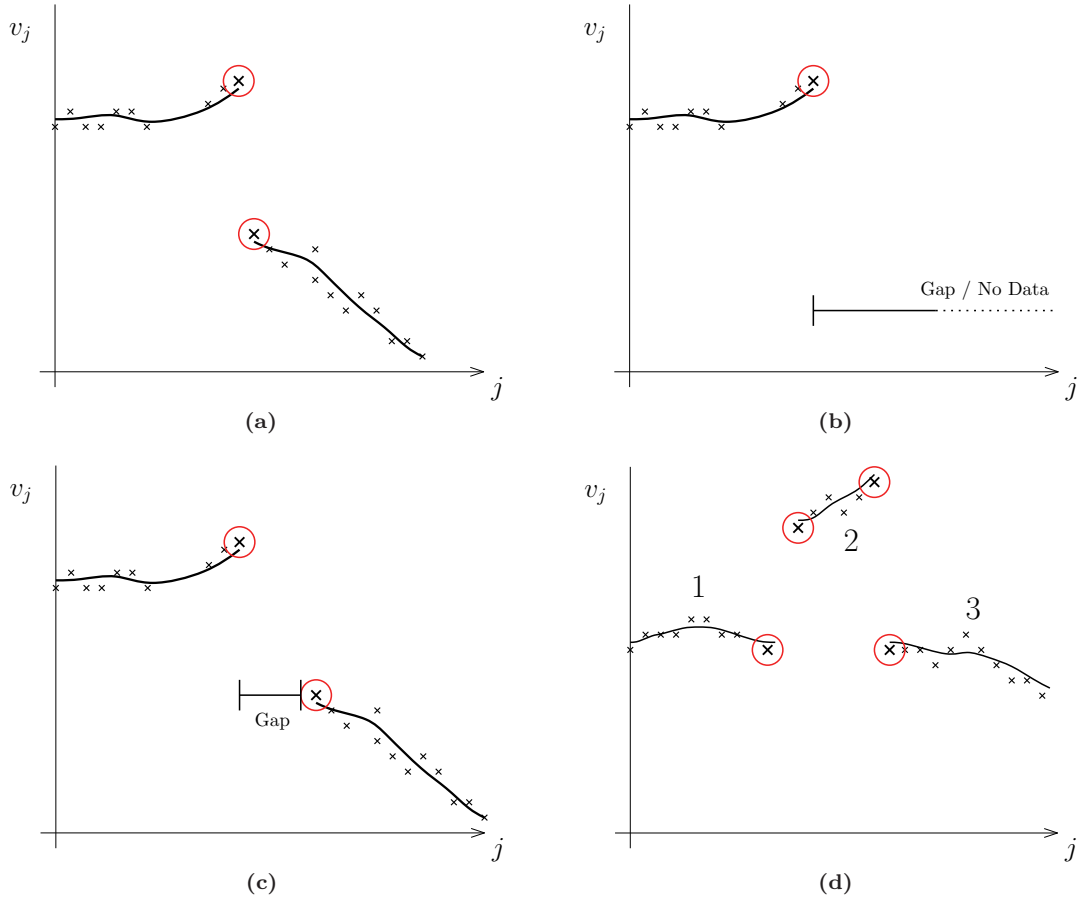


Figure 9.5: Edge Logic.

9.5 Parametrization

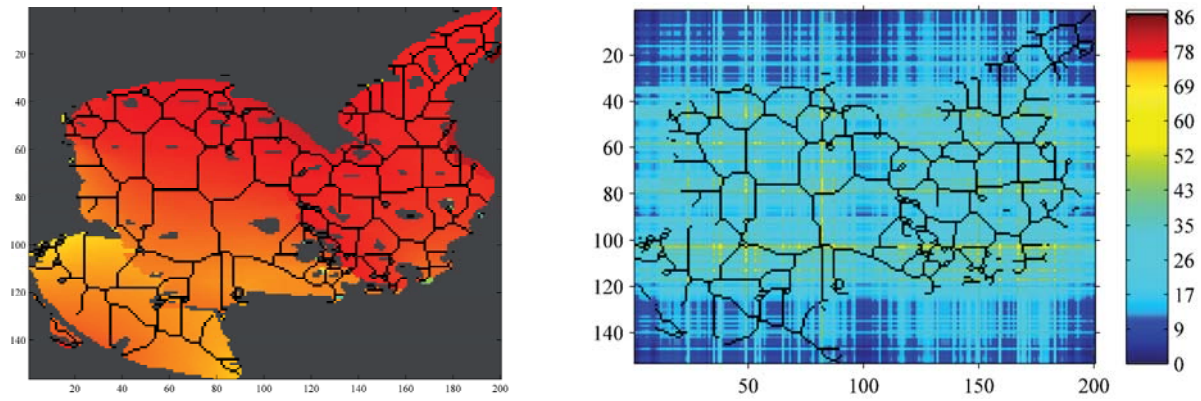
During the filtering process a set of discrete points is transformed to a set of functions, $f : \mathcal{V} \mapsto \mathcal{A}$. The set of analytic functions, $\mathcal{A}$ represents for each set member, a smooth curve on the surface, $\mathcal{S} \subseteq \mathbb{R}^3$. The work of Taubin (1991) is certainly considered fundamental in this area, especially the reasoning as to why parametrization of non-planar curves is essential for 3-D object segmentation. In opposition to the manufactured objects elaborated in the Taubin paper, natural objects are underlain by a high bias and variation of their surfaces; the challenge is to model surface curves with the minimum parameter count while retaining key surface features, such as curvature and orientation. The parametrization of the surface curve is described in the previous section; the next section explains the strategy of the filtering process to achieve the required quality using a minimal amount of processing time.

9.6 Filtering Strategy

The recursive filter's execution time is significantly higher than the execution time of "simple" filters based on convolution. The filtering strategy of convolution-based operators is the *brute force*[2] method, which processes each point individually. Depending on the convolution matrix, $C_{m,n}$ and the filter itself, each point is filtered at least once and it is involved in the process at least $m \times n$ times. Convolution operators are highly optimized and they perform very well in a real-time application; the processing time stays below the 20 ms mark for the whole image: $1560 \times 32\,[px]$. Achieving such processing speed and accuracy with a recursive filter requires careful selection of starting points and this is necessary for two reasons:

1. The first observations processed by the filter cannot be reliably tested for outliers, since the state vector is not computed until the observations and unknowns counts are not equal. In the beginning of the filtering process, the uncertainty is too large for a statistical test to deliver reliable results, however, with

[2] "Brute force is a crude algorithm which works through every possible answer until the correct one is found. Some problems can be solved this way but others- for instance obtaining the prime factors of large numbers or picking the best chess move, have far too many possibilities to be solved that way except in simple cases." About.com (2012)



(a) Raw Data and Skeleton Path. **(b)** Computation Density.

Figure 9.6: Filtering Strategy Decision Base.

an increasing number of valid observations the model is improved and data can reliably be tested for outliers; see the confidence interval margins, δ_{max} in Figure 9.4. The starting point of the filter and first observations are thus crucial to achieving reliable results.

2. An observation, if successfully filtered once, requires no further processing. An observation is successfully processed if the trace of the variance-covariance matrix, P is less than the predefined threshold $T_{\hat{\sigma}^2}$: $\sum \text{trace}(P) < T_{\hat{\sigma}^2}$.

The filtering process can fail; the strategy minimizes the probability of the filter failing; however it cannot be totally avoided. The recursive filter will fail if one of following situations occur:

1. The variance-covariance matrix has starting values which are too large: $\text{trace}(P) = \sigma_{i,i}^2 = \sigma_0^2$, $\hat{\sigma}_{i,i}^2 \ll \sigma_0^2$, where σ_0^2 denotes prior variances value (initial value) and $\hat{\sigma}_{i,i}^2$ denotes posterior variances along the diagonal of P. In a case when data is sparse and/or the noise is not white, the filter will not converge and the results will be useless.

2. If the initial state estimate vector contains arbitrary/wrong values and the filter's prior variance-covariance matrix is constrained by $\hat{\sigma}_{i,i}^2 \ll \sigma_0^2$, on the assumption that the initial values are accurate, the final state of the filter will not provide an optimal result or it will diverge.

3. The functional model does not fit the corresponding data.

The strategy foresees a two-stage solution. In the first stage, filter failure has to be minimized. The functional model of the data is determined to be a polynomial, $a(j) = c_0 j^0 + c_1 j^1 + \cdots + c_n j^n$ of an arbitrary degree n. The coefficients $c_0, c_1, \ldots, c_n$ are state vector variables and are estimated by the filter:

$$x = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} \tag{9.28}$$

The corresponding design matrix, H contains polynomial variables:

$$H = \begin{bmatrix} j^0 & j^1 & \cdots & j^n \end{bmatrix} \tag{9.29}$$

The *state transition* matrix of the filter is an identity matrix. The particular polynomial degree range is to be determined experimentally and three degrees are tested: 3, 4 and 5. The filtering process is bi-directional: forward, $j_{n+1} > j_n \Rightarrow \Delta j_f > 0$ and backward, $j_n > j_{n+1} \Rightarrow \Delta j_b < 0$. The values, $|\Delta j_f| = |\Delta j_b| = const.$ apply for the whole of the filtering process; for further details see Grewal and Andrews (2001, page 190 ff.).

The meadow data stored in the buffer is filtered in two perpendicular directions: x (row) and y (column). In each row or column the starting point of the filter is selected by the filter logic. The logic used aims to minimize the computation time by reducing the number of filter passes per direction, accelerating filter convergence by the selection of starting points with sufficient number of data points, and provide a system with quality indicators for each observation processed. To emphasize the significant advantage of this approach, the following example shows a situation where a standard convolution filter fails and a recursive filter succeeds. The primary goal

of data filtering, is the transition from discrete observations to an analytic function. The edge detector that results from the filtering process, is a side-effect; it helps the system to identify surface patches in more complex situations. Such a situation is shown in Figure 9.2a; it is a meadow patch with superimposed surface patches computed through a "standard" segmentation process: median filter, Canny edge detection, 2-D connected component labeling. Figure 9.2b shows eight objects extracted and enumerated. In the first segmentation step the system neglects all objects with an area smaller than the threshold area, as determined by the user. However, this step is not necessary for the process to work properly; it solely removes very small objects from the processing queue. As mentioned in Section 2.1, shape-based classification can only succeed if the resolution is sufficiently high.

Coarse segmentation was successful, but Object 2 is certainly not well segmented. The human eye immediately recognizes three different leaves; see Figure 9.2a for the position of Object 2. Five standard edge detection algorithms provided by MATLAB deliver the same results; no additional edges have been found in this situation; see Figure 9.7a showing *sobel*, *prewit*, *roberts*, *log* and *canny* images. White pixels represent edges. The $z-$values are color-coded, and warmer colors represent greater $z-$values. The "*custom*" edge detector is a morphological operation that analyzes an 8-point neighborhood, and if any observation is invalid in that area, the center of observation is considered as an edge. This range image is used as raw input for the recursive filter.

The filtering process (see Algorithms B.2-B.5) consists of a two-pass procedure. On the first pass "brute force filtering" is applied, during which each column and row must be processed at least once. On the second pass, the filter starting points are computed from the data footprint on the $xy-$plane. For the purpose of recursive filtering, an object O is extracted from the point cloud $\mathcal{P}$ through a bit-mask operation: $O = \mathcal{P} \vee \ , \ = label_o$, where is a bit-mask containing logical *true* on each position where point label equals to the object label. Remaining bit-mask points contain logical false. A subset for processing is established with that operation. In the next step, the optimal starting points for the filtering steps are estimated: the skeleton operation is executed on the object's footprint. The skeleton, S_o, shown in Figure 9.7b as black "veins", represents possible optimal starting points for the refinement filtering process; optimal, because the distance to the edge of the object on this position is the largest. Therefore, points of the skeleton are assumed to lie within the area which has sufficiently dense data and optimal conditions for the filter. The probability that initial observations have a low bias, is higher at the position of skeleton points than for other points near the edges of the surfaces. Furthermore, the largest distance to the nearest edges assures "no gaps" during the first filtering computations: the filter will converge quickly.

Each valid observation is filtered twice for each direction: x and y. The quality indicators matrices are:

Q_{pp_v} Quality indicator matrix of the vertical filter,

Q_{pp_h} Quality indicator matrix for the horizontal filter,

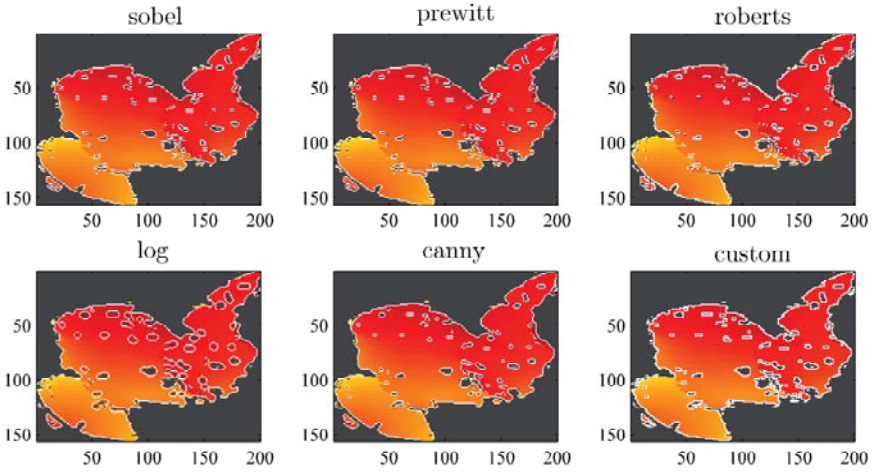
and contain on the position (x, y) of each observation, the lowest quality indicator value out of all the filter passes: $\min(\sum \text{trace}(P^i_{j_f}), \sum \text{trace}(P^i_{j_b}))$, where $P^i_{j_*}$ is a variance-covariance matrix, and i the number of passes, with the i^{th} pass for the "time point", t, from forward, or backward filtering. Figure 9.7c shows the quality of the results of the complete filtering process. The red rectangles in Figure 9.7c on the left image highlight the points in the data where the filter did not provide satisfactory results: $\sigma > 16$. This situation occurred only for the filter in $x-$direction; in the $y-$direction there are no such events. If the result is analyzed more carefully, the observations lie exactly between two edges comparing the horizontal filtering directions. The filtering failed on two spots due to a lack of valid observations in the horizontal direction; see red rectangles in Figure 9.7c.

The filter has successfully separated one leaf from the group, the lower left one; the separation of remaining two leaves failed just for a single line. The height difference is probably too low to be detected by the sensor and one leaf lies directly over another; see green rectangle in Figure 9.7b.

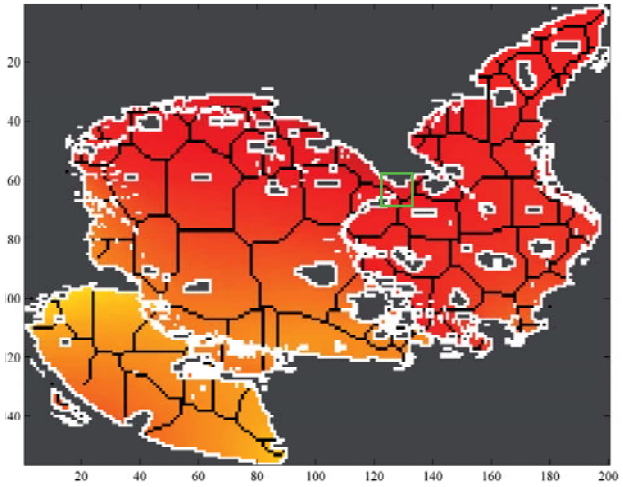
Conclusion

The filtering method described in the text above represents a very complex, and computationally intensive process of data smoothing and discontinuity detection. It is applicable to real-time systems under the condition that the underlying hardware has a sufficient amount of processors or other parallel HPC devices. The exact behavior and the measurements are shown in Part IV. Besides a computer-resource hunger, the algorithm satisfies the major goals required of a data model and edge detection algorithm; in addition, it also provides a quality or reliability indicator for each observation. The quality indicators allow additional logic that supports autonomous decisions of the system; to decide whether the discontinuity or analytic function should be accepted or not.

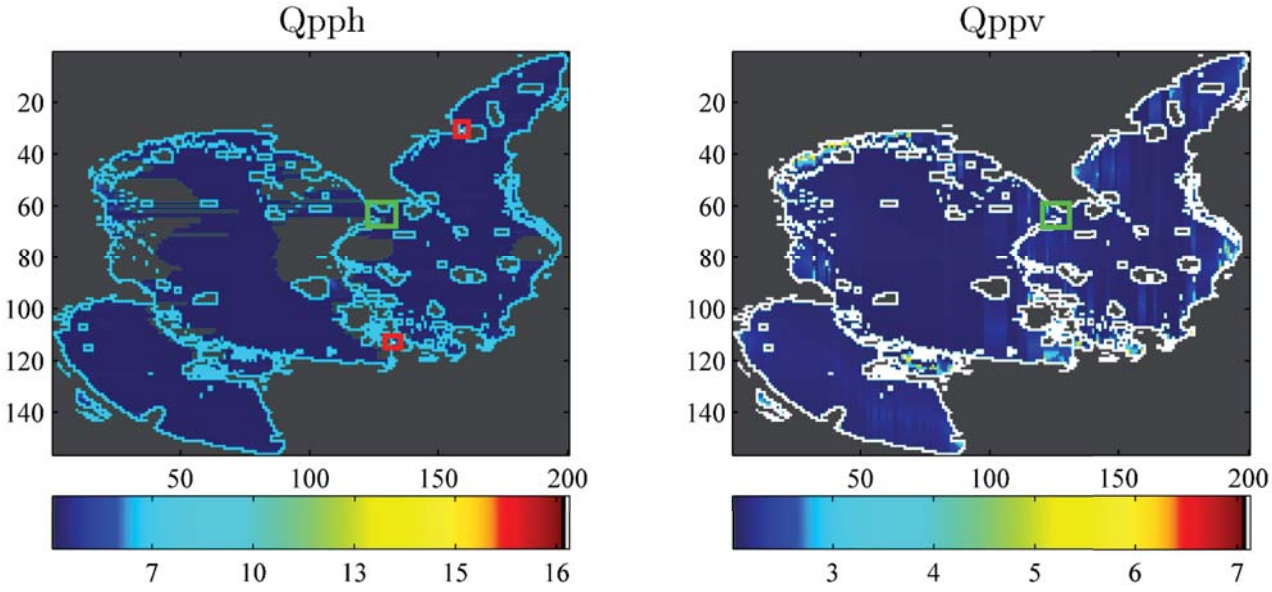
The polynomials allow efficient resolution scaling with a minimum of accuracy loss; one of the major advantages of a B-Spline representation of the intensity images; (see Unser et al., 1993b,a). Object defects such as holes, can be removed reliably. The flattening of the leaf can be computed in various ways and additional operations,



(a)



(b)



(c) The Quality of Data Derived From The State Covariance Matrix. Left: Filtering in x Directions, Right: Filtering in y Directions.

Figure 9.7: Comparison Between the Edge Detection With Standard Procedures and Recursive Filter Described in the Text.

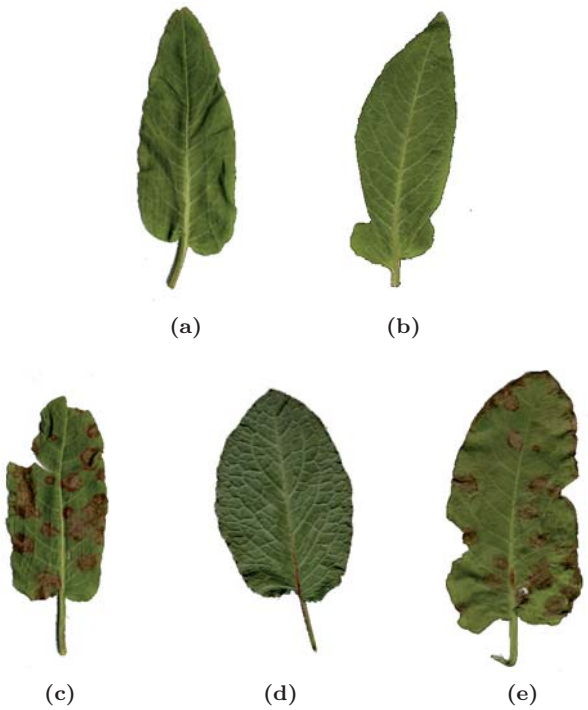


Figure 9.8: Scanned Rumex Leaves.

such as water evaporation corrections, can be applied. The resulting representation of plane leaves provides a very good estimate of the leaves as prepared with an original herbaria preparation process.

9.7 Leaf Flattening Using Filter Results

Each object's surface varies depending on parameters such as the plant water content, exposure to sunlight, the season, pests, and diseases. The cellular structure also hinders deformation free mapping onto the plane. The flattened shape is the basis for boundary extraction and thus this mapping process is a required step. In Section 4.5 some basic approaches are elaborated, although elegant projection to one of the surfaces (plane, cone or cylinder) does not deliver satisfactory results, especially if the projection is superimposed onto the image acquired from the flat-bed scanner. A major source of the discrepancy is the physical deformation of the leaf structure caused by its non-planar nature.

Figure 9.8 depicts four examples which collectively illustrate the major deformations which commonly occur, and one sample almost without any deformation. The leaf in Figure 9.8d is insignificantly altered and is a rare example of an approximately plane surface; the deformation caused by the leaf's non-planar nature is shown

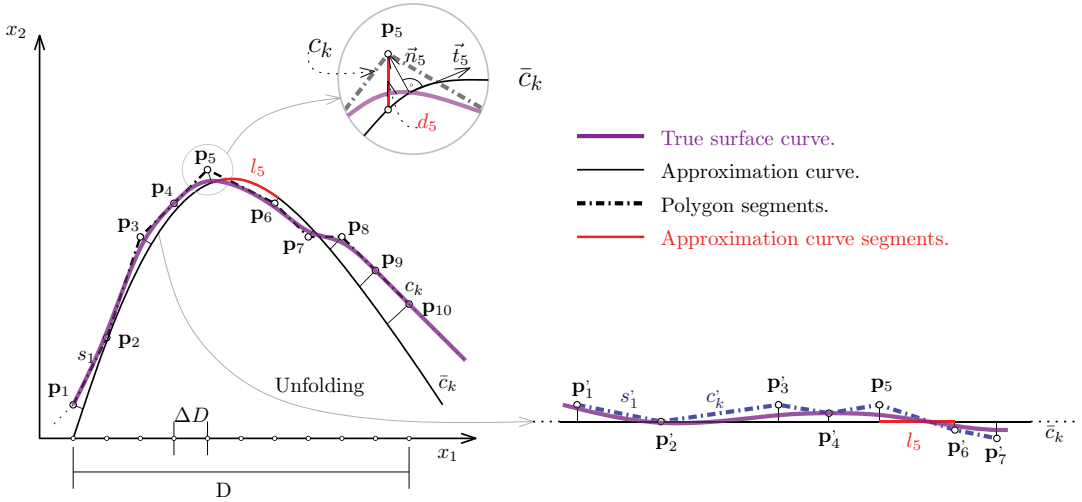


Figure 9.9: Unfolding of the Point Cloud Along the Curve c_k.

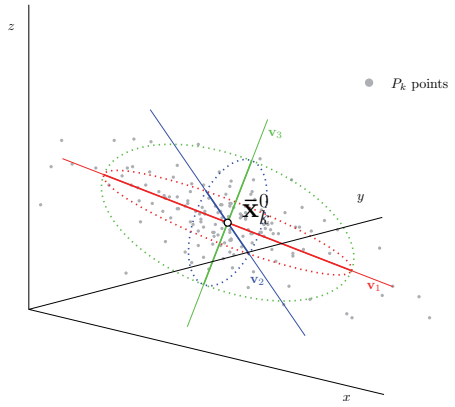


Figure 9.10: Eigenvalues and Eigenvectors of the Point Cloud $\mathcal{P}$.

in Figure 9.8a; In Figure 9.8 the natural diversity of plant leaves and their geometric deformation due their non-planar nature is shown; pest and mechanical damage can significantly change a leaf's shape, and this is shown in Figure 9.8c. Finally, pest influence and successive deformation caused by non-planar leaf nature is shown Figure 9.8e.

The challenge is to flatten the surface so that the resulting shape or footprint of the point cloud, $\mathcal{P}$ is as close as possible to the results achieved by scanning the shapes with a flat-bed scanner. For this the purpose following parameters are computed:

Moments $\mathbb{R}^3$

$$m,\, p,\, q = \{0,\, 1,\, 2\} \tag{9.30}$$

$$m_{pqs} = \sum_{xyz} x^p y^q z^s \, I(x,\, y,\, z)\, \Delta A \tag{9.31}$$

$$I(x,\, y,\, z) = \begin{cases} 1 & if\, valid\, point \\ 0 & otherwise \end{cases} \tag{9.32}$$

$$\Delta A = const \tag{9.33}$$

where $I(\cdot)$ denotes the intensity operator and ΔA denotes the area unit.

Centroid:

$$\bar{\mathbf{x}}_0 = \left\{ \bar{x} = \frac{m_{100}}{m_{000}},\, \bar{y} = \frac{m_{010}}{m_{000}},\, \bar{z} = \frac{m_{001}}{m_{000}} \right\} \tag{9.34}$$

Centralized moments:

The centroid values $\bar{\mathbf{x}}_0$ are used to compute centralized moments:

$$p,\, q,\, s = \{0,\, 1,\, 2\} \tag{9.35}$$

$$\mu_{pqs} = \sum_{xyz} (x - \bar{x})^p (y - \bar{y})^q (z - \bar{z})^s \, I(x, y, z)\, \Delta A \tag{9.36}$$

$I(\cdot)$ and ΔA from Eq. (9.31).

Distribution and Orientation:

Distribution and orientation of $\mathcal{P}$ in $\mathbb{R}^3$(see Figure 9.10) is extracted from the variance-covariance matrix, $Q_{\mathcal{P}\mathcal{P}}$ out of $\mathcal{P}$, which consists of centralized moments computed in Eq. (9.36):

$$Q_{\mathcal{P}\mathcal{P}} = \begin{bmatrix} \mu_{200} & \mu_{110} & \mu_{101} \\ \mu_{110} & \mu_{020} & \mu_{011} \\ \mu_{101} & \mu_{011} & \mu_{002} \end{bmatrix} \tag{9.37}$$

The matrix, $Q_{\mathcal{PP}}$ is symmetric. The eigenvalues and eigenvectors are computed by a singular value decomposition algorithm (SVD):

$$(\mathbf{V},\, \mathbf{v}) = \mathrm{SVD}(Q_{\mathcal{PP}}) \tag{9.38}$$

$$\mathbf{V} \;=\; \begin{bmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \\ n_1 & n_2 & n_3 \end{bmatrix} = [\mathbf{v}_1,\, \mathbf{v}_2,\, \mathbf{v}_3] \tag{9.39}$$

$$\|\mathbf{v}_1\| > \|\mathbf{v}_2\| > \|\mathbf{v}_3\| \tag{9.40}$$

$$\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^T,\; v_1 > v_2 > v_3 \tag{9.41}$$

where $\mathbf{V}$ is an eigenvector matrix and $\mathbf{v}$ is an eigenvalue vector. The geometrical interpretation of the computation is: the resulting vectors in $\mathbf{V}$ span around the centroid, $\bar{\mathbf{x}}_0$, in the "local" coordinate system of $\mathcal{P}$. The bounding box of $\mathcal{P}$ is defined by the central moments μ_{100}, μ_{010} and μ_{001}. The "local" coordinate system is oriented in $\mathbb{R}^3$ through $\mathbf{V}$: the three planes, which intersect in $\bar{\mathbf{x}}_0$, are $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$, with plane gradients $\mathbf{v}_3$, $\mathbf{v}_2$ and $\mathbf{v}_1$; see Figure 9.10. Each plane $\mathbf{P}_i$ intersects the surface represented by the point cloud, $\mathcal{P}$ and creates a *polygonal chain*, denoted as *path*: $c_k^i \supseteq \mathbb{R}^2 = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$, $n \in \mathbb{N}$. Since the surface can take on an arbitrary form of bending, orientation and position, the criteria for the optimal unfolding direction is crucial to the projection result. Before the flattening process is initiated, the point cloud, $\mathcal{P}$ is transformed to its coordinate system defined by $\mathbf{v}$:

$$\mathcal{P}' = (\mathcal{P} - \bar{\mathbf{x}}_0)\mathbf{V} \tag{9.42}$$

$$\mathcal{P} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix} \tag{9.43}$$

The surface is flattened along one of the eigenvector axes, using two criteria for the selection of the axis:

1. The surface is unfolded along the *longest continuous* path. The length of the path is computed as the sum of the segments' lengths: $l_{c_k}^i = \sum_{n=1}^{N-1} \|\mathbf{p}_{n+1} - \mathbf{p}_n\|$. The computation is straightforward and the CPUs inexpensive.

2. The surface is a unfolded leaf along the axis with the greatest average curvature. The average curvature is computed around each $\mathbf{p}_n$ in the one-ring neighborhood. The curvature computation is an approximation and is given by Eq. (4.41). Each valid $\mathbf{p}_n$ has a $\bar{\kappa}_n$, and the mean curvature for the axis is: $\bar{\kappa}_p = \frac{1}{N}\sum_{i=1}^{N} \kappa_i$. The computed curvature is an approximation, and as with any curvature computed using discrete data, the results may vary in accordance with the approach used. An alternative approach that can be used, is given by Schneider and Eberly (2002, page 800).

Since the computation of the curvature in each point is completely independent of the other points (see Section 4.5.3), it is ideal for parallel processing. To compute EFDs, it is only necessary to unfold boundary points. If a real analysis of the data is required, a whole point cloud needs to be unfolded. Since unfolding is a coordinate transformation in the wider sense, it is typically an SIMD operation; a repetitive matrix vector multiplication performed on a large amount of data (in this case 3-D coordinates).

9.7.1 Flattening by Parameter Line Fitting

The flattening process is divided into several steps to achieve optimal results. The distortion of the surface is inevitable for any process, since each surface patch is in all probability, *not* a regular surface. For a robust comparison between 3-D and 2-D data, the flattening process must deliver *similar* shape forms to those from the flat-bed scanner device, or the results from the herbaria; the distortion of the leaf form should be similar for both acquisition methods. In Figure 9.11 the simple and extracted 3-D leaf is observed from different view points (3-D representation of the Object 1 from Figure 9.2b). A vision camera might produce the same 2-D images; it is obvious that each shape extracted from the images, produces different shape descriptors, independent of the 2-D analysis method applied. Lastly, at this point the advantage of 3-D data analysis is disclosed: The two-manifold shape in 3-D space can be analyzed as a two-manifold shape in 2-D space, independent of the pose of the sensor relative to the shape object during data acquisition. From 3-D data, the optimal pose can be computed by transformations and approximations to minimize the effects of surface variations and the sensor pose at the moment of observation.

The projection process consists of the following steps:

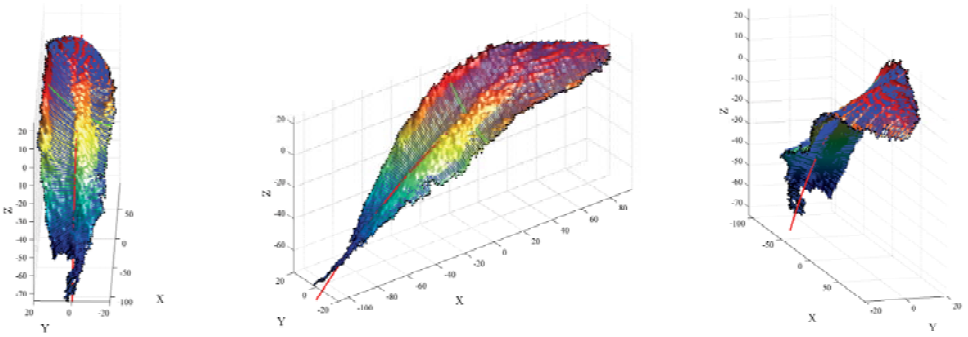


Figure 9.11: Original Data from Different Viewpoints.

1. Prior to projection, the data is filtered by a recursive filter and an *atlas* of the shape is created. The atlas consists of a variable number of *maps*; a set of maps is shown in Figure 9.12 - each blue line represents a single map.

2. The points of point cloud, $\mathcal{P}$ are transformed onto a coordinate system of their own eigenvectors. The eigenvector computation is explained in Section 9.7. The point cloud is shifted to its centroid; the semi-major axis of the confidence ellipsoid is then $u_0(x, y)$. Initially the transformation to the eigenvector axis is a single rotation around the $z-$axis. The restriction to the $xy-$plane minimizes the influence of measurement outliers on the transformation.

3. After rotation, the semi-major axis is collinear with $u_0(x, y)$ and the $x-$axis; the origin of the new coordinate system is the centroid of the shape (see result in Figure 9.14). The plane $\mathbf{\Pi_0} : \{u_0(x, y), z\}$ intersects the shape along its largest dimension. The u_0-parameter line is approximated by the polynomial: $u_0 = u_0(x, y = y_0 = const) = a_0 + a_1x + \cdots + a_nx^n$. The domain of the polynomial is the $x-$axis. In Figure 9.12 the red line represents the polynomial.

4. An atlas consists of maps which are exactly one pixel wide; hence each parameter line $v_i = v_i(x, y)$ is one map in the collection. Each parameter line complies with the following condition: $v(x, y), v(x, y) \perp u_0(x, y)$. The plane $\mathbf{\Pi}_{v_i} : \{v_i(x = const, y), z\}$ is the "slicing" plane of each map and planes are parallel to each other: $\mathbf{\Pi}_{v_1} \parallel \mathbf{\Pi}_{v_2} \parallel \cdots \parallel \mathbf{\Pi}_{v_i}$. For each parameter line, $v_i(x, y)$ the following constraints apply:

   (a) For each line there is at least one valid observation point.

   (b) Each observation is included in exactly one polynomial, $p_v(x = const, y|_{P_i}^{P_k}) = b_0 + b_1y + \cdots + b_ny^n$ that approximates the $z-$values of the shape slice between the points P_i and P_k, where $k \geqslant i$. The polynomial degree is arbitrary and the coefficients $b_0, b_1, \ldots, b_n$ are the result of the recursive filtering as described in Chapter 5.

   (c) The consecutive parameter lines v_i and v_{i+1} are connected on the closest point of both polynomials. More about the connection principle follows.

5. The parameter lines u_0 and v_i are unfolded to the plane, retaining the perpendicularity between them; thus parameter lines, u_0 and v_i are coordinate axes of a new 3-D Cartesian coordinate system. The distance between the consecutive points in new coordinate system is neither uniform nor constant; it is determined by the arc length of the polynomial that models the parameter line.

6. Each point in the point cloud, $\mathcal{P}$ is projected onto the plane surface defined by the parameter lines, u and v. Since the parameter lines are not necessarily straight, new coordinates are computed as follows:

$$P = \{x, y, z\}, P \in \mathcal{P} \tag{9.44}$$

$$u_0 = u_0(x, y) = p_{u_0}(x, y = y_0 = const) = a_0 + a_1x + \cdots + a_nx^n, n \in \mathbb{N} \tag{9.45}$$

$$v_i = v_i(x_i, y) = p_{v_i}(x_i, y|_{y_j}^{y_k}) = (b_0 + b_1y + \cdots + b_ny^n)|_{y_j}^{y_k}, i, j, k \in \mathbb{N} \wedge k > j \tag{9.46}$$

$$P_{\Pi_0} = \{\widehat{u_0(x, y)}|_{x_0}^{x_P}, \widehat{v_i(x_i, y)}|_{y_1}^{y_2}, z_P - p_{v_i}(x_i, y_P)\} \tag{9.47}$$

7. The parameter lines, u_0 and v_0 are the new major axes of the 3-D Cartesian coordinate system, spanning Π_0. The normal vector of Π_0 is the $z-$axis. New $z-$coordinates of the respective points P_{Π_0}, are Euclidean distances in the $z-$axis direction, from the v_0 parameter line and to the $z-$coordinate of the original point.

8. The longest polynomial on the surface patch in the v direction is v_0.

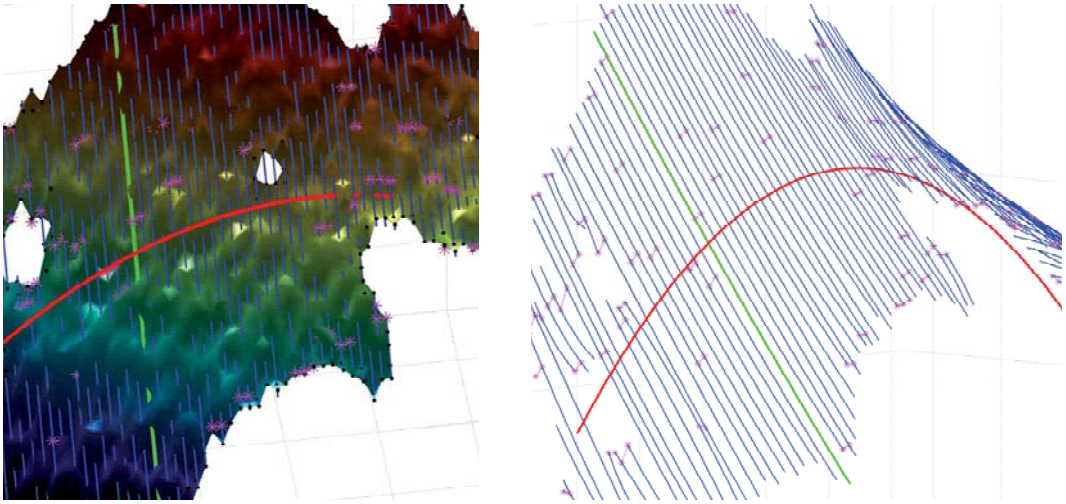


Figure 9.12: Parameter Lines. The detail shows the main parameter lines: u_0 red and v_0 green. The blue lines are the v slice lines. Magenta connections between the lines show connection points between the two slices (maps).

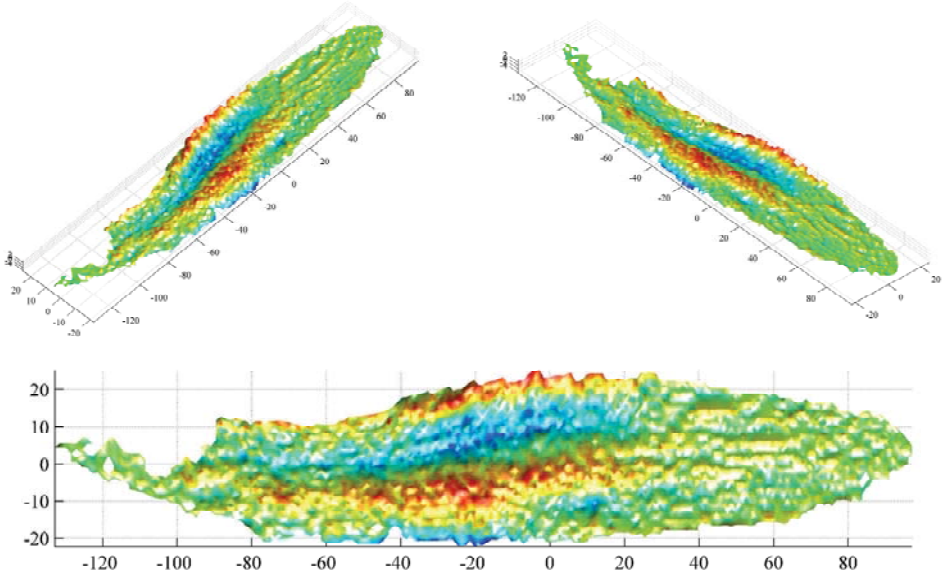


Figure 9.13: Flattened 3-D Leaf. The Height is Color-Coded, warmer colors represent higher points; colder the lower. The height range: $[-4, 2] \approx 2.3\,mm$.

The projection of a 3-D shape onto a plane as explained above, differs significantly from projections known in geodetic applications. Common projections such as UTM and Gauss–Krüger have been developed for everyday tasks where the surveyor does not continuously have to consider chart or grid borders when doing work in the field. The projection of the leaf has a different constraint, namely keeping the leaf shape as *near* as possible to the real shape of the leaf, flattened between two plates. Slicing a shape into narrow bands and transforming these bands onto the plane, creates the atlas of the shape. Unlike a UTM grid or Gauss–Krüger bands, each map is individual with regard to scale and position, and does not overlap with adjacent maps. Considering each parameter line (map) separately, the projected $z-$coordinates are residuals of the data; see the Figure 9.9. Each parameter line is a good approximation of the corresponding band on the flat leaf. The challenge remains that the atlas approximates the shape of the leaf equally well. The quality of the flattening process is proportional to the resolution; higher data resolution allows a higher quality of surface approximation. The flattening of the leaves, as shown in Figure 9.13, Figure 9.14 and Figure 9.15, were acquired at $\sim 2\,[px\,mm^{-1}]$; the resulting plane shapes represent a significant improvement on the footprints of the raw data shapes. In Section 8.3 an $\sim 7\times$ higher resolution is required if the smaller shapes are to be analyzed, since the leaf boundary, in case of coarse resolution, does not provide sufficient data for the reliable elliptic Fourier descriptors computation. Further investigation is necessary to determine the optimal resolution for the segmentation process, on the one hand, and for the shape analysis on the other.

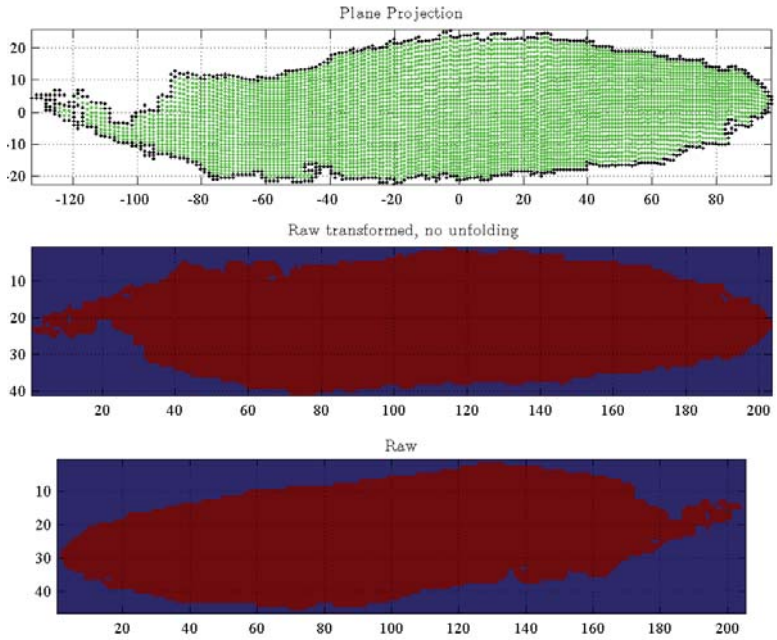


Figure 9.14: Flattened Leaf. From bottom to top: Raw data footprint. In the middle eigen axis transformation foot print with consecutive bicubic data interpolation. The flattened leaf on the top leaves the closest form to the expected one, if the leaf were put between two plane plates.

Conclusion

It is obvious that real-time single plant detection is challenging in at least three different areas: sensor technology, data modeling and data processing. Higher processing accuracy requires exponentially higher processing resources and even higher bandwidth on the data transfer buses. The methods proposed here represent scalable and extendable algorithms for data segmentation, feature extraction and classification for two-manifold in $\mathbb{R}^3$. The surface must be sampled at a predefined resolution, which has been determined as $\sim 400\,dpi = \sim 15.7\,pt/mm$ for an object of size $\geqslant 25\,mm$. Segmentation using a recursive filter outperforms discrete operators when considering object extraction accuracy; the required processing power however, is higher by at least two orders of magnitude. Therefore a cascading approach is applied where two-stage segmentation takes place: the coarse stage for noise reduction and small objects removal, and a fine stage using a recursive filter method for the final object extraction on a restricted area. In the end, the surface is flattened approximating the process of the dehydration of a leaf in a herbaria. The boundaries extracted from the 3-D data are compared with those obtained from a flat-bed scanner. The results are shown in Part IV: Evaluation and Experiments.

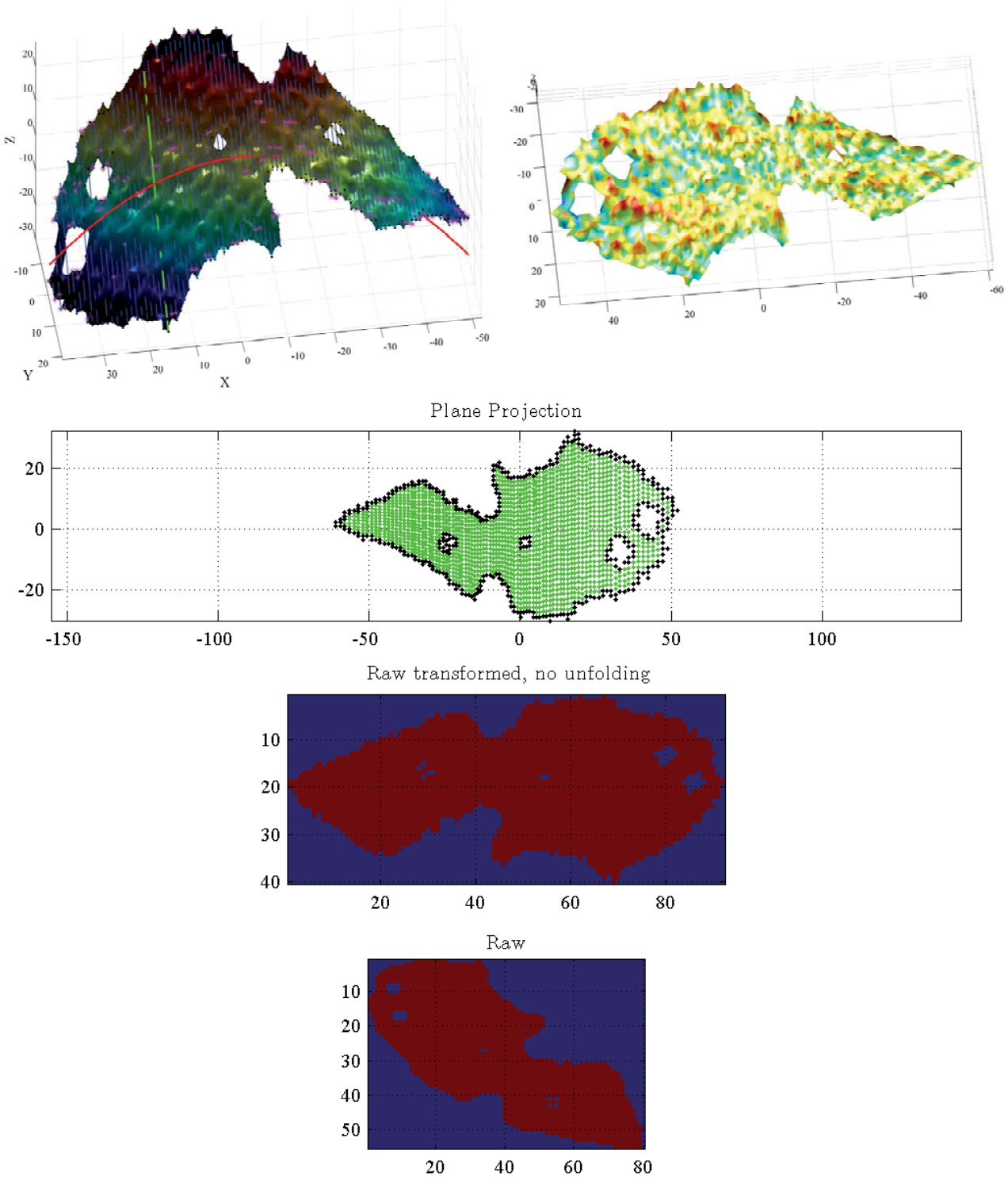


Figure 9.15: Second Leaf Example. Extracted part of Object 2 from Figure 9.2b.

Part IV

Evaluation and Experiments

Chapter 10

Experimental System

10.1 General System Information

The system is evaluated by two independent tests. The first is based on synthetic data with, and without added noise, and the second is a real-world test on a meadow using the SmartWeeder vehicle under different meteorological conditions; see Šeatović et al. (2010). The support vector machine classifier is trained differently for the synthetic and the real-world tests. Both tests are explained and commented on in their respective sectopms: synthetic tests in Section 12.3 and real-world experiments in Section 12.4. The SmartWeeder's key system components are described respectively in the following sections: the carrier with its coordinate system in Section 10.2; data flow in Section 11.1; and system and software, architecture with important realization elements in Section 11.2. For the evaluation of the algorithms, see Section 11.1.

10.2 Sensor System And Its Carrier

SmartWeeder, illustrated in Figure 10.1b[1], is a vehicle-drawn prototype, constructed to carry an arbitrary number of sensors; at present a frame-mounted Sick IVP camera which represents a type B sensor (see Section 2.1 and Sick (2013)), and two computers. The first computer is the hard real-time system responsible for sensor control and navigation, and the second one is the soft real-time system responsible for 3-D data processing. It has been built to be mechanically robust, suppress shocks, protect the computer equipment and minimize sunlight influence on the sensor; see Šeatović and Grüninger (2007) for a detailed description. The measurement capabilities of system are:

- Vehicle maximal speed 1 ms^{-1}.
- Resolution in object space $< 2\,mm^3$.
- Acquisition volume:
  - Width: 1.0 m, $x-$axis.
  - Height: 0.6 m, $z-$axis.
  - Length: ∞, $y-$axis[2].

SmartWeeder is intended to be a real-time data acquisition, processing and control system that recognizes plants immediately and marks them while being continuously drawn over vegetation. The SmartWeeder has no absolute positioning equipment such as a GPS receiver. It is designed for the immediate treatment of the plants and not for mobile mapping, but it can be modified to provide geo-referencing functionality. The treatment device consists of a line of spray nozzles installed at the back of the Smart Weeder. One spray nozzle covers an area of approximately $20 \times 20\,cm$. Therefore, the localization accuracy is not challenging at all. There are no additional supporting sensors mounted on the frame and navigation is realized on pure dead reckoning, stabilized by an extended Kalman filter. The SmartWeeder coordinate system and constraints are shown in Figure 10.1a:

W_L *and* W_R Middle points of the wheels. The wheel rotation axis is fixed exactly at these positions.

[1] Images courtesy of Dr. Thomas Anken, Agroscope Tänikon, Tänikon, Switzerland.

[2] Y coordinate axis is defined as the perpendicular vector to a wheels axis. The vectors orientation in space varies through the time but it is considered as constant for small time intervals.

r_L *and* r_R Wheel radii. $r_L \neq r_R$, but both values are considered to be constant after calibration.

$\{O, x', y'\}_{v_t}$ Vehicle Coordinate System (VCS). The VCS is constrained by the wheel axis between W_L and W_R. O_{v_t} is defined through the projected camera origin onto the object space. The index, t emphasizes that the absolute orientation of the vehicle and measurement system, vary over time. The plane, x', z' is identical to its laser plane $L = \pi_{x'z'}$; see Figure 8.9. The laser source produces a higher order surface and the plane defined here is only an approximation. By calibrating the system, projection errors are minimized (see Section 8.4). In Figure 10.1a the intersection of the laser plane is emphasized as a dash-dotted, dark red line.

M_{v_t} Center point of vehicle rotation. Rotation occurs around the z-axis located at this point.

$\vec{h_t}$ Heading vector of the vehicle system. This vector changes its bearing with every observation triggered by the system. Its length however, is constant over time.

$\vec{d_N}$ Vector between the origin of the sensor system and the origin of the treatment system ($\overrightarrow{O_{v_t}O_N}$). The following conditions apply for $\vec{d_N}$:

$$\vec{d_N} \equiv y_t^{'} \wedge \left|\vec{d_N}\right| > 0$$

$x_{v_t}^{"}$ Axis of the treatment system, where $x_{v_t}^{"} \shortparallel x_{v_t}^{'}$ and its origin is at O_N.

N_i Spray nozzle on the treatment system axis, where $i \epsilon \{0, \ldots, n\}$. The distance between subsequent nozzles are assumed to be identical.

All conditions mentioned above are constrained by the construction of the prototype. The wheel's radii can vary over the time, but the influence on the navigation accuracy and treatment system, is negligible.

(a) Vehicle Scheme. **(b)**

Figure 10.1: (a) Vehicle Scheme and (b) Actual Experimental Platform in the Field.

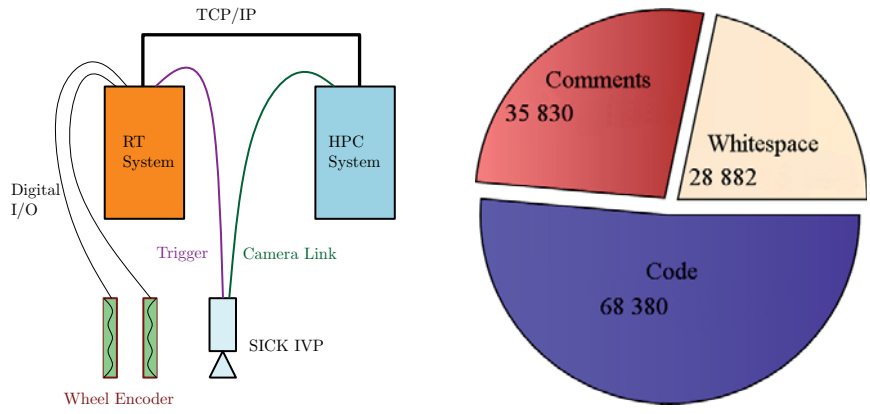
Chapter 11

The Data Processing Realization

11.1 Data Flow and Real Time Processing

Figure 11.1 shows the data processing scheme. Two processing paths lead to the final result; the blue path contains the sample acquisition used for the training of the classifier. The samples are manually scanned with an off-the-shelf flatbed scanner and prepared for training of the support vector machine. The green nodes show the two of processing steps of the real-time constrained, field system. The data acquired is three-dimensional and undergoes several processing steps (see Part II for details): data acquisition (1); basic filtering segmentation, shape and boundary extraction (3); and finally unfolding of the shapes and feature extraction (4) (see Section 9.2). The plane boundary is extracted and elliptic Fourier descriptors are computed (see Section 3.4.3), and classified by the previously trained, support vector machine (6).

11.2 Software Design and Key Components Description



(a) Computing Components and Sensors of the Evaluation Platform.

(b) Code Statistics.

Figure 11.2: System Components Scheme and Software Statistics

A performance analysis of the evaluation algorithms without a short excerpt on the system that embeds them, would be incomplete. The test platform is designed and realized in the research project SmartWeeder and elaborated on in the following publications: Šeatović and Grüninger (2007); Šeatović (2008); Šeatović et al. (2009, 2010). The system is schematically reduced to its essence in the Figure 11.2a.

RT System

The real-time system is realized as a QNX-OS computer with a digital I/O card for the navigation and measurement control with hard real-time constraints. The system computes the position and orientation of the main wheel axis on the fly: $L_t = \{x_t, y_t, \phi_t\}$, $t = 1, 2, \ldots, n$. The camera is triggered at fixed time or distance intervals. Distance-based triggering ensures resolution of the data in the travel direction; see $y-$axis in Section 10.2. The navigation messages are transferred over the Gbit Ethernet connection to the HPC system.

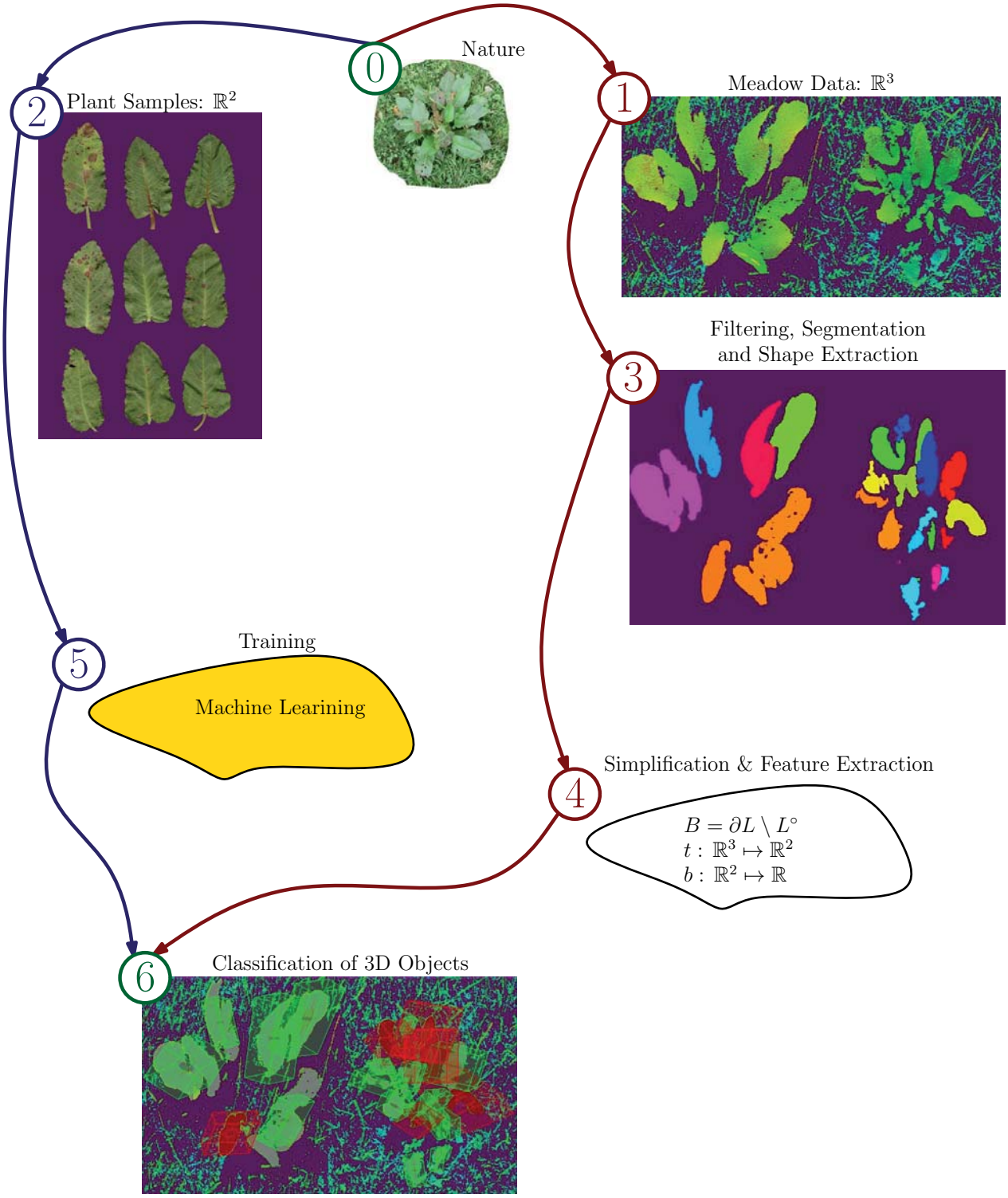


Figure 11.1: The Flow from the Raw-Data to Classified Plant Species.

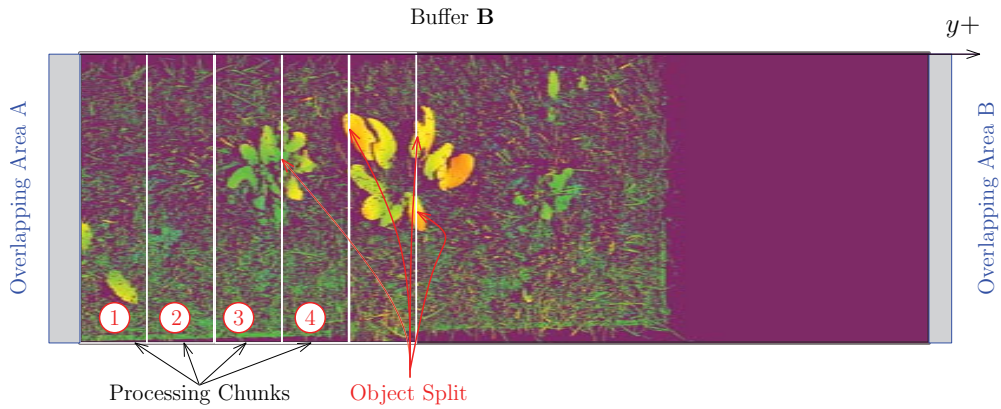


Figure 11.3: Ring Buffer Scheme with Processing Chunks and Overlapping Areas.

HPC System

This system is realized as an MS-Windows-based computer with two CPUs. The computing power of this at computer is insufficient to fulfill the minimal time constraints required for RT processing as described in the previous chapter, but it has been shown that, in principle, RT plant treatment is possible.

Full functional application on an HPC computer with all its components, results in approximately 68 000 lines of active C++ code (see Figure 11.2b), which is a mid-size software application; only relevant parts of the architecture will be explained:

1. The ring buffer.
2. The algorithms container.
3. The algorithms class.

Load balancing during an RT operation is a major problem, because the data processed is not independent at all. Consider the data in Figure 11.3 which shows the layout of a ring buffer that contains some data. The ring buffer contains two overlapping areas; one at the beginning of the buffer denoted by A, and one at the end of the buffer denoted B.

The ringed numbers denote *processing chunks*, subsets of points in the buffer that are processed in the *sequence* emphasized by the transparent white rectangles. The size of a single chunk is platform-dependent and is determined experimentally: the parameters are determined by the average speed of the vehicle, the single core processing speed and the number of CPU cores. The size of buffer is arbitrary but limited by the size of the RAM. The experimental parameter optimization is also called *system tuning*. With all CPU cores utilized evenly, the tuned system scales well. At this point the implementation details are neglected, but a few key classes are explained:

Buffer B is a class that provides raw and processed data storage (see Section 9.1) and encloses one container of *operators,* $\mathbf{C}_o = \{o_1, \ldots, o_n\}$, $n > 0$.

Operator o_n is a single computer algorithm. It operates on a subset of data in buffer **B**; see Section 9.2. The operators are designed and implemented to fulfill the following constraints:

1. The region of interest (ROI) or processing chunk is not limited by size or data format.
2. Operators can, but need not exchange results amongst themselves (master-slave relationship).
3. Operators are "injected" into the buffer; they are executed by the buffer instance. This design allows for maximum flexibility; operations can be joined into groups that operate on one buffer, used for the data exchange between buffers, or used as stand-alone operations on multiple disjoint buffers.

The main challenge is parallel processing of *dependent* data. The dependency is caused by the row sensor, because each object observed is measured only once; see Section 2.1. Strictly considered, the problem is sequential; each row is independently observed but subsequent rows are fused to the depth image in the buffer (see Figure 11.3). A predefined number of observed rows build one *chunk*. The objects that fit into the chunk raster require no additional logic. The majority of the objects however, require additional logic; this is explained in more detail here.

The red leader lines in Figure 11.3 show leaves are split between two processing chunks and that will be processed in successive time slots; see the layout in Figure 11.6. The low-level filters are designed to operate in the ROI,

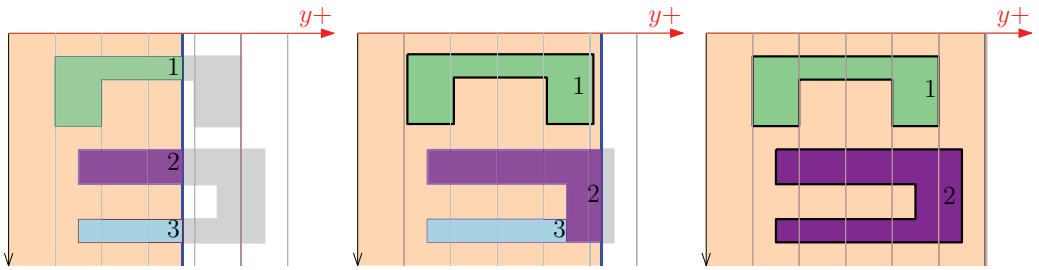


Figure 11.4: CCL Processing on the Set Open Towards $y+$ Direction.

therefore only the handling of the overlapping area requires slightly more logic; it is however, common sense and not worthy of detailed analysis. Object *extraction* on the other hand, is challenging.

Before object extraction is explained, the object constraints need to be defined:

1. An object is an *entity* when its boundary is closed.
2. The *area* of the object must be within a predefined interval: $A_{min} \leqslant A_o \leqslant A_{max}$. The area is derived from the sensor resolution; see Section 8.2.
3. The object length must not *exceed* the distance between the sensor and the nozzles; see Figure 10.1a.

The sensor delivers data in consecutive rows and they are gathered as processing chunks by the sensor driver. In the first step, low-level processing is initiated and its last step is object extraction; see Figure 11.5. Object extraction is dependent on connected-component labeling (CCL); only the objects that are confirmed as an entity are marked for further processing. Considering this problem from the image processing point of view: CCL is a per-image operation, and hence a finite data set. This is opposed to a regular digital image, where the scanned data is a depth image and it is open in one direction; in this case in $y+$ direction. Note that $y+$ can be virtually infinite.

In Figure 11.4 the computation of connected components is shown in three different consecutive stages. The processed row is denoted by the blue lines. The gray vertical lines represent the chunk boundaries. The far left image shows that the system currently has three components, all still "open" in the $y+$ direction.

The image in the middle shows the state of the system after some processing steps; the first object now has a closed boundary is marked for extraction and is allocated '1' (a unique label). Also in this stage, the second object overlaps with the third one, and the boundary is still not closed. For this case, the algorithm is implemented to label the remaining rows to the lower number and push the component(s) with higher number to the merging stack: Object number '3' becomes a child of the object number '2' in this particular case, which is then considered to be the parent. Once the boundary is closed, the algorithm assigns all child objects to the parent label; this is illustrated in the far right image of Figure 11.4. The label of the object is '2' in this example.

The next step is a fine-grained analysis of the extracted objects. Since the moving-window algorithms do not "see" the entire object, they usually neglect the edges and mark outlier points as edge points, even though they are not actually edge points. The recursive, filter which has been designed to compensate for this simplification, is then implemented; see Chapter 5. Each extracted object is the data source for the *one independent* recursive filter instance: This data separation then allows fully parallel and independent processing of each object extracted, see Figure 11.5.

Each recursive filter reprocesses the edges of the object, invokes connected-component labeling and final object extraction (see Figure 9.7b). In a subsequent step, but in same filter instance each object is treated as a separate point cloud of a 3-D surface. The point cloud is flattened to the plane by one of the projections elaborated in Section 4.5. Finally the plane shape described by its boundary, is ready for classification; the elliptical Fourier descriptors are computed for the shape boundary, and area descriptors for all shape points. The EFDs are classified by a support vector machine. In case that a shape is classified into the treatment group, the system marks the area around the centroid to be treated. The location is sent via TCP/IP to the RT computer, which activates the treatment when the system passes over the location.

Figure 11.7 shows the steps just described, as an image sequence. The aim of the experiment is to localize all Rumex Obtusifolius L leaves on that particular meadow patch. The localized leaves are colored red. Although there are obviously many more leaves of the same kind in the image, the system only located a few. The reason for this behavior lies in the settings of the decision component of the system which determines the probability threshold: the value that the classifier provides with each classification (see Section 6.4).

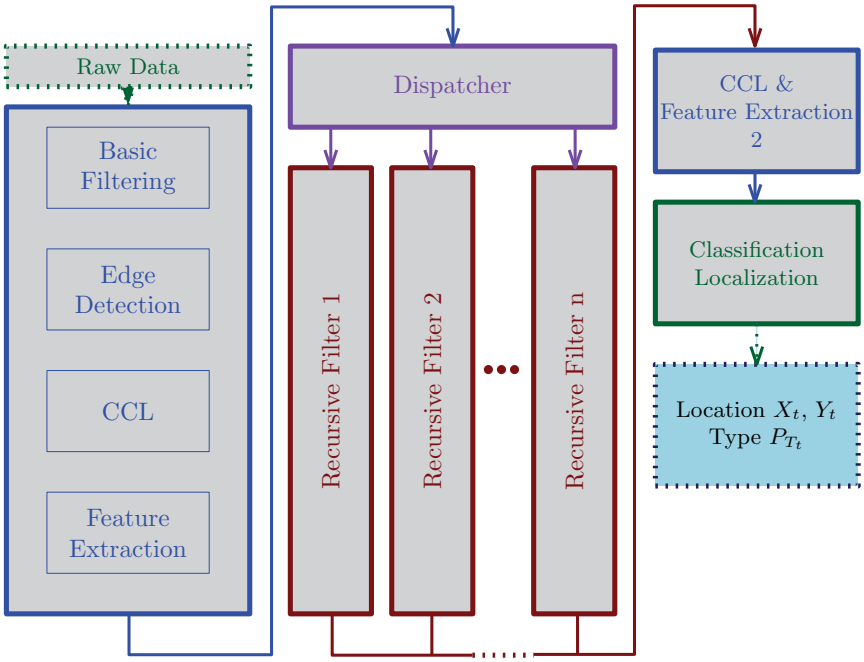


Figure 11.5: Processing Scheme with Thread-Blocks: each rectangle represents separate thread. Each thread operates independently and they exchange data through the adapted message passing interface (MPI). The data flow is mainly *sequential*, so several "tricks" are used to parallelize the processing.

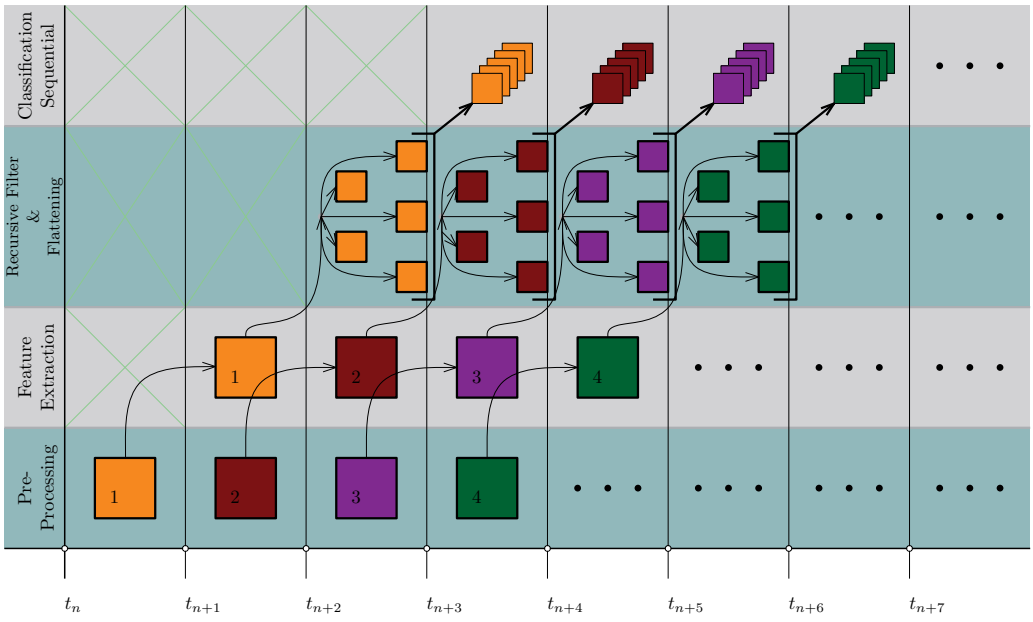
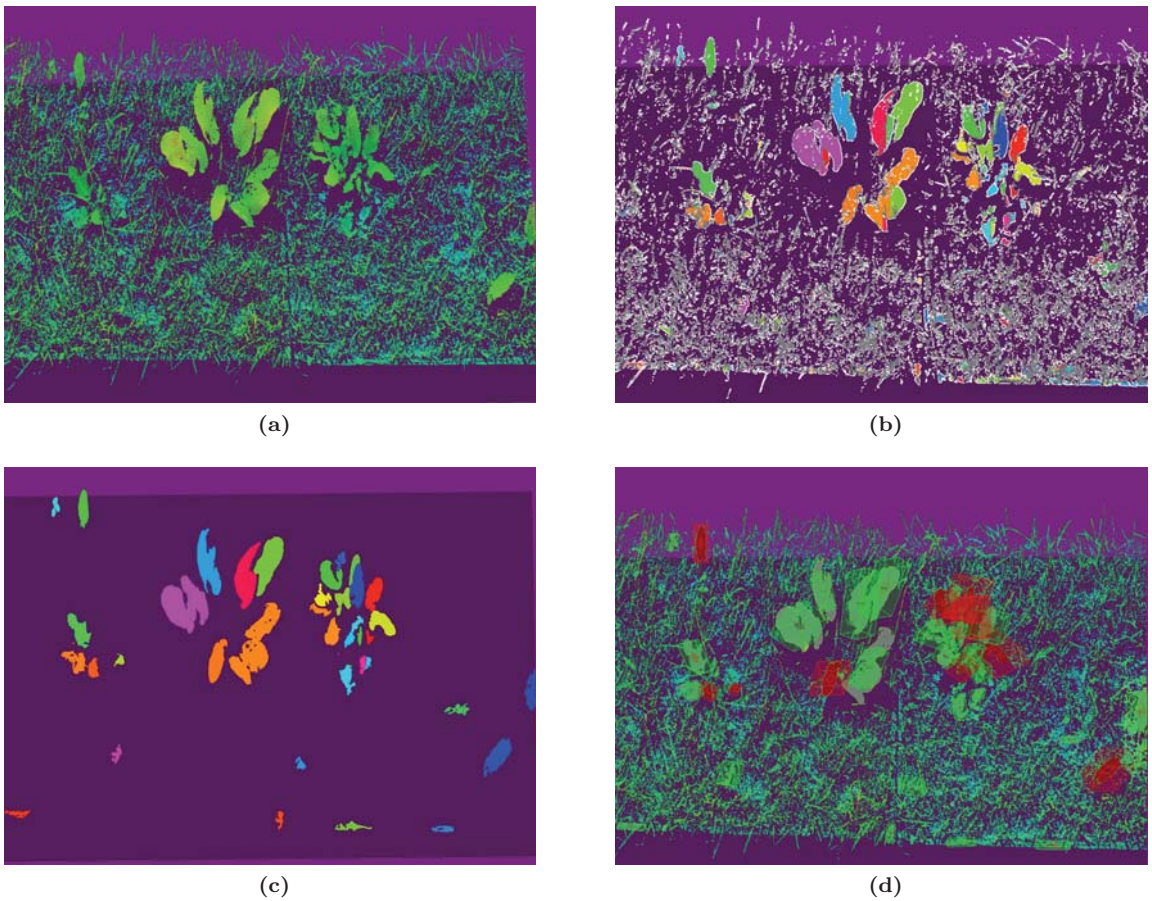


Figure 11.6: Load Balancing and Parallelization of the Processing.



(a) **(b)** **(c)** **(d)**

Figure 11.7: Illustration of Processing Sequence: (a) Raw Data, (b) Low-Level Segmentation, (c) Connected Component Labeling and (d) Classification Results.

Chapter 12

Experiments

12.1 General Remarks About Experiments

The experiments shown in this section evaluate the processing procedure on its accuracy, scalability and computation speed. The synthetic data with and without noise is created and processed. Such test data is created to be as similar as possible to the real data with one major advantage: The ground truth is 100% accurate. The evaluation consists of the comparison between the truth class and the one assigned through the SVM. The process is semi-automatic. The processing protocol is humanly readable and false classifications are emphasized for faster localization by the operator. The difference between field experiments and synthetic tests is in the source data. Field data initially has no real ground truth. The evaluation of the results is a manual process: The training of the SVM is based on a limited number samples acquired by the flat-bed scanner and they require manual correction, thus the operator improves classification results through the extension of the training data to include field data. The aim of the experiments is to show the capabilities of the three major processing components: recursive filter, surface parametrization and flattening, and real-time capabilities of the system.

12.2 Data Description

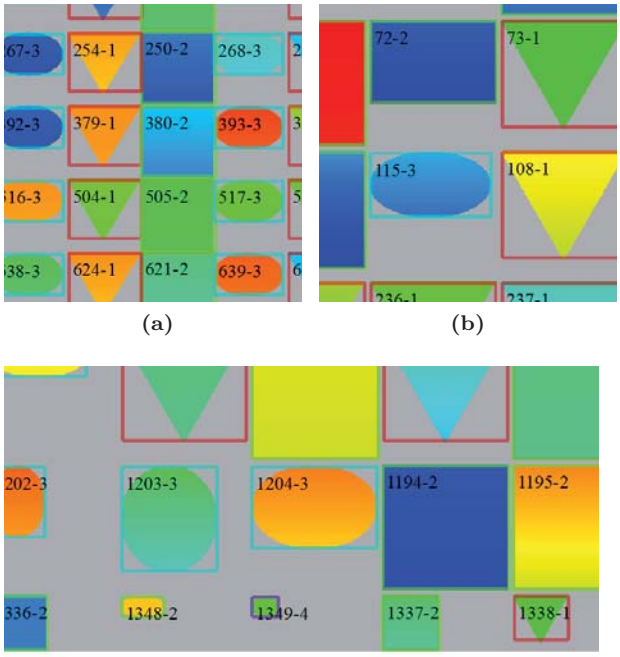
Since the real data is obtained on a real meadow patch, its description is obsolete. Each sample of field data has the same constraints: the geometry of an acquired depth image is limited in its width to $1\,m$ and its height to $\approx 0.35\,m$ (the length of the sample varies), as described in Section 10.2. The spatial resolution of data is determined by the system parameters and is limited by the camera: a width resolution ($x-$direction) of $0.65\,mm$ and a length resolution ($y-$direction) of $1\,mm$, yields a resolution $\approx 1.54 \times 1.00\,pt/mm = 39.1 \times 25.4\,dpi$. Using the rule of thumb from Eq. (8.7), the minimal object size is $39\,cm$. The conclusion from Section 8.2 changes the requirement of the sensor resolution to $400\,dpi$, which is ten times higher than is available. Considering Table A.2 one can see that filigree leaf shapes are the first that suffer under resolution decimation. Therefore the following complex leaf types are excluded from field experiments (see Figure A.1):

- 9. Alternate.
- 17. Bipinate.
- 27. Odd Pinate.
- 28. Tripinate.
- 31. Even Pinnate.

Potentially false classification of 23. Trifoliate as 24. Perfoliate is taken into account. This reduction of leaf shape classes results in a significantly lower required sensor resolution of $\approx 100\,dpi$, and a minimal object size of $\approx 150\,mm$. The downside of such a reduced resolution is that it limits the classifier a-priori; the occurrence of these excluded shapes in the data will inevitably lead to false detections, and hence classification errors. To achieve a spatial resolution of $\geqslant 400\,dpi$, covering the width of $1\,m$, is technically and financially a challenge currently (2012), and cannot be realized at this stage: the hardware demand of ten or more vision systems that provide a $1\,kHz$ data rate at that resolution is (at the present moment) an investment of more than 115 000 EUR. This is besides the technical challenge of transferring $\sim 30\,MiB\,s^{-1} = 10 \times 3072 \times 1000 \times 1024^{-2}\,bytes\,s^{-1}$ reliably and synchronously from the ten cameras to the computer system for real-time processing. The resulting point cloud requires a transfer and processing rate of $\sim 351.5\,MiB\,s^{-1}$, and with the DDR3-1333 memory having a theoretical transfer rate of $6400\,MiB\,s^{-1}$, just the transfer of data requires 18% of the memory bandwidth.

Total Objects	196
Correctly Classified	189
False Classified	7
Error Rate [%]	3.57

Table 12.1: Synthetic Data Test Summary.



(a) **(b)**

(c)

Figure 12.1: Synthetic Data: Only parts of the files are shown to improve readability. For full images see Section A.5. 12.1a Training set, manually classified. 12.1b Excerpt of the successful classification. 12.1c Classification errors due to low resolution of shapes.

The experimental system configuration is realized as described in Section 10.1 and Šeatović and Grüninger (2007).

The synthetic data sets are divided into two groups: simple planar shapes with constant and variable orientation, and complex non-planar shapes with variable orientation. Orientation of the shape is defined by the shape distribution in $\mathbb{R}^3$ and its moments; see Section 9.7 for details. Synthetic data is created to address each algorithm separately. The example of simple shapes is shown in Figure 12.1.

12.3 Synthetic Data and Results

A set of simple geometrical shapes is created to test the system performance. In Figure 12.1 a test set without any noise applied, is shown. In the first testing stage a system has to reliably segment and classify the noise and deformation-free test set. The classifier is trained only using ideal shapes. Tests on synthetic data benefit from being in controlled environments: the data can be scaled, and modified without effort, so the bottlenecks and application malfunction can be located efficiently.

The experiment shows the RT capability of the system and its performance considering the basic filtering, segmentation, object extraction, boundary extraction and classification. The aim of the tests with synthetic data is to prove the basic functionality, recognize issues and reproduce misbehavior of the software components; if this is achieved the aims will have been met. Current software implementation provides RT treatment of $0.67\,ms^{-1} = 2.4\,km/h.$; comparing to the minimal requirement of $1.4\,ms^{-1} = 5.0\,km/h$, given Table 8.1, yields an achievement level of 47%. Despite this low achievement level, the functionality of the system can still be analyzed and its performance evaluated.

Table 12.3 is an excerpt of the classification result table. The classifier was trained for four classes:

1. Triangle.
2. Square.

Label	Classification Machine	Classification Human	Probability %	X	Y	Z
1	1	1	97.09	58.0	50.0	180.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
! 1001	*1*	*1*	*48.84*	*742.0*	*1172.0*	*27.0*
*** 20**	**2**	**3**	**65.36**	**1494.0**	**22.0**	**397.0**
⋮	⋮	⋮	⋮	⋮	⋮	⋮
!* 123	**2**	**3**	**49.09**	**1498.0**	**245.0**	**101.0**
*** 613**	**2**	**3**	**77.66**	**1496.0**	**698.0**	**635.0**
*** 1010**	**2**	**3**	**74.97**	**1501.0**	**1153.0**	**31.0**
*** 1348**	**2**	**3**	**67.20**	**934.0**	**1496.0**	**534.0**
⋮	⋮	⋮	⋮	⋮	⋮	⋮
! 1198	*3*	*3*	*43.58*	*364.0*	*1405.0*	*27.0*
! 1207	*3*	*3*	*38.88*	*1403.0*	*1393.0*	*16.0*
! 1345	*3*	*3*	*48.71*	*588.0*	*1497.0*	*24.0*
⋮	⋮	⋮	⋮	⋮	⋮	⋮
!* 1341	**4**	**3**	**52.76**	**24.0**	**1493.0**	**567.0**
!* 1349	**4**	**3**	**36.56**	**1041.0**	**1496.0**	**380.0**

Legend:	
1111	Correct classification.
! 1111	Classification probability below threshold: $p < 65\%$
*** 1111**	**Classification error.**
! italic	Emphasizes correct classification with low probability.

Table 12.3: Excerpt of the Classification Results of Synthetic Data, see Figure 12.1.

3. Ellipse.
4. Others.

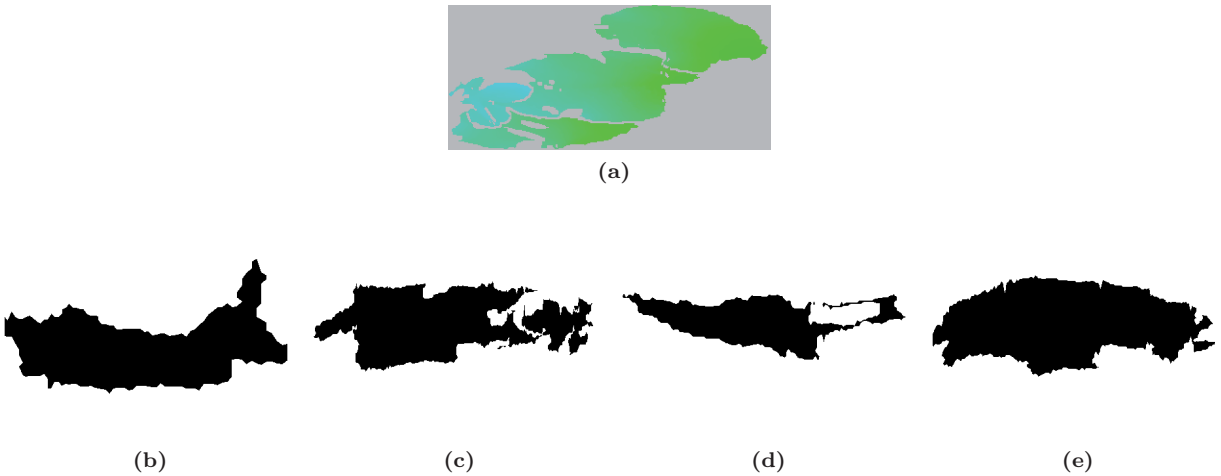
The training set is never processed in the test. All test files are different and differ from the training set in shape sizes and shape locations. In Table 12.3 parts of the classification result table are shown. The column *Classification* is sub-divided into *Machine* and *Human* columns. *Machine* contains the SVM classification results. Most of the correctly classified shapes receive a high *probability* from the classifier, although not *all*. A reasonable threshold that determines the minimum probability is necessary if the classification results are to be used for the treatment action.

Also significant, are the probabilities of the shapes that were classified into the wrong class. They depend on the sensor resolution: the smaller the shape, the higher the probability for classification error. Misclassified objects generally have low probabilities in all classes, though in this example shape 613 is classified with a probability of 78%. For each experiment, the quality of classification is determined by the classification probability. Shapes classified with probabilities *below* 65% are emphasized and marked with a '!' character preceding the label number. *Classification errors* are marked with a '*' preceding the label number and the font is bold. If *both* events occur then '!*' precedes the label number, see Table 12.3.

Artificial data tests have shown that the system can operate at $\sim .7\,ms^1$ and still reliably extract the shapes in numerous situations. The controlled environment results are to be considered carefully; they certainly do not justify a conclusion that the same performance can be expected in an unordered environment.

Plant Name	Training Samples
Dock	750
Clover	142
Dandelion	29

Table 12.4: Training Set Summary.



(a) **(b)** **(c)** **(d)** **(e)**

Figure 12.2: Segmentation Results Comparison: a) Original shape 2339; b)-e) Shapes extracted by recursive filtering.

12.4 Field Data and Results

The field data contains approximately $150\,m$ of different natural meadows around Tänikon, Switzerland. Since the data is from natural grassland, the ground truth is determined by a human operator. The camera covers the area of $\sim 1\,m$ which yields a resolution of $\sim 1.5\,pt/mm$. The resolution is approximately $10\times$ less than required; see Section 8.2. The software is expected to correctly classify objects larger than $25\,cm$. The setup of the experimental equipment is described in Section 10.2. The average speed of the vehicle is $0.7\,ms^{-1}$. The classifier is trained, using manually collected data listed in Table 12.4.

In order to test the RT performance, the experiment is built on the base system of SmartWeeder. In addition to the experiment using basic processing components, an additional processing step is implemented in MATLAB. This processing step consists of two processing components: recursive filter (Chapter 5) and flattening (Section 9.7). Since the algorithms' execution of these components are MATLAB scripts, their RT behavior is irrelevant. The next step, implementation of the algorithms in C/C++, is alleviated since the specification and the process is already tested in the script language. A skilled developer is able to implement optimized software components in approximately nine months. The proof of the concept is illustrated in following example.

12.4.1 Field Data Example 1

The following example demonstrates the improvement of recognition, leading to fewer *false* positive classifications. This false positive classification is caused by insufficiently accurate segmentation: the window-based edge detector is unable to locate all edges, and hence the analyzed boundary is the union of all the objects. The recursive filter has detected discontinuities in the data and has separated one shape into four new shapes. The original 3-D shape is shown in Figure 12.2a and is classified with $\sim 98\%$ probability as Rumex (1); recursive filter results yield four new shapes, which are shown in Figures 12.2b-12.2e. Each new shape's descriptors are re-classified and all of the leaves are classified as Taraxcum (20).

Figure 12.3 superposes the first 20 elliptic Fourier descriptor invariants of each new shape with the invariants of the original object 2339 (see Figure 12.2a). The first 20 invariants are significantly better separable than the higher order invariants. In Section 8.2 the correlation between the invariants for predefined shapes are shown in Figure 8.2b; obviously, correlation between feature vectors (vectors of invariants) increases with increasing order of invariants. The first ten descriptors have lowest correlation when compared to those of the other invariants. Figure 12.3 shows similar behavior, the first ten invariants differ significantly from each other; the last ten invariants differ significantly less in their magnitude.

Original shape descriptors are colored blue. Each new shape is denoted as "*New Descriptors 1-4*". The shapes are depicted in Figures 12.2b-12.2e. Corresponding colored bars are assigned in the same sequence as the shapes appear in Figure 12.2.The flattening (Section 9.7) procedure based on results of recursive filtering (Chapter 5) corrects the error of false positive classification for this case because:

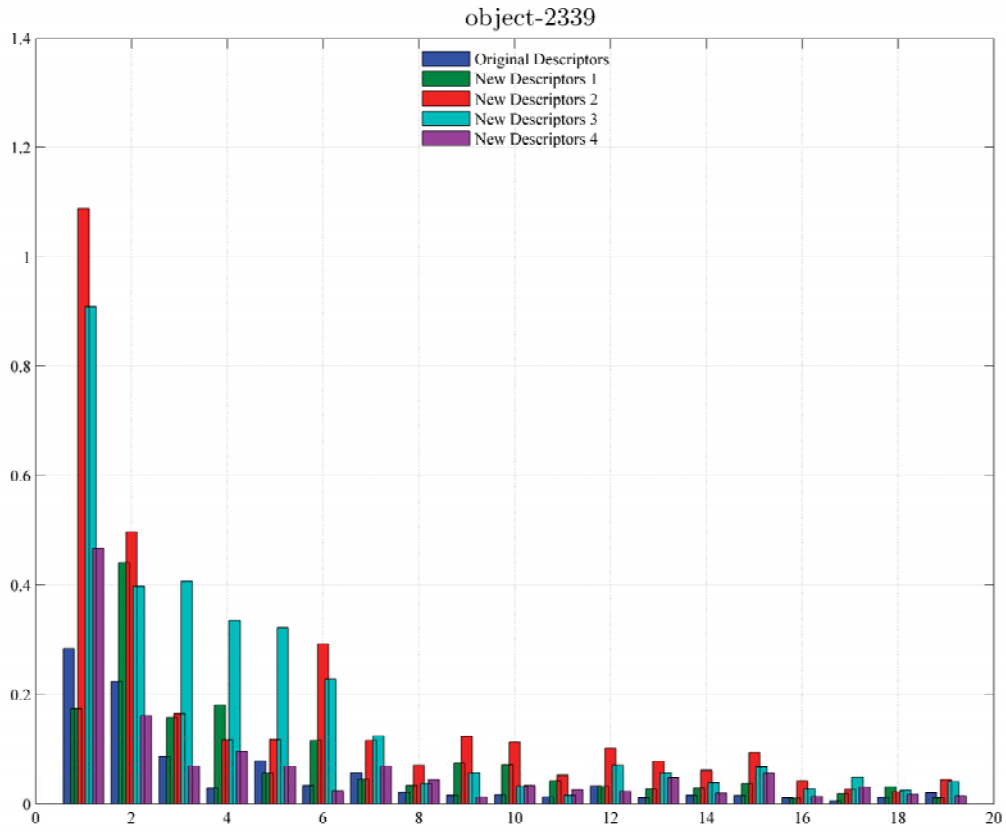


Figure 12.3: Elliptic Fourier Descriptors Differences.

- The classification of shape 2339 as Rumex is corrected, since shape 2339 is composed out of four disjoint shapes, hence the shape 2339 actually never existed. This experiment shows that more sophisticated filtering and segmentation algorithms such as one described in Chapter 9 are necessary to achieve more reliable results.
- Consecutive flattening process revealed that the new shapes are probably not Rumex, they rather belong to the class of Taraxcum. The result confirms that the training of support vector machine can be completed with less effort (grert samples) and still achieve acceptable results, presupposing that shapes are reliably extracted.

Additional computation effort required by a recursive filter is small price one has to pay, if compared to the effort required for collecting a large number of samples and training the classifier to achieve more reliable results. The effort increases if the plant diversity, which depends on the location, has to be modeled, too.

12.4.2 Field Data Example 2

In the Section 12.4.1 a successful correction of the classification results is shown. In this section the limitations of the process are shown. The limitations are significant. The advantages of 3-D data are negated if the sensor resolution is insufficient. The recursive filter improves the robustness to a certain extent, but it fails if height differences are below the sensor resolution. Consider the situation in Figure 12.4, especially the extracted composite objects 9450 and 9492. The leaves are connected to each other due to overlapping leaf blades. Each leaf blade is$\sim 0.2\,mm$ thick, just as thick as the sensor resolution is in the $z-$direction ($30\,cm/1024\,px \cong 0.29\,mm/px$). If two leaf blades are oriented so that the angle between the planes is $< 120°$ (empiric estimation) the chance to determine the separation line is higher than if the angle is greater than 120°. Edge detection fails in such situations; hence all successive steps will also fail since the shape will be analyzed as one leaf. In clover fields this situation occurs very often, because clover has thin leaves which cover the ground efficiently. The diversity of the resulting shapes is countless, and therefore any machine learning will fail if the shape boundary is the only data source. Segmentation can however, be significantly improved if multiple sensor systems are integrated; the topic is discussed in Chapter 13.

Table 12.5 shows an error rate of $\sim 27\%$, which is relatively high for the simple situation shown in Figure 12.4. Comparing the results to Table 2.2: Recognition Goal., where it is stated that more than 90% of 100% visible leaves should be correctly classified; the maximum error rate should not exceed 15%. In grassland a 100%

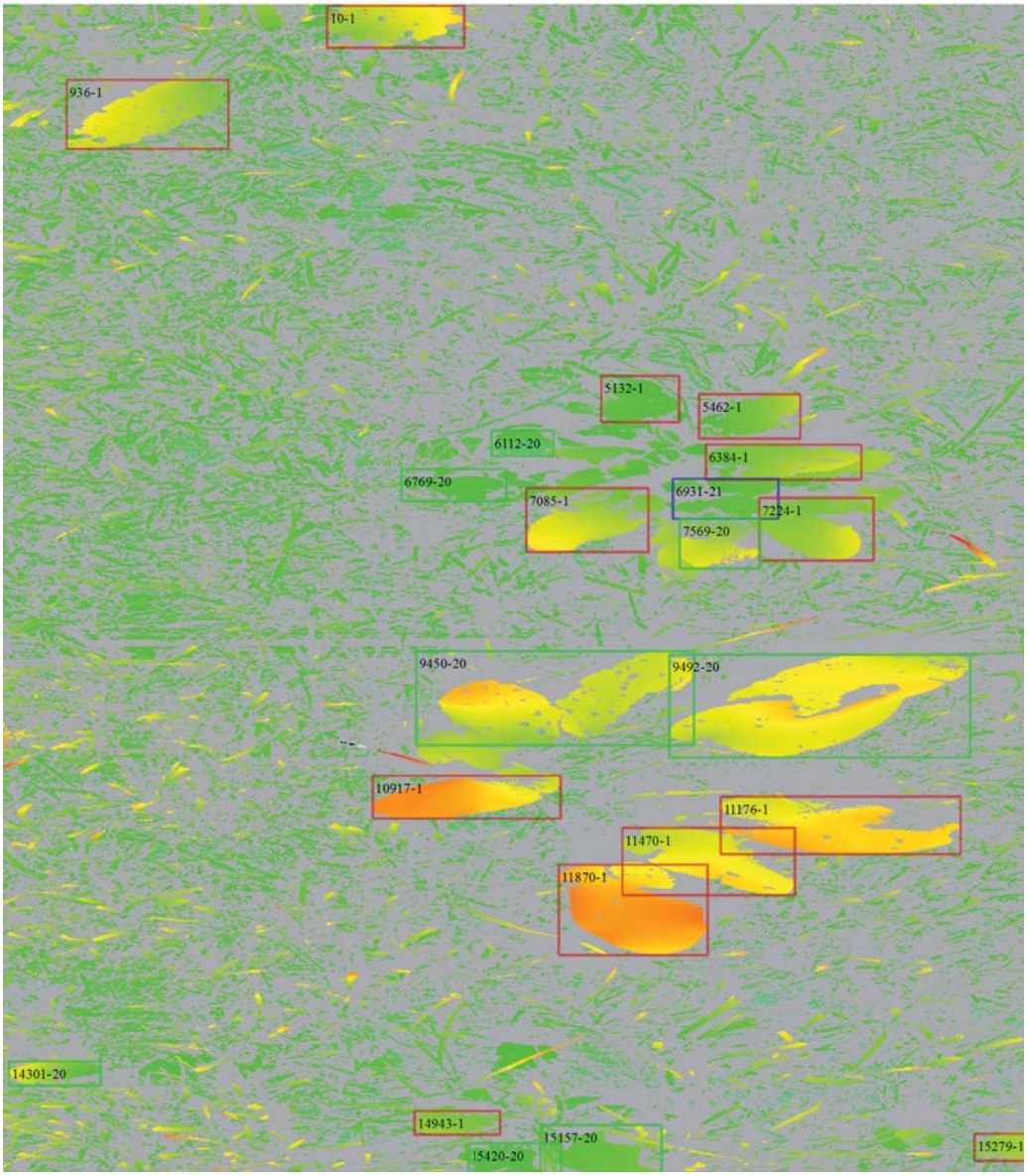


Figure 12.4: Field Test Situation.

(a) **(b)**

Figure 12.5: Recursive Filter Limits: Recursive filter fails to separate shapes in 12.5a due low sensor resolution, the transition from one shape to another is far too smooth do be detected. The overlapping leaves in 12.5b are even more challenging, they are treated as one damaged leaf, although these two leaves are completely intact.

Total Objects	22
Correctly Classified	16
False Classified	6
Error Rate	27.3%

Table 12.5: Summary of the Classification.

	Classification					
Label	Machine	Human	Probability %	X	Y	Z
10	1	1	97.44	635.0	25.0	472.0
936	1	1	99.99	351.0	126.0	460.0
5132	1	1	99.94	917.0	452.0	356.0
5462	1	1	98.53	1042.0	472.0	404.0
!* 6112	**20**	**1**	**60.27**	**782.0**	**503.0**	**280.0**
6384	1	1	96.38	1081.0	524.0	426.0
!* 6769	**20**	**1**	**48.11**	**702.0**	**551.0**	**315.0**
!* 6931	**21**	**1**	**56.71**	**1015.0**	**566.0**	**362.0**
! 7085	*1*	*1*	*51.56*	*856.0*	*591.0*	*453.0*
7224	1	1	88.26	1119.0	601.0	443.0
*** 7569**	**20**	**1**	**72.06**	**1008.0**	**618.0**	**444.0**
*** 9450**	**20**	**1**	**66.65**	**818.0**	**796.0**	**482.0**
*** 9492**	**20**	**1**	**66.65**	**1123.0**	**805.0**	**496.0**
10917	1	1	96.17	717.0	910.0	514.0
! 11176	*1*	*1*	*56.98*	*1147.0*	*942.0*	*511.0*
11470	1	1	99.21	995.0	984.0	487.0
11870	1	1	95.78	909.0	1039.0	540.0
! 14301	*20*	*20*	*58.92*	*245.0*	*1226.0*	*417.0*
14943	1	1	71.71	706.0	1283.0	418.0
! 15157	*20*	*20*	*61.74*	*872.0*	*1340.0*	*372.0*
15279	1	1	100.00	1378.0	1311.0	438.0
15420	20	20	74.41	772.0	1340.0	299.0

Legend:	
1111	Correct classification.
! 1111	Classification probability below threshold: $p < 65\%$
*** 1111**	**Classification error.**
! italic	Emphasizes correct classification with low probability.

Table 12.7: Classification Results of Field Situation. For field data see Figure 12.4.

visible leaf is rather seldom observed. More commonly in grassland, 80% and 50% visible leaves are seen; the corresponding maximal error margins are 25% and 30% respectively. In the situation shown in Figure 12.4, only 1 in 15 of the$\sim$ 100% visible leaves, was not correctly classified (14301).

The detailed classification results as shown in Figure 12.4/Table 12.7 are now briefly discussed. Misclassified objects generally have low probabilities for all classes, though in this example, the shape 613 is classified with a probability of 78%. For each experiment, the quality of classification is determined by the classification probability. Low probability is established to be *below* 65%. If shape classification is below that threshold the label is prefixed with a '!' character. Additional markers are shown in the table legend; see Table 12.7.

Segmentation can however, be significantly improved, especially if multiple sensor systems are integrated; the topic is discussed in Chapter 13

Size	CPU [s] Upload	Operations	$\sum$	GPU [s] Upload	Operations	$\sum$
1024×1024	0.0460	0.0620	0.1080	0.0000	0.0700	0.0700
2048×2048	0.1650	0.3430	0.5080	0.0000	0.2730	0.2730
3072×3072	0.3590	0.7890	1.1480	0.0000	0.5940	0.5940
4096×4096	0.6560	1.4290	2.0850	0.0000	1.0390	1.0390
5120×5120	1.0150	2.2740	3.2890	0.0000	1.6250	1.6250

Table 12.8: GPU vs. CPU.

12.5 Remarks About GPU Acceleration Potential.

Massively parallel platforms such as general-purpose graphics processing units (GP-GPUs) enable significant progress in HPC. The MATLAB scripts mentioned in the previous sections take a few minutes to a few tens of minutes to complete. The recursive filter has a high computation power demand and is necessary to accelerate execution of processing. One possible solution available is to use the GPU module of the OpenCV library, which provides a high-level C++ interface to the NVIDIA CUDA environment. The code fragment (see Listing B.2) was executed five consecutive times; the execution time is measured by processor clock and then converted to seconds.

The matrix size varies from 1024 to 5120 elements. The comparison between 1 CPU and 1 GPU shows a 50% performance increase; see Table 12.8. This straightforward evaluation discloses the potential of out-sourcing computations to a GP-GPU. Careful analysis is necessary though, to secure the algorithms' floating point precision and avoid rounding errors. Until the GP-GPU does provide double precision floating point arithmetic with same performance as single precision, one has count on a larger implementation effort and time, if an increase in acceleration of 3 to 5 times is to be realized.

12.6 Conclusion

The examples above illustrate the intended enhancement of the shape analysis using the recursive filter and flattening process. The approach must be improved if it is to be used in practical plant treatment. The following facts can be derived from the experiments:

- 70%-80% positive detections is not sufficient if the weed plants such as Rumex Obtusifolius L must be extinguished from the meadow.
- The processing speed is 50% slower than required for practical applications.
- In the next five years there will be probably a sensor available on the market, that is able to acquire 3-D data at the $400\,dpi$ resolution. Hence sensor resolution is not a research problem. The amount of data produced with such a sensor ($\sim 3.2\,GiB$)[1] and that needs to be processed within one second, represents the research problem.
- Novel processes of classifying 3-D shapes, based on 2-D scans have been developed; fine tuning of the learn $\triangleright$ detect $\triangleright$ treat sequence is however necessary to improve recognition rate.
- The GP-GPU encloses the potential for massive computation acceleration.

The experiments also show that the human work-force replacement goal is only partially achieved, so further research is absolutely necessary. Research areas such as machine learning, and short range ($\leqslant 2.5\,m$) data acquisition, are explored mainly by computer and image processing scientists; these areas seem to be only marginally explored by remote-sensing and geodetic scientists. It is however necessary to point out the problems stated from an engineering perspective. Remarks on further work, given in Part V might provide the impetus to solve the problem of single plant detection.

[1] $3.2\,GiB \approx 17^2[px/mm] \times 1,000,000\,[mm^2] \times 3 \times 4\,[bytes]$. Single precision floating point 3-D coordinates are assumed to be provided from the sensor.

Part V

Conclusion and Outlook

Chapter 13

Summary and Review

13.1 Review and Achievements

Machine vision systems acquire data fast and cover a wide range of the light spectrum. Furthermore, with a wide range of high resolution sensors available, and approximately 40 years of development in the field of image processing algorithms, vision systems are the first choice for many applications, including shape analysis. An intensity image as the data source for plant detection, provides satisfactory results if the *pose* of the camera to the object is optimal. In Section 8.2 such a situation is analyzed and the classification results show good separability of the shape features. However, a sub-optimal pose of the camera can significantly alter the projection of the shape on the sensor. In a worst case situation, shape alteration is so significant that correct classification is impossible. When grasslands are considered, the objects are mainly green, and segmentation procedures can fail as a result of low or absent contrast. Obviously 3-D data benefits from the additional dimension in such situations, although this demands more processing power. With 3-D, the segmentation results improve significantly when compared to those from intensity images, since the contrast loses its significance.

An additional advantage of a 3-D sensor over a vision system, is the greater tolerance of the the sensor pose to an object: the leaf observed by a 3-D sensor at an angle which would be disadvantageous for a vision system, will still become a 3-D point cloud whose shape is ready for analysis. When considering the different leaf types, the 3-D sensor also has its limitations. The low resolution of sensors currently available and a sub-optimal angle to the object can cause identical errors to those that occur in vision systems, despite a 3-D sensor covering a wide range of sensor-object poses. Even if the shape's 3-D data is acquired from a sub-optimal pose, broad leaves with a smooth boundary can still be recognized. Complex leaf boundaries unfortunately suffer a massive loss of information if the sensor resolution is not sufficient; this leads to the false classification of features (see Section 8.2).

Although the complexity of the algorithms increases by one order of magnitude, the additional dimension helps to speed up the preprocessing process. The solution outperforms any known system or algorithm; see Section 9.2.1, Chapter 12 and Šeatović and Grüninger (2007); Šeatović (2008). Besides the processing speed, the real advantage is achieved by the recursive filtering methods, by which overlapping leaves are more reliably separated; see Chapter 5. Each surface or surface patch is acquired as set of discrete 3-D points and parametrized by the recursive filter. More reliable boundary analysis is possible even if the object was *not* flattened: a boundary analysis is possible from the optimal perspective since a 3-D shape can be projected to an eigenvector plane computed from the shapes point cloud (see Section 9.7). The final, and probably the largest benefit of this research is the possibility to analyze the boundary shape *after* it has been flattened and projected onto the plane; see Section 9.7.

The algorithms developed in this thesis, enable more reliable shape classification, almost independently of the environment in which they are performing. They compensate for two major weaknesses of 2-D vision:

1. Poor performance under the low contrast conditions, such as grasslands where the color variation is low: typically a situation where there are green leaves on a green background.
2. The camera pose problem is resolved with help of a third dimension. Object edges are explicitly computed, in contrast to the intensity image where implicit edge computation is achieved by using intensity variations.

Therefore the decision to analyze the possibilities and limits of a pure 3-D approach, results in a faster and more robust segmentation algorithm. Furthermore, the abilities of real-time data acquisition, processing and classification have been realized, even though stated goals were not completely achieved. In Chapter 12 analyzed data is acquired with a customized indoor sensor. Sunlight significantly affects the observations and the applied filters are unable to compensate for this effect; see Šeatović and Grüninger (2007). Thus, the algorithms were evaluated under these sub-optimal, real-world, field conditions and still managed to yield reasonable results.

(a) (b)

Figure 13.1: RGB and NIR Images of a Plant.

13.2 Critics

The goal to replace human labor in the field, is obviously too ambitious to be reached in the short time that elapsed for the creation of this thesis. The major unresolved issue is the rather weak trade-off between the processing speed and its accuracy. Although the algorithms developed improve the segmentation process and enable more reliable shape extraction, they don't represent a silver bullet for single plant detection. Further research is necessary; single plant detection for weed treatment is one of the main goals in organic agricultural production (see Section 7.4) and some proposals for intermediate solutions and research are given in Section 13.3.

The preprocessing methods, basic segmentation and feature extraction developed are efficient but inaccurate. Recursive filtering methods are sufficiently accurate and reliable, but require far more computing power than expected and jeopardize the RT system. Algorithm implementation is definitely one cause of the processing power hunger, but the data flow and marginal computations should also be optimized. The hope that the next processor generation will solve the issue is unfounded. Alternative processing platforms and methods are to be evaluated; a promising candidate is the GP-GPU platform with accompanying library such as CUDA (see NVIDIA (2012)).

A radical review of the all-in-one real-time system requirement is necessary: simultaneous data acquisition, plant detection and treatment introduced a significant degree of complexity. Other problems encountered were technical restrictions regarding sensor technology and computer hardware; the latter having had a severely restrictive impact on algorithms and the software. The time constraints were far too tight, to achieve of the high reliability target. A compromise can be achieved if processing is decoupled from data acquisition and treatment; more about that later.

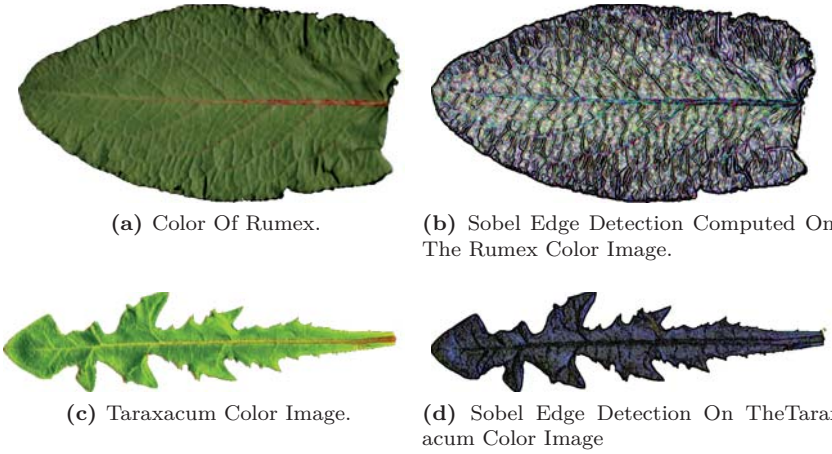
The machine learning library used for the classification of leaf shapes is integrated "as-is", without any changes. Therefore the performance of such an algorithm might not have been optimally applied. Certainly SVM is a state-of-the-art ML algorithm, even though it is used only for feature classification. Revision of the ML application and comparison to the alternatives, such as multilayer perceptron or neural networks needs to be investigated. Classification performance of the SVM library used for the experiments achieved $75 - 85\%$ and can be improved, the effort, however, is hard to estimate.

To achieve the goal of replacing human labor for single plant recognition, additional research based on the results achieved in this thesis, is necessary. In next section some few thoughts are presented with the hope of inspiring the next generation of researchers to contribute to the solution of single plant detection.

13.3 Further Work

In further research, the first step should focus on the improvement and extension of existing algorithms for:

1. Simultaneous processing of depth and intensity data.
2. Superposition of intensity and depth data. Besides the spatial information of each point, additional attributes can be stored. Existing data structures allow seamless and almost arbitrary extensions. The extended information can be obtained from following sources:
   (a) Reflection intensity provided by the 3-D scanner, which ideally should be equipped with a narrow band filter.
   (b) Visible color information such as RGB or gray images; see Figure 13.1a.



(a) Color Of Rumex.

(b) Sobel Edge Detection Computed On The Rumex Color Image.

(c) Taraxacum Color Image.

(d) Sobel Edge Detection On TheTaraxacum Color Image

Figure 13.2: Rumex Obstusifolius L and Taraxacum Officinale with Corresponding Textures / Venation Extracted by Sobel Edge Detection. Edge detection is computed with GIMP.

(c) UV or IR intensities; see Figure 13.1b.

3. Leaf venation represents the texture of the surface, which provides features that are a characteristic of a species. Venation structure, depending on the light wavelength observed, can differ in appearance; also, one cannot expect that venation will always be apparent in each or any image. In the case of missing information, the lack of venation also provides information which can be processed. An example of such a processing approach is shown in Figure 13.2. Consider that the figure represents only a coarse estimation of the possible solution; the robust solution is considered to be a research problem par excellence.

The leaf shape and leaf boundary contain sufficient information to identify the leaf of a plant specie, although venation can also contribute significantly to classification; see Section 3.3. The next logical step is extension of the existing boundary analysis algorithms with region analysis algorithms. Similar to the boundary elliptic Fourier descriptors invariants, region invariants also provide additional features about the area which could improve classification results. Zernike moments are such invariants; originally they were introduced for image processing purposes by Teague (1980). This method was used successfully for optical character recognition and handwriting recognition; a comprehensive survey is given by Trier et al. (1996). Region invariants are still a research topic and algorithms are continuously being improved; a publication by Belkasim et al. (2004) shows a proposal for a more effective computational approach.

Feature collection including EFD invariants, regional descriptors and texture information obtained by multi-spectral sensors, represents a complete set of information for leaf-based, single plant detection. A separate in-depth analysis for each kind of descriptors is probably necessary. Further research work at this point, branches in two directions:

1. The data analysis direction, where cascade feature extraction ▷ feature analysis ▷ feature classification is analyzed for new types of descriptors.
2. The sensor analysis direction, where high-resolution and high-accuracy multi-spectral sensor technology is provided as the research result.

New algorithms and sensor technology can be combined to provide autonomous, flexible, mobile units for single plant detection purposes. First steps towards such mobile units have already been taken; see Figure 13.3.

13.4 Final Conclusion

The problem elaborated in this thesis remains a research issue. There are some open questions left. The chances of solving this problem in next decade without strong financial support from funding organizations is unlikely. Projects such as SmartWeeder are an excellent starting point to cross-connect researchers from different disciplines. Unfortunately, branding the issue as pure research closes many doors to the industrial sector. Considering a paradigm shift from mass production of food, to organic food production with its higher revenue and profit expectations, the level of automation in food production will inevitably increase. The transition from mono-cultivation to heterogeneous cultivation will require adaptive systems, that provide autonomy and independence from human supervision; the automated production line comes to the crop field.

If the methods described here serve as one of basic building blocks for this future technology, this thesis will have contributed to that future.

(a) ZHAW Forbot With Measurement Pilar.

(b) University Of Hohenheim Unmanned Ground Vehicle (UGV) With ZHAW Measurement Pillar.

Figure 13.3: Mobile Mapping Vehicles.

Acknowledgments

On very first place, I want to thank my spouse Insa Will for supporting me during the writing of this text beyond any imaginable expectations and common sense.

My personal thanks belong to following persons and institutions:

Prof. Dr.-Ing. habil. Hansjörg Kutterer and Dr.-Ing. Hans Wernher van de Venn for their gentle, but steady guidance in the work.

Prof. Dr.-Ing. Uwe Sörgel and Prof. Dr.-Ing. Willfried Schwarz for their precious time spent on examining this thesis and for inspiring and helpful inputs.

The Gebert Rüf Stiftung that funded the SmartWeeder project and motivated me to continue to work on the issue even after the project was completed.

Colleagues Thomas Anken, Roy Latsch and Martin Holpp who have opened my eyes to the agricultural problems and the expectations of the people working on the field.

Paul Alves, Zorana Filak, Slaven Jurić, Vukašin Štrbac, Joachim Wirth, *Niels Wieffering*[1] and Benedikt Zebhauser for countless hours of review, discussion and argumentation.

Colleagues that had to finish their work without my support.

My very special thanks to my family who had to temporarily relinquish a father, spouse, son and brother. They have suffered as much as I did through the approximately 2500 hours of work that went into this thesis.

All the people who have developed the applications that have been used for the thesis deserve my deepest respect: foremost the LyX community and MikTex developers, followed by the IPE author(s), Inkscape authors(s), GIMP authors, and the Zotero women & men. Free software, which never failed to operate stably, was used to create 100% of the text and about 80% of the images. Many thanks to uncounted developers, and translators of the Open Source Community who spend their free time to make tools available for the scientific work. People, without you this text would never have been as good as it is now.

[1] Niels deserves very special regards and thanks for his text review. It was great pleasure to work with you!

List of Figures

1.1 Strawberry: (a) Whole Plant, (b) Represents Fruit Detail. . . . 3

2.1 The intensity image of Rumex Obstusifolius L and edge detection image. The error sources caused by the camera: Ill-pose of the sensor (A), the partly visible leaves (B), the bending of the leaf surface (C) and the leaf separation through the blade of grass (D). . . . 9
2.2 Desktop Computers: Theoretical CPU Performance Since 1989. . . . 10
2.3 The Processing Steps of Proposed System for Automatic Plant Detection. . . . 12
2.4 The Thesis Outline And Navigation. . . . 13

3.1 Translation (1), Rotation (2) and Scale Change (3) of a Shape. . . . 18
3.2 Simple Leaf Structure. . . . 19
3.3 Leaf Morphology[2]: (a) Leaf Shapes and Arrangement, (b) Margin and (c) Venation. From (de Bivort, 2013). . . . 20
3.4 Boundary Extraction. . . . 20
3.5 Shape and its Description Using Chain Codes. . . . 21
3.6 Polygonal Representation of the Boundary of a shape $\mathcal{S}$ and its discrete pendant $\mathbf{S}$. . . . 22
3.7 A Shape in the Cartesian Coordinate System. . . . 23
3.8 Curve $c(t)$ digitized by sensor of finite resolution. . . . 27
3.9 Computed Elliptic Fourier Descriptors for Shape S. . . . 28

4.1 Curve and Surface Definitions. . . . 29
4.2 Network types. Given a TRN, the function m converts TRN to a triangulated irregular network TIN. The inverse function m^{-1} restores the TRN from the TIN data. . . . 30
4.3 Transformation from TIN to TRN. Showing irregular triangulated network TRN (i: black circles) through the Delaunay triangulation (ii: red lines), definition of the regular grid in arbitrary density (iii: blue crosses), the subsequent interpolation using the grid results in triangulated regular network (iv: blue and red crosses, dotted lines in same colors). . . . 30
4.4 Basic Convolution Principle. . . . 32
4.5 Surface Patch. Purple dots depict the discrete points ($\mathcal{F}$) on surface Φ. Red and blue lines are parameter lines. Point $P \in \Phi$ is projected on Γ surface and then on the x, y plane (dark red arrow). The projection sketch represents an ideal case. . . . 34
4.6 Regular Surfaces. . . . 35
4.7 Cylinder Projection. . . . 35
4.8 Cone Projection Fundamentals. . . . 36
4.9 Discrete Curvature. . . . 38

5.1 Recursive Filter Types. . . . 42

6.1 Russell and Norvig (2009, 693 ff.) Artificial Intelligence Disciplines. . . . 45
6.2 (a) Learning from Samples (Statistical Learning Process) Scheme; the red lines represent the feature vector input into the ML algorithm. During the learning process the class of the feature vector is also required as an input. (b) Classification Process Scheme; the classification requires the feature vector only - the output of the ML algorithm is the feature class. . . . 46

6.3 (a) Maximum Margin Classifier Principle. Examples of Support Vector Machine: (b) Scheme of two-class problem where red circles and blue crosses represent classes. the margin boundaries are drawn in the class color. Decision boundary is drawn as thick black line between two margins. The purple circles emphasize support vectors. (c) The left image shows linearly non-separable data, if the problem is mapped to the higher dimensional space (feature space) the data might be more easily linearly separable into two classes (the right image). 48

8.1 Continuous Line Digitization: (a) Original (black) continuous line and its sensor values (red squares). (b) For 15° rotated continuous and corresponding sensor values (blue squares). (c) Superimposed sensor values with the original continuous line. 57
8.2 Simplified Analysis of Feature Vectors: (a) Normalized Mahalanobis Distance between the Feature Vector (Classes). (b) Graphical Illustration of Correlation Coefficients Between Single Features. . 58
8.3 Elliptic Fourier Descriptors of Similar Simple Leaf Shapes. 60
8.4 Distance and Angle Graph with k-Means Clusters. 60
8.5 k-Means Results; $x-$axis: simulated relative sensor resolution in %; $y-$axis: number of correctly distinguished leaf shapes. For each distance computation method different marker and color are shown. 61
8.6 Elliptic Fourier Descriptors for Complex Leaves. Large standard deviation of the features is caused by digitization errors: the tiny connections to petiole disappear in the digitized image. The shapes *9.Alternate* and *38.Opposite* are not affected: the connection is sufficiently thick to not to be affected by the sensor resolution. 61
8.7 Tripinate Shape and Boundary Approximation: (a) Original Shape. (b) Boundary approximation: red solid - longest boundary extracted; black dashed - classification boundary (20 harmonics); blue dashed - reconstructed boundary using 59 harmonics. 62
8.8 Sensor Scheme. 63
8.9 Laser Triangulation. 65
8.10 Profile Observation. 66

9.1 Range Image. 67
9.2 Standard Segmentation Results. 70
9.3 Raw Data and Region Of Interest Extraction Result. 71
9.4 Filter Functionality and Events. 73
9.5 Edge Logic. 75
9.6 Filtering Strategy Decision Base. 76
9.7 Comparison Between the Edge Detection With Standard Procedures and Recursive Filter Described in the Text. 78
9.8 Scanned Rumex Leaves. 79
9.9 Unfolding of the Point Cloud Along the Curve c_k. 79
9.10 Eigenvalues and Eigenvectors of the Point Cloud $\mathcal{P}$. 80
9.11 Original Data from Different Viewpoints. 82
9.12 Parameter Lines. The detail shows the main parameter lines: u_0 red and v_0 green. The blue lines are the v slice lines. Magenta connections between the lines show connection points between the two slices (maps). 83
9.13 Flattened 3-D Leaf. The Height is Color-Coded, warmer colors represent higher points; colder the lower. The height range: $[-4, 2] \approx 2.3\,mm$. 83
9.14 Flattened Leaf. From bottom to top: Raw data footprint. In the middle eigen axis transformation foot print with consecutive bicubic data interpolation. The flattened leaf on the top leaves the closest form to the expected one, if the leaf were put between two plane plates. 84
9.15 Second Leaf Example. Extracted part of Object 2 from Figure 9.2b. 85

10.1 (a) Vehicle Scheme and (b) Actual Experimental Platform in the Field. 90

11.2 System Components Scheme and Software Statistics 91
11.1 The Flow from the Raw-Data to Classified Plant Species. 92
11.3 Ring Buffer Scheme with Processing Chunks and Overlapping Areas. 93
11.4 CCL Processing on the Set Open Towards $y+$ Direction. 94

11.5 Processing Scheme with Thread-Blocks: each rectangle represents separate thread. Each thread operates independently and they exchange data through the adapted message passing interface (MPI). The data flow is mainly *sequential*, so several "tricks" are used to parallelize the processing. 95
11.6 Load Balancing and Parallelization of the Processing. 95
11.7 Illustration of Processing Sequence: (a) Raw Data, (b) Low-Level Segmentation, (c) Connected Component Labeling and (d) Classification Results. 96

12.1 Synthetic Data: Only parts of the files are shown to improve readability. For full images see Section A.5. 12.1a Training set, manually classified. 12.1b Excerpt of the successful classification. 12.1c Classification errors due to low resolution of shapes. 98
12.2 Segmentation Results Comparison: a) Original shape 2339; b)-e) Shapes extracted by recursive filtering. 100
12.3 Elliptic Fourier Descriptors Differences. 101
12.4 Field Test Situation. 102
12.5 Recursive Filter Limits: Recursive filter fails to separate shapes in 12.5a due low sensor resolution, the transition from one shape to another is far too smooth do be detected. The overlapping leaves in 12.5b are even more challenging, they are treated as one damaged leaf, although these two leaves are completely intact. 102

13.1 RGB and NIR Images of a Plant. 108
13.2 Rumex Obstusifolius L and Taraxacum Officinale with Corresponding Textures / Venation Extracted by Sobel Edge Detection. Edge detection is computed with GIMP. 109
13.3 Mobile Mapping Vehicles. 110

A.1 Predefined Leaf Shapes and Arrangement. (See also Chapter A: Results of the Analysis of the Predefined Shapes) 132
A.2 Elliptic Fourier Descriptors for Shapes 1-5. 133
A.3 Elliptic Fourier Descriptors for Shapes 6-10. 133
A.4 Elliptic Fourier Descriptors for Shapes 11-15. 134
A.5 Elliptic Fourier Descriptors for Shapes 16-20. 134
A.6 Elliptic Fourier Descriptors for Shapes 21-25. 135
A.7 Elliptic Fourier Descriptors for Shapes 26-30. 135
A.8 Elliptic Fourier Descriptors for Shapes 31-35. 136
A.9 Elliptic Fourier Descriptors for Shapes 36-40. 136
A.10 (a) Training Set and (b)Test File. 147

List of Tables

1.1 Influences on Features Selected for Analysis and Automatic Classification. 5

2.1 Sensor Requirements. 8
2.2 Recognition Goal. 10

8.1 Excerpt of Sensor Requirements from Table 2.1 with Reviewed Values. 62
8.2 Required Sensor Performance. 63

9.2 Complex Data Type: $d_{m,k}$. 69

12.1 Synthetic Data Test Summary. 98
12.3 Excerpt of the Classification Results of Synthetic Data, see Figure 12.1. 99
12.4 Training Set Summary. 100
12.5 Summary of the Classification. 103
12.7 Classification Results of Field Situation. For field data see Figure 12.4. 103
12.8 GPU vs. CPU. 104

A.1 Shapes and Sizes. 131
A.2 Scale Effect on k-Means Clustering. 143

List of Algorithms

B.1 Edge Detection Through Surface Normals. 153
B.2 Filter Reset. 154
B.3 Measurement Update. 154
B.4 Discontinuity Detected. 154
B.5 Recursive Least Squares Filter. 155

Nomenclature

[px]	Number (count) of picture elements.
AI	Artifical Intelligence
ANN	Artificial Neural Network(s)
API	Application Programming Interface
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CPU	Central Processor Unit
DSP	Digital Signal Processor
FPGA	Field Programmable Gate Array
GiB	gibibyte: 2^{30} bytes
GP-GPU	General Purpose Graphics Processing Unit
LVO	Last Valid Observation
MER	Maximum Error Rate
MiB	mebiyte: 2^{20} bytes
ML	Machine Learning
MPP	Minimum-Perimeter Polygon
NIR	Near Infra Red: Light wave length between 700 nm-1100 nm.
ROI	Region Of Interest
SE	Structured element
SMP	Symmetrical Multi Processing
STL	Standard Template Library
SVM	Support Vector Machine
TIN	Triangulated Irregular Network
TRN	Triangulated Regular Network
UGV	Unmanned Ground Vehicle
UV	Ultraviolet
WCS	World Coordinate System

Bibliography

About.com, 2012. Brute force - definition. http://cplus.about.com/od/glossar1/g/bruteforce.htm [Accessed on 12.5.2012]. URL http://cplus.about.com/od/glossar1/g/bruteforce.htm 75

AGG, E., 2012. ETH - applied geometry group - journal and conference papers. http://www.agg.ethz.ch/publications/journal_and_conference [Accessed on 4.2.2012]. URL http://www.agg.ethz.ch/publications/journal_and_conference 51

Agoston, M. K., 2005. Computer Graphics and Geometric Modelling: Implementation & Algorithms (v. 1). Springer. 33, 64

Aguado, A. S., Nixon, M. S., Montiel, M. E., 1998. Parameterizing arbitrary shapes via fourier descriptors for evidence-gathering extraction. Comput. Vis. Image Underst. 69, 202–221. URL http://portal.acm.org/citation.cfm?id=287277.287297 28

Alexandrescu, A., 2001. Modern C++ design: generic programming and design patterns applied. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 67

Arulampalam, M. S., Maskell, S., Gordon, N., Clapp, T., 2002. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. Signal Processing, IEEE Transactions on 50 (2), 174 –188. 41

Barto, A. G., Sutton, R. S., 1998. Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning). A Bradford Book. 47

Belkasim, S., Hassan, E., Obeidi, T., 2004. Radial zernike moment invariants. pp. 790–795. 109

Bennett, K. P., Mangasarian, O. L., 1992. Robust Linear Programming Discrimination Of Two Linearly Inseparable Sets. 49

Besl, P. J., Jain, R. C., 1988. Segmentation through variable-order surface fitting. Pattern Analysis and Machine Intelligence, IEEE Transactions on 10 (2), 167 –192. 3, 53

Bi, Z., Wang, L., 2010. Advances in 3D data acquisition and processing for industrial applications. Robotics and Computer-Integrated Manufacturing 26 (5), 403 – 413. URL http://www.sciencedirect.com/science/article/pii/S073658451000013X 31

Bishop, C. M., 2006. Pattern Recognition and Machine Learning. Springer. 45, 47, 48, 49, 50

Blais, A., Mertz, D., 2001. An introduction to neural networks. https://www.ibm.com/developerworks/library/l-neural/ [Accessed on 27.1.2012]. URL https://www.ibm.com/developerworks/library/l-neural/ 47

Borkowski, A., 2004. Modellierung von oberflächen mit diskontinuitäten. Ph.D. thesis, Fakultaet fuer Forst-, Geo- und Hydrowissenschaften der Technischen Universität Dresden, Deutsche Geodätische Kommission, Marstallplatz 8, D-80 539 München. 4

Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Levy, B., 2010. Polygon Mesh Processing. A K Peters. 31, 33, 34, 35, 38, 52

Bowsher, J., Johnson, V., Turkington, T., Jaszczak, R., Floyd, C.E., J., Coleman, R., 1996. Bayesian reconstruction and use of anatomical a priori information for emission tomography. Medical Imaging, IEEE Transactions on 15 (5), 673 –686. 52

Boxer, L., Miller, R., 2005. Algorithms Sequential & Parallel: A Unified Approach (Charles River Media Computer Engineering). Charles River Media. 71

Bradski, G., Kaehler, A., 2008. Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media. 48

Bresinsky, A., Körner, C., Kadereit, J. W., Neuhaus, G., Sonnewald, U., Strasburger, E., 2008. Strasburger - Lehrbuch der Botanik (German Edition). Spektrum Akademischer Verlag. 5, 18, 19

Brown, R. G., Hwang, P. Y. C., Wiley, J., 2005. Introduction to Random Signals and Applied Kalman Filtering (3 rd Edition) by. 41

Bruce, L. M., Tamhankar, H., Mathur, A., King, R., 2002. Multiresolutional texture analysis of multispectral imagery for automated ground cover classification. In: Geoscience and Remote Sensing Symposium, 2002. IGARSS '02. 2002 IEEE International. Vol. 1. pp. 312 – 314 vol.1. 17

Chang, C.-C., Lin, C.-J., 2001. LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm/ [Accessed on 26.11.2012]. URL http://www.csie.ntu.edu.tw/~cjlin/libsvm/ 47, 48, 49

Coeurjolly, D., Klette, R., 2004. A comparative evaluation of length estimators of digital curves. Pattern Analysis and Machine Intelligence, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2), 252 –258. 22

Cortes, C., Vapnik, V., 1995. Support-vector networks. In: Machine Learning. p. 273–297. 49

Crimmins, T. R., 1982. A complete set of fourier descriptors for two-dimensional shapes. Systems, Man and Cybernetics, IEEE Transactions on 12 (6), 848 –855. 28

de Bivort, B., 2013. Predefined Shapes. http://lab.debivort.org/.
URL http://lab.debivort.org/ 19, 20, 113

Dewhurst, S. C., 2005. C++ Common Knowledge: Essential Intermediate Programming. Addison-Wesley Professional. 67

do Carmo, M. P., 1976. Differential Geometry of Curves and Surfaces. Prentice-Hall, Englewood Cliffs, NJ. 33, 35, 37

Dummer, G. W. A., Amos, S. W., Amos, R. S., 1999. Newnes Dictionary of Electronics, 4th Edition. Newnes. 3

Durrant-Whyte, H., Jan. 2001. Introduction to Estimation and the Kalman Filter. No. 2.2 in Lecture Notes. The University of Sydney NSW 2006, Sydney. 41

EPFL, 2012a. Bluebrain | EPFL. http://bluebrain.epfl.ch/ [Accessed on 8.11.2012].
URL http://bluebrain.epfl.ch/ 46

EPFL, 2012b. LGG | research. http://lgg.epfl.ch/research.php [Accessed on 6.3.2012].
URL http://lgg.epfl.ch/research.php 51

Erickson, J., 2011. Computational geometry pages. http://compgeom.cs.uiuc.edu/~jeffe/compgeom/ [Accessed on 30.0.2011].
URL http://compgeom.cs.uiuc.edu/~jeffe/compgeom/ 33, 64

Gan-Mor, S., Clark, R. L., Upchurch, B. L., 2007. Implement lateral position accuracy under RTK-GPS tractor guidance. Computers and Electronics in Agriculture 59 (1-2), 31 – 38.
URL http://www.sciencedirect.com/science/article/B6T5M-4P0VCVV-1/2/b272b5264b2691d1ac4d5d6a3e6fe2cb 17

Ge, S., Xu, M., Anderson, G. L., Carruthers, R. I., 2007. Estimating yellow starthistle (centaurea solstitialis) leaf area index and aboveground biomass with the use of hyperspectral data. Weed Science 55 (6), 671–678.
URL http://dx.doi.org/10.1614/WS-06-212.1 17

Gebhardt, S., 2007. Automatic classification of grassland herbs in close-range sensed digital colour images. Ph.D. thesis, Mathematisch-Naturwissenschaftliche Fakultät, University Bonn. 4, 10, 17, 39, 52

Gebhardt, S., Kuehbauch, W., Apr. 2007. A new algorithm for automatic rumex obtusifolius detection in digital images using colour and texture features and the influence of image resolution. Vol. 8 of Precision Agriculture. Springer Netherlands, Institute of Crop Science and Resource Management-Crop Science and Plant Breeding, University of Bonn, Katzenburgweg 5, D 53115 Bonn, Germany, pp. 1–13. 17, 39, 52

Gelb, A., Kasper, Joseph F., J., Raymond, N. A. J., Price, C. F., Sutherland, Arthur A., J., 1974. Applied Optimal Estimation. 41

Gerhards, R., Christensen, S., 2003. Real-time weed detection, decision making and patch spraying in maize, sugar beet, winter wheat and winter barley 43. 10

Gonzalez, R. C., Woods, R. E., 2006. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA. 18, 21, 22, 32

Goodman, J. E., O'Rourke, J. (Eds.), Apr. 2004. Handbook of Discrete and Computational Geometry, Second Edition (Discrete Mathematics and Its Applications), 2nd Edition. CRC Press LLC, Boca Raton, FL.
URL http://www.crcpress.com/ 33

Google, C., 2012. Google maps. https://maps.google.com/ [Accessed on 26.6.2012].
URL https://maps.google.com/ 52

Granlund, G. H., 1972. Fourier preprocessing for hand print character recognition. IEEE Transactions on Computers C-21 (2), 195 –201. 23, 24

Grewal, M. S., Andrews, A. P., 2001. Kalman Filtering: Theory and Practice Using MATLAB, 2nd Edition. Wiley-Interscience.
URL http://www.worldcat.org/isbn/0471392545 41, 42, 43, 76

Hartley, R. I., Zisserman, A., 2004. Multiple View Geometry in Computer Vision, 2nd Edition. Cambridge University Press, ISBN: 0521540518. 64

Hilton, P. J., 2000. Laser-induced fluorescence for discrimination of crops and weeds. In: Gonglewski, J. D., Vorontsov, M. A., Gruneisen, M. T. (Eds.), High-Resolution Wavefront Control: Methods, Devices, and Applications II. Vol. 4124. SPIE, pp. 223–231.
URL http://link.aip.org/link/?PSI/4124/223/1 17

Ho, H. P., Wang, F., Papademetris, X., Blumberg, H., Staib, L., 2012. Fasciculography: Robust prior-free real-time normalized volumetric neural tract parcellation. Medical Imaging, IEEE Transactions on 31 (2), 217 –230. 52

Jazwinski, A. H., Apr. 1970. Stochastic Processes and Filtering Theory. Academic Press, published: Hardcover.

URL http://www.worldcat.org/isbn/0123815509 41

Josuttis, N. M., 1999. The C++ standard library: a tutorial and reference. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 67, 69

Kahmen, H., 2005. Angewandte Geodasie: Vermessungskunde (de Gruyter Lehrbuch) (German Edition). Walter de Gruyter. 34

Klette, R., Rosenfeld, A., 2004. Digital straightness–a review. Discrete Applied Mathematics 139 (1-3), 197 – 230, the 2001 International Workshop on Combinatorial Image Analysis.
URL http://www.sciencedirect.com/science/article/B6TYW-49YD4PJ-4/2/8a755750eee9d6517adbff2f20ee7dc2 22

Kobbelt, L., 2012. RWTH graphics: OpenMesh. http://www.openmesh.org/ [Accessed on 1.6.2012].
URL http://www.openmesh.org/ 31, 51

Kraus, K., 2007. Photogrammetry: Geometry from Images and Laser Scans. Walter de Gruyter, Berlin. 64

Krzysteczko, P., Münchenberger, J., Schäfers, M., Reiss, G., Thomas, A., 2012. The memristive magnetic tunnel junction as a nanoscopic synapse-neuron system. Advanced Materials 24 (6), 762–766.
URL http://dx.doi.org/10.1002/adma.201103723 46

Kuhl, F. P., Giardina, C. R., 1982. Elliptic fourier features of a closed contour. Computer Graphics and Image Processing 18 (3), 236 – 258.
URL http://www.sciencedirect.com/science/article/B7GXF-4D7JNT5-M/2/df4fbf50f43cb5917bcdb88fbaddfcc3 23, 24

Lapack, 2012. LAPACK — linear algebra PACKage. http://www.netlib.org/lapack/ [Accessed on 17.2.2012].
URL http://www.netlib.org/lapack/ 11

Laplante, P. A., 2004. Real-Time Systems Design and Analysis. Wiley-IEEE Press. 10

Latsch, R., Kaeser, A., Sauter, J., 2011a. Hot water steam in dock control. Landtechnik 66 (3), 170–172. 53

Latsch, R., Sauter, J., 2009a. Ergebnisse zur Ampferbekämpfung mittels Mikrowellentechnologie. Wissenschaftstagung Ökologischer Landbau: Werte - Wege - Wirkungen: 11.-13.02.2009, Zürich, Band 1: Boden, Pflanzenbau, Agrartechnik, Umwelt- und Naturschutz, Biolandbau international 10 (1), 163–16. 53

Latsch, R., Sauter, J., 2009b. Mikrowellentechnologie zur Bekämpfung des Stumpfblättrigen Ampfers. Agrarforschung Schweiz 1 (7-8), 170–172. 53

Latsch, R., Sauter, J., 2010. Microwave for dock control on grassland. In: Grassland in a changing world. Vol. 15 of Grassland Science in Europe. European Grassland Federation EGF, Kiel, pp. 169–171. 53

Latsch, R., Sauter, J., 2011. Ampferbekämpfung im Ökologischen Landbau. In: Landtechnische Lösungen zur Beikrautregulierung im Ökolandbau. Deutsches Institut für Tropische und Subtropische Landwirtschaft (DITSL) GmbH, Witzenhausen, pp. 251–264. 53

Latsch, R., Sauter, J., Hermle, S., Drr, L., Anken, T., 2001. Control of rumex obtusifolius l. in grassland using microwave technology. In: Tagung Landtechnik AgEng 2007. VDI-Berichte. VDI-Max-Eyth-Gesellschaft [Hrsg.], Hannover, pp. 501–506. 53

Latsch, R., Sauter, J., Kaeser, A., 2011b. Ampferkontrolle mittels Heissdampfinjektion. In: Band 1: Boden - Pflanze - Umwelt, Lebensmittel und Produktqualität. Vol. 1. Verlag Dr. Köster, Berlin, Giessen, pp. 115–118. 53

Latsch, R., Sauter, J., Knížatová, M., 2009. Ampferbekämpfungmpfung mittels Mikrowelle - energetische und monetäre Betrachtungen. Landtechnik 64 (5), 350–353. 53

Leica, 2010. Leica mojoRTK system - give your equipment some mojo! - leica geosystems - leica geosystems. http://www.leica-geosystems.com/en/Leica-mojoRTK-System_70426.htm?changelang=true [Accessed on 21.9.2010].
URL http://www.leica-geosystems.com/en/Leica-mojoRTK-System_70426.htm?changelang=true 17

Lengyel, E., 2002. Mathematics for 3D game programming and computer graphics. Charles River Media, Inc., Rockland, MA, USA. 31

Lomenie, N., Gallo, L., Cambou, N., Stamon, G., 2000. Morphological operations on delaunay triangulations. In: ICPR00. pp. Vol III: 552–555. 33

Loménie, N., Stamon, G., Jul. 2008. Morphological mesh filtering and alpha-objects. Pattern Recogn. Lett. 29 (10), 1571–1579.
URL http://dl.acm.org/citation.cfm?id=1377042.1377225 33

Luhmann, T., 2010. Nahbereichsphotogrammetrie, 3rd Edition. No. 3. Wichmann Verlag. 31, 64

MacQueen, J. B., 1967. Some methods for classification and analysis of multi-variate observations. In: Cam, L. M. L., Neyman, J. (Eds.), Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability. Vol. 1. University of California Press, pp. 281–297. 59

Mahalanobis, P. C., Apr. 1936. On the generalised distance in statistics. In: Proceedings National Institute of Science, India. Vol. 2. p. 49–55.
URL http://ir.isical.ac.in/dspace/handle/1/1268 58

Marsland, S., 2009. Machine Learning: An Algorithmic Introduction. CRC Press, New Jersey, USA. 47

Mathworld, W., 6 2013a. Convolution – from wolfram MathWorld. http://mathworld.wolfram.com/Convolution.html.
URL http://mathworld.wolfram.com/Convolution.html 31

Mathworld, W., 6 2013b. Cross-correlation – from wolfram MathWorld. http://mathworld.wolfram.com/Cross-Correlation.html.
URL http://mathworld.wolfram.com/Cross-Correlation.html 32

Mesa, 2011. Mesa imaging AG - SwissRanger SR4000 - miniature 3D time-of-flight range camera. http://www.mesa-imaging.ch/prodview4k.php [Accessed on 30.0.2011].
URL http://www.mesa-imaging.ch/prodview4k.php 64, 66

Meyers, S., 1995. More Effective C++: 35 New Ways to Improve Your Programs and Designs. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 67

Meyers, S., Jul. 2001. Effective STL: 50 Specific Ways to Improve the Use of the Standard Template Library. Addison-Wesley Professional Computing Series. Addison Wesley, Boston, published: Paperback.
URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0201749629 67

Meyers, S., 2005. Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition). Addison-Wesley Professional. 67

Microsoft, 2012. Hugues hoppe - homepage. http://research.microsoft.com/en-us/um/people/hoppe/ [Accessed on 15.3.2011].
URL http://research.microsoft.com/en-us/um/people/hoppe/ 51

Mthembu, L., Marwala, T., 2008. A note on the separability index.
URL http://arxiv.org/abs/0812.1107 50

Neto, J. C., Meyer, G. E., Jones, D. D., Samal, A. K., 2006. Plant species identification using elliptic fourier leaf shape analysis. Computers and Electronics in Agriculture 50 (2), 121 – 134.
URL http://www.sciencedirect.com/science/article/B6T5M-4HNYM8T-1/2/2f7f5d8dd3a49f8aec2cebf4e99d3920 17, 19, 22, 52

Nixon, M., Aguado, A. S., Jan. 2008. Feature Extraction & Image Processing, Second Edition, 2nd Edition. Academic Press, published: Paperback.
URL http://www.worldcat.org/isbn/0123725380 4, 21, 23, 28, 32

NVIDIA, I., 2012. Parallel programming and computing platform | CUDA | NVIDIA. http://www.nvidia.com/object/cuda_home_new.html [Accessed on 27.6.2012].
URL http://www.nvidia.com/object/cuda_home_new.html 108

Panneton, B., Guillaume, S., Roger, J.-M., Samson, G., 2010a. Improved discrimination between monocotyledonous and dicotyledonous plants for weed control based on the blue-green region of ultraviolet-induced fluorescence spectra. Appl. Spectrosc. 64 (1), 30–36.
URL http://as.osa.org/abstract.cfm?URI=as-64-1-30 17

Panneton, B., Guillaume, S., Samson, G., Roger, J. M., Longchamps, L., 2010b. Automatic detection of weeds in corn with induced fluorescence, 1–8. 17

PCL, X., 2011. About - point cloud library. http://pointclouds.org/ [Accessed on 25.8.2011].
URL http://pointclouds.org/ 31, 53

Perwass, C., Wietzke, L., 2012. Single lens 3d-camera with extended depth-of-field, 829108–829108–15.
URL http://dx.doi.org/10.1117/12.909882 64

Petrie, G., 2010. Mobile mapping systems. GEOInformatics.
URL http://web2.ges.gla.ac.uk/~gpetrie/Petrie_Mobile_Mapping_Systems_Jan-Feb_2010.pdf 52

Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P., Aug. 2007. Numerical Recipes 3rd Edition: The Art of Scientific Computing, 3rd Edition. Cambridge University Press, published: Hardcover. 47, 69

Reum, D., Zhang, Q., 2007. Wavelet based multispectral image analysis of maize leaf chlorophyll content. Computers and Electronics in Agriculture 56 (1), 60 – 71.
URL http://www.sciencedirect.com/science/article/B6T5M-4N08WW8-1/2/f0b701f028bfe5132fc3201ff795a366 17

Ristic, B., Arulampalam, S., Gordon, N., 2004. Beyond the Kalman Filter: Particle Filters for Tracking Applications. Artech House. 41

Russell, S., Norvig, P., 2009. Artificial Intelligence: A Modern Approach (3rd Edition). Prentice Hall. 4, 45, 46, 47, 113

Rusu, R. B., Oct. 2009. Semantic 3D object maps for everyday manipulation in human living environments. Ph.D. thesis, Computer Science department, Technische Universitaet Muenchen, Germany. 31, 53

Sachniss, C., Frese, U., Grisetti, G., 2012. OpenSLAM.org. http://openslam.org/ [Accessed on 29.4.2012].
URL http://openslam.org/ 41

Sattler, T., 2012. RWTH graphics: Image localization. http://www.graphics.rwth-aachen.de/index.php?id=406 [Accessed on 6.3.2012].
URL http://www.graphics.rwth-aachen.de/index.php?id=406 51

Sattler, T., Leibe, B., Kobbelt, L., Nov. 2011. Fast image-based localization using direct 2D-to-3D matching. In: Computer Vision (ICCV), 2011 IEEE International Conference on. pp. 667 –674. 51

Schneider, P., Eberly, D. H., 2002. Geometric Tools for Computer Graphics (The Morgan Kaufmann Series in Computer Graphics). Morgan Kaufmann. 33, 64, 81

Schölkopf, B., Smola, A. J., 2002. Learning with kernels : support vector machines, regularization, optimization, and beyond. MIT Press.
URL http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=8684 47, 49

Schuster, I., Nordmeyer, H., Rath, T., 2007. Comparison of vision-based and manual weed mapping in sugar beet. Biosystems Engineering 98 (1), 17 – 25.
URL http://www.sciencedirect.com/science/article/B6WXV-4PGPKY2-1/2/c3bddc4a10777eead1be99f658627c01 10

Scopigno, R., 2012. MeshLab. http://meshlab.sourceforge.net/ [Accessed on 1.6.2012].
URL http://meshlab.sourceforge.net/ 31

Šeatović, D., 2008. A segmentation approach in novel real time 3D plant recognition system. In: Gasteratos, A., Vincze, M., Tsotsos, J. (Eds.), Computer Vision Systems. Vol. 5008 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 363–372, 10.1007/978-3-540-79547-6_35.
URL http://dx.doi.org/10.1007/978-3-540-79547-6_35 4, 5, 8, 17, 70, 91, 107

Šeatović, D., Grüninger, R., Jun. 2007. Smart weeder: Novel approach in 3D object recognition, localization and treatement of broad dock in its natural enviroment. RAAD 2007. 5, 17, 89, 91, 98, 107

Šeatović, D., Kutterer, H., Anken, T., Sep. 2010. Automatic weed detection and treatment in grasslands. In: ELMAR, 2010 PROCEEDINGS. pp. 65 –68. 5, 17, 53, 70, 89, 91

Šeatović, D., Kutterer, H., Anken, T., Holpp, M., 2009. Automatic weed detection in grassland. In: Innovations to meet future challenges / Conference: Agricultural Engineering, Land.Technik - AgEng 2009. VDI-Berichte. Düsseldorf : VDI-Verl., Hannover, p. 530.
URL http://d-nb.info/999096710 17, 91

Sewell, M., 2011. Support vector machines. http://www.svms.org/ [Accessed on 27.1.2012].
URL http://www.svms.org/ 48

Shahidi, R., Tombropoulos, R., Grzeszczuk, R., 1998. Clinical applications of three-dimensional rendering of medical data sets. Proceedings of the IEEE 86 (3), 555 –568. 52

Shannon, C. E., 1949. Communication in the presence of noise. Proceedings of the IRE 37 (1), 10–21. 26

SI, 2012. BIPM - metre. http://www.bipm.org/en/si/si_brochure/chapter2/2-1/metre.html [Accessed on 27.9.2012].
URL http://www.bipm.org/en/si/si_brochure/chapter2/2-1/metre.html 63

Sick, 2013. 3D cameras. http://www.sick.com/ [Accessed on 09.06.2013].
URL http://www.sick.com/group/EN/home/products/product_portfolio/vision/Pages/3d_cameras.aspx 4, 66, 89

Simon, D., Aug. 2006. Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches, 1st Edition. Wiley & Sons, published: Gebundene Ausgabe.
URL http://www.worldcat.org/isbn/0471708585 41, 42

Sloboda, F., Sloboda, F., Klette, R., Klette, R., 1998. On the topology of grid continua. In: SPIE Vision Geometry VII. p. 52–63. 22

StatSoft, 2012. Neural networks. http://www.statsoft.com/textbook/neural-networks/ [Accessed on 27.1.2012].
URL http://www.statsoft.com/textbook/neural-networks/ 47

Steffen, S. J. G., Jul. 2006. Identification of broad-leaved dock (Rumex obtusifolius l.) on grassland by means of digital image processing. Vol. 7 of Precision Agriculture. Springer Netherlands, Institute of Crop Science and Resource Management -Crop Science and Plant Breeding, University of Bonn, Katzenburgweg 5, D 53115 Bonn, Germany, pp. 165–178. 17, 39, 52

Steinle, E., 2005. Gebäudemodellierung und -änderungserkennung aus multitemporalen Laserscanningdaten. Deutsche Geodätische Kommission bei der Bayerischen Akademie der Wissenschaften / C. Verlagd.Bayerischen Akademie der Wissenschaften.
URL http://books.google.ch/books?id=HNz6PgAACAAJ 52

Stroustrup, B., 2000. The C++ Programming Language, 3rd Edition. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 67, 68

Strubecker, Karl, P. D., 1964. Differentialgeometrie I, Kurventheorie der Ebene des Raumes. No. 1113/1113a in Sammlung Göschen. Walter de Gruyter & Co., Berlin. 33

Strubecker, Karl, P. D., 1969a. Differentialgeometrie II, Theorie der Flächenmetrik. No. 1179/1179a in Sammlung Göschen. Walter de Gruyter & Co., Berlin. 33, 37

Strubecker, Karl, P. D., 1969b. Differentialgeometrie III, Theorie der Flächenkrümmung. No. 1180/1180a in Sammlung Göschen. Walter de Gruyter & Co., Berlin. 33, 34, 35

Sutter, H., Alexandrescu, A., 2004. C++ Coding Standards: 101 Rules, Guidelines, and Best Practices (C++ in Depth Series). Addison-Wesley Professional. 67

Taubin, G., 1991. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on 13 (11), 1115 –1138. 52, 62, 75

Taubin, G., 2012. Publications by topic, brown university. http://mesh.brown.edu/taubin/publications-by-topic.html [Accessed on 6.3.2012].
URL http://mesh.brown.edu/taubin/publications-by-topic.html 52

Teague, M. R., 1980. Image analysis via the general theory of moments. Journal of the Optical Society of America (1917-1983) 70, 920–930.
URL http://www.opticsinfobase.org/ 109

Thornton, C., Qh, B., 1999. Truth-from-Trash Learning and the Mobot Footballer. 50

Trier, Ø. D., Jain, A. K., Taxt, T., 1996. Feature extraction methods for character recognition - a survey 29 (4), 641–662.
URL http://citeseer.ist.psu.edu/ 109

UniHohenheim, 2012. Research: Phytomedizin. https://phytomedizin.uni-hohenheim.de/69544?&no_cache=1&L=1 [Accessed on 7.3.2012].
URL https://phytomedizin.uni-hohenheim.de/69544?&no_cache=1&L=1 52

Unser, M., Aldroubi, A., Eden, M., 1993a. B-spline signal processing: Part II—Efficient design and applications. IEEE Transactions on Signal Processing 41 (2), 834–848. 77

Unser, M., Aldroubi, A., Eden, M., 1993b. B-spline signal processing: Part I—Theory. IEEE Transactions on Signal Processing 41 (2), 821–833, IEEE Signal Processing Society's 1995 best paper award. 77

Viau, A. A., Jang, J.-D., Payan, V., Devost, A., 2005. The use of airborne LIDAR and multispectral sensors for orchard trees inventory and characterization. Montpellier, France. 17, 62

Wang, L., 2005. Support Vector Machines: Theory and Applications. Vol. 177 of Studies in Fuzziness and Soft Computing. Springer Berlin, Heidelberg, Germany. 48, 49

Weis, M., 2010. An image analysis and classification system for automatic weed species identification in different crops for precision weed management. Ph.D. thesis, Stuttgart-Hohenheim, Institut für Phytomedizin. 52

Weis, M., Gerhards, R., 2007a. Feature extraction for the identification of weed species in digital images for the purpose of site-specific weed control. Vol. 6. Wageningen Academic Publishers, Wageningen, Netherlands. 52

Weis, M., Gerhards, R., 2007b. Identification of weeds from digital images. Verlag Grauer, Beuren, Stuttgart. 52

Weis, M., Gerhards, R., 2009. Detection of weeds using image processing and clustering. Vol. 69. Leibniz Institute for Agricultural Enineering (ATB), Potsdam-Bornim. 52

Weis, M., Sökefeld, M., 2010. Detection and identification of weeds. Precision crop protection - the challenge and use of heterogeneity Edition. Springer Verlag, Dordrecht, Heidelberg, London, New York. 52

Welch, G., Bishop, G., 2010. The kalman filter. http://www.cs.unc.edu/~welch/kalman/ [Accessed on 20.4.2010].
URL http://www.cs.unc.edu/~welch/kalman/ 41

Wilke, W., 2002. Segmentierung und Approximation großer Punktwolken. PhD thesis, Technische Universität Darmstadt, Darmstadt.
URL http://tuprints.ulb.tu-darmstadt.de/255/1/wilke_01.pdf 4, 53

Z+F, 2010. Z+F | 3D · laserscanner · laserscanning · software · modellierung · entwicklung · verkauf · dienstleistung. http://www.zf-laser.com/e_imager5006ex.html [Accessed on 1.6.2010].
URL http://www.zf-laser.com/e_imager5006ex.html 4, 66

Zoeller, F., 2012. Z+F USA, inc. » 2D profiling systems. http://www.zf-usa.com/2d-profiling-systems/ [Accessed on 29.6.2012].
URL http://www.zf-usa.com/2d-profiling-systems/ 8

Part VI

Appendices

Appendix A

Results of the Analysis of the Predefined Shapes

A.1 General

For each shape, this section shows how *rotation* or *decreasing* sensor resolution causes the computed elliptic Fourier descriptors of a shape to vary from the original feature vector, $\mathbf{x_0}$. The relative change in the euclidean distance, which is the difference between the original and the variation, expressed as a percentage, is computed as follows:

$$\varepsilon_i = \frac{\|\mathbf{x}_0 - \mathbf{x_i}\|}{\|\mathbf{x_0}\|} \cdot 100\% = \left(\frac{\sqrt{\sum_j (x_0^j - x_i^j)^2}}{\sqrt{\sum_j (x_0^j)^2}} \right) \cdot 100\%,\ i, j \in \mathbb{N} \wedge i \neq j \tag{A.1}$$

The figures and classification were computed with MATLAB script. The script is listed in Section A.4.1. Table A.1 contains the shape sizes in *mm* and the digitization resolution for each class; the analysis is based on this data set. The results however, can also be applied to other data sets provided that the relation between size and resolution is retained: $scale \cdot size \sim \frac{resolution}{scale}$.

Table A.1: Shapes and Sizes.

		Size [mm]		Resolution [pt/mm]		Ratio
Nr.	Class	Width	Height	Width	Height	W/H
1.	Acicular	59.1	4.1	20.03	20.03	14.25
2.	Falcate	48.6	12.8	20.01	20.01	3.79
3.	Orbicular	28.3	19.9	20.00	20.00	1.43
4.	Rhomboid	41.3	20.4	19.98	19.98	2.03
5.	Acuminate	51.1	15.6	20.03	20.03	3.28
6.	Flabellate	39.6	22.1	19.98	19.98	1.80
7.	Ovate	35.0	16.1	19.99	19.99	2.17
8.	Rosette	28.0	25.9	19.99	19.99	1.08
9.	Alternate	57.4	22.1	20.00	20.00	2.60
10.	Hastate	48.9	20.9	19.98	19.98	2.34
11.	Palmate	39.5	23.4	20.00	20.00	1.69
12.	Spatulate	48.7	14.0	20.00	20.00	3.48
13.	Aristate	47.6	16.0	20.02	20.02	2.98
14.	Lanceolate	54.0	11.5	19.96	19.96	4.71
15.	Pedate	41.5	29.6	19.99	19.99	1.40
16.	Spear Shaped	44.5	15.1	20.02	20.02	2.95
17.	Bipinnate	48.9	30.7	20.00	20.00	1.59
18.	Linear	51.9	9.5	20.04	20.04	5.44
19.	Pelitate	32.6	19.0	20.02	20.02	1.72
20.	Subulate	51.7	5.0	20.08	20.08	10.28
21.	Cordate	32.3	22.6	20.00	20.00	1.43
22.	Lobed	36.1	26.6	20.00	20.00	1.36
23.	Perfoliate	39.0	24.7	20.01	20.01	1.58
24.	Trifoliate	28.4	25.9	20.00	20.00	1.10
25.	Cuneate	48.4	14.8	20.00	20.00	3.28

Nr.	Class	Size [mm] Width	Height	Resolution [pt/mm] Width	Height	Ratio W/H
26.	Obcordate	39.1	15.6	20.01	20.01	2.50
27.	Odd Pinnate	41.9	25.4	20.00	20.00	1.65
28.	Tripinnate	49.5	31.7	19.99	19.99	1.56
29.	Deltoid	37.9	20.3	19.98	19.98	1.87
30.	Obovate	35.6	17.5	19.98	19.98	2.04
31.	Even Pinnate	34.4	24.2	20.02	20.02	1.42
32.	Truncate	35.4	18.0	19.99	19.99	1.97
33.	Digitate	39.6	31.3	20.01	20.01	1.27
34.	Obtuse	33.8	18.0	20.00	20.00	1.87
35.	Pinnatisect	50.2	17.8	19.99	19.99	2.83
36.	Unifoliate	47.4	17.7	19.99	19.99	2.68
37.	Elliptic	36.2	15.9	20.01	20.01	2.27
38.	Opposite	41.4	24.2	20.00	20.00	1.71
39.	Reniform	24.2	25.7	19.99	19.99	0.94
40.	Whorled	52.8	32.2	19.99	19.99	1.64
	Maximum Values	59.1	32.2	20.08	20.08	
	Minimum Values	24.2	4.1	19.96	19.96	

A.2 Effect of Altering Shape Orientation

The orientation of the shape relative to the sensor grid affects the quality of digitization. In Section 8.2 it is shown that some complex shapes suffer severe changes solely due to rotation. The full data set of 40 shapes is shown below. For clarity, only the first twelve elliptic Fourier descriptors are used. For each descriptor a mean value (a colored bar) and its standard deviation (red lines) have been computed and illustrated. Figure A.1 depicts analyzed shapes; the figures that follow refer to those leaf shapes.

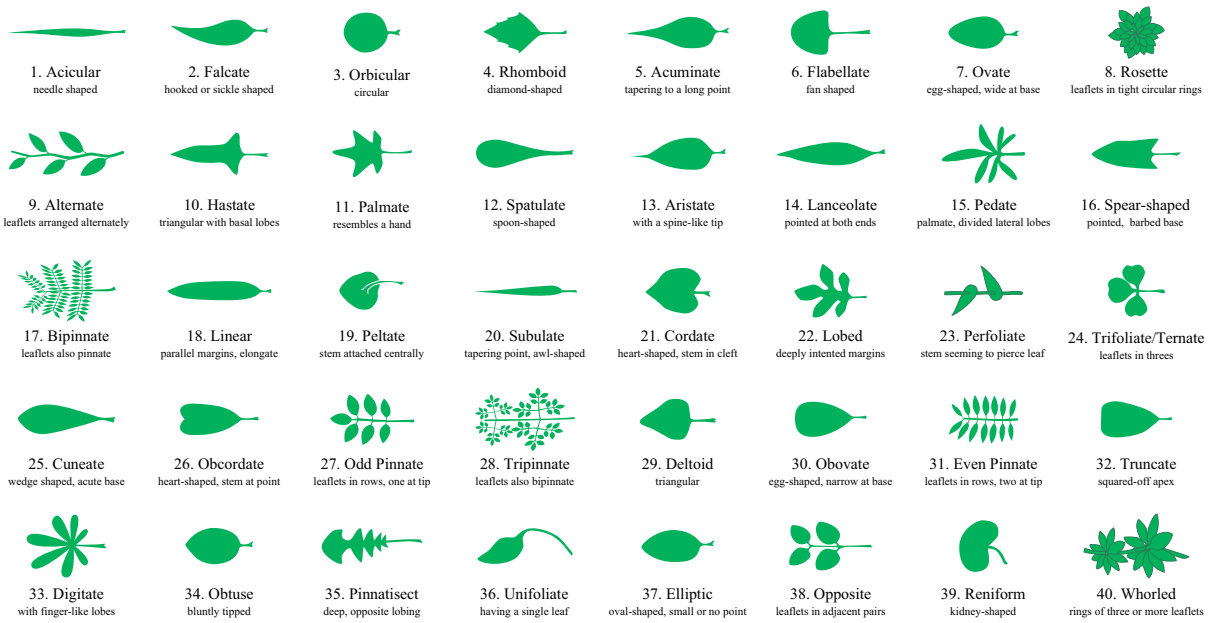


Figure A.1: Predefined Leaf Shapes and Arrangement. (See also Chapter A: Results of the Analysis of the Predefined Shapes)

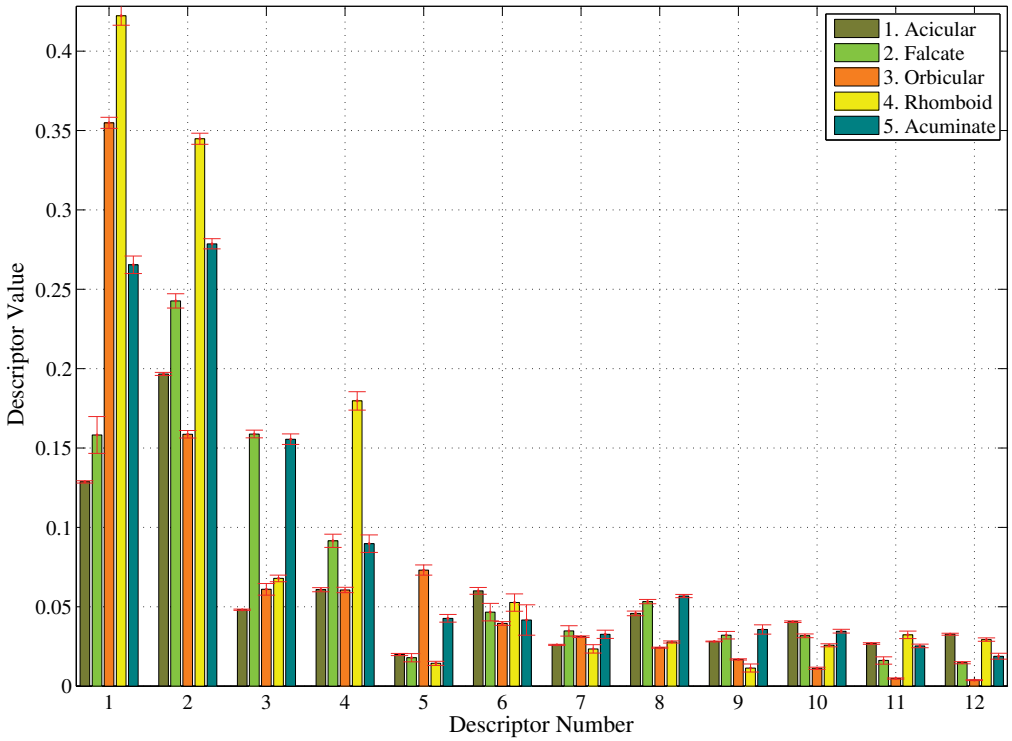


Figure A.2: Elliptic Fourier Descriptors for Shapes 1-5.

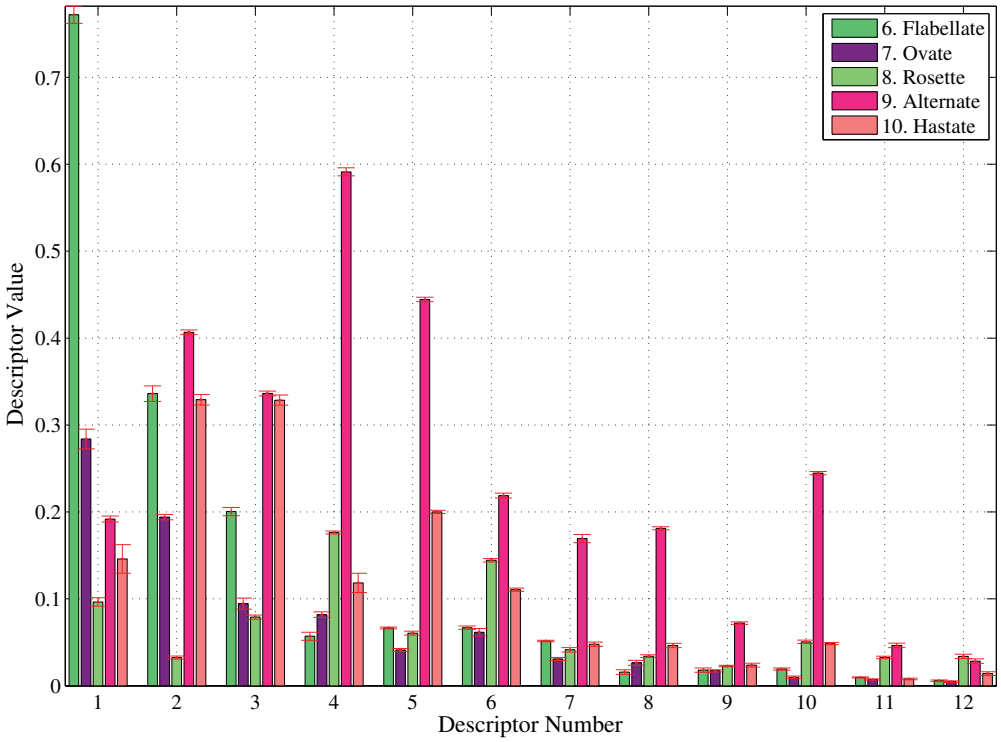


Figure A.3: Elliptic Fourier Descriptors for Shapes 6-10.

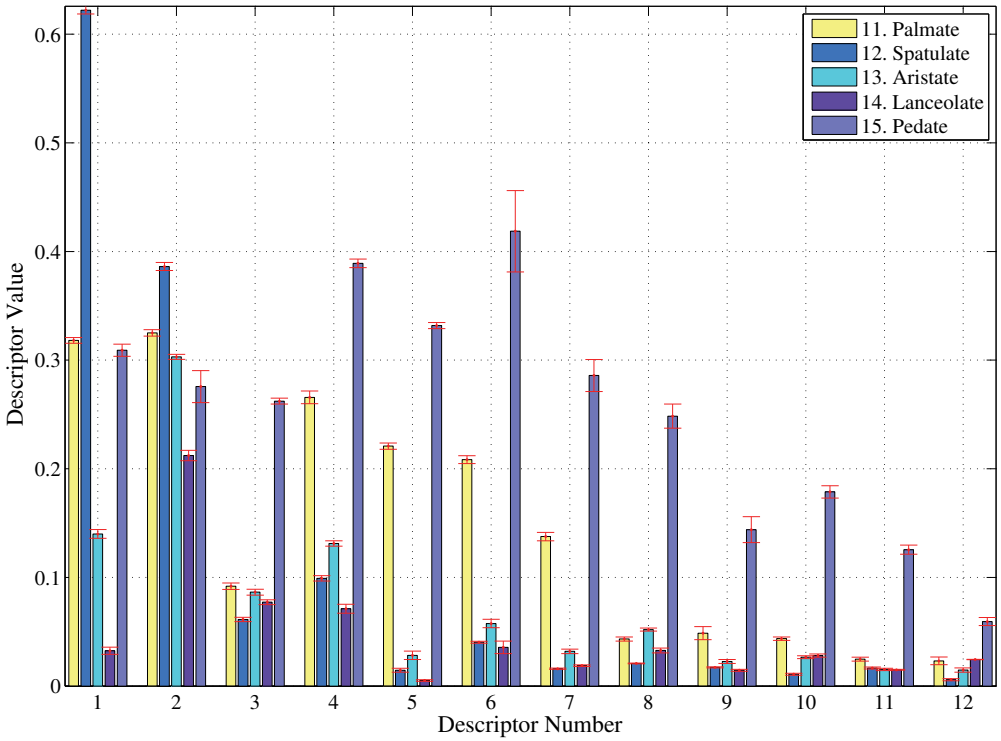


Figure A.4: Elliptic Fourier Descriptors for Shapes 11-15.

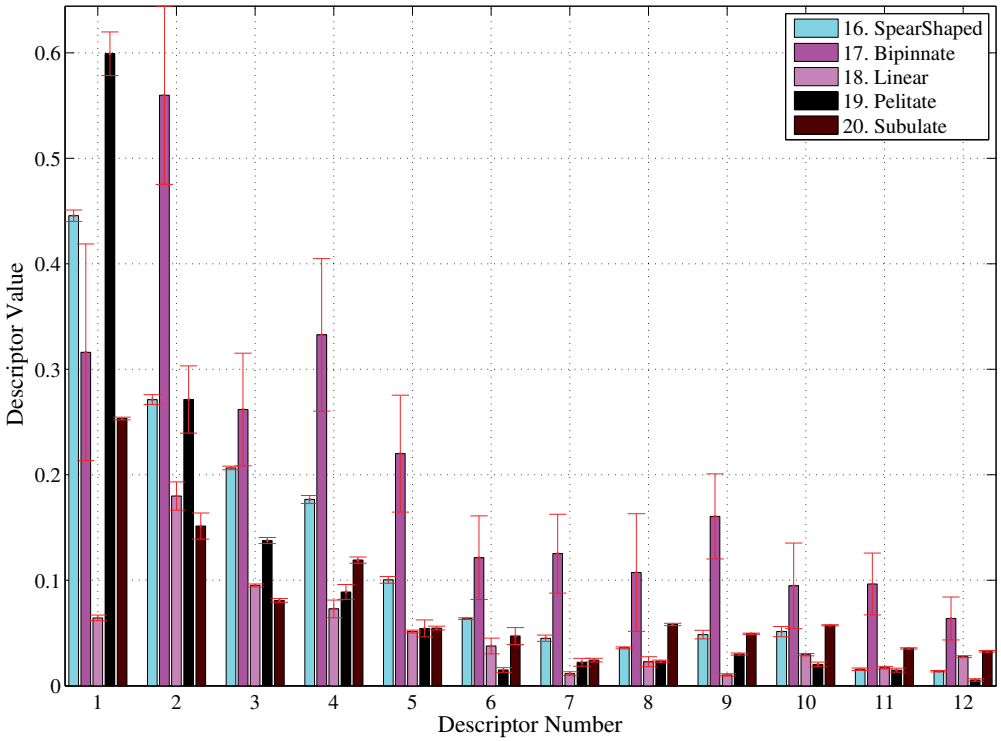


Figure A.5: Elliptic Fourier Descriptors for Shapes 16-20.

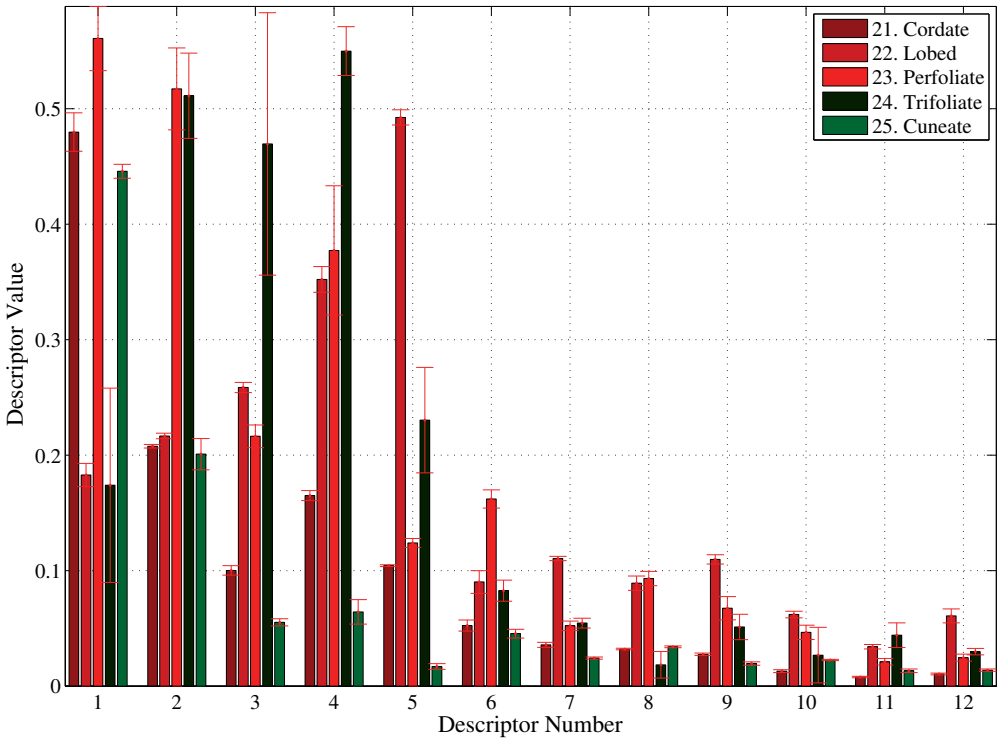


Figure A.6: Elliptic Fourier Descriptors for Shapes 21-25.

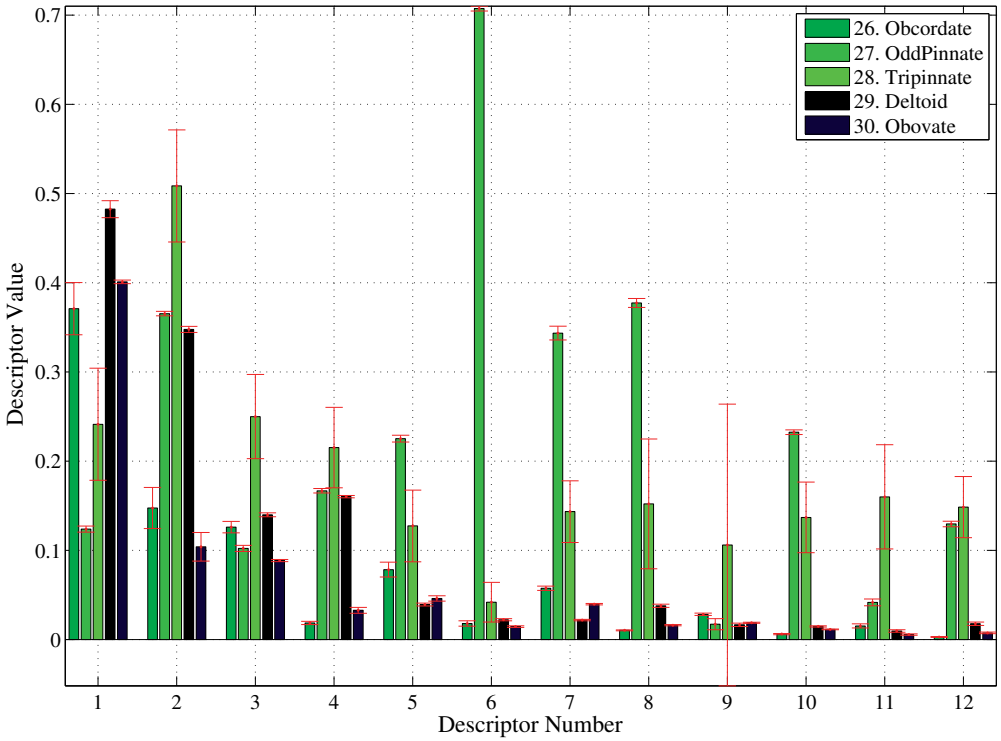


Figure A.7: Elliptic Fourier Descriptors for Shapes 26-30.

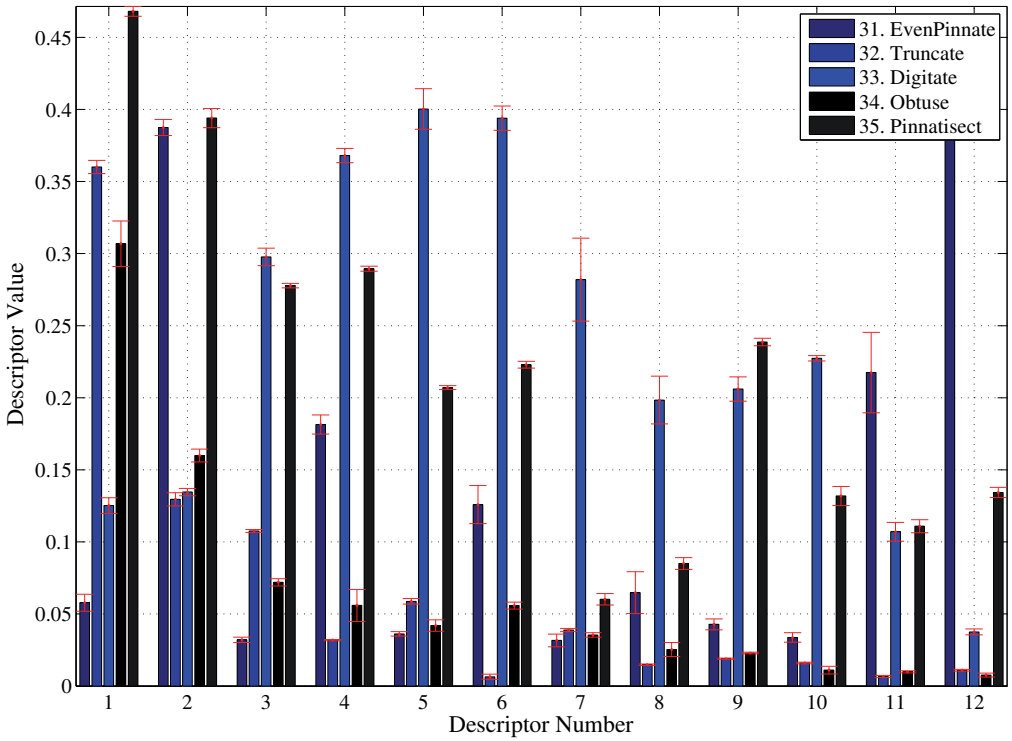


Figure A.8: Elliptic Fourier Descriptors for Shapes 31-35.

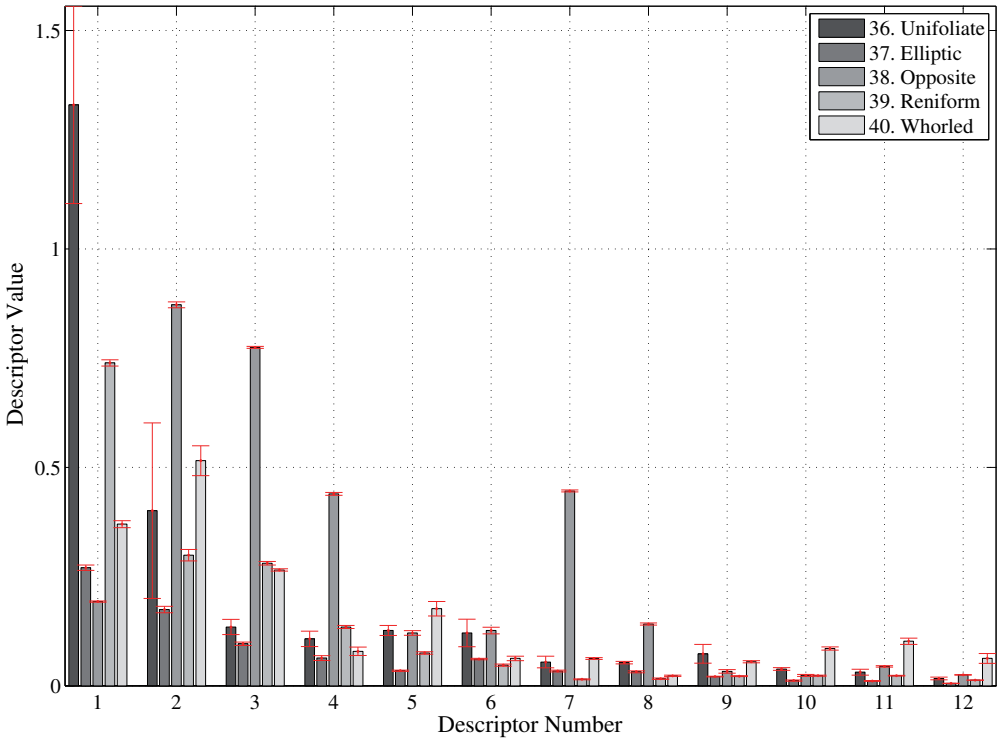


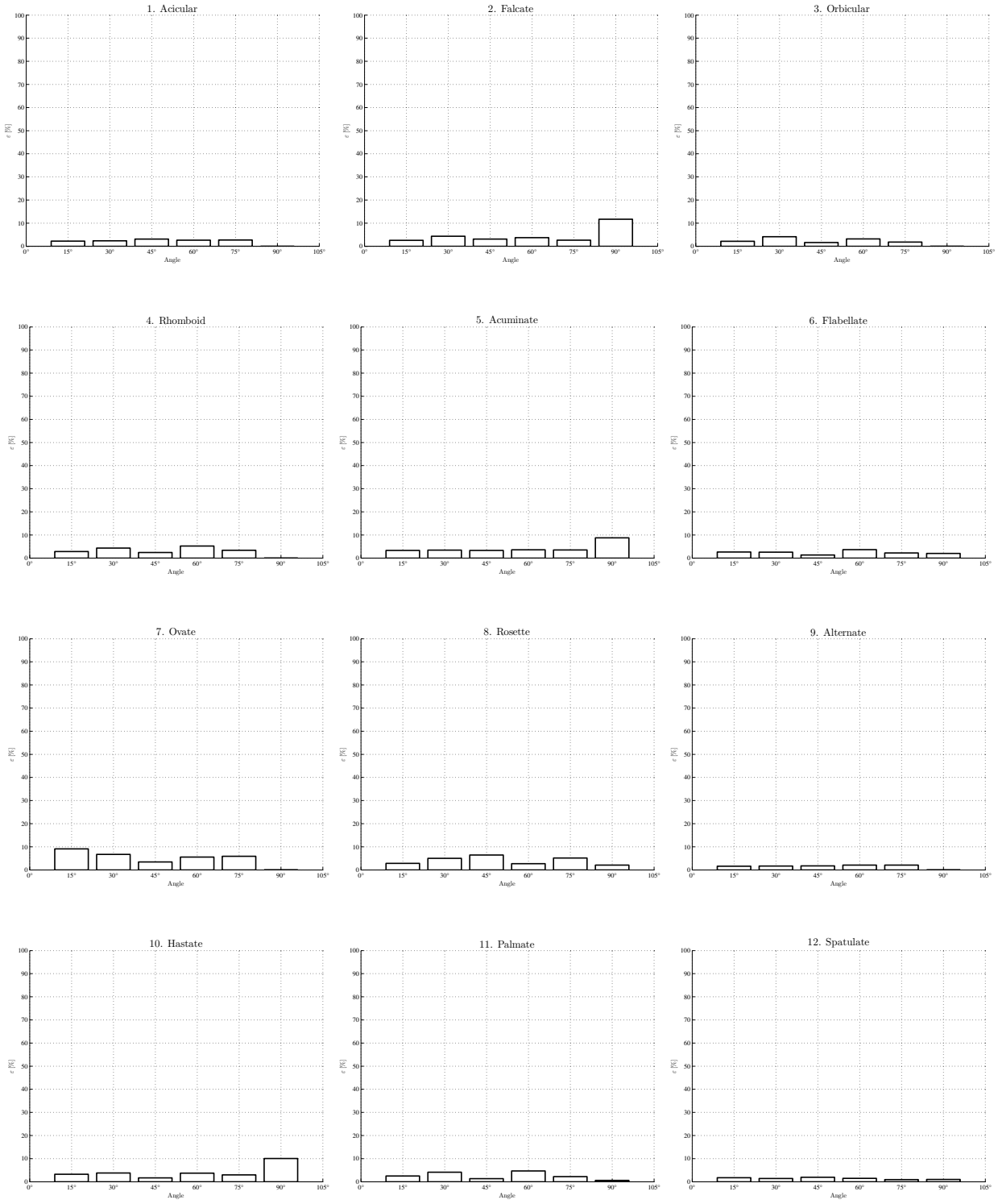
Figure A.9: Elliptic Fourier Descriptors for Shapes 36-40.

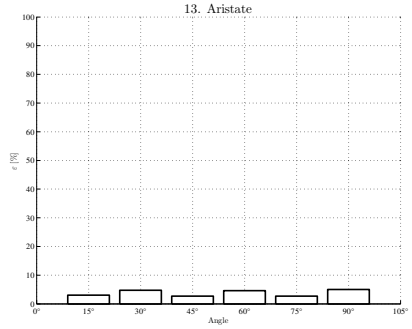
A.2.1 Differences Between Original Feature Vector and Vectors from Rotated Shapes

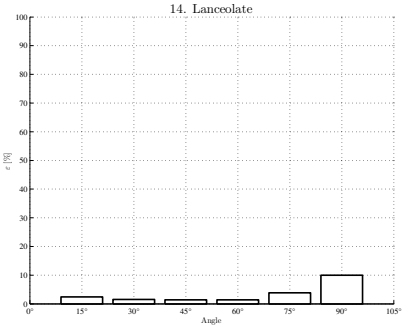
For the figures below the following applies:

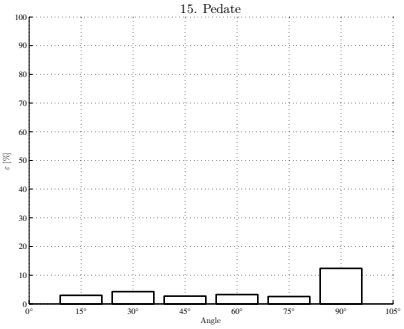
The abscissa: the rotation angle in fixed steps of 15° in the interval [0°, 90°].

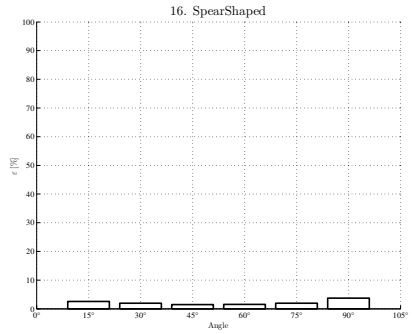
The ordinate: the change in the Euclidean distances as given by Eq. (A.1); the difference between the original vector and the vector computed from the rotated data.

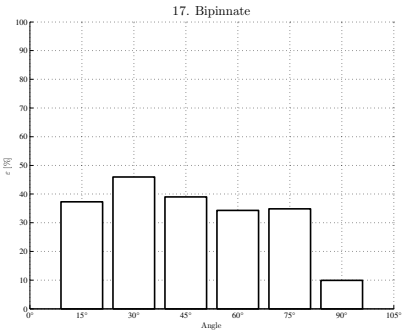


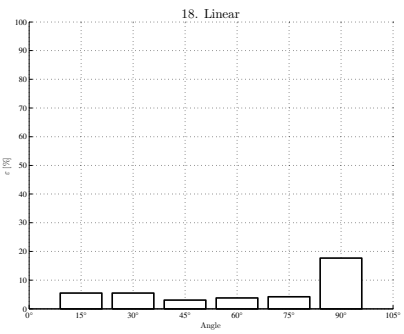

13. Aristate
ε [%]
Angle

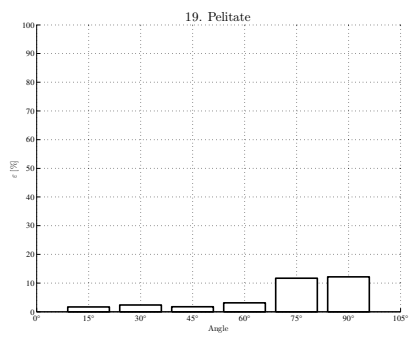

14. Lanceolate
ε [%]
Angle

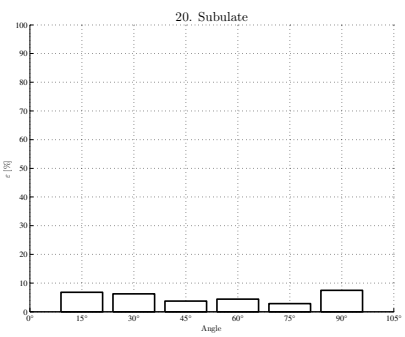

15. Pedate
ε [%]
Angle

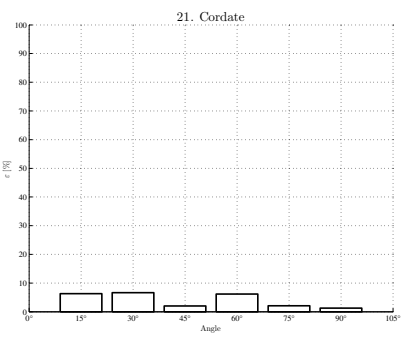

16. SpearShaped
ε [%]
Angle

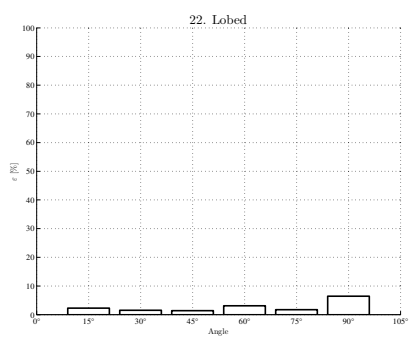

17. Bipinnate
ε [%]
Angle

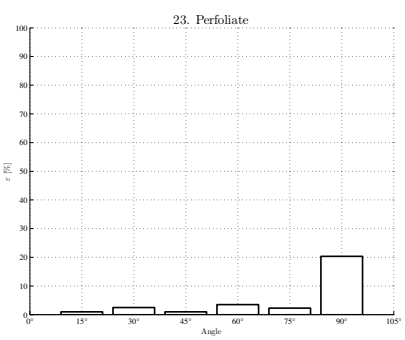

18. Linear
ε [%]
Angle

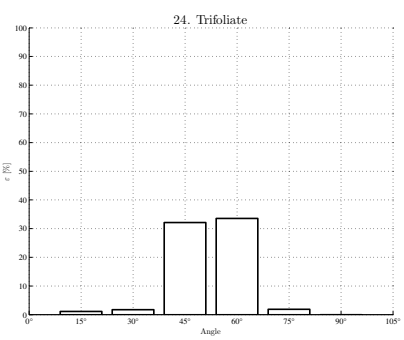

19. Pelitate
ε [%]
Angle

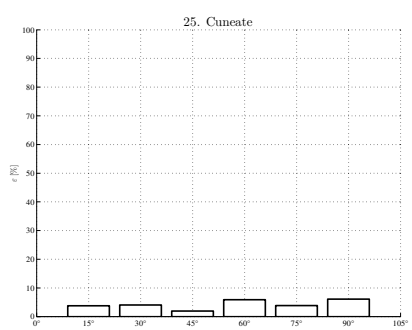

20. Subulate
ε [%]
Angle

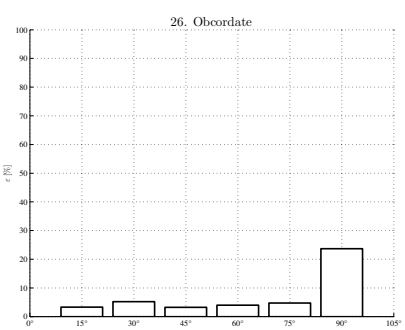

21. Cordate
ε [%]
Angle

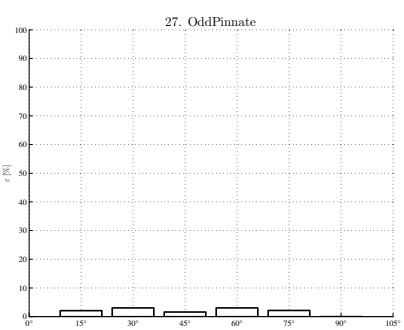

22. Lobed
ε [%]
Angle

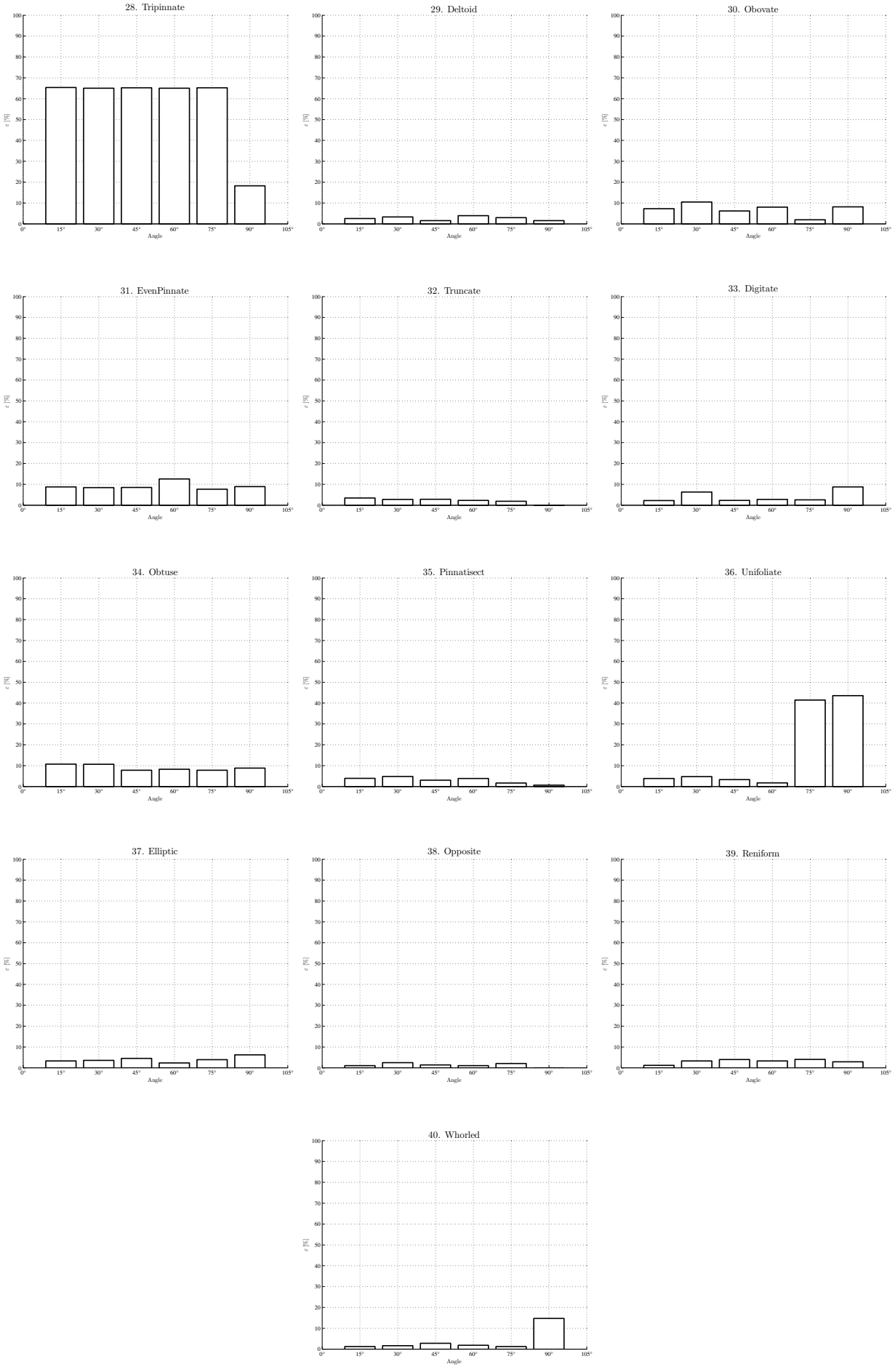

23. Perfoliate
ε [%]
Angle


24. Trifoliate
ε [%]
Angle


25. Cuneate
ε [%]
Angle


26. Obcordate
ε [%]
Angle


27. OddPinnate
ε [%]
Angle


28. Tripinnate
29. Deltoid
30. Obovate
31. EvenPinnate
32. Truncate
33. Digitate
34. Obtuse
35. Pinnatisect
36. Unifoliate
37. Elliptic
38. Opposite
39. Reniform
40. Whorled
ε [%]
Angle
0°
15°
30°
45°
60°
75°
90°
105°

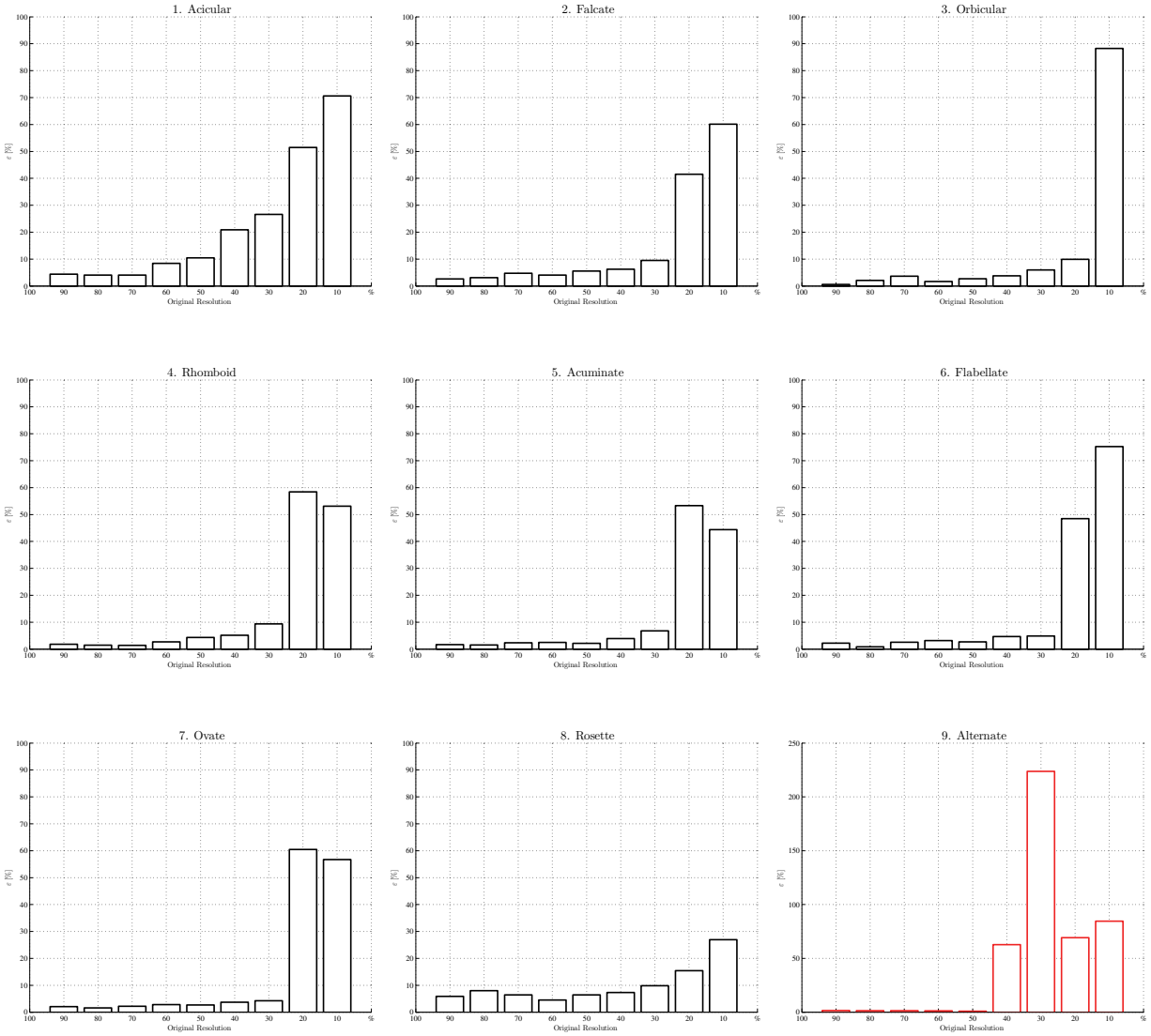
A.3 Effect of Decreasing Sensor Resolution

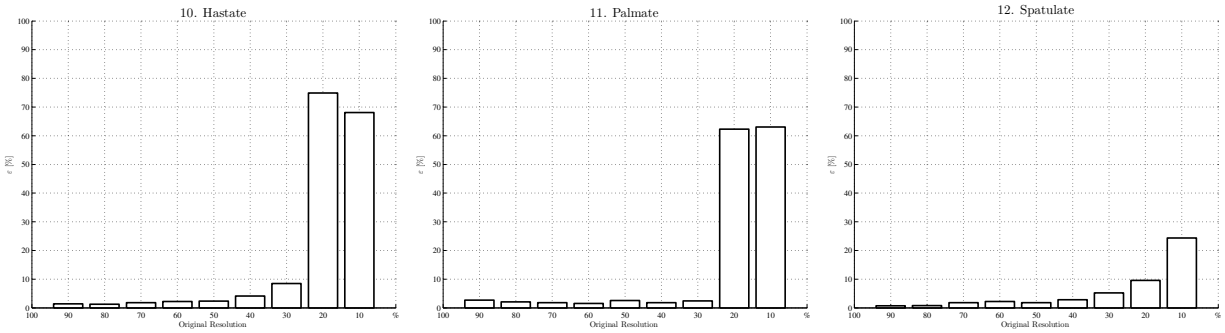
The computation results of the graphical representation gives a rough estimate of the required resolution. The aim of the analysis is to determine the minimal sensor resolution necessary to separate predefined shapes using k-Means clustering. When considering the figures below, the relationship between decreasing resolution and the corresponding change in the Euclidean distance is a significant and obvious common property; the lower resolution has a direct influence on the distance within the feature vector. For some shapes the computed change in Eucledian distance even exceeds 100% at lower resolutions. Shapes: 8, 9, 17, 27, 28, 31, 38 and 40 are representatives of complex leaf families; they have tiny connections between the leaflet and the petiole. These structures are the first details that disappear at a decreased sensor resolution and can change the shape of the object significantly. Sensor resolution can be considered the main influence on the classification results; a lower resolution causes an exponential increase of scatter by the shape descriptors - the figures in Section A.3.1 illustrate this.

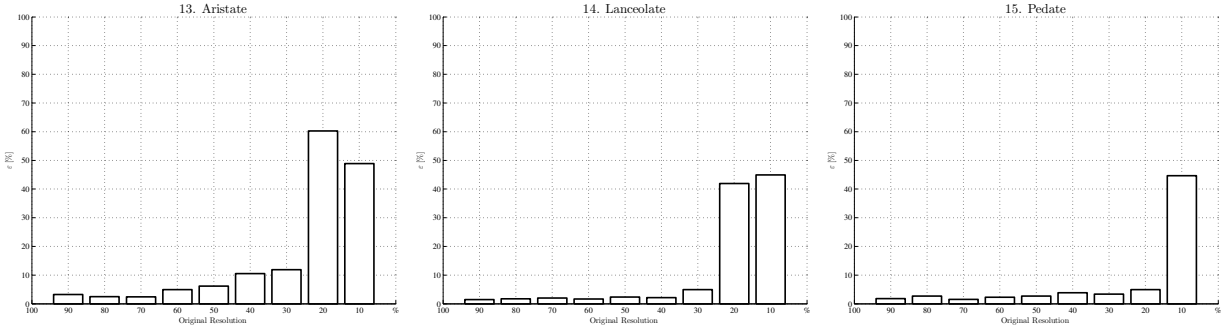
A.3.1 Differences Between the Original Feature Vector and Vectors Obtained from Lower Resolution Images

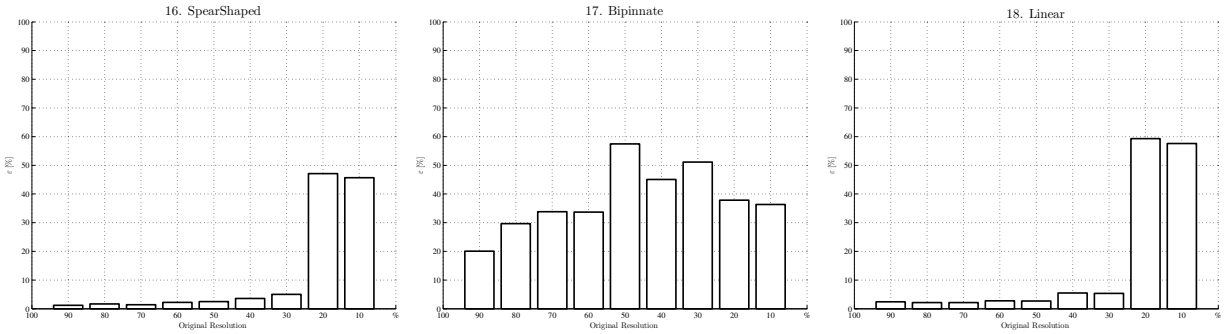
The Abscissa shows the decreasing resolution in percent. 100% denotes 600 dpi.

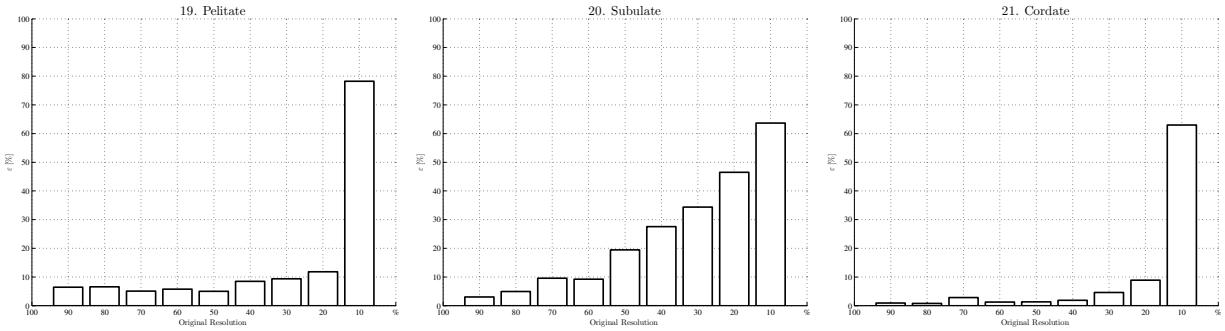
The Ordinate is the change in the Euclidean distance from Eq. (A.1); the difference between the original vector and the vector computed using lower resolution data.

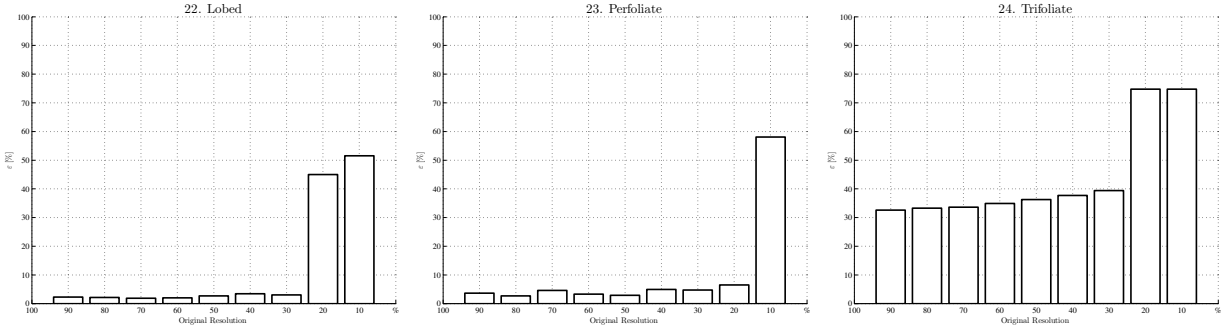


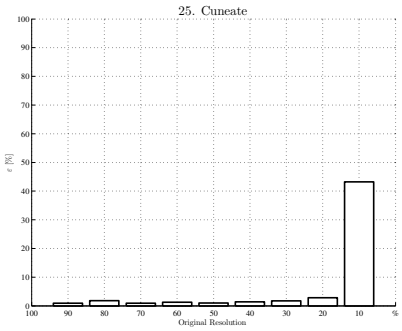

10. Hastate
11. Palmate
12. Spatulate
ε [%]
Original Resolution

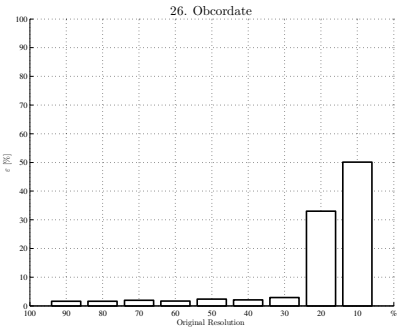

13. Aristate
14. Lanceolate
15. Pedate
ε [%]
Original Resolution

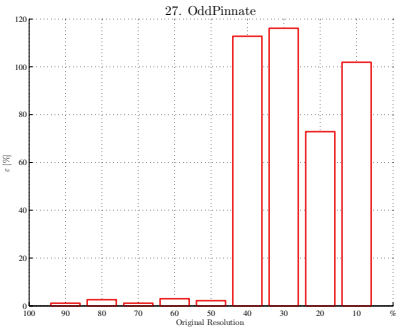

16. SpearShaped
17. Bipinnate
18. Linear
ε [%]
Original Resolution

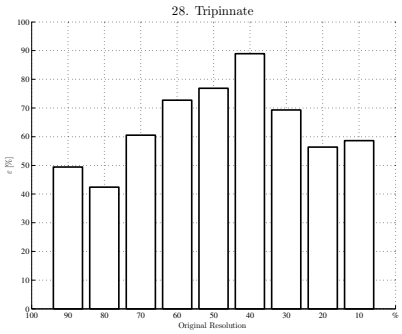

19. Pelitate
20. Subulate
21. Cordate
ε [%]
Original Resolution

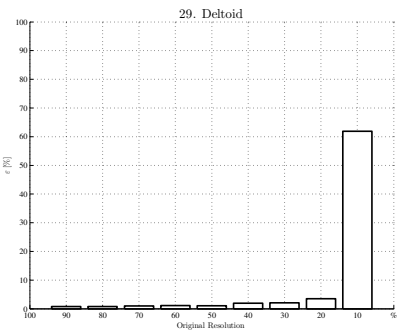

22. Lobed
23. Perfoliate
24. Trifoliate
ε [%]
Original Resolution

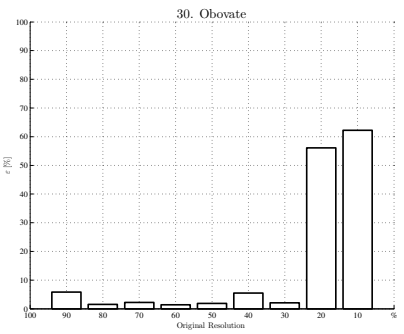

25. Cuneate
ε [%]
Original Resolution

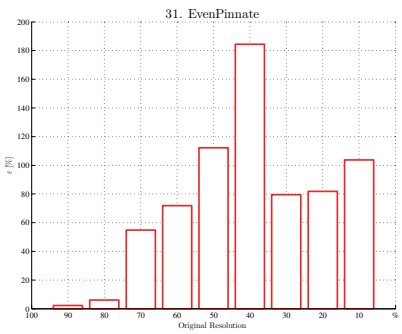

26. Obcordate
ε [%]
Original Resolution

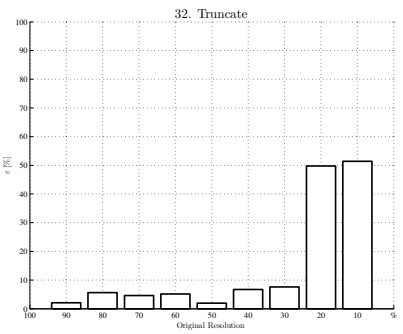

27. OddPinnate
ε [%]
Original Resolution

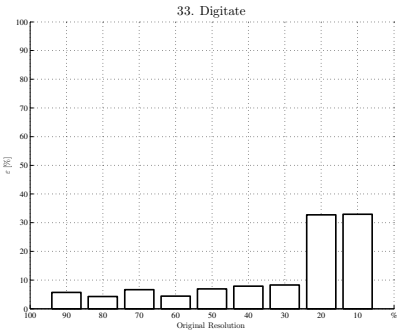

28. Tripinnate
ε [%]
Original Resolution

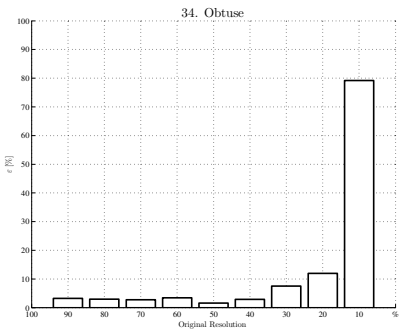

29. Deltoid
ε [%]
Original Resolution

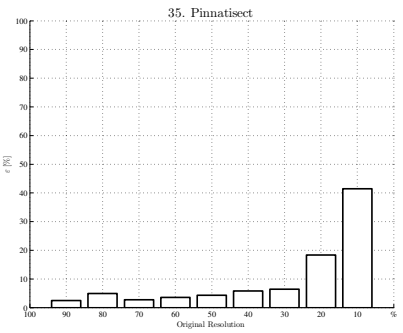

30. Obovate
ε [%]
Original Resolution

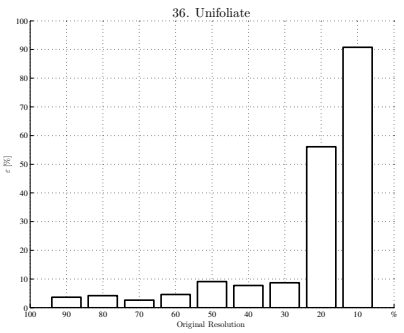

31. EvenPinnate
ε [%]
Original Resolution

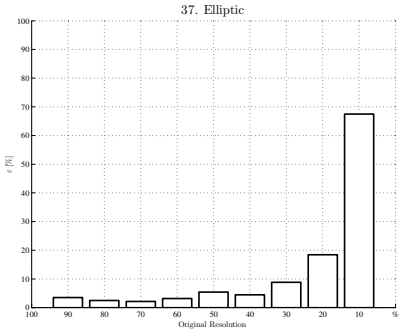

32. Truncate
ε [%]
Original Resolution

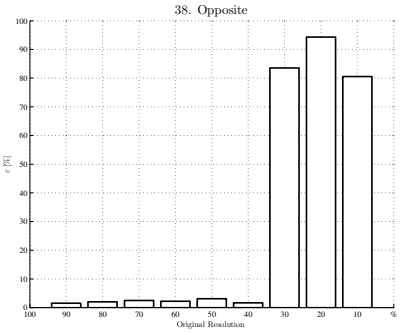

33. Digitate
ε [%]
Original Resolution

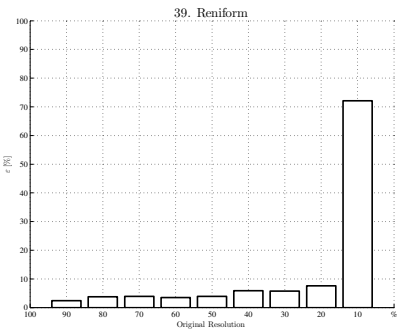

34. Obtuse
ε [%]
Original Resolution

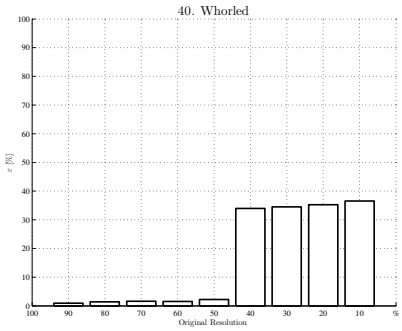

35. Pinnatisect
ε [%]
Original Resolution


36. Unifoliate
ε [%]
Original Resolution


37. Elliptic
ε [%]
Original Resolution


38. Opposite
ε [%]
Original Resolution


39. Reniform
ε [%]
Original Resolution



A.4 Shape Analysis Data, Facts and Tables

A.4.1 Separability Analysis and *k*-Means Clustering

Below are the results of the k-Means clustering computed using MATLAB. Table A.2 is the result of the "city block" distance between the feature vectors. Figure 8.5 illustrates the values as computed by the methods given in A.4.2 ro A.4.5, below the Table A.2

Table A.2: Scale Effect on k-Means Clustering.

Resolution	100	90	80	70	60	50	40	30	20	10	[%] [dpi]
Class	500	450	400	350	300	250	200	150	100	50	Centroid Shift [%]
1. Acicular	+	+	+	+	+	+	+	+	14	13	0.03
2. Falcate	+	+	+	+	+	+	+	+	7	7	0.06
3. Orbicular	+	+	+	+	+	+	+	+	+	14	0.04
4. Rhomboid	+	+	+	+	+	+	+	+	13	37	0.05
5. Acuminate	+	+	+	+	+	+	+	+	25	3	0.04
6. Flabellate	+	+	+	+	+	+	+	+	32	34	0.10
7. Ovate	+	+	+	+	+	+	+	+	18	18	0.26
8. Rosette	+	+	+	+	+	+	+	+	+	+	0.06
9. Alternate	+	+	+	+	+	+	10	38	23	18	0.07
10. Hastate	+	+	+	+	+	+	+	+	21	29	0.07
11. Palmate	+	+	+	+	+	+	+	+	10	10	0.06
12. Spatulate	+	+	+	+	+	+	+	+	+	25	0.04
13. Aristate	+	+	+	+	+	+	+	+	34	37	0.05
14. Lanceolate	+	+	+	+	+	+	+	+	18	+	0.01
15. Pedate	+	+	+	+	+	+	+	+	+	9	0.07
16. Spear Shaped	+	+	+	+	+	+	+	+	37	37	0.08
17. Bipinnate	+	+	+	+	+	16	23	35	40	24	0.48
18. Linear	+	+	+	+	+	+	+	+	14	13	0.03
19. Pelitate	+	+	+	+	+	+	+	+	+	18	0.06
20. Subulate	+	+	+	+	+	+	+	+	13	13	0.04
21. Cordate	+	+	+	+	+	+	+	+	+	37	0.08
22. Lobed	+	+	+	+	+	+	+	+	+	24	0.07
23. Perfoliate	+	+	+	+	+	+	+	+	+	25	0.11
24. Trifoliate	+	+	+	23	23	23	23	23	39	5	0.12
25. Cuneate	+	+	+	+	+	+	+	+	+	7	0.11
26. Obcordate	+	+	+	+	+	+	+	+	32	37	0.07
27. OddPinnate	+	+	+	+	+	+	35	23	1	6	0.09
28. Tripinnate	17	17	17	17	35	15	35	20	16	20	0.42
29. Deltoid	+	+	+	+	+	+	+	+	+	30	0.07
30. Obovate	+	+	+	+	+	+	+	+	34	18	0.06
31. Even Pinnate	+	+	+	+	40	15	23	13	13	19	0.10
32. Truncate	+	+	+	+	+	+	+	+	+	+	0.02
33. Digitate	+	+	+	+	+	+	+	+	15	22	0.09
34. Obtuse	+	+	+	+	+	+	+	+	+	14	0.05

Resolution	100	90	80	70	60	50	40	30	20	10	[%] [dpi]
Class	500	450	400	350	300	250	200	150	100	50	Centroid Shift [%]
35. Pinnatisect	+	+	+	+	+	+	+	+	+	11	0.09
36. Unifoliate	+	+	+	+	+	+	+	+	19	18	0.15
37. Elliptic	+	+	+	+	+	+	+	+	+	18	0.10
38. Opposite	+	+	+	+	+	+	+	6	18	3	0.07
39. Reniform	+	+	+	+	+	+	+	+	+	37	0.06
40. Whorled	+	+	+	+	+	+	+	+	10	5	0.05
Errors	1	1	1	2	3	4	6	7	25	37	

A.4.2 Euclidean Method

$\|x^j - \overline{x}\| \rightarrow \min$ Euclidean distance between the centroid and the feature vector.

Shape	100	90	80	70	60	50	40	30	20	10	[%]
1. Acicular	+	+	+	+	+	+	+	+	14	13	0.14
2. Falcate	+	+	+	+	+	+	+	+	7	7	0.08
3. Orbicular	+	+	+	+	+	+	+	+	+	18	0.10
4. Rhomboid	+	+	+	+	+	+	+	+	13	37	0.10
5. Acuminate	+	+	+	+	+	+	+	+	25	3	0.07
6. Flabellate	39	39	39	39	39	39	39	39	26	37	0.50
7. Ovate	+	+	+	+	+	+	+	+	18	18	0.09
8. Rosette	+	+	+	+	+	+	+	+	+	+	0.07
9. Alternate	+	+	+	+	+	+	40	6	23	18	4.00
10. Hastate	+	+	+	+	+	+	+	+	21	29	0.12
11. Palmate	+	+	+	+	+	+	+	+	10	10	0.17
12. Spatulate	+	+	+	+	+	+	+	+	+	4	0.08
13. Aristate	+	+	+	+	+	+	+	+	34	1	0.09
14. Lanceolate	+	+	+	+	+	+	+	+	18	1	0.06
15. Pedate	+	+	+	+	+	+	+	+	+	9	0.18
16. SpearShaped	+	+	+	+	+	+	+	+	37	37	0.08
17. Bipinnate	+	+	+	+	+	16	23	35	+	40	0.75
18. Linear	+	+	+	+	+	+	+	+	14	13	0.07
19. Pelitate	+	+	+	+	+	+	+	+	+	18	0.14
20. Subulate	+	+	+	+	+	+	+	1	1	13	0.15
21. Cordate	+	+	+	+	+	+	+	+	+	37	0.09
22. Lobed	+	+	+	+	+	+	+	+	24	24	0.20
23. Perfoliate	+	+	+	+	+	+	+	+	+	25	0.15
24. Trifoliate	+	+	+	+	+	+	23	23	39	5	1.30
25. Cuneate	+	+	+	+	+	+	+	+	+	37	0.06
26. Obcordate	+	+	+	+	+	+	+	+	37	18	0.09
27. OddPinnate	+	+	+	+	+	+	23	23	13	12	0.62
28. Tripinnate	17	17	17	17	15	15	6	26	5	10	1.17
29. Deltoid	+	+	+	+	+	+	+	+	+	37	0.07
30. Obovate	+	+	+	+	+	+	32	+	1	18	0.12
31. EvenPinnate	+	+	+	+	28	9	6	13	13	19	1.23
32. Truncate	+	+	+	+	+	+	+	+	37	18	0.12
33. Digitate	+	+	+	+	+	+	+	+	15	22	0.25
34. Obtuse	+	+	+	+	+	+	+	+	+	18	0.08
35. Pinnatisect	+	+	+	+	+	+	+	+	+	11	0.19
36. Unifoliate	+	+	+	+	+	+	+	+	19	18	0.21
37. Elliptic	+	+	+	+	+	+	+	+	+	14	0.08
38. Opposite	+	+	+	+	+	+	+	6	18	4	0.22
39. Reniform	+	+	+	+	+	+	+	+	+	37	0.14
40. Whorled	+	+	+	+	+	+	10	10	10	5	0.26

A.4.3 City Block Method

$\sqrt{\|x^j\|} \rightarrow \min$ "City block" : length of the feature vector.

Shape	100	90	80	70	60	50	40	30	20	10	[%]
1. Acicular	+	+	+	+	+	+	+	+	14	13	0.03
2. Falcate	+	+	+	+	+	+	+	+	7	7	0.06
3. Orbicular	+	+	+	+	+	+	+	+	+	14	0.04
4. Rhomboid	+	+	+	+	+	+	+	+	13	37	0.05
5. Acuminate	+	+	+	+	+	+	+	+	25	3	0.04
6. Flabellate	+	+	+	+	+	+	+	+	32	34	0.10
7. Ovate	+	+	+	+	+	+	+	+	18	18	0.26
8. Rosette	+	+	+	+	+	+	+	+	+	+	0.06
9. Alternate	+	+	+	+	+	+	10	38	23	18	0.07
10. Hastate	+	+	+	+	+	+	+	+	21	29	0.07
11. Palmate	+	+	+	+	+	+	+	+	10	10	0.06
12. Spatulate	+	+	+	+	+	+	+	+	+	25	0.04
13. Aristate	+	+	+	+	+	+	+	+	34	37	0.05
14. Lanceolate	+	+	+	+	+	+	+	+	18	+	0.01
15. Pedate	+	+	+	+	+	+	+	+	+	9	0.07
16. SpearShaped	+	+	+	+	+	+	+	+	37	37	0.08
17. Bipinnate	+	+	+	+	+	16	23	35	40	24	0.48
18. Linear	+	+	+	+	+	+	+	+	14	13	0.03
19. Pelitate	+	+	+	+	+	+	+	+	+	18	0.06
20. Subulate	+	+	+	+	+	+	+	+	13	13	0.04
21. Cordate	+	+	+	+	+	+	+	+	+	37	0.08
22. Lobed	+	+	+	+	+	+	+	+	+	24	0.07
23. Perfoliate	+	+	+	+	+	+	+	+	+	25	0.11
24. Trifoliate	+	+	+	23	23	23	23	23	39	5	0.12
25. Cuneate	+	+	+	+	+	+	+	+	+	7	0.11
26. Obcordate	+	+	+	+	+	+	+	+	32	37	0.07
27. OddPinnate	+	+	+	+	+	+	35	23	1	6	0.09
28. Tripinnate	17	17	17	17	35	15	35	20	16	20	0.42
29. Deltoid	+	+	+	+	+	+	+	+	+	30	0.07
30. Obovate	+	+	+	+	+	+	+	+	34	18	0.06
31. EvenPinnate	+	+	+	+	40	15	23	13	13	19	0.10
32. Truncate	+	+	+	+	+	+	+	+	+	+	0.02
33. Digitate	+	+	+	+	+	+	+	+	15	22	0.09
34. Obtuse	+	+	+	+	+	+	+	+	+	14	0.05
35. Pinnatisect	+	+	+	+	+	+	+	+	+	11	0.09
36. Unifoliate	+	+	+	+	+	+	+	+	19	18	0.15
37. Elliptic	+	+	+	+	+	+	+	+	+	18	0.10
38. Opposite	+	+	+	+	+	+	+	6	18	3	0.07
39. Reniform	+	+	+	+	+	+	+	+	+	37	0.06
40. Whorled	+	+	+	+	+	+	+	+	10	5	0.05

A.4.4 Cosine Method

$1 - \frac{x^t y}{\|x\|\|y\|} \rightarrow \min$ Cosine, angle between the feature vectors.

Shape	100	90	80	70	60	50	40	30	20	10	[%]
1. Acicular	+	+	+	+	+	+	+	+	14	14	0.30
2. Falcate	+	+	+	+	+	+	+	+	7	4	0.11
3. Orbicular	+	+	+	+	+	+	+	+	+	17	0.10
4. Rhomboid	+	+	+	+	+	+	+	+	13	7	0.13
5. Acuminate	+	+	+	+	+	+	+	+	25	25	0.09
6. Flabellate	+	+	+	+	+	+	+	+	16	30	0.13
7. Ovate	+	+	+	+	+	+	+	+	13	13	0.16
8. Rosette	+	+	+	+	+	+	+	+	+	+	0.18

9.	Alternate	+	+	+	+	+	+	18	5	5	23	0.21
10.	Hastate	+	+	+	+	+	+	+	+	7	7	0.15
11.	Palmate	+	+	+	+	+	+	+	+	22	10	0.22
12.	Spatulate	+	+	+	+	+	+	+	+	+	+	0.07
13.	Aristate	+	+	+	+	+	+	+	+	34	29	0.12
14.	Lanceolate	+	+	+	+	+	+	+	+	13	13	0.11
15.	Pedate	+	+	+	+	+	+	+	+	+	11	0.12
16.	SpearShaped	+	+	+	+	+	+	+	+	37	+	0.13
17.	Bipinnate	+	+	+	18	2	35	7	11	18	2	0.56
18.	Linear	+	+	+	+	+	+	+	+	14	13	0.16
19.	Pelitate	+	+	+	+	+	+	+	+	+	39	0.10
20.	Subulate	+	+	+	+	+	+	+	1	13	13	0.28
21.	Cordate	+	+	+	+	+	+	+	+	+	39	0.09
22.	Lobed	+	+	+	+	+	+	+	+	24	24	0.16
23.	Perfoliate	+	+	+	+	+	+	+	+	+	3	0.08
24.	Trifoliate	+	+	+	+	+	+	+	23	39	5	0.46
25.	Cuneate	+	+	+	+	+	+	+	+	+	4	0.05
26.	Obcordate	+	+	+	+	+	+	+	+	+	5	0.13
27.	OddPinnate	+	+	+	+	+	+	18	2	14	19	0.22
28.	Tripinnate	17	17	17	17	15	15	21	37	2	35	0.72
29.	Deltoid	+	+	+	+	+	+	+	+	+	32	0.06
30.	Obovate	+	+	+	+	+	+	32	+	7	5	0.17
31.	EvenPinnate	+	+	+	+	28	22	16	14	14	39	0.59
32.	Truncate	+	+	+	+	+	+	+	+	37	37	0.17
33.	Digitate	+	+	+	+	+	+	+	+	15	15	0.19
34.	Obtuse	+	+	+	+	+	+	+	+	+	13	0.13
35.	Pinnatisect	+	+	+	+	+	+	+	+	+	18	0.16
36.	Unifoliate	+	+	+	+	+	+	+	+	6	2	0.14
37.	Elliptic	+	+	+	+	+	+	+	+	+	13	0.13
38.	Opposite	+	+	+	+	+	+	+	12	35	12	0.14
39.	Reniform	+	+	+	+	+	+	+	+	+	29	0.08
40.	Whorled	+	+	+	+	+	+	18	18	18	+	0.22

A.4.5 Correlation Method

$1 - \frac{C_{ij}}{\sqrt{C_{ii}C_{jj}}} \rightarrow \min$ Correlation between the feature vectors.

Shape	100	90	80	70	60	50	40	30	20	10	[%]
1. Acicular	+	+	+	+	+	+	+	14	14	14	0.34
2. Falcate	+	+	+	+	+	+	+	+	7	4	0.16
3. Orbicular	+	+	+	+	+	+	+	+	+	13	0.11
4. Rhomboid	+	+	+	+	+	+	+	+	13	7	0.13
5. Acuminate	+	+	+	+	+	+	+	+	25	25	0.10
6. Flabellate	+	+	+	+	+	+	+	+	16	30	0.13
7. Ovate	+	+	+	+	+	+	+	+	13	13	0.35
8. Rosette	+	+	+	+	+	+	+	+	+	+	0.20
9. Alternate	+	+	+	+	+	+	18	5	5	23	0.19
10. Hastate	+	+	+	+	+	+	+	+	7	7	0.16
11. Palmate	+	+	+	+	+	+	+	+	22	10	0.23
12. Spatulate	+	+	+	+	+	+	+	+	+	+	0.07
13. Aristate	+	+	+	+	+	+	+	+	34	29	0.15
14. Lanceolate	+	+	+	+	+	+	+	+	13	13	0.12
15. Pedate	+	+	+	+	+	+	+	+	+	11	0.14
16. SpearShaped	+	+	+	+	+	+	+	+	37	+	0.15
17. Bipinnate	2	13	18	13	2	20	7	11	18	2	0.62
18. Linear	+	+	+	+	+	+	+	+	14	13	0.20
19. Pelitate	+	+	+	+	+	+	+	+	+	39	0.09
20. Subulate	+	+	+	+	+	+	23	2	13	13	0.28
21. Cordate	+	+	+	+	+	+	+	+	+	39	0.09
22. Lobed	+	+	+	+	+	+	+	+	24	24	0.17

23.	Perfoliate	+	+	+	+	+	+	+	+	+	3	0.08
24.	Trifoliate	+	+	+	+	+	+	+	23	39	5	1.00
25.	Cuneate	+	+	+	+	+	+	+	+	+	4	0.06
26.	Obcordate	+	+	+	+	+	+	+	+	+	5	0.14
27.	OddPinnate	+	+	+	+	+	+	17	2	14	19	0.58
28.	Tripinnate	35	17	17	35	15	15	21	39	2	10	1.36
29.	Deltoid	+	+	+	+	+	+	+	+	+	32	0.06
30.	Obovate	+	+	+	+	+	+	32	+	7	5	0.18
31.	EvenPinnate	+	+	+	+	+	22	16	14	14	39	0.61
32.	Truncate	+	+	+	+	+	+	+	+	37	37	0.19
33.	Digitate	+	+	+	+	+	+	+	+	15	15	0.21
34.	Obtuse	+	+	+	+	+	+	+	+	+	13	0.22
35.	Pinnatisect	+	+	+	+	+	+	+	+	+	13	0.53
36.	Unifoliate	+	+	+	+	+	+	+	+	6	2	0.13
37.	Elliptic	+	+	+	+	+	+	+	+	7	13	0.21
38.	Opposite	+	+	+	+	+	+	+	12	16	12	0.14
39.	Reniform	+	+	+	+	+	+	+	+	+	29	0.09
40.	Whorled	+	+	+	+	+	+	18	18	18	+	0.21

A.5 Synthetic Data Images

(a) **(b)**

Figure A.10: (a) Training Set and (b)Test File.

The following table shows the complete classification results of the synthetic test elaborated in Section 12.3.

Label	Machine	Human	Probability	X	Y	Z
1	1	1	97.09	58.0	50.0	180.0
599	1	1	96.96	628.0	734.0	119.0
3	1	1	97.38	400.0	50.0	93.0
381	1	1	96.25	1510.0	479.0	134.0
772	1	1	97.13	1426.0	962.0	134.0
6	1	1	97.47	742.0	50.0	250.0
7	1	1	97.33	856.0	50.0	373.0
8	1	1	97.44	970.0	50.0	424.0
771	1	1	96.90	1312.0	962.0	122.0
770	1	1	96.08	1084.0	962.0	138.0
442	1	1	96.99	58.0	620.0	207.0
1099	1	1	97.69	742.0	1304.0	80.0

Label	Machine	Human	Probability	X	Y	Z
1098	1	1	97.01	514.0	1304.0	81.0
443	1	1	96.95	172.0	620.0	315.0
600	1	1	97.11	742.0	734.0	633.0
605	1	1	97.21	1426.0	734.0	389.0
73	1	1	96.27	286.0	164.0	325.0
650	1	1	97.06	400.0	848.0	90.0
447	1	1	97.25	1312.0	620.0	122.0
448	1	1	95.45	1510.0	593.0	598.0
769	1	1	96.52	970.0	962.0	629.0
78	1	1	97.29	970.0	164.0	634.0
767	1	1	97.07	628.0	962.0	580.0
80	1	1	97.30	1198.0	164.0	354.0
81	1	1	96.98	1312.0	164.0	549.0
766	1	1	96.64	400.0	962.0	550.0
83	1	1	93.48	1510.0	137.0	550.0
1097	1	1	97.08	400.0	1304.0	34.0
765	1	1	96.98	286.0	962.0	72.0
108	1	1	96.15	286.0	278.0	490.0
109	1	1	96.97	628.0	278.0	622.0
110	1	1	97.56	742.0	278.0	53.0
654	1	1	97.21	1312.0	848.0	567.0
112	1	1	95.98	1198.0	278.0	481.0
652	1	1	96.93	628.0	848.0	586.0
1005	1	1	97.46	1426.0	1190.0	236.0
1004	1	1	96.89	1084.0	1190.0	225.0
1003	1	1	97.03	970.0	1190.0	406.0
1002	1	1	96.95	856.0	1190.0	363.0
! 1001	*1*	*1*	*48.84*	*742.0*	*1172.0*	*27.0*
1000	1	1	96.85	514.0	1190.0	70.0
651	1	1	96.90	514.0	848.0	586.0
235	1	1	96.94	58.0	392.0	416.0
236	1	1	97.05	172.0	392.0	371.0
237	1	1	96.99	286.0	392.0	167.0
603	1	1	97.05	1198.0	734.0	602.0
239	1	1	96.19	970.0	392.0	76.0
240	1	1	97.09	1084.0	392.0	277.0
241	1	1	97.32	1312.0	392.0	81.0
242	1	1	93.06	1510.0	365.0	244.0
999	1	1	96.92	400.0	1190.0	382.0
998	1	1	97.19	286.0	1190.0	554.0
996	1	1	97.00	58.0	1190.0	498.0
860	1	1	97.03	1312.0	1076.0	237.0
857	1	1	97.00	742.0	1076.0	646.0
855	1	1	97.02	286.0	1076.0	368.0
596	1	1	97.51	286.0	734.0	546.0
649	1	1	97.71	286.0	848.0	229.0
373	1	1	96.72	400.0	506.0	177.0
374	1	1	96.88	514.0	506.0	343.0
375	1	1	96.48	628.0	506.0	252.0
376	1	1	97.02	742.0	506.0	546.0
598	1	1	97.23	514.0	734.0	555.0
378	1	1	97.22	1084.0	506.0	367.0
379	1	1	97.05	1198.0	506.0	162.0
1100	1	1	97.31	970.0	1304.0	235.0
1102	1	1	96.99	1198.0	1304.0	174.0
1189	1	1	97.11	58.0	1418.0	129.0
1334	1	1	95.16	370.0	1505.0	544.0
1338	1	1	94.22	1282.0	1505.0	358.0
11	2	2	98.34	1426.0	58.0	575.0
*** 20**	**2**	**3**	**65.36**	**1494.0**	**22.0**	**397.0**
71	2	2	98.32	58.0	172.0	660.0

Label	Machine	Human	Probability	X	Y	Z
72	2	2	97.96	172.0	154.0	28.0
74	2	2	98.24	514.0	172.0	207.0
75	2	2	98.19	628.0	172.0	427.0
76	2	2	98.29	742.0	172.0	300.0
77	2	2	98.32	856.0	172.0	254.0
79	2	2	98.31	1084.0	172.0	26.0
82	2	2	98.22	1426.0	172.0	372.0
107	2	2	98.51	58.0	283.0	52.0
111	2	2	98.21	856.0	286.0	122.0
113	2	2	98.27	1426.0	286.0	356.0
!* 123	**2**	**3**	**49.09**	**1498.0**	**245.0**	**101.0**
238	2	2	98.27	742.0	400.0	529.0
371	2	2	98.33	58.0	514.0	113.0
372	2	2	98.31	286.0	514.0	494.0
377	2	2	98.28	970.0	514.0	524.0
380	2	2	97.97	1312.0	514.0	293.0
444	2	2	98.32	286.0	628.0	229.0
445	2	2	98.33	1084.0	628.0	342.0
446	2	2	97.87	1198.0	628.0	520.0
595	2	2	98.23	172.0	742.0	323.0
597	2	2	98.33	400.0	742.0	86.0
601	2	2	98.20	856.0	742.0	141.0
602	2	2	98.35	970.0	742.0	193.0
604	2	2	98.31	1312.0	742.0	330.0
*** 613**	**2**	**3**	**77.66**	**1496.0**	**698.0**	**635.0**
647	2	2	98.28	58.0	856.0	638.0
648	2	2	98.34	172.0	856.0	345.0
653	2	2	98.36	1198.0	856.0	342.0
655	2	2	97.58	1510.0	826.0	527.0
764	2	2	98.26	58.0	970.0	126.0
768	2	2	98.31	856.0	970.0	603.0
773	2	2	98.33	1510.0	940.0	660.0
856	2	2	98.31	628.0	1084.0	317.0
858	2	2	98.23	970.0	1084.0	367.0
859	2	2	98.33	1084.0	1084.0	436.0
861	2	2	98.28	1426.0	1084.0	471.0
997	2	2	98.23	172.0	1198.0	79.0
*** 1010**	**2**	**3**	**74.97**	**1501.0**	**1153.0**	**31.0**
1095	2	2	98.26	172.0	1312.0	618.0
1096	2	2	98.33	286.0	1312.0	226.0
2	2	2	98.30	286.0	58.0	108.0
1101	2	2	98.34	1084.0	1312.0	483.0
4	2	2	98.34	514.0	58.0	131.0
1103	2	2	98.34	1312.0	1312.0	234.0
1104	2	2	97.86	1510.0	1282.0	648.0
5	2	2	98.25	628.0	58.0	92.0
1190	2	2	98.31	172.0	1426.0	462.0
1191	2	2	98.36	514.0	1426.0	492.0
1192	2	2	98.33	628.0	1426.0	126.0
1193	2	2	98.34	742.0	1426.0	73.0
1194	2	2	98.31	1198.0	1426.0	29.0
1195	2	2	98.32	1312.0	1426.0	527.0
1333	2	2	98.05	256.0	1510.0	307.0
9	2	2	97.99	1084.0	39.0	22.0
1335	2	2	97.95	484.0	1510.0	544.0
1336	2	2	98.24	826.0	1510.0	88.0
1337	2	2	97.87	1168.0	1510.0	235.0
10	2	2	98.35	1312.0	58.0	397.0
1339	2	2	97.75	1396.0	1510.0	152.0
1340	2	2	98.07	1510.0	1510.0	251.0
*** 1348**	**2**	**3**	**67.20**	**934.0**	**1496.0**	**534.0**

Label	Machine	Human	Probability	X	Y	Z
863	3	3	99.30	154.0	1070.0	419.0
864	3	3	99.18	373.0	1052.0	577.0
865	3	3	98.83	493.0	1055.0	160.0
867	3	3	98.93	853.0	1059.0	500.0
870	3	3	99.39	1192.0	1084.0	190.0
872	3	3	94.08	1499.0	1050.0	65.0
245	3	3	98.46	619.0	373.0	158.0
660	3	3	91.08	936.0	845.0	252.0
244	3	3	99.42	511.0	391.0	299.0
243	3	3	99.30	391.0	380.0	433.0
121	3	3	99.28	1304.0	275.0	302.0
120	3	3	96.57	1053.0	274.0	149.0
119	3	3	87.33	933.0	266.0	316.0
117	3	3	97.97	507.0	258.0	135.0
116	3	3	90.33	366.0	255.0	327.0
115	3	3	98.52	170.0	261.0	99.0
1007	3	3	99.53	626.0	1198.0	302.0
1008	3	3	99.34	1194.0	1189.0	114.0
1009	3	3	98.75	1284.0	1189.0	590.0
661	3	3	99.09	1056.0	842.0	244.0
663	3	3	99.06	1414.0	835.0	535.0
458	3	3	98.96	1396.0	612.0	306.0
86	3	3	92.22	385.0	136.0	533.0
17	3	3	99.01	1179.0	43.0	494.0
12	3	3	99.33	168.0	40.0	398.0
453	3	3	98.71	710.0	593.0	451.0
452	3	3	99.17	626.0	607.0	498.0
451	3	3	98.77	478.0	605.0	541.0
455	3	3	99.51	965.0	629.0	431.0
450	3	3	99.19	381.0	624.0	493.0
1105	3	3	99.38	49.0	1292.0	642.0
1108	3	3	99.40	620.0	1305.0	560.0
1109	3	3	72.90	843.0	1277.0	520.0
1112	3	3	96.29	1394.0	1286.0	101.0
388	3	3	99.45	1410.0	506.0	406.0
385	3	3	94.24	854.0	483.0	596.0
383	3	3	88.73	151.0	478.0	85.0
611	3	3	89.60	1074.0	706.0	640.0
775	3	3	94.73	171.0	940.0	76.0
776	3	3	99.10	510.0	953.0	529.0
777	3	3	98.95	719.0	939.0	135.0
1197	3	3	99.08	283.0	1401.0	637.0
! 1198	*3*	*3*	*43.58*	*364.0*	*1405.0*	*27.0*
1202	3	3	97.45	825.0	1404.0	583.0
1203	3	3	99.35	957.0	1419.0	230.0
1204	3	3	99.42	1084.0	1409.0	558.0
! 1207	*3*	*3*	*38.88*	*1403.0*	*1393.0*	*16.0*
1208	3	3	83.88	1502.0	1389.0	151.0
779	3	3	83.98	1192.0	936.0	293.0
249	3	3	98.37	1411.0	368.0	523.0
454	3	3	99.36	843.0	612.0	325.0
248	3	3	99.18	1169.0	377.0	48.0
606	3	3	93.60	47.0	706.0	308.0
658	3	3	99.49	730.0	846.0	55.0
247	3	3	99.18	828.0	386.0	438.0
659	3	3	99.32	848.0	835.0	36.0
1342	3	3	96.90	133.0	1507.0	395.0
! 1345	*3*	*3*	*48.71*	*588.0*	*1497.0*	*24.0*
1346	3	3	94.60	706.0	1504.0	285.0
862	3	3	97.43	26.0	1077.0	424.0
!* 1341	**4**	**3**	**52.76**	**24.0**	**1493.0**	**567.0**

Label	Machine	Human	Probability	X	Y	Z
!* 1349	**4**	**3**	**36.56**	**1041.0**	**1496.0**	**380.0**

Legend:	
1111	Correct classification.
! 1111	Classification probability below threshold: $p < 65\%$
*** 1111**	**Classification error.**
! italic	Emphasizes correct classification with low probability.

Appendix B

Algorithms And Sources

B.1 Low Level Feature Extraction Algorithm

Algorithm Algorithm B.1 shows the pseudo code for the edge detection algorithm described in Section 9.2.2.

Algorithm B.1 Edge Detection Through Surface Normals.

check neighborhood of P_i using conditions described by Eq. (9.15), condition one.

one b_i is 0, then mark P_i as an edge, otherwise go for full computation

Full computation:

compute All **n** and **n'**
compute All $\measuredangle$between **n** and **n'**
analyze All computed $\measuredangle$. Conditions two and three in `Eq. (9.15).`
analyze $\triangle$in neighborhood, compute ratios between $\frac{\triangle_i}{\sum \triangle_j - \triangle_i}$, condition four in `Eq. (9.15)`
mark P as an edge if any of conditions met.

Run Connected Component Labeling:

compute Labels and mark objects.

Notify Object extraction process that data is ready.

filter Objects according to the size limitation.
remove Small objects.
copy Accepted objects.

Notify Recognition tasks.

B.2 Recursive Filter Algorithms

The recursive filter algorithm explained in Chapter 5 is reduced to the essential parts and presented here in pseudo code. The main flow is shown in Algorithm Algorithm B.5. Filter reset, measurement update and handling of the discontinuity detection are shown in separate pseudo code algorithms, see Algorithm B.2, Algorithm B.3 and Algorithm B.4.

Algorithm B.2 Filter Reset.

$\hat{x}_0 = E(x) \rightarrow$ Set the state of the system to the possible expectation value, the values are initialized to zeros if the initial expectation value is unknown.

$P_0 = E[(x - x_0)(x - x_0)^T] \vee \infty \cdot I \rightarrow$ Initialize variance-covariance matrix to the values that are valid for the system state (empirical values for example) or initialize variances to ∞ and covariances to 0 if there is no a priori knowledge about the measurement data.

$\Delta t_o := \Delta t_g := 0 \rightarrow$ Initialize the counters to 0.

Algorithm B.3 Measurement Update.

$K_t = P_{t-1} H_t^T (H_t P_{t-1} H_t^T + R_t)^{-1} \rightarrow$ Compute gain matrix.

$\hat{x}_t = \hat{x}_{t-1} + K_t(y_t - H_t \hat{x}_{t-1}) \rightarrow$ Update state of the system.

$P_t = (I - K_t H_t) P_{t-1} \rightarrow$ Update variance covariance matrix.

$t^- := t \rightarrow$ Set current observation time to the last valid observation.

Algorithm B.4 Discontinuity Detected.

$a_n = \hat{x}_t \rightarrow$ Analytic function a_n is the observation approximation for the observations in interval $[t_{start} := \top t, t_{end} := t]$.

$d_n = [t_{start} := \top t, t_{end} := t] \rightarrow$ Assign domain to the analytic function a_n.

$n := n + 1$

Algorithm B.5 Recursive Least Squares Filter.

$n := 1$, $t := 1$

Reset filter: **invoke** Alg. Algorithm B.2

$\forall\, t < t_{max} \rightarrow$ Loop over the measurements domain.

predict $y_t = H_t x_{t-1} + v_t \rightarrow$ Compute the value of observation y_t. This is the observation prediction.

. $y_t^* \rightarrow$ Obtain "real" observation from the sensor.

? $y_t^* \in \{1, \ldots, y_{max}\}$

no

.. $\Delta t_g := \Delta t_g + 1$

.. Continue on **predict** step.

end

. $\Delta t_g := 0 \rightarrow$ Reset gap count to 0.

. $c_o := c_o + 1 \rightarrow$ Increase the observations count.

. $r = t - c_o \rightarrow$ Compute redundancy based on observations count.

? $r > 0 \rightarrow$ Test if the redundancy condition is achieved.

true

.. $\Delta y = y_t - y_t^* \rightarrow$ Compute the difference between the measurement model and current observation.

.. $H_0 : \Delta y = 0 \rightarrow$ Establish hypothesis, that the measurement model equals observed value.

.. $H_1 : \Delta y > 0 \rightarrow$ Establish alternative hypothesis, that the measurement model does not equal observed value, hence it is an outlier concerning current state $\hat{x}_{t-1}$.

.. $t_t = t(y_t, , y_t^*, \Delta y, \sigma^2, r) \rightarrow$ Compute test value for the observation.

.. $z_t = c(y, y^*, r) \rightarrow$ Compute critical value z_t.

? $t_t > z_t$

true

... $\Delta t_o := \Delta t_o + 1$

? $\Delta t_o + \Delta t_g > \Delta t_{max}$

yes **Stop** filtering, continue on **Discontinuity detected**.

end

... Continue on **predict** step.

end

end
$\Delta t_o := 0 \rightarrow$ Reset the outlier count to 0.
invoke Alg. Algorithm B.3
end loop $\rightarrow$ Assigns next value to t.

Discontinuity detected **invoke** Alg. Algorithm B.4

? $t < t_{max}$

yes Continue on **Reset** filter.

end

B.3 Matlab script for the resolution and orientation analysis

Listing B.1: AnalysisEFDS.m

```
% This file analyses features computed as EFD.
%
% set(0, 'DefaultAxesFontName', 'Times New Roman')
clear
close all;
configfile = inputdlg( 'Enter the orientation file-list file name:','File Name:', 1, {'
    lsaaefds.txt'} );
files = parse_filelist( configfile{ 1 } );
count = max( size( files ) );
efds = cell( 1, count );
names = extract_names( files );
for ii = 1:count
    if( ( names{ ii }( 1 ) - 'a' ) >= 0 )
        names{ ii }( 1 ) = char( 'A' + names{ ii }( 1 ) - 'a' );
        names{ ii } = sprintf( '%d. %s', ii, names{ ii } );
    end
end

%% Load data and determine ranges
%
for ii = 1:count
    efds{ ii } = load( files{ ii }, '-ascii' );
end

% Get features size
%
[rw cl ] = size( efds{ 1 } );

%% Compute feature separability for all EFDs
%
means = zeros( count, cl );
medians = zeros( count, cl );
stddevs = zeros( count, cl );
variances = zeros( count, cl );
for ii = 1:count
    means( ii, : ) = mean( efds{ ii } );
    medians( ii, : ) = median( efds{ ii } );
    variances( ii, : ) = var( efds{ ii } );
    stddevs( ii, : ) = std( efds{ ii } );
end

%% Compute covariance matrix
I = efds{ 1 };
for ii = 2:count
    I = [ I; efds{ ii } ];
end
Qxx = cov( I );
P = inv( Qxx );
g_mean = mean( I );
% I = I - repmat( g_mean, size( I, 1 ), 1 );
mn = medians;

% I1 = I * P;
% [idx, c, sumd ] = kmeans( I, count, 'distance','cosine','emptyaction', 'drop', 'start', mn )
    ;
success = false;
clusters = count;
while( ~success )
    try
        [idx, c, sumd ] = kmeans( I, clusters, 'distance', 'cityblock', 'start', mn );
%         [idx, c, sumd ] = kmeans( I, clusters, 'distance', 'cosine' );
        success = true;
    catch err
       clusters = clusters - 1;
    end
end
%% Mahalonobis distance
%
distMahMean = zeros( count, count );
for ii = 1:count
    for jj = ii : count
        diff = means( ii, : ) - means( jj, : );
        distMahMean( ii, jj ) = sqrt( diff * P * diff' );
        distMahMean( jj, ii ) = distMahMean( ii, jj );
```

```
    end
end
figure( 'name', 'Mahalanobis' );
colors = jet( 8 );
colors = colors( length( colors ):-1:1, : );
distance_matrix_plot( distMahMean / max( max( distMahMean ) ), colors, true );
% latex_style( gca );
% print_figure( gcf, 'MahalanobisDistances.pdf' );
presentation_style( gca );
print_figure_presentation( gcf, 'MahalonobisDistance' );

latex_style( gca );
figure( 'name', 'Qxx' );
distance_matrix_plot( Qxx / max( max( Qxx ) ), jet( 10 ), true );
latex_style( gca );
print_figure( gcf, 'Qxx.pdf' );

Rxx = zeros( size( Qxx ) );
for ii = 1:cl
    for jj = ii:cl
        Rxx( ii, jj ) = Qxx( ii, jj ) / sqrt( Qxx( ii, ii ) * Qxx( jj, jj ) );
        Rxx( jj, ii ) = Rxx( ii, jj );
    end
end
figure( 'Name', 'Correlation' );
distance_matrix_plot( Rxx, jet(10), true );
latex_style( gca );
print_figure( gcf, 'Correlation.pdf' );
presentation_style( gca );
print_figure_presentation( gcf, 'Correlation' );

%% Plot data in x y diagram
%
plots = cell( 1, 1);
mina = 4*pi;
maxa = -4*pi;
mind = 1e+308;
y = g_mean;% * P;
for ii = 1:count
    sets = size( efds{ ii }, 1 );
    angles = zeros( 1, sets );
    distances = zeros( 1, sets );
    for jj = 1:sets
        x = efds{ ii }( jj, : ) - g_mean;
%         x = x * P;
        angles( jj ) = acos( x * y' / ( sqrt( x*x' ) * sqrt( y*y' ) ) );
        distances( jj ) = sqrt( sum( x .^ 2 ) );
    end
    mina = min( [ angles mina ] );
    maxa = max( [ angles maxa ] );
    mind = min( [ distances mind ] );
    plots{ ii } = [ angles; distances ] ;
end
figure( 'name', 'Features' );
cmap = colorcube( count+1 );

interval = maxa - mina;
scalea = 2 * pi / interval;
markers = 'sdo';
for ii= 1:count
    angles = ( plots{ ii }( 1, : ) - mina ) * scalea;
    distances = plots{ ii }( 2, : );
    x = distances .* cos( angles );
    y = distances .* sin( angles );
    marker = markers( mod( ii, 3 ) + 1 );
    plot( x , y, marker, 'MarkerSize', 10,'MarkerEdgeColor', [ 1 1 1 ] - cmap( ii,: ), '
        MarkerFaceColor', cmap( ii, : ), 'LineWidth', 2 );
%     pl = mmpolar( angles, distances, 's' );
%     set( pl, 'MarkerSize', 6,'MarkerEdgeColor', 'k', 'MarkerFaceColor', cmap( ii, : ) );
hold on;
end

grid off;
axis equal
axis off
leg = legend( names, 'Location', 'EastOutside' );
latex_style( gca );
set( leg, 'Box', 'off' );
```

```
orient( gcf, 'landscape' );
print_figure( gcf, 'FeaturesDistribution.pdf' );

%% plot k-means
% centroids
%
y = g_mean;% * P;
for ii = 1:clusters
    sbl = l( idx==ii, : );
    x = median( sbl ) - g_mean;
%     x = medians( ii, : ) - g_mean;
    angle = acos( x * y' / ( sqrt( x*x' ) * sqrt( y*y' ) ) );
    angle = ( angle - mina ) * scalea;
    distance = sqrt( sum( x .^ 2 ) );
    cx = distance * cos( angle );
    cy = distance * sin( angle );

    vx = sbl - repmat( g_mean, size( sbl, 1 ), 1 );
    szvx = size( vx, 1 );
    for jj = 1:szvx
        x = vx( jj, : );
        angle = acos( x * y' / ( sqrt( x*x' ) * sqrt( y*y' ) ) );
        angle = ( angle - mina ) * scalea;
        distance = sqrt( sum( x .^ 2 ) );
        vxx = distance * cos( angle );
        vxy = distance * sin( angle );
        plot([cx vxx] , [cy vxy] , '-', 'Color', cmap( ii, : ) );
    end
    plot( cx, cy, 'o', 'MarkerSize', 13, 'MarkerEdgeColor', [1 1 1] - cmap( ii, : ), '
        MarkerFaceColor', cmap( ii, : ), 'LineWidth', 2 );
    plot( cx, cy, 'x', 'MarkerSize', 13, 'MarkerEdgeColor', [1 1 1] - cmap( ii, : ), '
        LineWidth', 2 );
end
print_figure( gcf, 'KMeans.pdf' );

%% Scale data to [-1,+1] interval
%

minval = ones( 1, cl ) * 1e+308;
maxval = ones( 1, cl ) * -1e+308;
reffeat = zeros( rw, cl );
for ii = 1:count
    minval = min( [ efds{ ii }; minval ] );
    maxval = max( [ efds{ ii }; maxval ] );
    reffeat( ii, : ) = efds{ ii }( 1, : );
end

pl = 1;
ii = 1;
rest = mod( count, 10 );
whole = ( count - rest )  / 10;
if( rest == 0 )
    ends = (1:whole)*10;
else
    ends = [ (1:whole)*10 whole*10+rest ];
end

nr = 1:25;
splefd = 1:0.25:25;
figure( 'name', 'Descriptors' );
clear bar_data
bar_data = reffeat( 1, : );
step = 5;
efdstoplot=12;
% for ii = 1:max(size(ends))
for ii = 1:step:count-1
    cc = 1;
    for jj = ii:ii+step-1
        bar_data( cc, : ) = means( jj, : );
        cc = cc + 1;
    end
    bar( bar_data(:, 1:efdstoplot)', 'grouped' );
    colormap( cmap(ii:ii+step-1,:) );

    xlabel( 'Descriptor Number', 'FontSize', 16 );
    ylabel( 'Descriptor Value', 'FontSize', 16 );

     leg = legend( names{ ii :ii + step-1 }  );
```

```
     fnt = findobj( leg, 'type', 'text' );
     set( fnt, 'FontSize', 14 );
     set( gca, 'FontSize', 14 );
    hold on

    shift = min(0.8, step/(step+1.5));
    for jj = 1:step
        x = (1:efdstoplot)-shift*.5 + (2*jj-1)*shift/(2*step);
        errorbar( x, means( ii+jj-1, 1:efdstoplot ), stddevs( ii+jj-1, 1:efdstoplot ), 'r.' );
    end
    grid on;
    axis tight
    orient( gcf, 'landscape' );
    print_figure( gcf, sprintf('EfdsGroup-%d.pdf', pl ) );
    pl = pl + 1;
    clf
end

%% Simple leaves
idx=[25 30 32 34 37];

bar_data = means( idx, : );
bar( bar_data(:, 1:efdstoplot)', 'grouped' );
colormap( cmap(idx,:) );

xlabel( 'Descriptor Number', 'FontSize', 16 );
ylabel( 'Descriptor Value', 'FontSize', 16 );

leg = legend( names{ idx } );
fnt = findobj( leg, 'type', 'text' );
set( fnt, 'FontSize', 14 );
set( gca, 'FontSize', 14 );
hold on

step = length( idx );
shift = min(0.8, step/(step+1.5));
for jj = 1:step
    x = (1:efdstoplot)-shift*.5 + (2*jj-1)*shift/(2*step);
    errorbar( x, means( idx(jj), 1:efdstoplot ), stddevs( idx(jj), 1:efdstoplot ), 'r.' );
end
grid on;
axis tight
orient( gcf, 'landscape' );
print_figure( gcf, 'SimpleLeaves.pdf' );
clf
%% Filigrane leaves
idx=[9 17 28 38];
cc = 1;
bar_data = means( idx, : );
cc = cc + 1;
bar( bar_data(:, 1:efdstoplot)', 'grouped' );
colormap( cmap(idx,:) );

xlabel( 'Descriptor Number', 'FontSize', 16 );
ylabel( 'Descriptor Value', 'FontSize', 16 );

leg = legend( names{ idx } );
fnt = findobj( leg, 'type', 'text' );
set( fnt, 'FontSize', 14 );
set( gca, 'FontSize', 14 );
hold on
step = length( idx );
shift = min(0.8, step/(step+1.5));
for jj = 1:step
    x = (1:efdstoplot)-shift*.5 + (2*jj-1)*shift/(2*step);
    errorbar( x, means( idx(jj), 1:efdstoplot ), stddevs( idx(jj), 1:efdstoplot ), 'r.' );
end
grid on;
axis tight
orient( gcf, 'landscape' );
print_figure( gcf, 'FiligraneLeaves.pdf' );
clf

%% Variance plot
%
variance = diag( Qxx );
lgscales = log10( variance );
% mx = max( variance );
```

```
% scale = 1/mx;
% variance = variance * scale;
figure( 'Name', 'Variance Graphs' );

x = 1:cl;
[ A h1 h2 ] = plotyy( x, variance, x, lgscales, 'plot','semilogy' );
hold on;
axis tight
grid on;
latex_style( A(1) );
latex_style( A(2) );
% set( A(1),'YColor', 'k', 'YTick', [0 0.25 0.5 0.75 1.0] );
set( A(1),'YColor', 'k' );

set( A(2),'XTickLabel', [], 'XTick', [], 'YColor', 'r', 'YGrid', 'on' );
set( h1, 'Marker', 'x', 'LineWidth', 2, 'MarkerSIze', 8, 'Color', 'b' );
set( h2, 'Marker', '.', 'LineWidth', 2, 'MarkerSIze', 8, 'Color', 'r' );
print( gcf, '-r300', '-dpdf', 'SeparabilityGraph.pdf' );
```

B.4 GPU Performance Analysis Source

Listing B.2: GP-GPU Code Fragment.

```
// Fill the matrices with random values
//
rng.fill( _A, cv::RNG::UNIFORM, 0, 1 );
rng.fill( _B, cv::RNG::UNIFORM, 0, 1 );

// Enqueue upload process
//
stream.enqueueUpload( _A, A );
stream.enqueueUpload( _B, B );

// Emulate few intensive matrix operations
//
cv::gpu::multiply( A, B, res, stream );
cv::gpu::transpose( A, B, stream );
cv::gpu::multiply( B, res, A, stream );
cv::gpu::absdiff( res, B, A, stream );

// Download result
//
stream.enqueueDownload( A, gpu_res );
```

Curriculum Vitae

Dejan Šeatović

14.02.1969	Geboren in Nova Gradiška, Kroatien
	Ausbildung
1993 - 2000	Studium der Geodäsie Karlsruher Institut für Technologie (KIT) Karlsruhe
1992 - 1993	Erlernen der deutschen Sprache Studienkolleg der Universität Karlsruhe Karlsruher Institut für Technologie (KIT) Karlsruhe
1989 - 1991	Studium der Geodäsie Universität Zagreb Zagreb, Kroatien Kein Abschluss, Abbruch des Studiums wegen des Bürgerkrieges
	Beruflicher Werdgang
2006 -	Zürcher Hochschule für Angewandte Wissenschaften (ZHAW) Winterthur, Schweiz Wissenschaftlicher Mitarbeiter
2002 - 2006	Leica Geosystems AG Corporate Technology and Innovation Center (CTI) Heerbrugg, Schweiz Senior Engineer & Software Architekt
2000 - 2002	Leica Geosystems AG Software Division Heerbrugg, Schweiz Softwareentwicklungsingenieur
1994 - 2000	Betreuung der Ingenieurbüros ÖbVI Seitz & Stark, Offenburg ÖbVI Ortmann, Bühl ÖbVI Hess, Bretten Architekturbüro Fischer, Freiburg in Breisgau. Verschiedene fachbezogene Tätigkeiten.