3D Reconstruction of Plant Architecture by Grammar-based Modeling and Markov Chain Sampling

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Florian Erwin Schöler

aus

Siegen

Bonn, 2014

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn.

Erstgutachter: PD Dr. Volker Steinhage, Bonn
Zweitgutachter: Prof. Dr. Reinhard Klein, Bonn
Tag der mündlichen Prüfung: *20.08.2014*
Erscheinungsjahr: *2014*

Abstract

The world population is growing rapidly. As a consequence increasingly more products of crop plants are needed. Here, plant breeding gains a special significance. The goal of plant breeding is to develop new cultivars that need less nutrients and at the same time are more resistent to diseases and pests. A deciding factor is the mapping from gene analysis to the phenotype of the plant, i.e., all traits that can be observed from the outside. A major impairment is the so-called *phenotyping bottleneck*, i.e., the fact that phenotyping is too time-consuming and that current, manual methods are subjective and imprecise.

This thesis presents a procedure that is a step toward eliminating the phenotyping bottleneck. A method is presented for the reconstruction of plant architecture from sensor data with the goal of applying it in an automated phenotyping. Several contributions are presented: The reconstruction procedure has three sub-steps that allow phenotyping in increasingly fine detail. A model of the investigated plant, which comprises components, topology and geometry, serves as a compact description of all allowed structures and makes the intrinsic complexity and variability manageable. Above that, the parameters of the model are assigned valid value ranges by an automated skeletonization method. The introduction of the model into the reconstruction allows to regard not only the sensor data as source of evidence, making possible the handling of massive occlusions. In phenotyping coarse categorizations can be replaced by quantified measures and new traits can be investigated.

The grapevine plant, especially the grape cluster, serves as application. The grape cluster is a highly complex construct of branches and berries, whose different development stages offer different challenges. In addition, the grape cluster enables a yield estimate and the detection of diseases. The presented procedure is applied on grape clusters in three different development stages and on the venation of vine leaves. A detailed presentation and evaluation is given for the grape cluster development stage of full ripeness. In principle, the presented procedure is extensible to other plants that, like the grape cluster, can be defined by a tree-like branching structure.

Zusammenfassung

Die Weltbevölkerung wächst rasant. Dies hat zur Folge, dass immer mehr Nutzpflanzenprodukte benötigt werden. An dieser Stelle erhält die Pflanzenzüchtung eine besondere Bedeutung. Ihre Aufgabe ist es, Pflanzensorten zu entwickeln, die weniger Nährstoffe benötigen und gleichzeitig resistenter gegen Schädlinge und Krankheiten sind. Ein entscheidender Faktor ist die Abbildung von Genanalyse auf den Phänotyp der Pflanze, also alle von außen beobachtbaren Merkmale. Ein großer Schwachpunkt dabei ist der sogenannte *Phänotypisierungs-Flaschenhals*, also die Tatsache, dass die Phänotypisierung zu zeitaufwendig ist und mit den aktuellen, eher händischen Methoden subjektiv und ungenau ist.

Diese Arbeit präsentiert ein Verfahren, das einen Schritt auf dem Weg zur Entfernung des Phänotypisierungs-Flaschenhals darstellt. Vorgestellt wird eine Methode zur Rekonstruktion von Pflanzenarchitektur aus Sensordaten, mit dem Ziel des Einsatzes in der automatisierten Phänotypisierung. Es werden mehrere Beiträge vorgestellt: Das Rekonstruktionsvefahren besteht aus drei Teilschritten, die eine Phänotypisierung in feiner werdenden Detailstufen erlaubt. Ein Modell der untersuchten Pflanze, das Komponenten, Topologie sowie Geometrie umfasst, dient als kompakte Repräsentation aller erlaubten Strukturen und macht die naturgegebene Komplexität und Variabilität der Pflanzenstruktur handhabbar. Darüber hinaus werden die Parameter des Modells durch ein automatisiertes Skelettierungsverfahren mit gültigen Werten belegt. Die Einführung des Modells in die Rekonstruktion ermöglicht, dass die Sensordaten nicht als einzige Informationsquelle dienen, sodass auch massive Verdeckungen behandelt werden können. In der Phänotypisierung können grobe Kategorisierungen durch quantifizierte Maße ersetzt und neue Merkmale untersucht werden.

Als Anwendungsfall dient die Weinrebe und im Besonderen die Weintraube. Die Weintraube ist ein hochkomplexes Konstrukt aus Ästen und Beeren, deren unterschiedliche Entwicklungsstadien unterschiedliche Herausforderungen offerieren. Des Weiteren dient die Traube zur Ertragsabschätzung sowie zur Erkennung von Krankheitsbildern. Das vorgestellte Verfahren wird angewendet auf Weintrauben in drei unterschiedlichen Entwicklungsstadien, sowie auf die Aderstruktur von Weinblättern. Detailliert vorgestellt und evaluiert wird das Entwicklungsstadium Vollreife der Weintraube. Im Grundsatz ist die vorgestellte Methode erweiterbar auf Pflanzen, die sich, ähnlich der Weintraube, in Form von baumartigen Verzweigungsstrukturen darstellen lassen.

Acknowledgments

First of all, I would like to thank PD Dr. Volker Steinhage for giving me the opportunity to work on this project and for all the time he had (or took) for discussions and advice about my work.

This work was part of and funded by the CROP.SENSe.net project and especially of the subproject D2. Thank you to all members of that project for discussions and common work. I give special thanks to the working group of Prof. Dr. Heiner Kuhlmann for providing the sensor data used in this thesis and to the working group of Prof. Dr. Reinhard Töpfer for carefully selecting and providing the plants that were used in this thesis.

A great thank-you goes also to Dr. Jens Behley for motivating me in times when things just did not want to work out, and for discussions and ideas about my work. Thanks also to the people who proofread my thesis and gave hints for its improvement: Jenny Balfer, Dr. Jens Behley, Dr. Ribana Roscher, Christoph Schmalhofer, Vanessa Straub, and Sevi Trakossa.

A heartfelt thank-you to my parents who supported me throughout my life and never let me down. Without them I would not be where I am now. Thanks also to my sister and my girlfriend for their encouragement and support.

Contents

Chapter 1

Introduction

1.1 Motivation

In today's world plant breeding is of utmost importance in order to fulfill the demands of an ever-growing population for plant products, which are often categorized as food, feed, fiber, and fuel. The aim of plant breeding is to develop new cultivars of crop plants that are more resistant to diseases and at the same time produce more yield and need less nutrition. This circumvents the need for using pesticides and fungicides, or for using more and more land for these plants. It is a long-term process to breed a new cultivar with beneficial properties; some crop plants need several years before they first bear any fruit and the sheer number of possible crossings alone prohibits an exhaustive search. One of the major problems in plant breeding is called *phenotyping bottleneck* and will be discussed in the next paragraphs.

Plant breeders describe a cultivar by its *genotype* and by its *phenotype* (Sapp [1983]). The genotype is the genetic buildup of an organism, while the phenotype is the set of traits of an organism that can be observed from the outside. The genotype of an organism is the major influence of its phenotype, environmental factors are another influence. While on the one hand there is this *genotype-phenotype distinction*, on the other hand there is also a *genotype-phenotype map* (Alberch [1991]). This map from genotype to phenotype is not trivial, it is rather a map of a set of genotypic traits to a set of phenotypic traits. Plant breeders are interested in this map, because it helps them in predicting the outcome of new breeds. Nowadays, phenotyping, i.e., determining the phenotype is mostly done by trained personnel that has to actually go into the field and look at the plants to be investigated. This is extremely time-consuming and restricts the number and accuracy of phenotypic traits and the number of plants to be investigated within one growth cycle. With manual inspection the traits can only be categorized into rather coarse value regions – otherwise the phenotyping would take even longer – and some categorizations are subject to human judgement and the experience of the particular person. These are reasons why plant breeders speak of the *phenotyping bottleneck*. But phenotyping is important, since the genotype-phenotype map provides a better comprehension of the plant genome. However, phenotyping is well suited for automization. One could imagine an autonomous robot that moves through the field and takes sensor measurements for later processing. Unlike human personnel, the robot could do this theoretically day and night. This holds the potential of a faster and more precise determination of phenotypic traits and also of possible correlations between them.

One part of the phenotype that is of great importance is the *plant architecture*. This is a term that describes the topology – like the number of plant components – and the geometry – like length of components – of the plant's basic structure. The architecture is important for a precise genotype-phenotype map, because it helps in estimating the yield or the detection of susceptibilities to certain diseases. Consider the grapevine plant or more specifically the *grape cluster*. This is a complex structure of stems and berries with certain characteristics regarding, e.g., its outer form. Shavrukov et al. [2004] have found that at a certain state of development, namely anthesis, there are five phenotypic traits of the grape cluster architecture that differ greatly between compact[1] and loose grape clusters and that the more compact clusters are more susceptible to Botrytis cinerea, a disease that causes the berries to rot.

This thesis focuses on an approach to the automated reconstruction of plant architecture for the calculation of phenotypic traits in order to widen the phenotyping bottleneck. It is part of the research network CROP.SENSe.net[2] that has the aim ”to non-destructively and quantitatively analyse and screen plant phenotype throughout plants' lifecycles.”

1.2 Problem Statement

Being part of the CROP.SENSe.net subproject D2 ”Interpretative 3D Plant Architecture,” the aim of this thesis is to develop an automatic, sensor-based method for the reconstruction of plant architecture for the purpose of a detailed phenotyping. This goal raises several challenges:

- **Complexity**: Plant architectures are very complex due to different components, their connections and the complexity in the branching. This requires the determination of the components that build a plant and their topological interrelations. Furthermore, the components do not follow such strict rules as man-made objects like buildings do.

[1]There is no one single definition of compactness but in general terms this describes the portion of the grape cluster volume taken by beries.

[2]`http://www.cropsense.de` (10.06.2014)

- **Variability**: There are many different developments of the architecture. Even within a specific cultivar, there are differences in the actual appearance of exemplars. Again, the variability can be much larger than for man-made objects. This results in a space of possible reconstructions too large to search exhaustively.
- **Disconnectedness**: The used sensor data are laser rangefinder generated point clouds. As such they are a set of unconnected point measurements on the surface of the scanned object. Therefore neither the kind of plant component, nor their connections are encoded in the data and this assignment has to be found.
- **Occlusions**: Parts of the plant or of a specific plant component can not be captured by the sensor, because they are hidden from sight by other plant parts. Therefore, a way has to be found to fill the gaps in the data.

Depending on the actual plant and its state of development, these challenges can be of greater or lesser concern. Here the focus is put on grapevine and in particular on grape clusters. They are highly complex structures with lots of branches in a small space, which makes the disconnectedness even more of a problem since the individual components can even touch. Also the variability in grape clusters is very high, even more so when regarding different cultivars. Occlusions take shape to many different extents. In later stages of development almost the complete interior is hidden from sight by the berries, while in earlier stages almost the entire interior can be seen. Aside from those facts, the components of the grapevine plant that mostly determine the yield are the grape clusters. Also many diseases, like the above-mentioned Botrytis cinerea, manifest on them. Furthermore, the other parts of the grapevine are usually trained into a certain appearance without much natural growth, leaving the grape clusters with the most potential for phenotyping. All these reasons predestine a deep investigation of grapevine grape clusters.

1.3 Concept and Contributions

In this thesis an approach to the automatic, sensor-based reconstruction of plant architecture for the purpose of phenotyping is presented that is a step toward overcoming the problem of the phenotyping bottleneck. A framework is described that is general enough that it may be applied to different plant components or development stages. Figure 1.1 shows an overview of the overall approach. It consists of mainly two parts: Modeling of the plant architecture at hand and reconstruction by incorporating the model and the according sensor data.

To handle the complexity and variability of plant architectures a detailed model of the plant at hand is introduced that captures components, topology, and geometry. Details on the model are presented in Chapter 3, “The Model.” Based on a set of point clouds the model is established in a training phase and consists of two parts. The first part is a structural model that defines of which components the plant comprises and how they are topologically related.

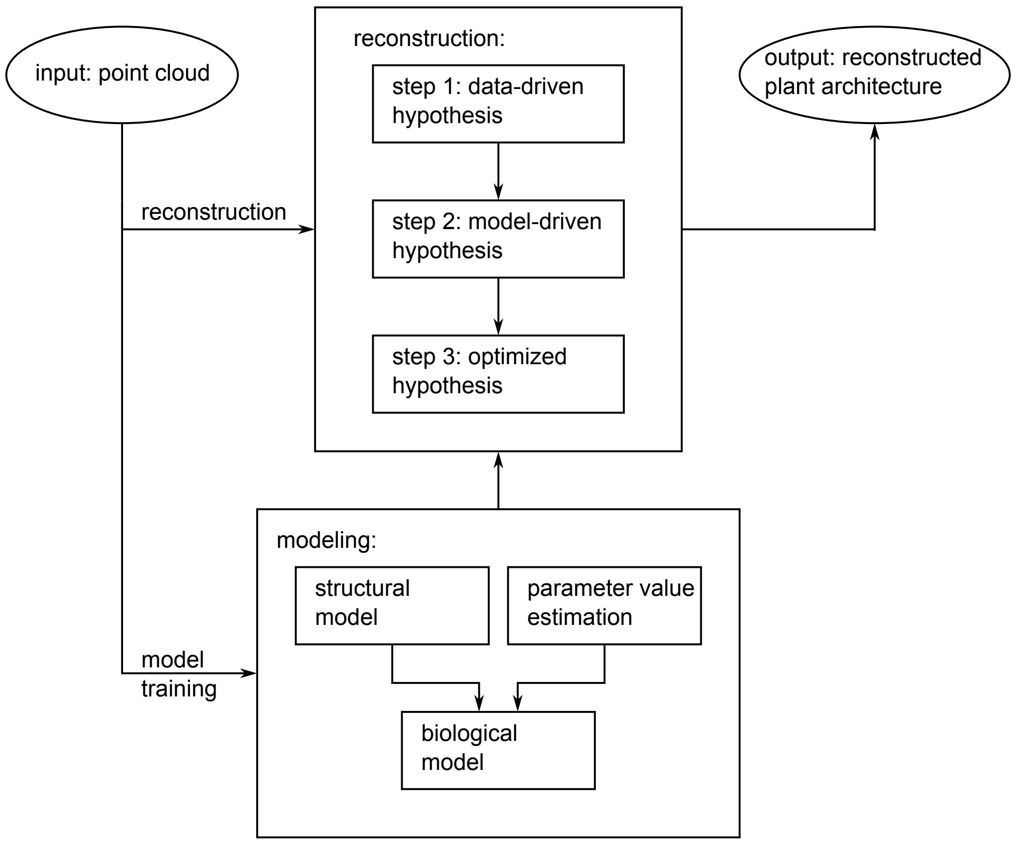

input: point cloud
reconstruction
reconstruction:
step 1: data-driven hypothesis
step 2: model-driven hypothesis
step 3: optimized hypothesis
output: reconstructed plant architecture
model training
modeling:
structural model
parameter value estimation
biological model

Figure 1.1: *The basic concept of the approach. (1) Based on an input point cloud the plant architecture is reconstructed. The reconstruction consists of three steps in a coarse-to-fine strategy. (2) The structural model describes the allowed topologies. A set of point clouds is used as input to an automatic parameter value estimation for assigning value distributions to the model's parameters. Together these methods form the biological model. (3) The output is the final reconstructed architecture.*

The second part consists of determining which parameters describe the components and allocating value distributions to the model's parameters by using the point clouds as input to an automatic parameter value estimation. Together these two parts form the biological model. This biological model serves as a compact description of the allowed variations and it helps in overcoming occlusions by making suggestions which parts are allowed where in space.

Although the model presents a way to handle and represent the complexity and variability of plant architectures and helps in handling occlusions and the disconnectedness of the sensor data, it does not give itself the means for selecting the correct characteristic of the model with respect to the sensor data. This is done in the reconstruction process that is introduced in a general formulation in Chapter 4, "The Reconstruction Approach," and demonstrated in different variations in Chapter 5, "Applications of the Reconstruction Approach." That process has three steps in a coarse-to-fine strategy. The first step enables a coarse derivation of characteristic traits of the plant by a data-driven reconstruction of the visible plant parts. This allows further steps to not only rely on the disconnected points but also on already reconstructed components. In the second step these visible parts can be connected to a complete, global reconstruction hypothesis by using the prior knowledge encoded in the plant model. This allows the derivation of more complex traits of the whole structure. The third step introduces local refinements for an optimization of the global structure and allowing a detailed phenotyping.

The herein presented approach on the reconstruction of plant architecture is in several ways a step toward a precise and automated phenotyping. Now not only established phenotypic traits can be investigated, but their coarse categorizations can be replaced by quantified measures. Traits can be derived faster and more traits can be derived at the same time. Based on the reconstruction results new traits can be found and correlations between traits can be investigated. The approach is specific enough that it, by means of the model, can capture detailed information on the structure of the investigated plant. But it is also general enough that the specifics can be exchanged by those of a different plant, given that they have a tree-like branching structure. This is shown by the applications given in Chapter 5, "Applications of the Reconstruction Approach," where the approach is applied to grapevine grape clusters and vine leaves, but also by the detailed investigation of results of the application to grape clusters in one of the investigated growth stages, summarized in Chapter 6, "Evaluation." The approach allows a level of detail that has not been reached before, even in the case of massive occlusions.

To summarize, the contributions of this work are:

- **Component-based plant model**: The plant model captures components, topology, and geometry and serves as a compact description of all allowed variations, thus reducing the space of reconstruction hypotheses.
- **Automatic parameter value estimation**: In a training phase the value distributions for the parameters of the model are estimated in an automated fashion. Existing distributions can be updated with new measurements and extended to other growth stages or plant parts.
- **Automatic reconstruction**: The three-step reconstruction allows phenotyping in an increasing level of detail.
- **Handling of occlusions**: The interplay of model and reconstruction allows overcoming even massive occlusions.
- **Basis for phenotyping**: The results can be used for a precise and automated phenotyping in high detail. Coarse categorizations of established traits can be replaced by quantified measures and new traits can be discovered.
- **Applicability to other plants**: Plant specifics can be exchanged by those of other plants, given that they have a tree-like branching structure.

1.4 Structure of the Thesis

The remaining part of this thesis is structured as follows. Chapter 2, “Foundations,” introduces basic knowledge about the grapevine plant and the used sensor, and presents an overview of related work. The main contributions of this work are presented in the following four chapters. Chapter 3, “The Model,” introduces an overview of plant modeling methods and the model used in this thesis. Chapter 4, “The Reconstruction Approach,” gives the general formulation of the reconstruction process. Chapter 5, “Applications of the Reconstruction Approach,” shows how the overall approach is used in different cases. A detailed evaluation of one of the cases is given in Chapter 6, “Evaluation.” The main content of this thesis is closed with Chapter 7, “Summary and Outlook,” that gives a summary of the work and possible future research directions. Furthermore, there is an appendix containing the fundamentals of probability theory. Readers who are not familiar with the often occurring biological terms find a glossary of the most important ones at the end of the thesis.

Chapter 2

Foundations

This chapter gives an overview of basic facts needed for this work. The first section contains an explanation of the most important terms concerning the grapevine plant. In the second section the sensor and sensor data utilized in later chapters are introduced. Last, an overview of related work is given, followed by a short summary.

2.1 The Grapevine Plant

The grapevine has a long-standing cultural heritage. For several thousand years wine has been produced in many different cultures and nations. Correspondingly, knowledge about the grapevine plant and about wine and winemaking in general is very diverse. Parts of the plant, however, are still not fully understood or have not even been studied for long. For example, the grape cluster has not gained much attention so far, due to its fine and complex structure that is hard to investigate by hand. On the other hand, Section 1.2 showed that grape clusters actually are very important, for example, for yield estimation and disease detection. Figure 2.1 shows a grape cluster on the right and a view on a whole vineyard on the left.

The exact appearance of a grapevine depends on the training system it has been grown into. Training system is an abstract term that describes the form into which a viticulturist wants the grapevine to grow. This training system is implemented by the trellis system on the one hand and pruning on the other. The trellis system is a structure of posts and wires onto which the grapevine grows and that determines the form of the grapevine. Pruning is the process of cutting parts of the grapevine that deviate from the desired growth form. Nevertheless, there are three basic terms to describe the structure of a grapevine, they are depicted in Figure 2.2. The trunk is the basic above-ground structure that holds the remaining components. From

Figure 2.1: *Left: Picture of a vineyard (Image courtesy of Patou / FreeDigitalPhotos.net) Right: Picture of a grape cluster (Image courtesy of rakratchada torsap / FreeDigitalPhotos.net)*

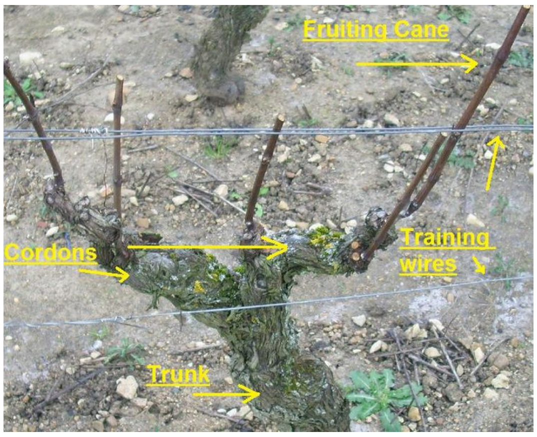


Figure 2.2: *Picture of the basic components of the grapevine. The trunk holds the grapevine and branches into cordons from which canes emerge. Also, the plant is grown into a system of training wires called trellis system. Image source:* `https://en.wikipedia.org/wiki/File:Structure_of_a_grape_vine.jpg` *(22.01.2014)*

the trunk the cordons branch. They are the holding structure for the fruiting canes, which are described in more detail in the next paragraph. The lengths, angles and numbers of those components depend on the training system of the vineyard. The composition of above-ground components is called foliage and the set of leaves are called leafy foliage. This is in contrast to most other plants, where the leafy foliage is called the foliage.

The terms (fruiting) cane, leaf, tendril, and grape cluster will be explained in short here, partially based on Mullins et al. [1992] and Hellman [2003]. Canes are the branches from which leaves, grape clusters, and tendrils emerge, see Figure 2.3 for an illustration. Depending on the pruning there might be a varying number of canes on a single grapevine. Since the canes are restricted in size by the pruning and trellising there are usually one to three grape clusters on a single cane of cultivated grapevines. There are certain rules to the

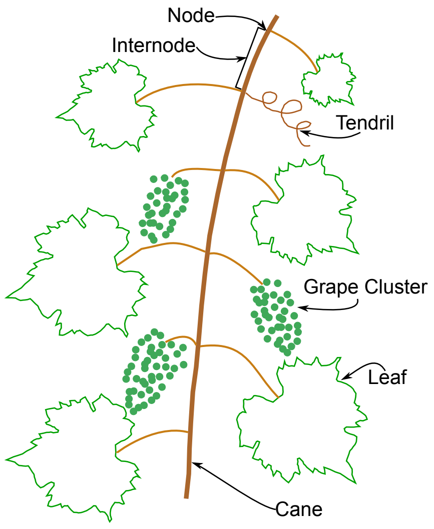


Figure 2.3: *A sketch of a grapevine cane. The cane is the central axis that holds the leaves, the clusters, and the tendrils. Nodes are the points where a branch is located and an internode is the space between two nodes.*

growth of those components. For example, tendrils and clusters occur opposite leaves. While clusters typically branch at the third to sixth node – a branching point –, tendrils emerge in a pattern where every third node lacks a tendril. In early stages of development, when grape clusters do not bear the berries but the flowers, they are called inflorescence. These inflorescences may hold several hundred flowers, of which 70% - 80% do not develop into berries, a process which is called *fruit set*. Leaves consist of the blade and the veins, of which there are five forming the main structure.

Due to the pruning and natural development, not all components are there at all times. Rather, the plant follows a certain growth pattern with certain development stages. These stages are named and classified by the BBCH (Biological Institute of Agriculture and Forestry, Federal Office for Plant Varieties, and Chemical Industry - in German: Biologische Bundesanstalt für Land- und Forstwirtschaft, Bundessortenamt und Chemische Industrie) Lorenz et al. [1994] (German) and Lorenz et al. [1995] (English). For comparing different grapevine cultivars there are mainly three institutes that work on a set of common descriptors for grapevine in its different growth stages and for different components. These are:

1. Organisation Internationale de la Vigne et du Vin (OIV)[1], Paris, France.
2. International Union for the Protection of New Varieties of Plants (UPOV)[2], Geneva, Switzerland.

[1]`http://www.oiv.int` (10.06.2014)
[2]`http://www.upov.int/` (10.06.2014)

3. International Plant Genetic Resources Institute (IPGRI), Rome, Italy, newly renamed to Bioversity International[3].

OIV released their own catalog of descriptors (OIV [2009]), but there is also a common release of all three institutes (IPGRI et al. [1997]).

2.2 Sensor Data

The used sensor data are dense three-dimensional point clouds. The sensor used is a Perceptron ScanWorks V5[4] model. This is a 2d line scanner that produces 7640 range measurements at a frequency of 60 Hz with an average point-to-point resolution of 0.0137 mm and a measurement accuracy of 0.024 mm. The scanner was mounted onto a Romer Infinite 2.0[5] articulated arm. While the operator moves the arm and therefore the sensor, the software that comes with the scanner aggregates all the line scans into a single unified coordinate frame, resulting in a dense 3D point cloud.

For the reconstruction of grape clusters the used grapevine cultivar is Riesling and point clouds of grape clusters were taken at three development stages. These stages are located at the beginning, the middle, and the end of the development process of the grape clusters after pollination and therefore offer different amounts of occlusions and different parameter values. The three stages are:

- BBCH89: Berries are ripe for harvest. Berries have gained their maximum size and weight. Depending on the density of the specific cultivar's grape clusters, the interior – the stem system – can be completely occluded by the berries.
- BBCH81: Berries are at the beginning of ripening. Berries have almost gained their maximum size and weight. Depending on the density of the specific cultivar's grape clusters, the stem system is mostly occluded by the berries.
- BBCH73: Berries are groat-sized. Berries are still growing, most of the stem system is visible.

For every development stage there are 20 point clouds available. Each time they were taken from 10 different canes and always the lowest two grape clusters were taken. The grape clusters were cut in such a way that a part of the cane could be used for fixation with the clusters hanging down. The sizes of the grape clusters typically range between 7 cm and 13 cm.

[3]`http://www.bioversityinternational.org/` (10.06.2014)
[4]`http://www.perceptron.com` (10.06.2014)
[5]`http://www.hexagonmetrology.us/` (10.06.2014)

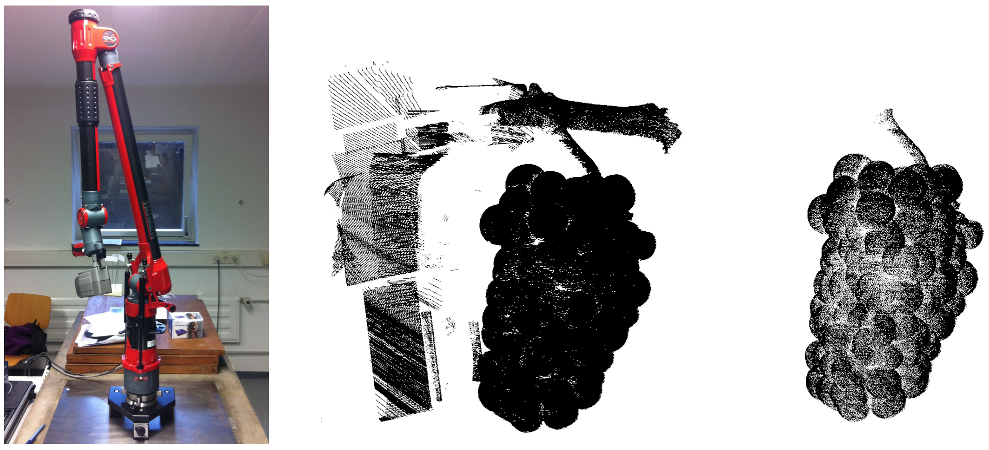


Figure 2.4: *Left: The Romer Infinite 2.0 articulated arm with the Perceptron ScanWorks V5 mounted to its end. Middle: Raw point cloud of a ripe grapevine grape cluster of the cultivar Riesling including parts of the cane and the bearer. Right: Thinned point cloud without cane and bearer.*

In the left image of Figure 2.4 the sensor equipment is shown. The middle image depicts an example point cloud of a Riesling grape cluster including the cane and the bearer that was used for fixating the cluster. In the right image cane and bearer are cut out and the point cloud is thinned to ca. 250,000 points. Thinning is done by taking every n-th of the input points. Since the laser scanner generates line scans of 7640 points each, taking every n-th point preserves the distribution of points on the surface.

In a further application the reconstruction of the veins of vine leaves is investigated. For this there are a total of 22 point clouds of leaves from five different cultivars.

2.3 Related Work

This is a general overview of related work in the fields of reconstruction, plants and plant modeling, and phenotyping. More specific references are discussed in the different subparts of this work.

There has been much work dedicated to reconstructing objects from sensor data. It can be distinguished by the used sensor, the investigated objects and, of course, the reconstruction method itself. Plants, for example, have always been of much interest, especially the reconstruction of trees, either in laser range data (Binney and Sukhatme [2009], Raumonen et al. [2013], Yan et al. [2009]) or based on images (Shlyakhter et al. [2001]). All the mentioned approaches differ from this work by regarding either no or only little occlusions and that they use no or only little domain knowledge. Also in contrast to this work, they do not consider the different components a plant consists of. The approach of Ning et al. [2009] for the segmentation of trees from background in pointclouds can be seen as a precursor for reconstruction methods. Similar to trees, there are medical applications of reconstructing branching tubular surfaces (Hijazi et al. [2010]).

Another well studied object for the reconstruction are buildings, where often similar sensor data and methods are used. Buildings can be reconstructed from laser range data (Ochmann et al. [2014]), aerial images (Vanegas et al. [2010]), digital surface models (Wahl et al. [2008]), or from multi-sensor inputs (Reisner-Kollmann et al. [2011]). More general approaches try to fit geometric primitives in point clouds (Li et al. [2011], Schnabel et al. [2009], Schnabel et al. [2007]), use structured light (Weinmann et al. [2012]), special tree graphs (Chen et al. [2010]), or aim for a global optimization of some energy model (Kolev et al. [2009]). Musialski et al. [2012] give an overview of reconstruction methods in an urban environment.

Plant architecture in general has received a lot of attention. A review on plant architecture and morphology in general terms is given in Barthélémy and Caraglio [2007]. Guédon et al. [2001] investigated branching sequences for recurring patterns. Roth-Nebelsick et al. [2001] specifically investigated the architecture of leaf veins. For comparing the similarity of the architectures of two plants or plant components, Ferraro and Godin [2000] and Boudon et al. [2013] made proposals for distance measures. Also, there are different ways to store a plant architecture in a computer's memory. Godin [2000] gives an overview of such methods. In Godin and Caraglio [1998] and Godin et al. [1999] a specific data structure, the mulitscale tree graph, is introduced. While these methods investigate plants in general, several researchers also examine specific plants. For grapevine, for example, there are analyses of cane development and architecture (Lebon et al. [2004], Louarn et al. [2007]), cluster architecture (Shavrukov et al. [2004], Vail and Marois [1991]), or the influence of trellis systems (Louarn et al. [2008]).

For plant modeling there are fundamental works on how to model a plant in general (Jaeger and Reffye [1992], Prusinkiewicz [1998], Prusinkiewicz et al. [2001]) and more specific investigations, for example, for trees (Honda [1971]), or inflorescences (Frijters [1978]). In the early years of plant modeling most models considered only the plant's functions, like photosynthesis or nutrient transport, or their structure, i.e., their architecture. In later works also the interactions between structure and function were modeled (Fourcaud et al. [2004], Fourcaud et al. [2008], Godin and Sinoquet [2005], Jirasek et al. [2000]). And since plants do not grow in separated spaces, their interactions with the environment also have to be regarded (Pirk et al. [2012]).

Several generative plant models have been developed using different formalisms. Barley was considered in Buck-Sorlin et al. [2008] as a Relational Growth Grammar (RGG) and in Dornbusch et al. [2007] using Matlab. Rice was modeled by Watanabe et al. [2005] as a Lindenmayer System and by Xu et al. [2009] as a Relational Growth Grammar. Another RGG model was done for poplar in Buck-Sorlin et al. [2006]. Norway spruce was investigated in Colin and Houllier [1991] with a special software called Simqua. The model plant of this work, grapevine, was the key interest in Pallas et al. [2009] and grape clusters were considered in Huang et al. [2013b]. Especially the last one should be considered for an improvement of the model that is currently used in this work. While these models need extensive manual labor, in Schöler et al. [2013] a method for automatically generating plant models is introduced.

Since the phenotyping bottleneck is such a huge problem, many research groups are working on solutions. Alenya et al. [2011] and Chéné et al. [2012] present methods to select leaves in time-of-flight camera data. This is used, for example, for a calculation of the leaf area, leaf curvature, or leaf orientation. In Nuske et al. [2011], Kicherer et al. [2013], and Roscher et al. [2014] methods are presented for the detection of wine berries in images for an estimation of the yield on different scales. Yield estimation can also be done with the work presented in this thesis. A comparison of results would be interesting. Hall et al. [2003] also consider grapevine. They developed a method to extract single vines in airborne digital images and can calculate the local size and shape of canopies. Another image-based method, termed PhenoPhyte, is presented in Green et al. [2012] for analysis of growth or leaf area. Huang et al. [2013a] measure the length of rice panicles in a dual-camera setup. On a larger scale, especially LemnaTec and their "scanalyzer" platforms[6] have shown practical applicability of automated phenotyping. All these methods rely on 2D image data from only one view-point. In contrast, in this work 3D point clouds with a surround view are used, which have the potential of greater accuracy. Point clouds are also used in Paulus et al. [2013], but with a different aim. They want to classify plant organs in the point clouds, for example, for the separation of wheat ears from other organs, for a calculation of kernel numbers and weights. Busemeyer et al. [2013] went one step further and presented a tractor-pulled phenotyping platform called BreedVision for an application in the field instead of laboratory conditions.

2.4 Summary

The first section of this chapter gave an introduction to the basic terms to describe the grapevine plant and its components. This was followed by a description of the used sensor and its resulting data, as well as an overview of the data actually used in this work. The chapter closed with an overview of related work in the fields of reconstruction, plant modeling, and phenotyping. In the next chapters the actual reconstruction approach will be presented, starting with the modeling part in the following chapter.

[6]`http://www.lemnatec.com/` (10.06.2014)

Chapter 3

The Model

This chapter details the first part of the overall reconstruction method: Plant modeling. At first a short survey of popular plant modeling technologies is given. Then the two parts that form the biological model are shown starting with a description of the structural model. Afterward a method for the automatic parameter value estimation is presented that can assign value distributions to the parameters of the structural model. All three sections then culminate in the implementation of the biological model of grape clusters.

3.1 Grammar-based Plant Modeling

There are many tools and formalisms for the modeling of plants. In earlier days these were divided into mainly two categories. On the one hand there were functional models. Those methods were used to model the inner workings of a plant, like the flow of nutrients or photosynthesis. On the other hand some modelers merely considered a plant's structure, describing, for example, internode lengths. In both directions a lot of improvements were made. Still, in recent years plant modelers discovered that it is useful to regard both aspects at the same time. This gave rise to the Functional Structural Plant Models (FSPM).

The next two subsections introduce the two most used general purpose FSPM formalisms, both of which are categorized as grammar-based approaches. They are termed Lindenmayer System and Relational Growth Grammar. Grammar-based approaches to plant modeling work by describing the state of a plant with a certain data structure and repeatedly reorganizing that data structure by adding, deleting, or substituting elements by rewriting rules. Opposed to other formal grammars, in such a grammar-based modeler all rewriting rules that can be applied at a certain point in time are applied to the data structure in parallel. This feature stems from the fact that a biological system is represented by the data structure and biological systems do not change their structure sequentially and in certain parts but in a parallel fashion and possibly on all parts. What makes these approaches practical for the usage in plant modeling is that even with only a few simple rewriting rules the development of a plant can be handled and very complex structures can be created. Lindenmayer System and Relational Growth Grammar mainly differ in their choice of data structure, but this has far-reaching implications for the actual implementation of a model. Relational Growth Grammar was developed later and can be seen as a superset of Lindenmayer System. For this work, Relational Growth Grammar was chosen for the modeling part, since it not only has the features of Lindenmayer System and all its major extensions, but by the choice of data structure offers even more modeling power. For example, in Section 5.1.3 one can see how in a Relational Growth Grammar it is possible to rewrite several different parts of the data structure in just one step. In a Lindenmayer System this would produce a less elegant implementation with several substeps and organization overhead. Also, the standard software tool for implementing a Relational Growth Grammar is well-suited for an application in this work, since it allows an easy extension by self-programmed plug-ins. In a step-wise introduction and for a better understanding of Relational Growth Grammar this section starts with an explanation of Lindenmayer System.

3.1.1 Lindenmayer System

Aristid Lindenmayer first developed his formulation for describing the development of simple filamentous organisms in the late 1960's (Lindenmayer [1968a], Lindenmayer [1968b]). It was later termed Lindenmayer System or L-System. The basic idea of the formalism is to use a string of characters to describe the current structure of the organism, where each character represents a single cell. Interactions are described by a character's context, i.e., characters to its right or left. For branches the special characters "[" and "]" are introduced. Changes of the structure are implemented by rewriting the string with rewriting rules. These rules substitute one character with a string, which results in a new string. Three components are needed to describe an L-System. First, an *alphabet* of allowed characters is needed, denoted by $\mathcal{A}$. Second, the *axiom* – or start word – ω gives the initial string, from which the formalism starts. Third, there is the set of rewriting rules, $\mathcal{R}$.

As an example, consider the alphabet that consists of the characters s, b, and the branching characters [, and]. The character s represents a stem and the character b represents a bud, i.e., a growth unit of a plant. The example starts with the axiom string sb, and it has just one rewriting rule, where a bud is replaced by a branching stem and a new growth section consisting of a stem and a new bud. This L-System can be formalized as follows:

$$\begin{aligned} \mathcal{A} &= \{s, b, [,]\} \\ \omega &= sb \\ \mathcal{R} &= \{b \rightarrow [s]sb\} \end{aligned}$$

According to the rewriting rules in every iteration the character b is replaced, while the s characters remain. Therefore, this example simulates the growth and branching of a plant. The first few iterations of this L-System are the following:

$$\begin{aligned} &0) \quad sb \\ &1) \quad s[s]sb \\ &2) \quad s[s]s[s]sb \\ &3) \quad s[s]s[s]s[s]sb \end{aligned}$$

This is still rather abstract since the result is merely a string. That is why over the years many extensions to the basic formalism were developed. The most important one for the purpose of visualization is the introduction of special characters that can be interpreted by so-called turtle graphics. Others are Parametric L-Systems, Stochastic L-Systems, or Table L-Systems. Most of them are covered in Prusinkiewicz and Lindenmayer [1990]. Other extensions introduce the control of the growth (Ijiri et al. [2006], Jirasek et al. [2000]), running L-Systems on parallel computer architecture (Lipp et al. [2009]), or the interaction of plants with their environment (Měch and Prusinkiewicz [1996], Prusinkiewicz et al. [1994]). Another direction for controlling the growth of plants is the interactive editing and arrangement of plant models (Ganster and Klein [2008], Power et al. [1999]). Chen and Colomb [2003] propose a coupling of L-Systems and databases. The generality of the L-System formalism is demonstrated by articles interpreting L-Systems as music, simulating fire or others (Goel and Rozehnal [1991], Prusinkiewicz [1986], Worth and Stepney [2005], Zaniewski and Bangay [2003]). The most wide-spread software to implement and visualize L-Systems is called L-Studio[1] (Prusinkiewicz et al. [2000]), an alternative is L-Py, which is part of the plant modeler OpenAlea[2].

3.1.2 Relational Growth Grammar

Relational Growth Grammar (RGG), presented in Kniemeyer [2008], Kniemeyer et al. [2008], and Hemmerling et al. [2008], works similar to L-System: A data structure is iteratively changed by rewriting rules. For Relational Growth Grammars this data structure

[1]`http://algorithmicbotany.org/lstudio/` (10.06.2014)
[2]`http://openalea.gforge.inria.fr/dokuwiki/doku.php` (10.06.2014)

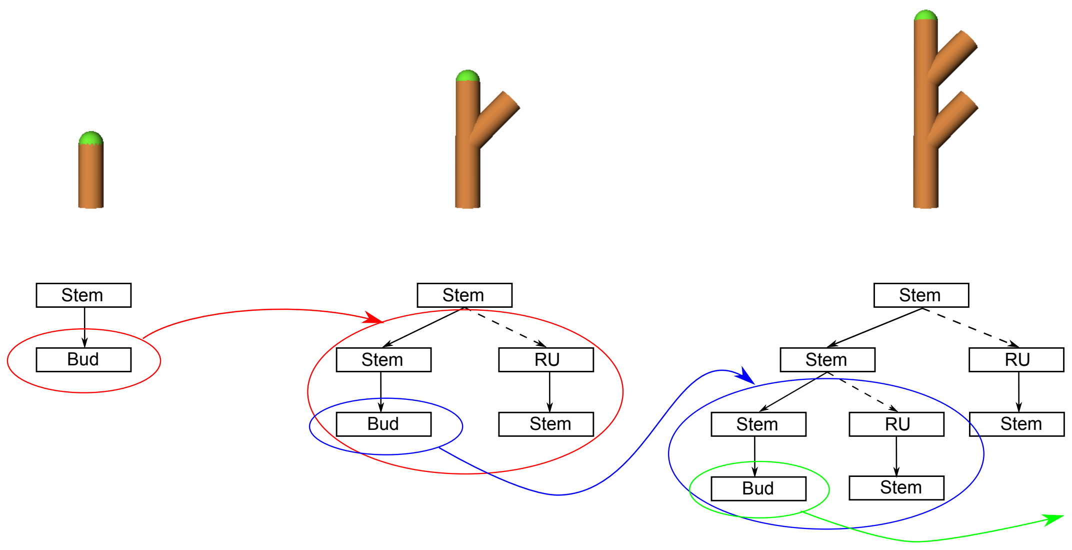


Figure 3.1: *Example RGG with one rewriting rule: Bud ==>[RU(45) Stem] Stem Bud; The replacements are highlighted with colored ellipses.*

is a graph, consisting of nodes, which are also called modules, and edges. The modules, which are objects in the sense of Object Oriented Programming, represent components and transformations of an object and the edges represent relations between them. In an RGG it is allowed to not only rewrite a single module but a subgraph or even a set of subgraphs within just one rewriting rule. Since the whole graph can be inspected, also the context of a rewriting rule can be a set of subgraphs. The RGG formalism was developed with L-System in mind. Therefore, while all important extensions of the L-System formalism are captured, Relational Growth Grammars grant much more modeling power through the use of a graph as the data structure.

An RGG is defined by three components. The rewriting starts at the initial graph, called axiom. The graph can be constructed from a set of allowed modules. Finally, there is a set of rewriting rules. See Figure 3.1 for an example of an RGG. Depicted are three steps of the early development of a simple plant. Module names are capitalized. The top row shows visualizations while the bottom row depicts the graphs. The solid arrows represent the relation of direct succession, the dashed arrow stands for a branch relation. In the left column the axiom is shown; it consists of a Stem, which is visualized as a cylinder, and a Bud, which is visualized as a sphere. The graph view shows the Stem and the Bud modules. There is only one rewriting rule:

```
Bud ==> [RU(45) Stem] Stem Bud;
```

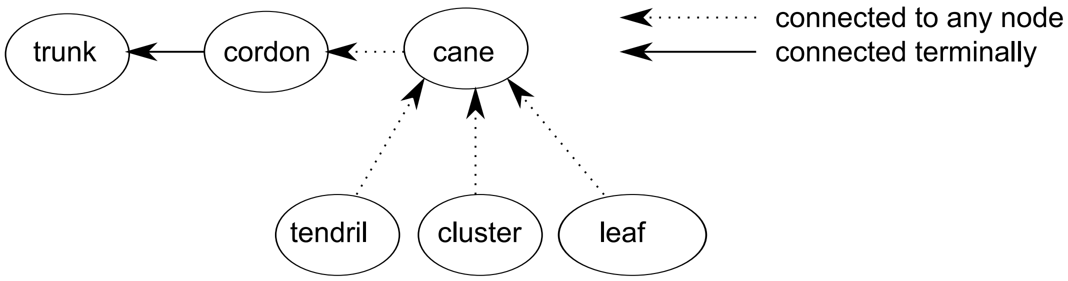


Figure 3.2: *Plant connectivity graph of grapevine: From the trunk cordons branch, from which canes branch, which in turn hold the leaves, the clusters, and the tendrils.*

This rule searches for occurrences of Bud modules. Each such occurrence is replaced by the graph on the right hand side of the arrow. The part between the [, and] symbols represents a branch. It consists of a rotation module, RU(45), and a new Stem module. Also, to model the growth of the plant a new Stem is inserted followed by a new Bud. In the figure each replacement is highlighted by the colored ellipses.

Simultaneously to Relational Growth Grammar the software GroIMP[3] (Growth Grammar-related Interactive Modeling Platform) was developed that is used to implement an RGG. It comes with features such as a compiler for the special purpose language XL and 3D rendering via OpenGL. It also has a simple plug-in interface that allows the user to write their own additions for GroIMP. In fact, for this work, for example, a plug-in for reading and processing 3D point clouds was written. GroIMP is constantly extended, for example, for specifying differential equations (Hemmerling et al. [2013]).

3.2 The Structural Model

In Section 1.3 the importance of a detailed, component-based model was pointed out. In short, the reasons were that an automatic, sensor-based reconstruction is needed for the purpose of a detailed phenotyping, and that the model determines the possible reconstruction hypotheses, thus also enabling the algorithm to handle massive occlusions. This section specifies the structural grapevine model. In the next two sections this is extended to become the biological model. The model and all the assumptions integrated into it are based on an empirical study of the grape clusters available for this work.

See Figure 3.2 for an abstract graph of possible connections of the above-ground grapevine components, following Section 2.1, that from now on will be called *plant connectivity graph*. This graph is an abstract representation of the allowed connections of components. A grapevine consists of the six components trunk, cordon, cane, tendril, grape cluster, and leaf,

[3]`http://sourceforge.net/projects/groimp/` (18.03.2014)

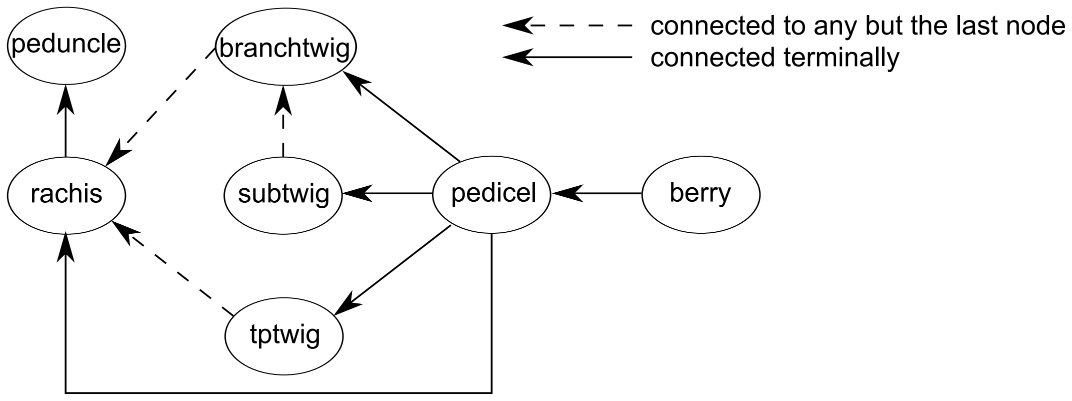


Figure 3.3: *Plant connectivity graph of grape cluster. The arrows indicate where the branches may occur. For example, pedicels are only allowed to occur terminally, i.e., at the end of a structure, while terminal pedicel twigs (tptwigs) are only allowed to branch non-terminally from the rachis. This will be used in Section 5.1.2*

.

but components are neither allowed to branch from an arbitrary component nor everywhere on a component. For example, the cordons emerge terminally, i.e., from the end of the trunk, represented by the solid arrow. The canes emerge from the cordons at any node, i.e., branching point, represented by the dotted arrow. The canes hold the leaves, grape clusters, and tendrils at any node, again represented by the dotted arrows. Imagine the ellipses as a high-level representation of the component that ignores the subcomponents of which it comprises. For example, the grape cluster itself is a highly complex structure but only represented by a simple ellipse; details are given in the next paragraph. As another example, canes consist of nodes and internodes, where a node is a point on the cane from which other components branch and internodes are the parts of the cane between nodes. These nodes and internodes are hidden by the high-level representation, since for the purpose of this graph it does not matter at which node a specific cluster, tendril, or leaf branches from the cane. The plant connectivity graph represents the fact that these components are only allowed to branch from any node of the cane but not, for example, from the trunk, or a cordon. The ellipses also ignore of how many subcomponents the components actually consist.

Since the focus of this work is on grape clusters, Figure 3.3 illustrates the plant connectivity graph of grape clusters. This is a detailed view of the cluster ellipse of Figure 3.2, which will be used in Section 5.1.2 for a first reconstruction hypothesis. There are five components: peduncle, rachis, twig (of which there are three different types), pedicel, and berry. The peduncle is a short stem that connects the grape cluster to the cane. The rachis is connected terminally to it and consists of nodes and internodes. Furthermore, there are three types of twigs: branchtwigs, subtwigs, and terminal pedicel twigs (short: tptwigs). A tptwig is a twig that branches from the rachis and only holds pedicels – short stems that hold the berries – that branch at its end, i.e., terminally. A branchtwig also branches from the rachis and holds

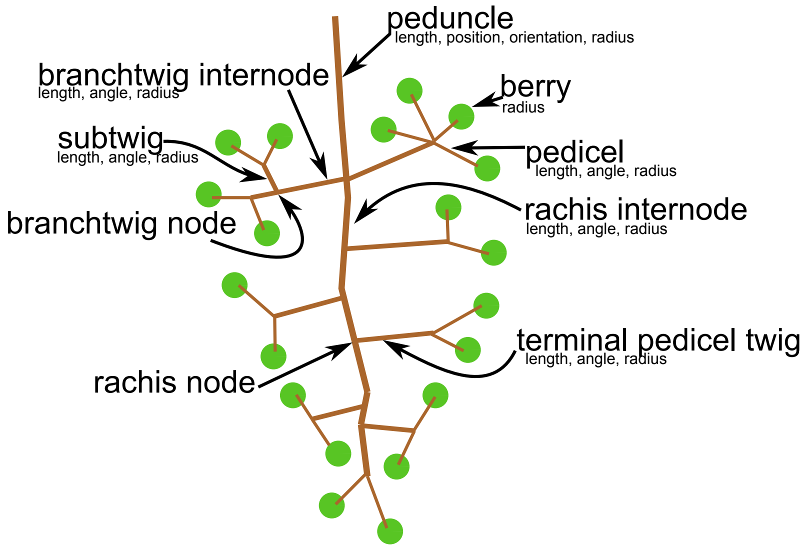


Figure 3.4: *This figure shows the basic components of a grapevine grape cluster according to the model: peduncle, rachis, three kinds of twig, pedicel, berry.*

terminal pedicels as well as a certain number of subtwigs. The subtwigs in turn hold only terminal pedicels. Subtwigs are not allowed to branch from the rachis and they are described by different parameter values than tptwigs. A berry is always attached to a pedicel, which in turn is always attached terminally to a twig or the rachis.

On the basis of the abstract grape cluster representation in form of the plant connectivity graph, Figure 3.4 shows one example structure that can be derived from the plant connectivity graph. It also shows the descriptive parameters of the components. The brown parts in that sketch form the stem system. At the top there is the peduncle. It is described by the position of the topmost point (the actual connection point to the cane), its length, its radius, and its orientation. Since the grape clusters were cut off from the plant, this orientation is in relation to the coordinate frame defined by the sensor. The orientations of all other components are always relative to the orientation of the parent component. After the first branch on the peduncle it is called rachis and it then consists of nodes and internodes, as do the branchtwigs. Subtwigs and tptwigs, however consist of only one stem. Terminally attached to twigs and rachis are the pedicels, which in turn hold the berries. There are a few more assumptions on the appearance only implicitly captured in the sketch.

- Berries whose pedicels branch from the same node form a *berry group*. The node these pedicels branch from is called *pedicel origin*.
- Although tptwigs are allowed to appear anywhere on the rachis, branchtwigs are only allowed to branch from the upper half of the rachis, i.e. on the first half of the path from the peduncle to the furthest berry group.

- The rachis is approximately located in the middle of the cluster, i.e., two twigs that branch close to each other from the rachis have approximately the same length.
- Besides the parameters of the single components and of the angles between them, further parameters are the frequencies of occurrence of the components, that is, of the different twigs and of the berries.

3.3 Automatic Parameter Value Estimation

Now the parameters of the structural model have to be given possible value distributions. A method for the generation of skeletons was used to automize the process. The skeletonization of 3D point clouds has the aim of reducing the complex and unconnected set of points to a simpler representation of line segments, or edges, that reflects the original structure inherent in the point cloud. The considered parameters are the length, angle and frequentness of grape cluster components.

Skeletons are very well suited for that purpose, since they reduce the point clouds to the minimally needed graph structure that, along with a semantic annotation, allows the derivation of parameters in the first place. For this purpose the berries were manually removed from the available grape clusters leaving behind the stem system. There are many different approaches for the computation of skeletons based on point clouds. For example, in Bucksch et al. [2009] and Bucksch [2011] an octree-variant is used to subdivide the space taken by the point cloud to create an initial skeleton by connecting neighboring octree cells with edges. Then iteratively specially configurated edges, so-called E-Pairs and V-Pairs, are contracted, which finally converges to the resulting skeleton. In a work by Cao et al. [2010] the points of the point cloud are iteratively moved towards the object's center, until the point cloud encloses no further volume. A third example of yet another approach is the one by Livny et al. [2010]. Like the method of Bucksch [2011] it specializes in computing skeletons of trees. Here the initial graph is a weighted Minimum Spanning Tree of the point cloud. In the iteration process points are moved by a chain of global least-squares optimizations, which already encode some assumptions about the structure of a tree. Cornea et al. [2007] have put together a summary of skeletonization approaches, sorted them into categories, and worked out a common vocabulary of properties.

Here the method presented in Balfer [2012] and Balfer et al. [2013] is used. This approach enhances the method of Livny et al. [2010]. The method was chosen because it specializes in generating skeletons from point clouds of trees and already yields directed, acyclic, and connected skeletons; the other investigated approaches did not have all these properties. The extended approach of Balfer [2012] and Balfer et al. [2013] introduces a model of grapevine stem systems. This introduction has two effects. (1) It is possible to annotate the different parts of the skeleton with semantic labels. (2) By help of the semantic labels the resulting

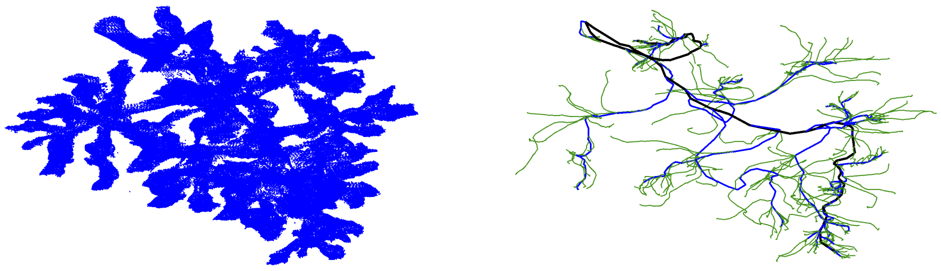


Figure 3.5: *Point cloud and final annotated skeleton of one Riesling exemplar. The rachis is drawn in black, the branches in blue, and the pedicels in green.*

skeletons could be improved in a post-processing step. See Figure 3.5 for a visualization. In the left there is an example point cloud and the right part depicts the skeleton. The rachis is drawn in black, the branches in blue, and the pedicels in green.

For actually computing value distributions, a set of stem system point clouds was used to produce the corresponding skeletons. From those skeletons distributions for parameter can be calculated. For example, for the total length of the rachis in the development stage BBCH89 ”ripe berries” the result is a Normal distribution with a mean of 11.53 cm and a standard deviation of about 1.74 cm.

3.4 Implementation of the Biological Model

As stated at the beginning of the chapter, Relational Growth Grammar will be used for the implementation of the grape cluster model, due to the greater modeling power based on the graph data structure and the software tool GroIMP that is easy to extend via plug-ins. Coming back to the grape cluster model of Section 3.2 and taking a closer look at Figure 3.4 one can see that a grape cluster has already a graph structure, therefore it is relatively easy to translate this structure into an RGG graph. The modules of that RGG graph are the plant components on the one hand and the transformations on the other hand. Rotation modules in the graph implement the assumption that orientations are relative to parent rotations. In RGG models the plant components are often represented by basic geometric primitives. This is done here as well and two primitives are used: frustum and sphere.

Table 3.1 shows the mapping of the grape cluster components – as described in Section 3.2 – to RGG modules and the respective geometric primitives, which represent the components graphically. Notice that RGG modules are capitalized, this notation will be used throughout the thesis. For example, the grape cluster component peduncle is mapped to the RGG module Peduncle, which is graphically represented as a frustum. More generally speaking, components that are kinds of stems are represented by frustums, whereas nodes and berries are

grape cluster component	RGG module	geometric primitive
peduncle	Peduncle	frustum
rachis internode	Rachis	frustum
rachis node	Rachis_Node	sphere
terminal pedicel twig	Tptwig	frustum
branchtwig internode	Branchtwig	frustum
branchtwig node	Branchtwig_Node	sphere
subtwig	Subtwig	frustum
pedicel	Pedicel	frustum
berry	Berry	sphere

Table 3.1: *Mapping of grape cluster components to RGG modules and geometric primitives for graphical representation.*

represented as spheres. There are essentially two relations between grape cluster components. A component can branch from its parent component or it can be its successor. These two relations directly translate to the corresponding relations in an RGG graph. Branches are represented by branch edges and successions are represented by successor edges.

An example of the representation of a grape cluster as an RGG graph is shown in Figure 3.6. In the left part of the figure there is a reduced exemplar of a grape cluster that consists of a short rachis and only one terminal pedicel twig. The right part shows the corresponding RGG graph. In general, modules with a solid border are grape cluster components and modules with a dashed border are transformations. In the root of the example graph there is a *Translate* module, that translates the top of the peduncle in relation to the global origin, as defined by the sensor. The modules *RU* and *RL* are rotations around the local y-, and x-axis, respectively. This is then followed by the actual Peduncle, represented by a frustum in the visualization. The Peduncle has two children with two different types of relation. The solid arrow depicts a successor and the dotted arrow a branch. This means that following solid arrows one walks along the rachis, while following the dotted arrow one ends up in the branching terminal pedicel twig, which again can be walked along by following the twig's solid arrows. This basic structure is repeated throughout the entire graph. The Tptwig and the Rachis are rotated by RU and RL modules relative to the parent component and are followed by successor and/or branch edges, until the last internode on a twig or the rachis is reached. Then this is followed by a Pedicel_Origin, from which the Pedicels branch, again rotated by RU and RL modules. Finally, there are Berries attached to the Pedicels.

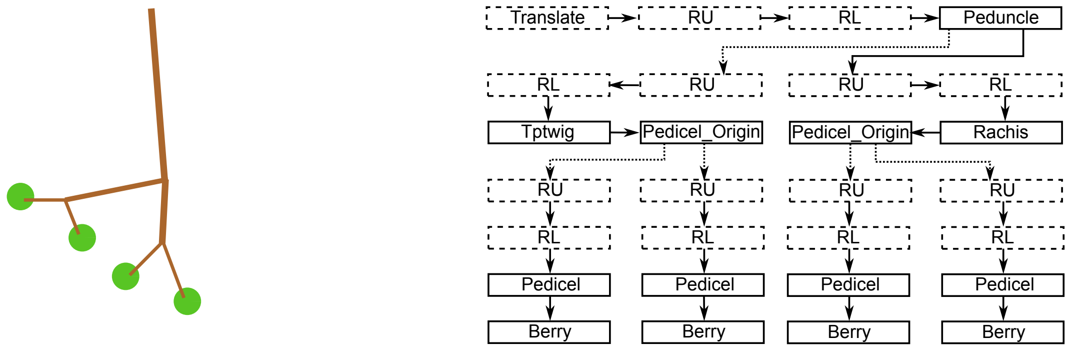


Figure 3.6: *Representation of the grape cluster model as an RGG graph. Transformations are marked with dashed boxes, geometric primitives with solid boxes. Solid arrows indicate successor relation and dotted arrows branching relation.*

3.5 Summary

In this chapter the biological model for grapevine grape clusters was introduced. First, an overview of popular grammar-based plant modeling approaches was given. After that the structural model of grapevine and in particular of grape clusters was introduced in the form of plant connectivity graphs. Next, a method for an automated finding of value distributions for the model parameters was presented. Finally, the implementation of the model as a Relational Growth Grammar was shown. The next chapter shows how that model is integrated into the reconstruction method.

Chapter 4

The Reconstruction Approach

In Section 1.3 the concept of this work was introduced. In short, the idea was to use a model of grapevine grape clusters to determine the possible hypotheses for the reconstruction. The structural model was enriched with parameter value distributions through a skeletonization algorithm and implemented in the Relational Growth Grammar formalism. This chapter deals with the remaining part of the concept, the reconstruction.

The model from Chapter 3, "The Model," has several parameters, like the frequencies of occurrrence of components or their lengths. All allowed allocations of values to those parameters lead to different characteristics of the model. The set of all those characteristics forms the space of allowed reconstruction hypotheses. Figure 4.1 illustrates this by means of grape clusters. One can see that there are different amounts of berries and twigs, the berry groups have different sizes, and the angles and lengths of components differ.

The goal is to find the best of all those hypotheses with regards to a certain sensor input. This is done in a coarse-to-fine strategy where each step introduces more details of the structure. In a first step – which is driven by the sensor data – the visible plant components are detected. For grape clusters, for example, it depends on the development stage and the density of the specific cultivar which components are actually visible, but always the set of outer berries can be detected. This first step already narrows down the search to a certain part of the hypothesis space, where the parameters equal those of the detected components. All other components are still free to chose. Since this first step usually does not yield a complete hypothesis, a second, more model-driven step is introduced. This step bases on the detected components of the first step and uses model knowledge to insert the occluded parts of the structure and generate a complete reconstruction hypothesis. But this is only one hypothesis out of the whole hypothesis space and not neccessarily the best with regards to the sensor data and the model. Therefore, in a third step, a method is needed that is able to refine the

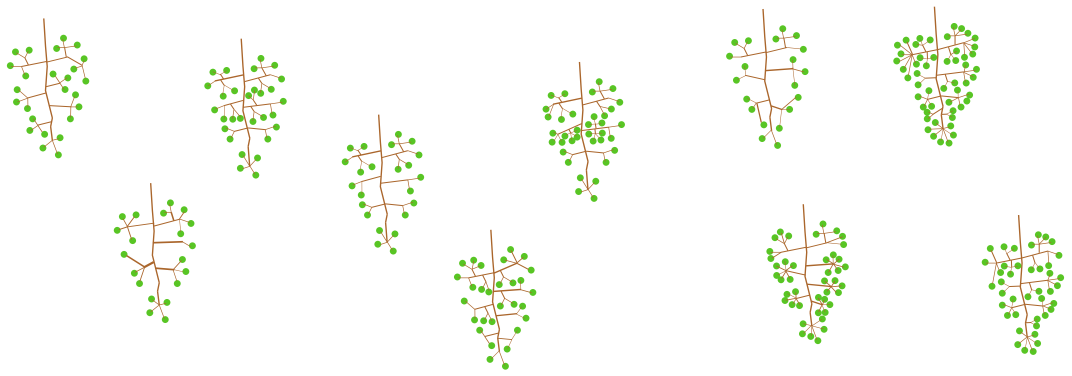


Figure 4.1: *Excerpt of the space of allowed reconstruction hypotheses, as defined by the grape cluster model. The hypotheses differ in the number of occurrences of components and their lengths and angles. For example, the hypothesis in the top right has many more berries than all the others.*

hypothesis in such a way that it better explains the sensor data with regard to the model. This method must be able to handle a space of reconstruction hypotheses that is of infinite size and that contains hypotheses of different dimensions. Due to the infinite size this method can not simply visit every hypothesis, but it must explore the hypothesis space in a smart way to visit likely hypotheses. For this one can define a probability distribution over the elements of the hypothesis space, where each hypothesis receives a value of how well this hypothesis explains the sensor data and satisfies model constraints. Then the problem is to find the hypothesis with the maximum value. The method of choice for this is Reversible Jump Markov Chain Monte Carlo. This is a method to generate samples from an arbitrary probability distribution and it is able to handle the size of the hypothesis space and the varying dimensions of hypotheses.

In a step-wise introduction of the reconstruction algorithm, the following sections will first introduce basic Markov Chain theory, followed by an explanation of Markov Chain Monte Carlo and the Metropolis-Hastings algorithm, which then leads to an explanation of Reversible Jump Markov Chain Monte Carlo and Simulated Annealing. Last, the actual reconstruction algorithm is explained.

4.1 Markov Chain Sampling

Markov Chain Monte Carlo (MCMC) is a method to draw samples from an arbitrary probability distribution $\mathcal{P}(\cdot)$ for which it is not possible to draw samples directly, for example, due to their complexity or high dimensionality. Sampling means generating values x_i that are distributed according to $\mathcal{P}(\cdot)$, written as $x_i \sim \mathcal{P}(\cdot)$. In MCMC the sampling is done by constructing a Markov Chain, consisting of states, state transitions, and state transition

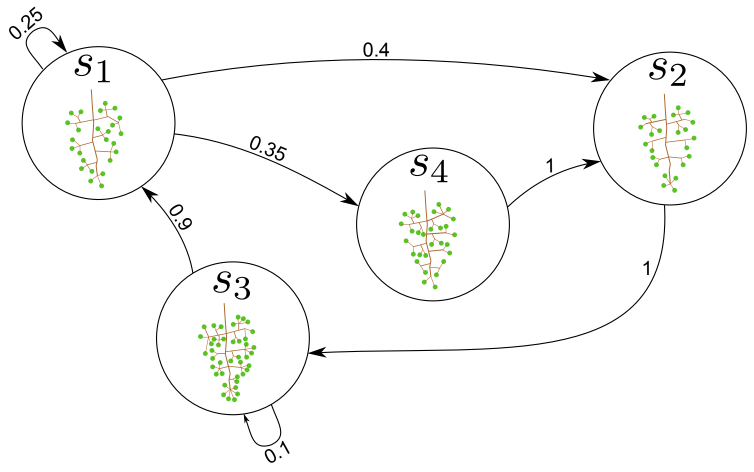


Figure 4.2: *A Markov Chain consisting of four states. The arrows indicate possible state transitions, the values on the arrows indicate the respective transition probabilities.*

probabilities and taking a random walk over that chain. Random walk means that when at a certain point in time the current state is x, the next state of the chain will be x' according to the state transition probabilities of state x. The Markov Chain is constructed in such a way, that due to the random walk, it has $\mathcal{P}(\cdot)$ as its stationary distribution. If that is reached, the following states of the random walk are considered to be distributed according to $\mathcal{P}(\cdot)$. It is then said that samples are collected from the Markov Chain. The next sections give a more detailed view of the theory and present two algorithms for actually implementing the theory.

4.1.1 Markov Chain Theory

This section gives an overview of the most important terms regarding Markov Chains and shows connections of those terms to the task of plant reconstruction. For a more comprehensive overview see, for example, Koller and Friedman [2009] or Bishop [2007]. A Markov Chain is defined by two components: (1) The *state space* $\mathcal{S}$ and (2) the *transition model* $\mathcal{T}$. The state space encompasses all possible values for the states of the Markov Chain. The transition model specifies the probabilities of going from the current state x to an arbitrary state x' in $\mathcal{S}$. Figure 4.2 shows an example of a Markov Chain where the states are exemplars from the space of reconstruction hypotheses. In fact, for the purpose of reconstruction the state space $\mathcal{S}$ equals the space of reconstruction hypotheses. The example consists of four states. For state s_1, for example, the probability to go to state s_2 is 0.4, to go to state s_4 it is 0.35, and with a probability of 0.25 the random walk remains at state s_1. The state s_3 can not be directly reached from s_1.

Due to the probabilistic nature of the transition model and the random choice of initial state, the result of the random walk is a sequence of states $x^{(0)}, x^{(1)}, x^{(2)}, \ldots$ One can see the state of the random walk at time t as a random variable $X^{(t)}$. The transition model $\mathcal{T}$ is a probability distribution $p(X^{(t+1)} \mid X^{(t)})$. This is also called the *Markov property* and it states that the probability for the value of the next random variable $X^{(t+1)}$ only depends on the value of the current random variable $X^{(t)}$, not on the previous ones. The possible values of $X^{(t)}$ at time t are described by a probability distribution $p^{(t)}(X^{(t)})$. The next state probability distribution is defined as

$$p^{(t+1)}(X^{(t+1)} = x') = \int p^{(t)}(X^{(t)} = x)\mathcal{T}(x' \mid x)\, dx.$$

The probability of x' being the next state of the Markov Chain is determined by the probability of being in any state x multiplied by the probability of transitioning from x to x' and integrating over all states x.

Considering again the example in Figure 4.2 and assuming that s_1 is the initial state, the first seven of the series of probability distributions looks like this (rounded to five decimal places):

$$\begin{aligned}
p^{(0)}(X^{(0)}) &= [1.00000, 0.00000, 0.00000, 0.00000]\\
p^{(1)}(X^{(1)}) &= [0.25000, 0.40000, 0.00000, 0.35000]\\
p^{(2)}(X^{(2)}) &= [0.06250, 0.45000, 0.40000, 0.08750]\\
p^{(3)}(X^{(3)}) &= [0.37563, 0.11250, 0.49000, 0.02188]\\
p^{(4)}(X^{(4)}) &= [0.53506, 0.17238, 0.16150, 0.13169]\\
p^{(5)}(X^{(5)}) &= [0.27912, 0.34571, 0.18853, 0.18727]\\
p^{(6)}(X^{(6)}) &= [0.23945, 0.29892, 0.36457, 0.09769]
\end{aligned}$$

Over time the series of probability distributions $p^{(0)}(X^{(0)}), p^{(1)}(X^{(1)}), \ldots$ might converge and this is then called a *stationary distribution* of the Markov Chain. The maximum of this stationary distribution is what has to be found in the reconstruction process. A distribution $\pi(X^{(t)})$ is stationary with respect to a Markov Chain, if it is invariant according to the step of the random walk, i.e., if it does not change anymore as new steps of the random walk are integrated:

$$\pi(X^{(t+1)} = x') = \int \pi(X^{(t)} = x)\mathcal{T}(x' \mid x)\, dx$$

For the Markov Chain of Figure 4.2 with initial state s_1 the later distributions $p^{(t)}(X^{(t)})$ are (rounded to five decimal places):

$$\begin{aligned}
p^{(28)}(X^{(28)}) &= [0.34058,\ 0.25603,\ 0.28390,\ 0.11949]\\
p^{(29)}(X^{(29)}) &= [0.34066,\ 0.25572,\ 0.28442,\ 0.11920]\\
p^{(30)}(X^{(30)}) &= [0.34114,\ 0.25547,\ 0.28416,\ 0.11923]
\end{aligned}$$

Therefore convergence can be assumed – at least to a certain degree. In this case s_1 has the highest value and should therefore be the output of the reconstruction method. The number of iterations until the stationary distribution is reached up to a user-defined tolerance ϵ is called *mixing time*. The mixing time might depend on the initially chosen distribution. Another term in this context is the *burn-in time*. This is the number of iterations that have to pass until samples are collected from the Markov Chain. Of course, the burn-in time should be equal to or greater than the mixing time, or otherwise the first samples were not collected from the stationary distribution.

A Markov Chain is called *reversible*, if it satisfies the *detailed balance* condition

$$\pi(x)\mathcal{T}(x' \mid x) = \pi(x')\mathcal{T}(x \mid x') \quad \forall x,\, x' \in \mathcal{S}$$

that states that for any pair of states x and x' the probability of being in x and transitioning to x' is the same as being in x' and transitioning to x. Reversibility, i.e., satisfying the detailed balance condition implies that the Markov Chain has been constructed so that $\pi(\cdot)$ is a stationary distribution. A Markov Chain may have more than one stationary distribution and is then called *reducible*. A Markov Chain is called *ergodic*, if for $t \to \infty$ the distributions $p^{(t)}(X^{(t)})$ converge to the required invariant distribution $\pi(X^{(t)})$, no matter the choice of initial distribution $p^{(0)}(X^{(0)})$, making $\pi(\cdot)$ the unique stationary distribution. Of course, the mixing time might still differ greatly for different initial distributions.

Taking another look at the example of Figure 4.2, one can observe for different initial reconstruction hypotheses a convergence to approximately the same distribution after only 30 iterations (rounded to five decimal places):

$$\begin{aligned}
p^{(0)}(X^{(0)}) = [1,\ 0,\ 0,\ 0] &\to p^{(30)}(X^{(30)}) = [0.34114,\ 0.25547,\ 0.28416,\ 0.11923]\\
p^{(0)}(X^{(0)}) = [0,\ 1,\ 0,\ 0] &\to p^{(30)}(X^{(30)}) = [0.34066,\ 0.25598,\ 0.28388,\ 0.11948]\\
p^{(0)}(X^{(0)}) = [0,\ 0,\ 1,\ 0] &\to p^{(30)}(X^{(30)}) = [0.34066,\ 0.25575,\ 0.28437,\ 0.11923]\\
p^{(0)}(X^{(0)}) = [0,\ 0,\ 0,\ 1] &\to p^{(30)}(X^{(30)}) = [0.34138,\ 0.25551,\ 0.28368,\ 0.11943]
\end{aligned}$$

The transition probabilites are usually unknown, therefore the goal of Markov Chain Monte Carlo algorithms is to construct a Markov Chain, i.e., its transition probabilities over a given state space that has a unique stationary distribution $\pi(\cdot)$ with $\pi(\cdot) = \mathcal{P}(\cdot)$, where $\mathcal{P}(\cdot)$ is the distribution from which samples are to be drawn. In the case of plant reconstruction

Algorithm 4.1 The Metropolis-Hastings Algorithm

```
Require: initial state $x_0$
1: for $i = 1 \rightarrow N$ do
2:     $x' \sim q(x' \mid x_i)$
3:     $\alpha = \mathcal{A}(x' \mid x) = \min\left(1, \frac{\mathcal{P}(x')}{\mathcal{P}(x)} \frac{q(x|x')}{q(x'|x)}\right)$
4:     if $\alpha > 0 \wedge a \sim U(0, 1) \leq \alpha$ then
5:         $x_{i+1} = x'$
6:     else
7:         $x_{i+1} = x_i$
8:     end if
9: end for
```

the state space equals the hypothesis space and the distribution $\mathcal{P}(\cdot)$ is the distribution of probabilities of the hypotheses with regards to the data. One of the most popular algorithms for constructing Markov Chains is the Metropolis-Hastings (MH) algorithm. It is a simple, yet powerful algorithm for designing Markov Chains with a specific stationary distribution. Also the MH algorithm was used as the basis in the formulation of the Reversible Jump Markov Chain Monte Carlo (RJMCMC) algorithm, which increases the capabilities by not only sampling from a specific distribution but also selecting the best distribution. For a better understanding, the MH algorithm is introduced first, followed by the RJMCMC algorithm.

4.1.2 The Metropolis-Hastings Algorithm

The Metropolis-Hastings (MH) algorithm is an approach for generating samples from an arbitrary, yet specific distribution of a known dimension. It constructs a Markov Chain, i.e., its transition probabilities over a given state space by taking a random walk over that space that finally leads to a convergence of the Markov Chain to the desired distribution. Since it is not possible to sample directly from the desired distribution, a so-called proposal distribution is used. This proposal, as the name suggests, proposes the next state of the random walk. The proposed state is then either accepted as the next state or rejected, depending on an acceptance probability. The MH algorithm is more of a guideline, since it leaves the designer with the freedom to create their own proposal distribution.

This is a short overview of the method, summarized in Algorithm 4.1, further details can be looked up in, e.g., Andrieu et al. [2003], Chib and Greenberg [1995], Hastings [1970], Metropolis et al. [1953], or Koller and Friedman [2009]. Algorithm 4.1 requires as the only input the initial state of the random walk, labeled x_0. Then the algorithm basically performs N many iterations of a loop consisting of three steps. In the first step in line 2 a new state x' is proposed by drawing it from the proposal distribution $q(\cdot)$. The proposal might be conditioned on the current state x_i as in the pseudo code but that is not mandatory. In the second step in line 3 the acceptance probability $\mathcal{A}(x' \mid x)$ is calculated. The current and the proposed states are evaluated according to the desired distribution $\mathcal{P}(\cdot)$ and the proposal distribution $q(\cdot)$. In the third and final step in lines 4 through 8 the acceptance probability is used to accept or reject the proposed state. The result of the algorithm is a sequence of states $x^{(0)}, x^{(1)}, x^{(2)}, \ldots, x^{(N)}$ of the state space. All states $x^{(i)}$, where $\eta \leq i \leq N$ and η the burn-in time, are considered samples drawn from the desired distribution $\mathcal{P}(\cdot)$.

The acceptance probability is derived as follows. For $\mathcal{P}(\cdot)$ to be the unique stationary distribution, the Markov Chain has to be ergodic and the detailed balance condition has to be satisfied. Since a step of the random walk is now suggested to consist of two steps, proposing a new state and accepting or rejecting it, the transition probability becomes

$$\mathcal{T}(x' \mid x) = q(x' \mid x)\mathcal{A}(x' \mid x)$$

and the detailed balance equation can be rewritten:

$$\begin{aligned}
& & \pi(x)\mathcal{T}(x' \mid x) &= \pi(x')\mathcal{T}(x \mid x') \\
&\Leftrightarrow & \mathcal{P}(x)q(x' \mid x)\mathcal{A}(x' \mid x) &= \mathcal{P}(x')q(x \mid x')\mathcal{A}(x \mid x') \\
&\Leftrightarrow & \frac{\mathcal{A}(x' \mid x)}{\mathcal{A}(x \mid x')} &= \frac{\mathcal{P}(x')}{\mathcal{P}(x)}\frac{q(x \mid x')}{q(x' \mid x)}
\end{aligned}$$

To fulfill detailed balance, in the MH algorithm $\mathcal{A}$ is chosen to be

$$\mathcal{A}(x' \mid x) = \min\left(1, \frac{\mathcal{P}(x')}{\mathcal{P}(x)}\frac{q(x \mid x')}{q(x' \mid x)}\right)$$

and implemented as follows. If $\alpha = \mathcal{A}(x' \mid x)$ is in the interval $(0, 1)$, x' is accepted as the new state with probability α. In the extreme cases $\alpha = 0$ and $\alpha = 1$ it is always rejected or accepted, respectively.

The distribution $\mathcal{P}(\cdot)$ from which the samples are to be drawn can be arbitrarily chosen. In the context of reconstruction a Bayesian approach can be used, where $\mathcal{P}(\cdot)$ is formulated according to Bayes' formula:

$$\mathcal{P}(x_i) = p(x_i \mid \mathcal{D}) = \frac{p(x_i, \mathcal{D})}{p(\mathcal{D})} = \frac{L(\mathcal{D} \mid x_i) \cdot p(x_i)}{p(\mathcal{D})} \propto L(\mathcal{D} \mid x_i) \cdot p(x_i).$$

The distribution $\mathcal{P}(\cdot)$ is replaced by a distribution stating how good the state of the random walk, i.e., the current reconstruction hypothesis is, given the data. By Bayes' formula this is transformed into two terms: (1) The likelihood $L(\mathcal{D} \mid x_i)$ that states how likely it is to observe the sensor data given the current state. (2) The prior $p(x_i)$ that states how probable the current state is, unaware of the data. The term $p(\mathcal{D})$, which is the prior for the sensor data, can be dropped since it is the same for all states.

The only shortcoming of Metropolis-Hastings is the fact that the target distribution has to be fixed. For the reconstruction this means that only one characteristic of the plant model, i.e., one specific combination of components can be regarded. The algorithm then samples over the geometric parameters, i.e., the lengths and angles. But since the model can have many different combinations of components, a more powerful algorithm is needed, that is capable of handling these different characteristics. This algorithm is Reversible Jump Markov Chain Monte Carlo and presented in the next section.

4.1.3 Reversible Jump Markov Chain Monte Carlo

The Metropolis-Hastings algorithm presented in the previous section is a useful tool for sampling from an arbitrary distribution in cases when the distribution to sample from is known. But when the distribution to sample from is unknown, standard Metropolis Hastings cannot be used. It is only designed to sample from one specific distribution of a known dimension. Here Reversible Jump Markov Chain Monte Carlo (RJMCMC) offers a solution. RJMCMC is an extension to the previously described Metropolis-Hastings algorithm for cases where the invariant distribution is a further unknown. The algorithm works by introducing jumps that can be used to switch between distributions and to change parameters of a distribution. The algorithm was originally presented in Green [1995] and in a reformulated version in Green [2003].

RJMCMC works on a countable set of models $\mathcal{M} = \{M_1, M_2, \ldots\}$. A configuration is described in terms of the chosen model M_k and its parameter vector θ_k. Each θ_k might have a different dimension, termed n_k. Therefore a configuration is written as $x = (k, \theta_k)$, denoting the model index k as well as the parameter vector θ_k for the model M_k. For a jump from a configuration (k, θ_k) in model M_k with dimension n_k to a configuration $(k', \theta'_{k'})$ in Model $M_{k'}$ with dimension $n_{k'}$ one has to generate auxiliary vectors u of length m_k and u' of length $m_{k'}$. One of the vectors, or both, may have length 0. Then a jump is a function

$$\begin{aligned} f : \mathbb{R}^{n_k+m_k} &\to \mathbb{R}^{n_{k'}+m_{k'}} \\ (\theta_k, u) &\mapsto (\theta'_{k'}, u') \end{aligned}$$

that transfers the current configuration (k, θ_k) into a new configuration $(k', \theta'_{k'})$ by help of the random vectors u and u'. This function f has to satisfy two constraints:

- It has to be deterministic. The generation of the vectors u and u' might be randomized, but the function itself must be deterministic.
- It must be a diffeomorphism: (a) It has to be bijective. (b) The function itself and its inverse function have to be differentiable, making certain that it has a non-zero Jacobian determinant.

Algorithm 4.2 Reversible Jump Markov Chain Monte Carlo Algorithm

```
Require: initial configuration $(k,\ \theta_k)_0$
 1: for $i = 1 \to N$ do
 2:     $m \sim j(1, \ldots, J)$
 3:     $u \sim q_m(u \mid (k,\ \theta_k)_i)$
 4:     $(\theta'_{k'},\ u') = f_m((\theta_k,\ u))$
 5:     $\alpha = \frac{\mathcal{P}(k', \theta'_{k'} | \mathcal{D})}{\mathcal{P}(k, \theta_k | \mathcal{D})} \cdot \frac{j_{m'}(\theta'_{k'})}{j_m(\theta_k)} \cdot \frac{q_{m'}(\theta_k, u | \theta'_{k'}, u')}{q_m(\theta'_{k'}, u' | \theta_k, u)} \cdot \left| \frac{\partial(\theta'_{k'}, u')}{\partial(\theta_k, u)} \right|$
 6:     if $\alpha > 0 \wedge a \sim U(0,\ 1) \leq \alpha$ then
 7:         $(k,\ \theta_k)_{i+1} = (k',\ \theta'_{k'})$
 8:     else
 9:         $(k,\ \theta_k)_{i+1} = (k,\ \theta_k)_i$
10:     end if
11: end for
```

For the function f to be a diffeomorphism, the so-called *dimension matching* condition must hold, i.e., $n_k + m_k = n_{k'} + m_{k'}$. The number and design of the jumps is up to the designer of the actual sampler and has to be adapted to the problem at hand. It is important that for every jump m there is a reverse jump m' that is able to transform $(k',\ \theta'_{k'})$ back into $(k,\ \theta_k)$.

The algorithm changes only a little compared to standard MH, see algorithm 4.2. The input is an initial configuration, here stated as $(k,\ \theta_k)_0$. Again the algorithm is basically just a loop of N iterations. There are two new steps involved at the beginning of the loop. The very first step in line 2 is to decide which jump is to be performed, so the index m into the set of jumps is drawn at random from a distribution $j(1, \ldots, J)$, where J is the total number of jumps. Second, according to the chosen jump an auxiliary vector u is drawn randomly from a distribution q_m in line 3. The following steps are similar to the ones in standard Metropolis-Hastings. The proposal is calculated by the jump f_m in line 4, followed by the computation of the acceptance probability in line 5. Here already the more specific case of the probability is given, where the desired distribution is $\mathcal{P}(k,\ \theta_k \mid \mathcal{D})$ instead of a general formulation. The term $j_m(\theta_k)$ gives the probability of chosing jump m in configuration $(k,\ \theta_k)$, q_m is the proposal distribution for the vector u and the last term is the determinant of the Jacobian matrix of the diffeomorphism from $(k,\ \theta_k)$ to $(k',\ \theta'_{k'})$. The last step in lines 6 through 10 again decides whether to accept or reject the proposed configuration.

In the context of reconstruction an RJMCMC model from the set of models $\mathcal{M}$ refers to a characteristic of the plant model, i.e., one specific configuration of its components. The distribution to sample from is $\mathcal{P}(k,\ \theta_k \mid \mathcal{D})$. By Bayes' Theorem this can be reformulated as follows:

$$\mathcal{P}(k,\ \theta_k \mid \mathcal{D}) = \frac{L(\mathcal{D} \mid k,\ \theta_k) \cdot p(\theta_k \mid k) \cdot p(k)}{p(\mathcal{D})} \propto L(\mathcal{D} \mid k,\ \theta_k) \cdot p(\theta_k \mid k) \cdot p(k).$$

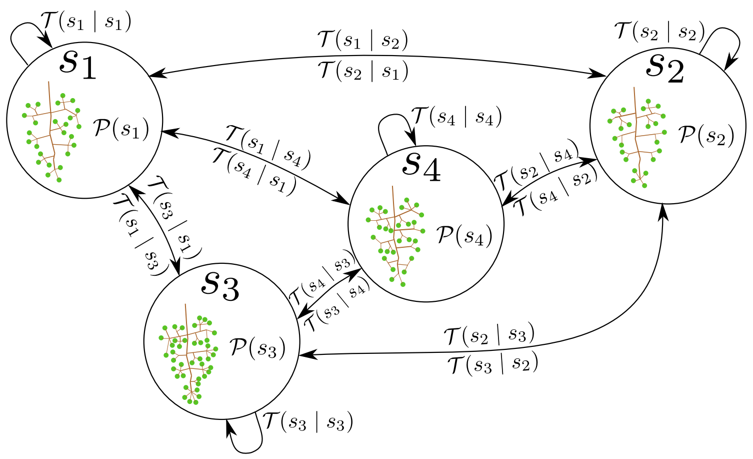

$\mathcal{T}(s_1 \mid s_1)$
s_1
$\mathcal{P}(s_1)$
$\mathcal{T}(s_1 \mid s_2)$
$\mathcal{T}(s_2 \mid s_1)$
$\mathcal{T}(s_2 \mid s_2)$
s_2
$\mathcal{P}(s_2)$
$\mathcal{T}(s_4 \mid s_4)$
s_4
$\mathcal{P}(s_4)$
$\mathcal{T}(s_1 \mid s_4)$
$\mathcal{T}(s_4 \mid s_1)$
$\mathcal{T}(s_2 \mid s_4)$
$\mathcal{T}(s_4 \mid s_2)$
$\mathcal{T}(s_3 \mid s_1)$
$\mathcal{T}(s_1 \mid s_3)$
s_3
$\mathcal{P}(s_3)$
$\mathcal{T}(s_4 \mid s_3)$
$\mathcal{T}(s_3 \mid s_4)$
$\mathcal{T}(s_2 \mid s_3)$
$\mathcal{T}(s_3 \mid s_2)$
$\mathcal{T}(s_3 \mid s_3)$

Figure 4.3: *Reversible Jump Markov Chain Monte Carlo calculates transition probabilities, such that the random walk leads to a convergence to the desired distribution $\mathcal{P}(\cdot)$.*

The likelihood $L(\mathcal{D} \mid k, \theta_k)$ states how likely the data are, given the chosen model with the chosen parameter values. The term $p(\theta_k \mid k)$ states the probability of the parameter values for the chosen model, unaware of the data. Finally, $p(k)$ is the prior probability of the chosen model, i.e., the probability of the model unaware of the data. Reversible Jump Markov Chain Monte Carlo calculates transition probabilities, such that the random walk leads to a convergence to the desired distribution $\mathcal{P}(\cdot)$, as shown in Figure 4.3. Each of the arrows corresponds to an RJMCMC jump and its reverse jump, each with their own probability. The selection of jumps defines how much of the hypothesis space can be explored, which transitions are allowed, and if regions of the hypothesis space with hypotheses of high probability are visited more often than those with low probability. There does not neccessarily have to be a direct transition between any two hypotheses, it is sufficient if a hypothesis can be reached within a finite number of transitions.

Several extensions to the basic formulation have been proposed, for example, for the efficient construction of proposals (Brooks et al. [2003]), for improving the acceptance rate (Al-awadhi et al. [2004]), and even in the direction of an automated design of RJMCMC (Hastie [2005]). While they yield results for relatively simple problems, these extensions cannot be used in the context of this thesis, due to the complexity of the task at hand. RJMCMC has been successfully applied to object tracking (Khan et al. [2005]), analysis of mixture distributions (Richardson and Green [1997]), the reconstruction of buildings (Brenner and Ripperda [2006], Ripperda [2008], Ripperda and Brenner [2009], Lafarge et al. [2008], Dick et al. [2004], Fan and Brooks [2000]), and image analysis in microscopy data (Al-awadhi et al. [2011], Schlecht et al. [2007]). A tutorial on RJMCMC applied on QTL mapping – the mapping of regions on the chromosome to the variation of quantitative phenotypic traits; hence quantitative trait loci (QTL) – is given in Waagepetersen and Sorensen [2001].

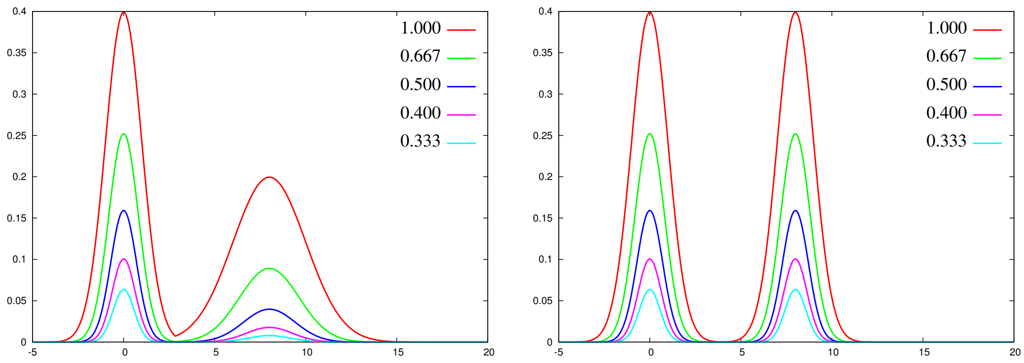


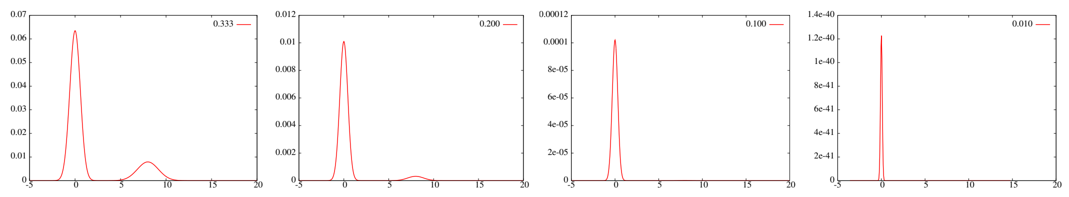
Figure 4.4: *Effects of Simulated Annealing on two functions. Colors represent temperatures. In both cases, as the temperature decreases the functions get concentrated around the global maxima.*

4.2 Simulated Annealing

The previous sections showed how Reversible Jump Markov Chain Monte Carlo can be used for a random walk over the space of reconstruction hypotheses. In the presented formulation, the algorithm generates a set of samples of the hypothesis space. It is not trivial to select the best hypothesis from that set. Instead, it is helpful to direct the RJMCMC sampling toward an optimum of the underlying distribution. This is done by introducing a temperature that tends toward 0. As it does so, less and less proposals get accepted. This is called Simulated Annealing.

Simulated Annealing simulates the physical process of heating up a system and then cooling it down in order to get to the desired state (Kirkpatrick et al. [1983], Cerny [1985]). Originally, the Metropolis algorithm (Metropolis et al. [1953]) was used as the basis for the optimization scheme and applied to the Traveling Salesman problem. The idea is to describe the condition of a system by an energy function that is added a temperature factor. This factor makes sure that at high temperatures even large state changes are allowed, while at low temperatures only small state changes are allowed. Thus, when starting with a high temperature and allowing large state changes in order to explore the state space and then iteratively cooling down the system and only allowing small state changes, a point will be reached, at a low temperature, where almost no state changes are accepted any more and the state merely fluctuates around some state. In the best case that state is the global optimum.

In Figure 4.4 an example is illustrated. Plotted on the left is the maximum of two Normal distributions with different means and variances. As such, this function has two maxima, one of which is merely a local maximum. The function is added a temperature exponent $^1/_t$ and exponentiated with five different values, namely 1, 1.5, 2, 2.5, and 3 corresponding to temperatures 1, 0.667, 0.5, 0.4, and 0.333, respectively. With a temperature of 1 the



(a) Temperature 0.333 (b) Temperature 0.2 (c) Temperature 0.1 (d) Temperature 0.01

Figure 4.5: *Development of the left function of Figure 4.4 for temperatures from* 0.333 *to* 0.010. *At first the local maximum is still visible, but it decreases more and more.*

distribution is unchanged, but if the temperature is decreased to $0.\overline{6}$, and thus the exponent is increased to 1.5, the distribution is "lowered," but more so at the local than at the global maximum. Also it gets more concentrated around the global maximum. This effect of concentrating around the global maximum increases with sinking temperatures. In the right part of the figure there are two global maxima, both of which are retained during annealing.

In the limit (i.e., $t \to 0$) this leads to a degenerate distribution with all its mass at the global optima. See Figure 4.5 for the development of the left function from Figure 4.4. At a temperature of 0.333 the local maximum is still visible, while at temperatures 0.2 and 0.1 it is almost gone. Jumping to a temperature of 0.01, one can see that the distribution is almost completely centered around the global maximum.

There are two parameters that influence the annealing process, namely when the system is cooled down and by what scheme it is cooled. For the first parameter, for example, one could cool in every iteration, in every n-th iteration, or irregularly after achieving some goal. For the second parameter the typically used implementation is geometric annealing. Others are, for example, logarithmic annealing and linear annealing. In geometric annealing the new temperature t_{i+1} is calculated by multiplying the current temperature t_i with a constant $c \in (0, 1)$:

$$t_{i+1} = c \cdot t_i.$$

The closer c is to 1 the slower the annealing takes place and the more iterations the algorithm needs. But on the other hand a slower annealing might be better to get out of local optima and to explore the hypothesis space better. These are all standard annealing schemes but, of course, one can also implement their own, arbitrary scheme. Hajek [1988] gives a further investigation on annealing schedules.

4.3 Combining RGG and RJMCMC

At several points in the previous sections connections between the general Markov Chain methods and the task of plant reconstruction were pointed out. This section serves as a summary of those connections and adds Relational Growth Grammar. The state space $\mathcal{S}$ is the set of valid reconstruction hypotheses. This set is defined by the grape cluster model of Chapter 3, "The Model." More precisely, the set $\mathcal{M}$ of RJMCMC models is defined by the possible combinations and arrangements of components of the biological model. For each component – tptwig, pedicel, etc. – there is a minimum and maximum allowed number. Combinations of these numbers and the arrangements of the components define the RJMCMC models. Taking into account the parameter values for each of the RJMCMC models, the state space can be divided into subspaces, where each subspace contains all possible parameter value allocations for a specific RJMCMC model. Each subspace – and therefore also the complete space $\mathcal{S}$ – is of infinite size, since the parameters allow values in a continuous range.

There are two ways to look at a single hypothesis. In an abstract view a hypothesis is represented by a vector containing the lengths and angles for the specific components. In the implementation view the current hypothesis is the current RGG graph, with the mapping presented in Table 3.1. To construct transition probabilities such that the distribution $\mathcal{P}(k,\, \theta_k \mid \mathcal{D})$ is the invariant distribution, a means to alter the current hypothesis is needed. For this the rewriting rules of the RGG grammar are used. They implement the state transitions and enable the random walk on the space of reconstruction hypotheses. However, not all generally possible transitions are useful. This is one key design aspect: the selection and design of jumps, such that the hypothesis space can be explored, unlikely regions are not visited too often, and the sampler does not get stuck in any region. Also there has to be a balance between jumps that propose only small changes in order to refine a hypothesis in detail and those jumps that propose large changes in order to get to other regions of the hypothesis space fast.

In order to direct the random walk toward the (global) optimum of the distribution $\mathcal{P}(\cdot)$, Simulated Annealing is included, see Section 4.2. The inclusion of Simulated Annealing into the RJMCMC algorithm is done by adding a temperature exponent, which results in the following updated acceptance probability:

$$\alpha = \frac{\mathcal{P}^{1/t_i}(k',\, \theta'_{k'} \mid \mathcal{D})}{\mathcal{P}^{1/t_i}(k,\, \theta_k \mid \mathcal{D})} \cdot \frac{j_{m'}(\theta'_{k'})}{j_m(\theta_k)} \cdot \frac{q_{m'}(\theta_k,\, u \mid \theta'_{k'},\, u')}{q_m(\theta'_{k'},\, u' \mid \theta_k,\, u)} \cdot \left| \frac{\partial(\theta'_{k'},\, u')}{\partial(\theta_k,\, u)} \right|$$

The exponent is added to the distribution $\mathcal{P}$, so in fact the sampling is not from the distribution $\mathcal{P}(\cdot)$ but from $\mathcal{P}_i(\cdot) = \mathcal{P}^{1/t_i}(\cdot)$, where i is the current iteration of the algorithm.

4.4 The Reconstruction Algorithm

Having introduced the methods and their connections, this section describes the actual reconstruction algorithm. Earlier results of this approach were published in Schöler and Steinhage [2012] and Steinhage et al. [2012]. The algorithm is depicted in Algorithm 4.3. It has a total of three inputs. The first one is $\mathcal{D}$, the set of input points as generated by the sensor. The second and third inputs, t_0 and T, are the start temperature and end temperature for the Simulated Annealing. The start temperature is used as initialization for the geometric annealing, the end temperature is used as a break condition in order not to use too many iterations. As described in Section 1.3, at first the visible plant components are detected by a data-driven procedure, represented by the method call in line 1. Then these components are used to derive a model-driven hypothesis by the method call in line 2. Both of these methods are problem-specific and therefore not discussed here. Details can be found in Chapter 5, "Applications of the Reconstruction Approach." The global hypothesis is then used as the initial configuration for the RJMCMC-based optimization. The following steps are repeated until the end temperature is reached, i.e., if the system is cooled down enough that convergence toward the optimum can be assumed. As usual with Reversible Jump Markov Chain Monte Carlo the loop itself consists of only a few steps, which will be discussed now.

At first in line 5 the current state of the random walk, i.e., the current reconstruction hypothesis, i.e., the current RGG graph is saved. Next, in line 6 the probabilities for the jumps are calculated. This is needed because not every jump is possible in every loop iteration due to possible constraints on their applicability. For example, if of one component there is already the maximum allowed number in the hypothesis, a jump to add another of this component would be assigned probability 0. Then, in line 7 one of the jumps is selected according to the distribution defined in the previous step. Remember that the jumps are the state transitions in the RJMCMC framework and are implemented as RGG rewriting rules. Therefore, after drawing the auxilliary vector u in line 8, the jump, i.e., the rewriting rule in order to create a proposal reconstruction hypothesis $(\theta'_{k'}, u')$ is executed in line 9. In the next step in line 10 the acceptance probability is calculated. This value is then used to determine whether or not to accept the proposal hypothesis in lines 11 through 16. It is accepted if α is greater than 0 and if a value a, sampled uniformly at random from (0, 1), satisfies $a \leq \alpha$. Otherwise the current hypothesis is also the new hypothesis and the previously saved RGG graph is restored. The second to last step is to update the temperature for the Simulated Annealing in line 17. In Algorithm 4.3 this is presented in a generic way, but as previously mentioned in Section 4.2 there are different ways to update the temperature. Here geometric cooling is used. Finally, lines 18 to 20 guarantee that further iterations are canceled if over the course of the last 1000 iterations less than ten jumps were accepted. In this case it is assumed that the current hypothesis is close enough to the desired result.

Algorithm 4.3 The Reconstruction Algorithm

```
Require: set of points $\mathcal{D} = \{(x_0, y_0, z_0), (x_1, y_1, z_1), \dots, (x_n, y_n, z_n)\}$
    $t_0$, start temperature
    $T$, end temperature

 1: $\mathcal{V} = \text{step}_1(\mathcal{D})$, set of visible components
 2: $(k, \theta_k)_0 = \text{step}_2(\mathcal{V})$, model-driven hypothesis
 3:
 4: while $t_i > T$ do
 5:     save_state()
 6:     $\mathcal{J} = [j_0, j_1, \dots, j_J]$: current jump probabilities
 7:     select jump index $m$ from $[0, 1, \dots, J]$ according to $\mathcal{J}$
 8:     $u \sim q_m(u \mid (k, \theta_k)_i)$
 9:     $(\theta'_{k'}, u') = j_m((\theta_k, u))$
10:     $\alpha = \frac{\mathcal{P}^{1/t_i}(k', \theta'_{k'}|\mathcal{D})}{\mathcal{P}^{1/t_i}(k, \theta_k|\mathcal{D})} \cdot \frac{j_{m'}(\theta'_{k'})}{j_m(\theta_k)} \cdot \frac{q_{m'}(\theta_k, u|\theta'_{k'}, u')}{q_m(\theta'_{k'}, u'|\theta_k, u)} \cdot \left|\frac{\partial(\theta'_{k'}, u')}{\partial(\theta_k, u)}\right|$
11:     if $\alpha > 0 \wedge a \sim U(0, 1) \leq \alpha$ then
12:         $(k, \theta_k)_{i+1} = (k', \theta'_{k'})$
13:     else
14:         $(k, \theta_k)_{i+1} = (k, \theta_k)_i$
15:         restore_state()
16:     end if
17:     $t_{i+1} = \text{sa\_update}(t_i)$
18:     if $i > 1000 \wedge$ accepted jumps in last 1000 iterations $< 10$ then
19:         break
20:     end if
21: end while
```

4.5 Summary

This chapter covered the details of the reconstruction and optimization algorithm. After stating the coarse behavior of the method, the theoretic background on Markov Chain Monte Carlo and Simulated Annealing were explained. This led to a description of the combination of the biological model and the methods of this chapter. The next chapter will show how this general formulation can be applied to leaves and three different development stages of grape clusters.

Chapter 5

Applications of the Reconstruction Approach

This chapter presents applications of the previously described reconstruction algorithm. As stated in Section 1.2 the focus of this work is on grapevine and in particular on grape clusters. The first section gives a detailed description of the application to grape clusters in development stage BBCH89 'ripe berries,' as this is the most challenging development stage. Due to the densitiy of the used cultivar (Riesling) the interior stem system is almost completely occluded by the berries. Each of the three reconstruction steps is explained in detail. In the second section an overview of further applications of the reconstruction approach to grape clusters in different development stages and vine leaves is given, which is based on student's theses that were developed in connection with this work.

Implementations were mostly done in the software environment GroIMP that specifies the programming language XL for implementing Relational Growth Grammars. Since XL is based on the Java programming language, plug-ins to GroIMP are written in Java and can be directly imported in any Relational Growth Grammar.

5.1 Reconstruction of Grape Clusters in BBCH89

The development stage BBCH89 'ripe berries' of Riesling is the most challenging one. Due to the cultivar's tendency to produce very dense grape clusters only the outer berries are observable, while the inner berries and the stem system are mostly occluded. Therefore the coarse-to-fine reconstruction strategy is interpreted as an outside-to-inside strategy as follows and detailed in the next three section.

The first reconstruction step for the data-driven hypothesis uses the sensor data as the basis for finding the visible plant components: berries and peduncle. Outer berries of the grape clusters are detected by a sphere detector. The inner berries, which are not represented in the sensor data are integrated into the hypothesis in the third reconstruction step. Grapevine berries can have very different appearances. Many cultivars have berries of ellipsoidal shape, but according to OIV [2009] other forms are, for example, cylindric, ovoid, horn-shaped, or finger-shaped. For the cultivar at hand, namely Riesling, the description says (oblate) globose. Therefore in this work spheres are used as an approximating representation of berries. For other cultivars a different representation might have to be used. In most cases also the peduncle is partially visible. At the moment the peduncle is inserted manually, but this could, for example, be replaced by a cylinder detector. There is no general rule where to find the peduncle. In some cases it is clearly visible, in other cases it is densely surrounded by berries. The following two reconstruction steps work on the results of the berry detector and the peduncle as anchor points and ignore the raw sensor data, since they are already explained through the detected berries and the peduncle.

The second reconstruction step for the model-driven hypothesis uses the detected berries as anchor points and generates a complete reconstruction hypothesis based on the model from Chapter 3, “The Model.” This is done in a bi-directional manner. First, the berries are grouped into berry groups and connected to pedicel origins. Second, a rachis is inserted. Third, pedicel origins and rachis are connected via terminal pedicel twigs. This yields a complete reconstruction hypothesis that is valid according to the biological model. However, this step ignores the distribution of berries and instead inserts, for example, a fixed number of berry groups. This grouping of berries alone is not obvious and from the distribution of outer berries no definite answer can be given. With the introduction of inner berries this becomes more obvious, but their insertion is also dependent on the inserted twigs and rachis. Therefore the third step optimizes the structure.

The third reconstruction step for the optimized hypothesis uses the reconstruction hypothesis from the second step as input and introduces local refinements for adapting the initial hypothesis to the actual distribution of berries. There are two categories for these refinements. One category increases or decreases the numbers of certain components by, for example, combining two twigs into one or adding a berry. The other category adjusts the describing parameters of the components, i.e., their lengths and angles. This yields the final reconstruction result that is the basis for phenotyping.

5.1.1 Step 1: Data-driven Hypothesis

As discussed in the previous section, berries are represented as spheres. For the detection of the spheres in the input point cloud, the method described in Jelden [2014], which is based on Schnabel et al. [2007], is used. See Figure 5.1 for an example. In its essence, the method

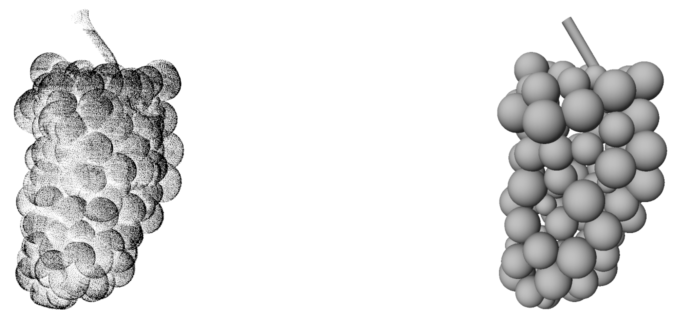


Figure 5.1: *The left image shows a reduced version of the input point cloud. The right image depicts the berries and the peduncle detected in the point cloud.*

is a variant of RANSAC (random sample consensus, Fischler and Bolles [1981]) for the detection of spheres. It is adapted to the detection of berries in that it is known which sphere radii can be expected. This yields a higher accuracy and higher performance, since not the whole point cloud has to be considered in every step. The following explanation imparts only the idea of the algorithm. At a high level, the algorithm has two steps: (1) Generate a set of sphere hypotheses. (2) Select the best hypotheses.

Step (1) iterates over the whole point cloud and generates for every point p_1 a number of hypotheses. For each hypothesis a second point p_2 is needed. Since it is known, how large a berry can be, p_2 is not selected from the whole point cloud but only from points in the allowed range. For both, p_1 and p_2, normals d_1 and d_2 are computed. Next, the shortest line c between the lines through d_1 and d_2 is calculated and the center point M of c is taken as the center for the sphere hypothesis. Last, the radius for the sphere hypothesis is the average of the distances from p_1 and p_2 to M. For step (2) the quality of a hypothesis is defined as the number of points of the point cloud, whose distance to the surface of the hypothesis is at most some ϵ divided by the area of the surface. Then the algorithm iteratively selects hypotheses by their quality.

Figure 5.1 shows an example for a BBCH89 grape cluster. On the left there is an input point cloud and on the right the detected spheres and peduncle. Most of the visible berries are found. Others are only visible to a very small degree or not at all. Those can not be detected.

5.1.2 Step 2: Model-driven Hypothesis

The general idea of the generation of the model-driven hypothesis is to take the detected outer berries and the peduncle and use them as anchor points for a model-driven bi-directional reconstruction. The basis for this reconstruction is the plant connectivity graph of grape

Algorithm 5.1 BBCH89 Model-driven Hypothesis

```
Require: set of Berries $\mathcal{V} = \{(p_0, r_0), (p_1, r_1), \ldots, (p_n, r_n)\}$, where $p_i$ = position of Berry
    $i$, and $r_i$ = radius of Berry $i$
    $P$ : the Peduncle
    $k$ : number of Pedicel_Origins
    $r$ : number of rachis levels
 1: $C = \{c_1, c_2, \ldots, c_k\}$ = find_pedicel_origin_positions($\mathcal{V}$, $k$)
 2: for $c_i \in C$ do
 3:     insert a Pedicel_Origin at position $c_i$
 4: end for
 5: for $b \in \mathcal{V}$ do
 6:     connect Berry $b$ to the closest Pedicel_Origin with a Pedicel
 7: end for
 8: insert_rachis($\mathcal{V}$, $P$, $r$), see Algorithm 5.2
 9: for Pedicel_Origin $p$ do
10:     connect $p$ to the closest Rachis_Node with a Twig
11: end for
```

clusters as shown in Figure 3.3, as this specifies which components are allowed to branch from which components.

The procedure is summarized in Algorithm 5.1. Keep in mind that RGG modules are capitalized. The algorithm's inputs are the set $\mathcal{V}$ of detected Berries, the detected Peduncle, P, the desired number of initial Pedicel_Origins, k, and the number of rachis levels, r. The latter parameter is used for Algorithm 5.2 and will be described there. Figure 5.2 summarizes the whole process visually. Bear in mind that in the figure the spheres are reduced in size for visualization, while computations were done on the original size. In line 1 the algorithm computes a set of locations for Pedicel_Origins by using the k-means algorithm (MacQueen [1967]), see Figure 5.2(a). This algorithm collects nearby Berries into clusters on the basis of the Berries' centers and assigns the position of the Pedicel_Origin as the cluster center. The parameter k is set to 20, the average number of berry groups observed in the available exemplars. Due to the sizes of the berries, mostly the cluster centers are located within a Berry, which is not allowed. The following method is used to solve that. A regular voxel grid with an edge length of 1 mm for each voxel is put around the grape cluster and for every voxel it is checked whether it is free or occluded. A voxel is considered occluded, when a ray from the center of mass of the grape cluster through the voxel's center intersects a Berry after leaving the voxel. Voxels whose centers are located within a Berry are not considered occluded. Since in general a Pedicel_Origin is occluded by its Berries, this is now used to move those preliminary locations of Pedicel_Origins that are located within a Berry to the closest occluded voxel center, because pedicels always point in an inward direction.

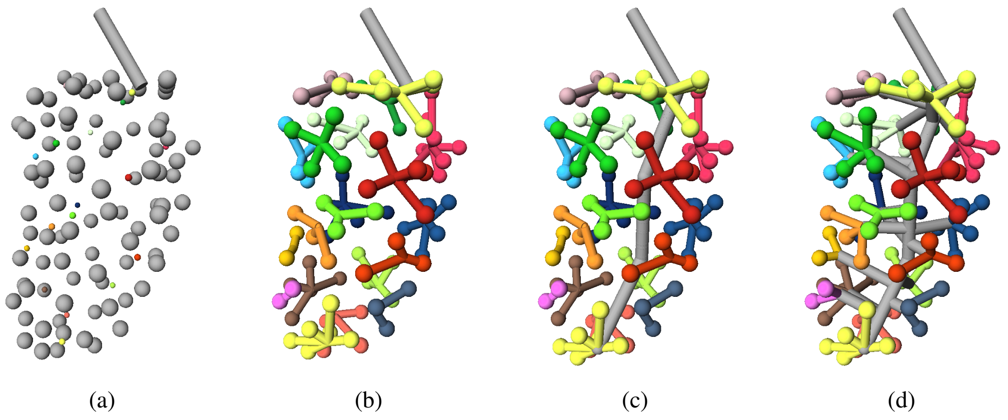


Figure 5.2: Generation of the model-driven hypothesis in four steps. The Berries are reduced to a smaller uniform size for the visualization, computations were done on the original sizes. Colors encode berry groups. (a) Cluster centers computed by k-means and moved inward. (b) Berries grouped together with their closest Pedicel_Origin. (c) Insertion of the rachis. (d) Inserting Tptwigs between rachis and berry groups.

If the Pedicel_Origin were not occluded, then the sensor would have been able to register that part and there would be points in the point cloud that could be used for a detection of the Pedicel_Origin. Voxels that are not occluded need not be considered here. The same voxel grid is later used in an RJMCMC jump that inserts new Berries to the hypothesis. In lines 2 to 4 of the algorithm, at those k positions Pedicel_Origins are actually inserted. In lines 5 through 7 every Berry is connected to its closest Pedicel_Origin by inserting a Pedicel between the two. In line 8 a rachis is inserted, detailed in Algorithm 5.2, which consists of Rachis modules and Rachis_Node modules. In lines 9 to 11 every Pedicel_Origin is connected to the closest, not yet used Rachis_Node by inserting a Tptwig between the two. The result is a reconstructed grape cluster that is valid according to the model and that is later used as the basis for the local refinements.

Algorithm 5.2 explains how the rachis for the model-driven hypothesis is generated. Figure 5.3 gives an overview of the process. The algorithm has three inputs: The Berries, the Peduncle, and the number of desired rachis levels. The algorithm makes use of the fact that during scanning the grape clusters were mounted in such a way that they hung down from the cane, because this downward direction corresponds to the negative z-direction of GroIMP's coordinate frame. Therefore the topmost point – the point where the cluster was connected to the cane – has the largest z-value. Since the biological model states that the rachis is approximately located in the middle of the cluster, i.e., two twigs that branch close to each other from the rachis have approximately the same length, the idea is to take equidistant, parallel slices of the grape cluster and for each slice find the central point. For this the algorithm takes the z-value of the end point of the Peduncle and the z-value of the position of the Pedicel_Origin that has the largest distance to the Peduncle, since these are

Algorithm 5.2 BBCH89 Model-driven Hypothesis: Insert rachis

```
Require: set of Berries $\mathcal{V} = \{(p_0, r_0), (p_1, r_1), \ldots, (p_n, r_n)\}$, where $p_i$ = position of Berry
    $i$, and $r_i$ = radius of Berry $i$
    $P$ : the Peduncle
    $r$ : number of rachis levels
 1: $C$ = list containing end point of Peduncle
 2: for $r$ equidistant planes between the Peduncle and the furthest Pedicel_Origin relative to
    the Peduncle do
 3:     $\mathcal{K}$ = set of intersection circles of intersections of the plane with the Berries
 4:     $\mathcal{K}'$ = set of centers of the intersection circles in $\mathcal{K}$
 5:     $c$ = center of mass of convex polygon of the points in $\mathcal{K}'$
 6:     add $c$ to the end of $C$
 7: end for
 8: add center of furthest Pedicel_Origin to the end of $C$
 9: for $c \in C$ do
10:     insert a Rachis from the previous center in $C$ (or the Peduncle) to the current center
        (or the furthest Pedicel_Origin)
11: end for
12: for all Pedicel_Origin $po$ do
13:     Find point $p$ on the Rachis modules that has the shortest distance to $po$
14:     Replace the according Rachis by two new ones with the same total length as the
        original and add a Rachis_Node in between
15: end for
```

the end points of the rachis. Along the z-axis the parameter r is used to insert r equidistant planes with the z-axis as their normal. For each plane the intersections with the Berries are computed (Figure 5.3(b)). Each intersection is either a circle or a single point (a degenerate circle). From each circle the center is taken and then the set of centers for that plane is used to generate their convex polygon (Figure 5.3(c)) and, finally, the polygon's center of mass (Figure 5.3(d)). The final result is an ordered set of points with decreasing z-values. In the next step Rachis modules are inserted between each two closest of those points, including the end point of the Peduncle and the furthest Pedicel_Origin, resulting in the rachis (Figure 5.3(e)). In the last step, for each Pedicel_Origin the point p on the Rachis modules is searched that has the lowest distance to the Pedicel_Origin. The according Rachis module is replaced by two new Rachis modules whose total length is the same as the length of the previous one. The two new Rachis meet at p and a Rachis_Node is inserted at p. This is needed for the last step of Algorithm 5.1, the insertion of Twigs between Rachis and Pedicel_Origin.

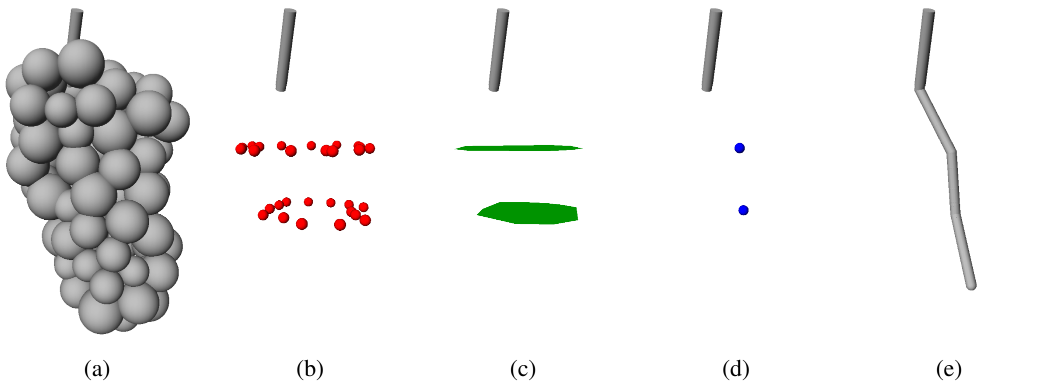


Figure 5.3: *Calculation of the rachis. For r equidistant planes along the z-axis their intersections with the Berries are calculated (b). These are then used to determine a convex polygon (c), the centers of mass of those polygons (d) and, finally, the rachis (e).*

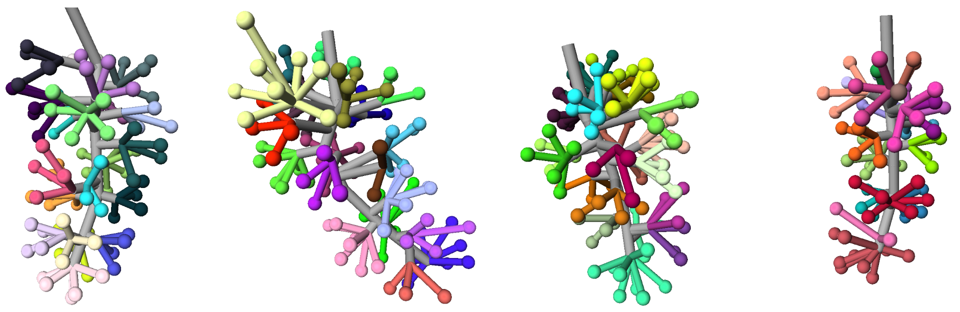


Figure 5.4: *Model-driven hypotheses of four inputs. Colors encode berry groups. The Berries are reduced to a smaller uniform size for the visualization. This is the basis for the optimization process.*

Figure 5.4 shows the model-driven hypotheses of four grape clusters. They can be used for a coarse phenotyping of, for example, the form (cylindrical, conical, and funnel shaped; OIV [2009]). The next reconstruction step are the local refinements. Before giving an overview of the RJMCMC jumps, the next section introduces the acceptance probability that is used to accept or reject a jump, because the jump descriptions will refer to the acceptance probability.

5.1.3 Step 3: Optimization

This section gives an overview of the local refinements for the optimization of the initial hypothesis that was generated in the last two sections. First, the RJMCMC acceptance probability is introduced, which is used to evaluate a proposal hypothesis. Second, the list of

RJMCMC jumps is given, which makes references to the acceptance probability. Third, one of the jumps is explained in detail. Last, some results are shown.

Acceptance Probability

Every jump execution, i.e., every newly proposed reconstruction hypothesis has to be tested for acceptance. As stated in Section 4.3 the formula for this is

$$\alpha = \frac{\mathcal{P}^{1/t_i}(k', \theta'_{k'} \mid \mathcal{D})}{\mathcal{P}^{1/t_i}(k, \theta_k \mid \mathcal{D})} \cdot \frac{j_{m'}(\theta'_{k'})}{j_m(\theta_k)} \cdot \frac{q_{m'}(\theta_k, u \mid \theta'_{k'}, u')}{q_m(\theta'_{k'}, u' \mid \theta_k, u)} \cdot \left| \frac{\partial(\theta'_{k'}, u')}{\partial(\theta_k, u)} \right| .$$

This section gives insight into the individual terms, starting with the distribution $\mathcal{P}(\cdot)$. As stated in Section 4.1.3 this is formulated as the product of three terms:

$$\mathcal{P}(k, \theta_k \mid \mathcal{D}) \propto \mathcal{L}(\mathcal{D} \mid k, \theta_k) \cdot p(\theta_k \mid k) \cdot p(k)$$

The likelihood $\mathcal{L}(\mathcal{D} \mid k, \theta_k)$ describes how likely the data are, given the current hypothesis. In the presented case of ripe and dense grape clusters, the sensor data are completely described by the outer berries and the peduncle – which remain fixed –, therefore the likelihood $\mathcal{L}$ can be assumed as 1 and be left out. In other cases, where parts of the stem system are visible, a likelihood function has to be introduced, for example, as the distance of points of the input point cloud to the reconstruction hypothesis. The model prior $p(k)$ gives a marginal probability for a specific RJMCMC model regardless of data or selected parameter values. An RJMCMC model is defined by a specific combination of grape cluster components. Therefore the model prior has to evaluate the general structure of the hypothesis, regardless of the actual geometric characteristic. The parameter prior $p(\theta_k \mid k)$ states how probable the chosen parameter values are for the selected model. Each model is described by a specific set of parameters, for example, the lengths of components. Each combination of values for the parameters has a different probability, reflected in the parameter prior. The model and parameter priors each consist of several subcomponents. The model prior has the following subcomponents:

- M1 The total number of berries (outer berries and newly inserted ones).
- M2 The number of nodes on the rachis.
- M3 The total number of twigs branching from the rachis.
- M4 The number of branchtwigs.
- M5 The number of subtwigs.
- M6 The number of tptwigs.
- M7 The number of pedicel origins.

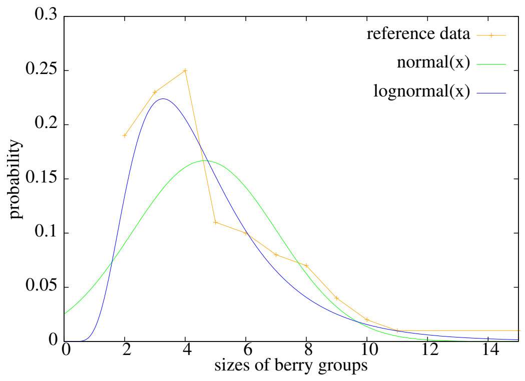


Figure 5.5: *Normal distribution compared to Lognormal distribution for parameter sub prior (P7) sizes of berry groups. The Lognormal distribution (blue) is a better approximation to the real data (orange) than the Normal distribution (green).*

The parameter prior has the following subcomponents:

P1 The angles of twigs (all twig types).
P2 The lengths of branchtwigs.
P3 The lengths of subtwigs.
P4 The lengths of tptwigs.
P5 The lengths of pedicels.
P6 The angles of pedicels.
P7 The sizes of the berry groups.

The final priors are the products of their according subcomponents. This, of course, assumes independence of the subcomponents. Though this assumption might be false in general, it reduces computational efforts and is precise enough for a first implementation. In Chapter 6, "Evaluation," this subject is picked up again. While some of those priors are modeled as Normal distributions, in most cases the Lognormal distribution gives a more natural approximation. See, for example, in Figure 5.5 how the Lognormal distribution approximates the distribution of berry group sizes better than the Normal distribution. The berry group size is the number of berries in a berry group. The x-axis holds the different berry group sizes (observed sizes range from 2 to 15) and the y-axis holds the relative frequency of each size. The orange line depicts the actual distribution of observed sizes. In green one can see the estimated Normal distribution and in blue the estimated Lognormal distribution. The most often occurring values for berry group sizes are 2, 3, and 4 (all other values are much lower), yet the mean of the Normal is at approximately 4.6486, while the mode of the Lognormal is at approximately 3.2683, and therefore between the most often occurring values. Also it is obvious from visual inspection already that the distribution of the observed berry group sizes is not symmetric, which is reflected in the Lognormal as well. The Normal distribution

Prior	Distribution	μ	σ
M1 Number of berries	Lognormal	4.5807	0.1115
M2 Number of rachis nodes	Lognormal	2.3678	0.2452
M3 Number of twigs	Lognormal	2.5547	0.1431
M4 Number of branchtwigs	Lognormal	1.2509	0.2447
M5 Number of subtwigs	Lognormal	2.0207	0.4083
M6 Number of tptwigs	Lognormal	2.1811	0.2760
M7 Number of berry groups	Lognormal	3.0829	0.1852
P1 Angles of twigs	Normal	84.6931	12.5842
P2 Lengths of branchtwigs	Lognormal	2.8178	0.3134
P3 Lengths of subtwigs	Lognormal	0.9858	0.3629
P4 Lengths of tptwigs	Lognormal	1.4986	0.5384
P5 Lengths of pedicels	Lognormal	2.1647	0.1998
P6 Angles of pedicels	Normal	56.9156	26.2828
P7 Sizes of berry groups	Lognormal	1.4191	0.4846

Table 5.1: *Mapping of subcomponents to the used distribution. In both cases the values for μ and σ are given, but keep in mind that those are interpreted differently in the two distributions.*

even allows negative values, which is contradictory to the fact that there cannot be a negative number of berries in a berry group.

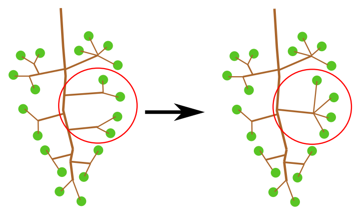
Table 5.1 gives an overview of the priors, their distributions and the distributions' parameter values. Not every subcomponent is computed in every jump, only those that may have changed during jump execution. This is done to save computation time and because those unchanged priors would not add any information.

The jump probabilities are computed rather simple. In every iteration the number of available jumps is computed. It might happen that a jump is unvavailable in an iteration. For example, there is a jump that combines two tptwigs into a single new one, if certain criteria are met. If they are not met, the jump is not available. Then, to every available jump the probability $\frac{1}{\#available\_jumps}$ is assigned. The proposals are different for every jump and cannot be explained in a general manner. An example is given after the overview of jumps. The determinant of the Jacobian matrix is always exactly 1, due to permutations that leave exactly one 1 in every row and column of the Jacobian matrix. Again, an example is given after the overview of jumps.

Jumps: Overview

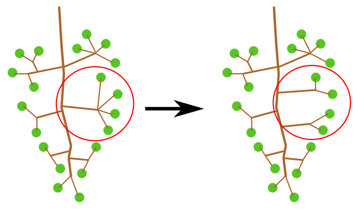
To recapitulate, the grape cluster model from Chapter 3, “The Model,” defines the state space for the reconstruction algorithm. Starting at the initial hypothesis, the RJMCMC jumps have to make certain that it is possible to visit the whole hypothesis space. This means that there have to be model-changing jumps and parameter-changing jumps. Model-changing jumps either reduce or expand the current hypothesis by deleting or adding components, thus changing the dimension of the describing vector θ_k, or they reassign a grape cluster component to a different parent component, thus preserving the dimension of θ_k but restructuring the parameters. An example for a jump that increases the dimension of the vector is ’add berry,’ which adds a new berry in the occluded space of the grape cluster. An example for a dimension-preserving jump is ’reassign berry,’ which assigns a berry to a different berry group. In contrast, parameter-changing jumps simply take a subset of the parameter vector θ_k and change their values. For example, in the jump ’move subtwig root,’ which moves the origin of a subtwig to a different position on the according branchtwig.

The selection of jumps used in this work defines for the components berry, subtwig, branchtwig, and tptwig jumps for increasing or decreasing their number. Therefore every characteristic of the biological model can be explored. For the components subtwig, branchtwig, and tptwig there are jumps to move the root and end positions, which are used to refine a hypothesis. In the current implementation the outer berries remain fixed, pedicel root positions are moved indirectly in jumps of other components, and the rachis is fixed. In the following list first the model-changing jumps are presented, then the parameter-changing jumps. For every jump a short description is given and the subcomponents are named that need to be computed for that jump. Each subcomponent is affected in at least two jumps. While this is a rather informal overview of the jumps, supplemented by example sketches, afterwards the reader will find a detailed description of one of the jumps. Again, keep in mind that RGG modules are capitalized. All numbers that are mentioned in the following are encoded in the biological model and were determined in the parameter value estimation.

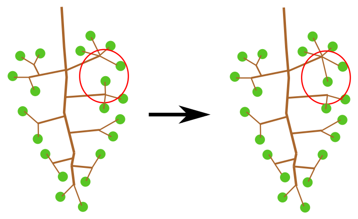


Combine Tptwigs: This jump takes two Tptwigs whose origins on the rachis are at most 1 cm apart and have in total less than the maximum allowed number of attached Berries per berry group (10). They are replaced with a single new Tptwig that has its root randomly located around the middle of the old two origins with a maximum distance of 0.5 cm to that middle point. The Berries that were connected to the old Tptwigs are now all connected to the new Tptwig by new Pedicels and the Tptwig ends at the center of mass of the end positions of the old Tptwigs. The result is again a Tptwig branching from the rachis. Since the rest remains unchanged, a valid hypothesis was generated. The reverse jump is 'Split Tptwig.' The red ellipses in the sketch mark the areas of change. Affected subcomponents: M2, M3, M6, M7, P1, P4, P5, P6, P7.

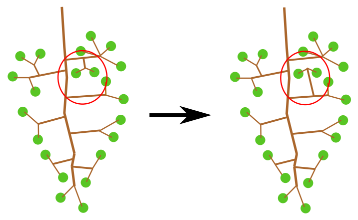
Split Tptwig: A single Tptwig is taken that has at least twice the minimum number of Berries per berry group (2) attached to it. It is then replaced with two new Tptwigs whose root and end points are sampled at random. The root points are at most 0.5 cm away from the old root point. The end points are sampled on the surface of a sphere with random radius. The Berries are distributed among the two Tptwigs randomly under the condition that in the end each of the new Tptwigs holds at least the minimum number of Berries per berry group (2). The result are two new Tptwigs branching from the rachis. Since the rest remains unchanged, a valid hypothesis was generated. The reverse jump is 'Combine Tptwigs.' The red ellipses in the sketch mark the areas of change. Affected subcomponents: M2, M3, M6, M7, P1, P4, P5, P6, P7.



Reassign Berry: When this jump is executed, a Berry is chosen at random from a berry group that has more than the minimum number of Berries per berry group (2). Next, another berry group is chosen randomly where less than the maximum number of Berries per berry group (10) are attached and that satisfies certain distance criteria with respect to the minimum (0.5 cm) and maximum (1.5 cm) pedicel length. Last, the randomly chosen Berry is attached to the randomly chosen berry group by a new Pedicel. Here a Berry is attached to another Pedicel_Origin and the rest is kept unchanged, therefore a valid hypothesis was generated. The reverse jump is the jump itself. The red ellipses in the sketch mark the areas of change. Affected subcomponents: P5, P6, P7.

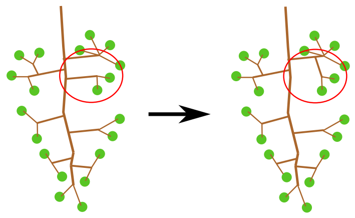


Reassign Subtwig: A Subtwig and an arbitrary Twig (Tptwig or Branchtwig) are chosen randomly under the condition that the distance of the Twigs' root points on the rachis to the root of the rachis are at most half the length of the rachis, since Branchtwigs are only allowed in the upper half of the grape cluster. Then the randomly chosen Subtwig is deleted and replaced by a new Subtwig between the old Subtwig's end position and a randomly selected position on the randomly selected Twig. Here one Subtwig is attached to another Twig and the rest is

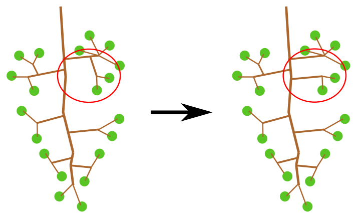


kept unchanged, therefore a valid hypothesis was generated. The reverse jump is the jump itself. The red ellipses in the sketch mark the areas of change. Affected subcomponents: M4, M6, P1, P2, P3, P4.

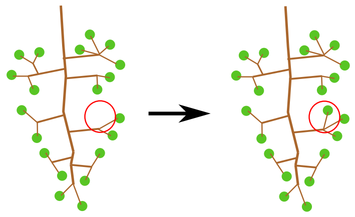
Attach Tptwig as Subtwig: This jump selects a Tptwig and another arbitrary Twig (Tptwig or Branchtwig) at random under the condition that they are located in the upper half of the rachis, i.e., that the distances of their origins on the rachis to the root of the rachis are at most half the length of the rachis. Then the Tptwig is deleted and replaced by a Subtwig between the Tptwig's end position and a randomly chosen position on the randomly chosen Twig. Here one Tptwig is deleted, replaced with a Subtwig, and the rest is kept unchanged. This results in a valid hypothesis. The reverse jump is 'Detach Subtwig as Tptwig.' The red ellipses in the sketch mark the areas of change. Affected subcomponents: M2, M3, M4, M5, M6, P1, P2, P3, P4.



Detach Subtwig as Tptwig: Here a Subtwig is chosen at random as well as a position in the upper half of the rachis such that the distance on the rachis from that point to the root of the rachis is at most half the length of the rachis. Then the Subtwig is deleted and replaced by a new Tptwig that starts at the randomly chosen position on the rachis and ends at the former Subtwig's end position. Here a Subtwig is deleted and a new Tptwig is inserted, the rest is unchanged. This results in a valid hypothesis. The reverse jump is 'Attach Tptwig as Subtwig.' The red ellipses in the sketch mark the areas of change. Affected subcomponents: M2, M3, M4, M5, M6, P1, P2, P3, P4.

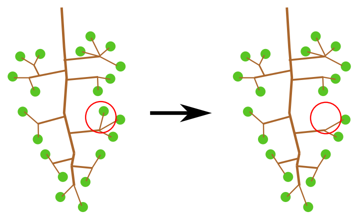


Add Berry: Before a new Berry can be added to the structure, possible places where a Berry can be added must be determined. For this, the voxel grid presented in Section 5.1.2 is used to calculate the occluded voxels. Then one of those occluded voxels is selected randomly and the center of that voxel is considered the center of the new Berry. Next, a Pedicel_Origin is drawn randomly from those that hold less than the maximum number of Berries per berry group (10). The Berry radius is drawn uniformly at random from the interval (0.38, 0.85). Additionally the new Berry is marked as such. Here a new Berry is added, which results in a valid hypothesis.

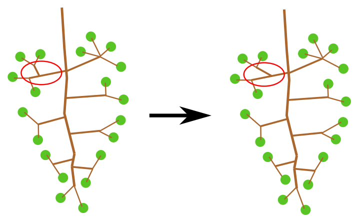


The reverse jump is 'Remove Berry.' The red ellipses in the sketch mark the areas of change. Affected subcomponents: M1, P5, P6, P7.

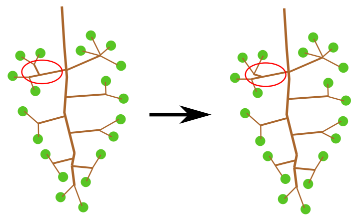
Remove Berry: For removing, one of the Berries that are marked as new Berries in the jump 'Add Berry' is selected and removed from the structure along with its Pedicel, given that at least the minimum number of Berries per berry group (2) remain at the according Pedicel_Origin. Here a Berry is removed that did not belong to the original set of Berries, while the rest stays unchanged. Therefore a valid hypothesis is generated. The reverse jump is 'add berry.' The red ellipses in the sketch mark the areas of change. Affected subcomponents: M1, P5, P6, P7.



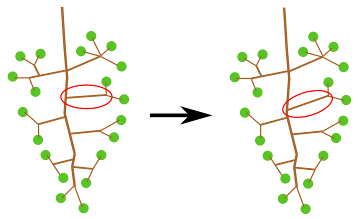
Move Subtwig root: In this jump a Subtwig is chosen at random. The new root position for that Subtwig is chosen randomly on the two Branchtwig modules between which the Subtwig branches. Since the model is not changed here and there are no restrictions on the angles of branches, a valid hypothesis is generated. The reverse jump is the jump itself. The red ellipses in the sketch mark the areas of change. Affected subcomponents: P1, P3.

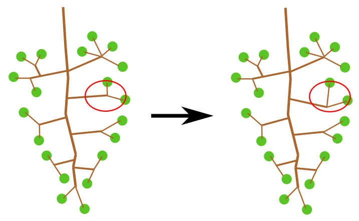


Move Subtwig end: Here again a Subtwig is chosen at random and the new end position of the Subtwig is determined by coordinate-wise sampling from a Normal distribution around the end position of the old Subtwig. Since the model is not changed here and there are no restrictions on the angles of branches, a valid hypothesis is generated. The reverse jump is the jump itself. The red ellipses in the sketch mark the areas of change. Affected subcomponents: P1, P3, P5, P6.

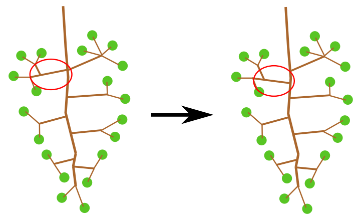


Move Tptwig root: This jump chooses a Tptwig at random and samples the new root position on the two Rachis modules between which the Tptwig branches. Since the model is not changed here and there are no restrictions on the angles of branches, a valid hypothesis is generated. The reverse jump is the jump itself. The red ellipses in the sketch mark the areas of change. Affected subcomponents: P1, P4.

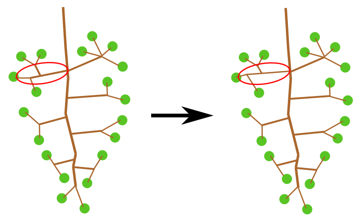




Move Tptwig end: For this jump a Tptwig is selected randomly and the new end position of the Tptwig is determined by coordinate-wise sampling from a Normal distribution around the end position of the old Tptwig. Since the model is not changed here and there are no restrictions on the angles of branches, a valid hypothesis is generated. The reverse jump is the jump itself. The red ellipses in the sketch mark the areas of change. Affected subcomponents: P1, P4, P5, P6.



Move Branchtwig root: When this jump is chosen a Branchtwig is selected at random and the new root position is selected randomly on the two Rachis modules between which the Branchtwig branches. The roots of the Subtwigs that branch from that Branchtwig are also adjusted accordingly. Since the model is not changed here and there are no restrictions on the angles of branches, a valid hypothesis is generated. The reverse jump is the jump itself. The red ellipses in the sketch mark the areas of change. Affected subcomponents: P1, P2, P3:



Move Branchtwig end: A Branchtwig is selected at random and the new end position is determined by sampling from a Normal distribution around the end position of the old Branchtwig. Since the model is not changed here and there are no restrictions on the angles of branches, a valid hypothesis is generated. The reverse jump is the jump itself. The red ellipses in the sketch mark the areas of change. Affected subcomponents: P1, P2, P3, P5, P6.

Details of Jump 'Combine Tptwigs'

In this section one of the jumps is described in detail, namely 'Combine Tptwigs,' where two Tptwigs are merged into a single new Tptwig. Keep in mind that in the following code any special cases at the beginning and at the end of the rachis are left out. Also not every line of the code is repeated here for the sake of brevity. The code is written in the language XL. It is based on the Java programming language, which was specifically designed for implementing Relational Growth Grammars and is inherent in the software platform GroIMP. At first suitable pairs of Tptwigs have to be found – assuming, for the sake of this description, that there actually is at least one such pair. In XL this looks as follows:

```
01  Rachis_Node [RU RL t1:Tptwig po1:Pedicel_Origin],
02  Rachis_Node [RU RL t2:Tptwig po2:Pedicel_Origin],
03  (
04    length_before(t1) <= length_before(t2)
05    && (count((* po1 [RU] *)) + count((* po2 [RU] *))
06       <= BERRYGROUP_SIZE_MAX)
07    && length_before(t2) - length_before(t1)
08       <= MAX_DIST_ON_RACHIS_BETWEEN_TPTWIGS
09  )
10  ::>
11  {
12    t1_ids.add(t1.getId());
13    t2_ids.add(t2.getId());
14  }
```

Lines 01 and 02 query all pairs of subgraphs – separated by the comma – that start at a Rachis_Node module and have a branch edge – symbolized by the bracket characters [, and] – to a successor-edge-chain – indirectly symbolized by the spaces – of module types RU, RL, Tptwig, and Pedicel_Origin. This formulation queries two subgraphs within one rewriting rule. As mentioned in Section 3.1, this is one advantage of RGG over L-System. The two Tptwig modules and the two Pedicel_Origin modules have been assigned identifiers for later reference. The part in the brackets describes a terminal pedicel twig. The code in the parentheses from lines 03 to 09 defines the further conditions such a pair of terminal pedicel twigs has to satisfy. The first condition in line 04 describes that Tptwig t1 has to be closer to the Peduncle than t2. The method length_before() calculates the length on the rachis from the branching point of a Tptwig (or any other module) to the Peduncle (not including the Peduncle). If it was not checked that t1 is above t2 this XL query would also yield the redundant result where t2 is above t1. Lines 05 and 06 make sure that the Tptwigs hold in total at most the maximum number of berries that are allowed in a single berry group. Finally, lines 07 and 08 ensure that the Tptwigs are not too far apart on the rachis. In the code execution part, lines 11 to 14, the module ids of the Tptwigs are stored in two separate lists t1_ids and t2_ids. From these Tptwig pairs one has to be selected. This is done by selecting uniformly at random one of the pairs with ids t1_id and t2_id. In the next step the actual rewriting rule is started, where at first the two Tptwigs from before are selected as well as a Rachis module that will be split for the new Tptwig. The selection code looks like this:

```
15  RU RL t1:Tptwig po1:Pedicel_Origin,
16  RU RL t2:Tptwig po2:Pedicel_Origin,
17  rl:RL r:Rachis rn:Rachis_Node,
18  (
19    t1.getId() == t1_id
20    && t2.getId() == t2_id
21    && length_before(rl) <= length_before(t1) + distance
22    && length_before(rn) >= length_before(t1) + distance
23  )
```

Lines 15 and 16 again select two subgraphs that represent terminal pedicel twigs. Line 17 selects a successor-edge-chain of modules RL, Rachis, and Rachis_Node. Lines 18 through 23 again state a condition to the application of the rule. At first, lines 19 and 20 make certain that of all the possible matches to the query of lines 15 to 17 only the ones get selected where t1 and t2 have the ids that were sampled before. Furthermore, lines 21 and 22 ensure that the correct Rachis is selected. The value stored in the variable distance is the distance from the center point between the branching points of t1 and t2 on the rachis to the point on the rachis where the new tptwig is to be installed. The selection condition makes certain that there is only one combination of the named subgraphs to match the query. Now comes the actual graph rewriting with the following piece of code.

```
24  cut,
25  cut,
26  rl
27  Rachis(length1, RACHIS_RADIUS, RACHIS_RADIUS)
28  Rachis_Node(RACHIS_RADIUS)
29  [
30    RU(rp.yaw) RL(rp.pitch)
31    Tptwig(rp.length, TWIG_RADIUS, TWIG_RADIUS)
32    Pedicel_Origin(TWIG_RADIUS)
33    for(int i = 0; i < berries.size(); ++i)
34    (
35      {
36        Berry b = berries.get(i);
37        RotationParameters pedicel_rp
38        = new RotationParameters(twig_rot, po_center, location(b));
39      }
40      [
41        RU(pedicel_rp.yaw)
42        RL(pedicel_rp.pitch)
43        Pedicel(pedicel_rp.length, PEDICEL_RADIUS, PEDICEL_RADIUS)
44        Berry(b[radius], b[original])
45      ]
46    )
47  ]
48  RU RL
49  Rachis(length2, RACHIS_RADIUS, RACHIS_RADIUS)
50  rn;
```

Lines 24 and 25 delete the old terminal pedicel twigs from the graph. In lines 26 through 50 the Rachis selected in line 17 is split and the new terminal pedicel twig is inserted. This procedure starts with reinserting the RL node with identifier rl in line 26. Next comes the first of the two new Rachis modules in line 27 followed by a Rachis_Node in line 28. Line 29 now introduces a new branch where at first rotation nodes RU and RL are inserted (with precomputed angles) in line 30, followed by a new Tptwig and Pedicel_Origin in lines 31 and 32. The loop in lines 33 to 46 inserts the berries of the old Tptwigs – which were previously stored in the list called 'berries' – at the new Pedicel_Origin. Lines 48 and 49 insert new rotation nodes (with angle 0) and the second of the two new Rachis nodes, while in line 57 the Rachis_Node from before gets reinserted. As usual in RJMCMC, once the new hypothesis is created, it has to be evaluated and then decided to accept it as the new current hypothesis or reject it. This is done in the following lines of code.

```
51  double j_combine = jump_probabilities[JUMP_COMBINE_TPTWIGS];
52  double q_combine = 1d /
53      count((* rn1:Rachis_Node [RU RL t1:Twig po1:Pedicel_Origin],
54               rn2:Rachis_Node [RU RL t2:Twig po2:Pedicel_Origin],
55               (
56                 length_before(t1) <= length_before(t2)
57                 && (count((* po1 [RU] *)) + count((* po2 [RU] *))
58                    <= BERRYGROUP_SIZE_MAX)
59                 && length_before(rn2) - length_before(rn1)
60                    <= MAX_DIST_ON_RACHIS_BETWEEN_TPTWIGS
61               )
62          *));
63  derive();
64  double j_split = get_jump_probabilities()[JUMP_SPLIT_TPTWIG];
65  double q_split = 1d /
66      count((* Rachis_Node [RU RL t:Twig po:Pedicel_Origin],
67            (count((* po [RU] *))
68              >= 2 * Parameters.BERRYGROUP_SIZE_MIN)
69          *));
70
71  double p_fraction =
72       (new_p_num_nodes / current_p_num_nodes)
```

```
73     * (new_p_num_twigs / current_p_num_twigs)
74     * (new_p_num_tptwigs / current_p_num_tptwigs)
75     * (new_p_num_berrygroups / current_p_num_berrygroups)
76     * (new_p_twig_angles / current_p_twig_angles)
77     * (new_p_tptwig_lengths / current_p_tptwig_lengths)
78     * (new_p_pedicel_lengths / current_p_pedicel_lengths)
79     * (new_p_pedicel_angles / current_p_pedicel_angles)
80     * (new_p_berrygroups / current_p_berrygroups)
81     * (new_p_intersection / current_p_intersection);
82  double j_fraction = j_split / j_combine;
83  double q_fraction = q_split / q_combine;
84
85  double acceptance_probability =
86      Math.pow(p_fraction, 1d / current_temperature)
87        * j_fraction
88        * q_fraction;
89  accept_reject(acceptance_probability);
```

Before discussing the individual terms of the acceptance probability, a specialty of XL / GroIMP has to be mentioned. In line 63 the method derive() executes the rewriting rule as it was described in the previous paragraphs. Without calling that method, the rewriting rule would only be executed after the end of the overall jump method, i.e., after line 89 of the above code block. As a result the proposed hypothesis would not have been generated and the results for the acceptance probability would be invalid. The first part of the acceptance probability is the fraction

$$\frac{\mathcal{P}^{1/t_i}(k', \theta'_{k'} \mid \mathcal{D})}{\mathcal{P}^{1/t_i}(k, \theta_k \mid \mathcal{D})}.$$

This is computed in lines 71 to 81. The values with the prefix new_p describe the proposal hypothesis, while the terms with the prefix current_p describe the current hypothesis. The latter ones are calculated separately before line 63 of the above code block. The calculations are left out here for the sake of brevity. An example is current_p_berrygroups for the sizes of berry groups:

$$\text{current\_p\_berrygroups} = \prod_{i=1}^{n} \ln\mathcal{N}(\text{children}(po_i); \mu, \sigma)$$

Here $\ln\mathcal{N}()$ is the Lognormal distribution, n is the number of berry groups, and children(po_i) counts the number of children of Pedicel_Origin po_i in the RGG graph, i.e., the number of Berries attached to po_i. The other subcomponents are calculated similarly. The new_p terms are calculated between lines 63 and 64. In line 81 a term is calculated that penalizes unallowed intersections of components. For example, it is allowed for a Pedicel to intersect the Berry that is attached to it, but it is not allowed that a Pedicel intersects a Rachis module. The unallowed intersections are penalized with a factor γ^n, where $\gamma \in (0, 1)$ and n the number of unallowed intersections. Having calculated the individual parts, the p_fraction can be determined. The reason for splitting this into several factors is that if the denominator and the numerator each were computed at once, there could easily be zero values in either or both and therefore invalid results. For the calculation of

$$\frac{j_{m'}(\theta'_{k'})}{j_m(\theta_k)},$$

line 51 computes $j_m(\theta_k)$, the probability of the selected jump, and line 64 computes $j_{m'}(\theta'_{k'})$, the probability to select the reverse jump when being at the proposed hypothesis. The fraction itself is constructed in line 82. The factors for the fraction

$$\frac{q_{m'}(\theta_k, u \mid \theta'_{k'}, u')}{q_m(\theta'_{k'}, u' \mid \theta_k, u)}$$

are computed in lines 52 through 62, and lines 65 through 69. In lines 52 through 62, the proposal probability for the proposed hypothesis is calculated as 1 divided by the number of possible pairs of terminal pedicel twigs. Lines 65 through 69 state the probability of proposing the current hypothesis when being at the proposed hypothesis, or in other words the proposal of the reverse jump. It is 1 divided by the number of Tptwigs with enough Berries. The fraction itself is constructed in line 83. After the computation of all fractions the final acceptance probability is computed in lines 85 through 88, where the p_fraction is exponentiated with the reciprocal of the current temperature from the Simulated Annealing. The last line of code then calls the method that decides to accept or reject the constructed hypothesis based on the herein calculated acceptance probability. What was left out in the computation of the acceptance probability is the absolute value of the determinant of the Jacobian matrix

$$\left|\frac{\partial(\theta'_{k'}, u')}{\partial(\theta_k, u)}\right|.$$

The following description shows how this value is always exactly 1 for the jump 'Combine Tptwigs,' but the same formulation applies to all other jumps. In Figure 5.6 the process is visualized. Before applying the jump, the current hypothesis is described by a parameter vector θ_k of length n_k. That vector holds all lengths and angles of all components. The proposal hypothesis is described by a vector $\theta'_{k'}$ of length $n_{k'}$. Since two Tptwigs were combined into one, it holds that $n_{k'} < n_k$. In order for the jump to be a diffeomorphism, auxilliary vectors u of length m_k and u' of length $m_{k'}$ must be introduced, such that $n_k + m_k = n_{k'} + m_{k'}$. Here u is chosen such that it comprises the angles and length of the new Tptwig, and u' such that it comprises the angles and lengths of the two Tptwigs that are deleted. Strictly speaking, the vectors θ_k, $\theta'_{k'}$, u, and u' must also hold the angles and lengths of the Pedicels of the affected Tptwigs, but for the sake of brevity they are left out in this description

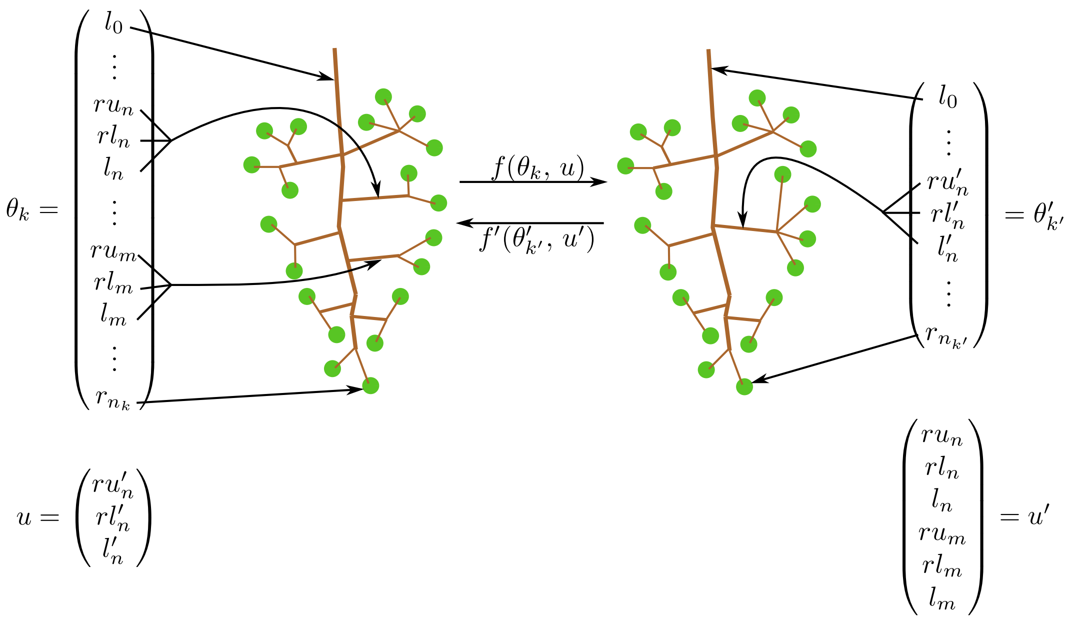


Figure 5.6: *Parameter and auxilliary vectors for the jump 'Combine Tptwigs.' The vectors u and u' are chosen such that the dimension matching condition holds, and the absolute value of the determinant of the Jacobian matrix computes to 1.*

and in the figure. Since most parts of the vectors θ_k and $\theta'_{k'}$ are the same and the other parts are formed by substituting the according values from the auxilliary vectors u and u', the Jacobian matrix consists of exactly one 1 in every row and every column. The determinant of such a matrix is either -1 or 1 and therefore the absolute value is always 1. Therefore in this jump – and in all other jumps – one can leave out the computation of the determinant of the Jacobian matrix and simply assume they are 1.

5.1.4 Reconstruction Results

Figure 5.7 shows the results of four Riesling grape clusters after ca. 8000 RJMCMC iterations. This is a first overview of results, the next chapter gives a detailed evaluation. The figure shows the results of one exemplar per row. In the left column there are the thinned input point clouds. The middle column shows the final reconstruction results as generated by the algorithm. Berries of one color belong to the same berry group. Due to the clusters' densities one cannot look into the cluster through the berries. Therefore in the right column the Berries have been deleted. There one can see that twig lengths are well adapted to the clusters' forms. For example, in the third and fourth example there are wider areas in the top of the clusters accompanied by longer twigs.

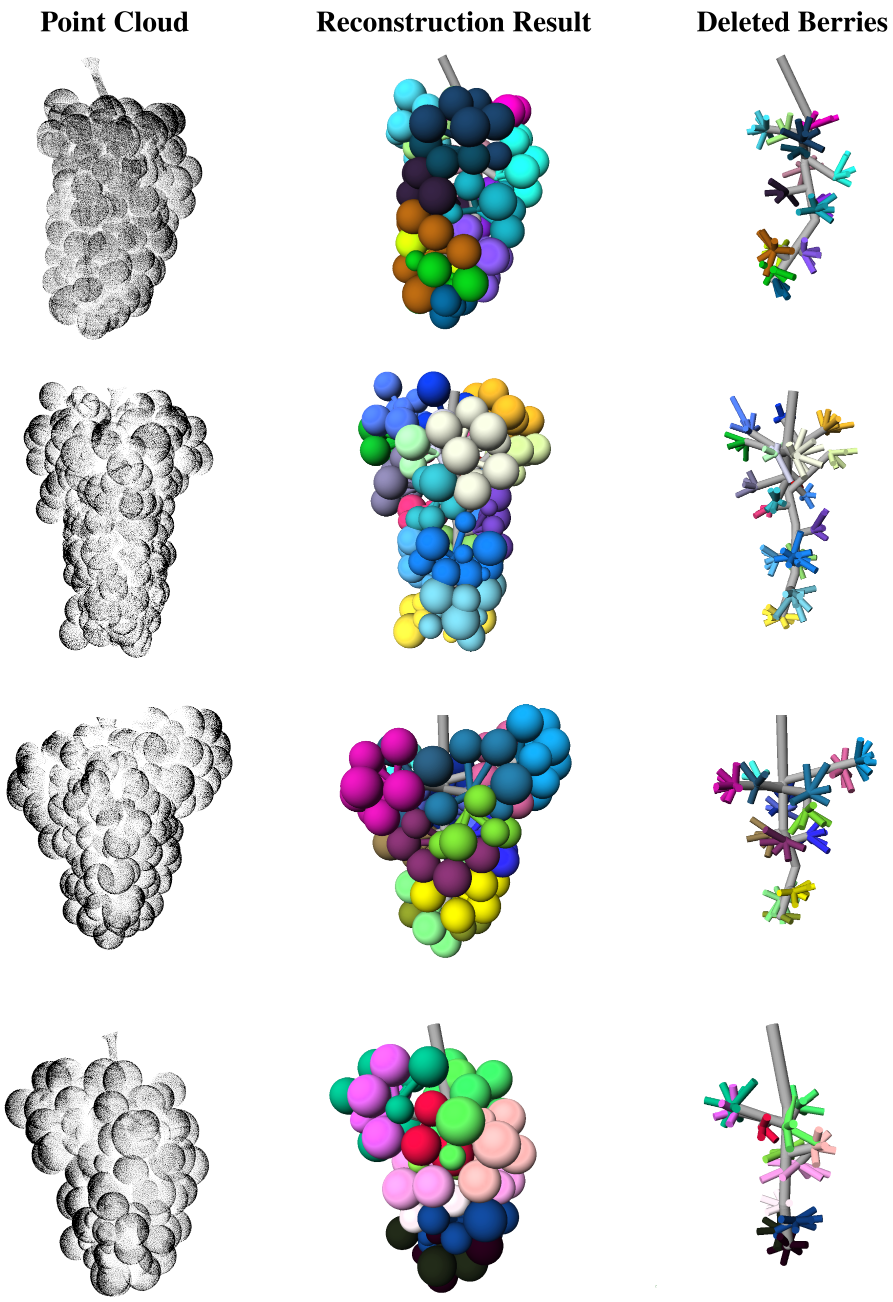


Figure 5.7: *Reconstruction results of four Riesling grape clusters in the development stage BBCH89. Colors encode berry groups. The left column shows the point cloud. The middle column holds the reconstruction results. Because that does not give much insight into the cluster, in the right column the Berries are removed.*

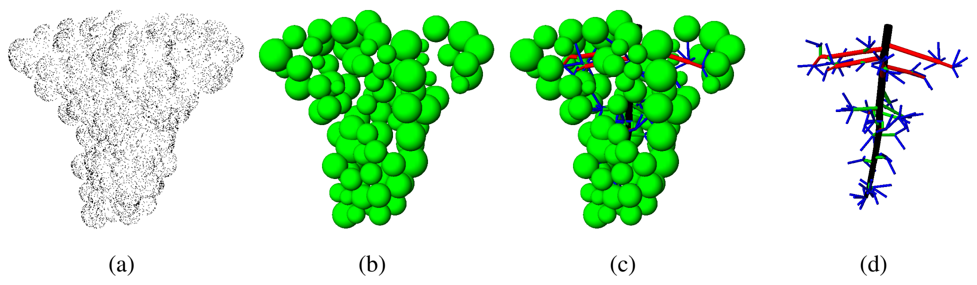


__Figure 5.8:__ Result of the BBCH81 reconstruction on one example grape cluster. From the point cloud (a) the berries are detected (b). This is used for the reconstruction (c). In (d) the berries are removed for a better view.

5.2 Further Applications

In Section 1.3 it was said that the presented reconstruction approach is applicable to other plants, given that they have a tree-like branching structure. Examples for further applications were investigated in student's theses (diploma, bachelor, master) and are shortly summarized in the following sections. Other development stages of grape clusters were examined in two applications, while the venation of vine leaves was the focus of a third application. Common ground for all applications was the same model-based three-step reconstruction that was introduced in this thesis and shown in detail in this chapter. Each of the different applications has different pre-conditions and therefore a different emphasis on the three steps. For example, when a larger part of the structure is visible, there is more concentration on the data-driven aspects of the approach. The following sections show the special conditions for the different approaches and summarize the specific behavior of the algorithms.

5.2.1 Reconstruction of Grape Clusters in BBCH81

In Jelden [2014] the development stage BBCH81 "beginning of ripening" was investigated. Figure 5.8 exemplifies a result. In this stage the berries are almost the same size as in the approach of this thesis, meaning that the stem system is mostly occluded. The general formulation and the combination of methods of the application are the same as in this thesis, starting with the same sphere detection for the first reconstruction step of reconstructing the visible components. The result is also a set of spheres, representing the berries that can be seen from the outside. The differences lie in the other two steps. The model-driven hypothesis is generated in a forward fashion and the local refinements for the optimized hypothesis are done by a different set of RJMCMC jumps. The following two paragraphs summarize these differences.

For the model-driven hypothesis the Berries are used as targets and the generation is done in a forward manner. The core of the approach is the so-called BranchExplorer. This is a sphere whose surface is divided into sectors and which is used for growing the rachis and the twigs. After the detection of the Berries and the start point of the rachis, the BranchExplorer is inserted at the start point for growing the rachis. The growth is guided into the right direction by adding strings from the Berries to the BranchExplorer. Each such string intersects a specific sector on the surface of the sphere. The actual direction of growth is determined by the length of the strings. The longer a string, the higher its weight for the direction. Now the rachis is iteratively prolonged by a certain length and the strings are updated to the new center of the BranchExplorer. This is done for the upper part of the rachis where there are branchtwigs allowed. Along that way branches are detected by investigating the distribution of string intersections on the surface. If a branch is detected the twig is not directly grown but its start point and direction are memorized. In the next substep for each branching twig a new BranchExplorer is inserted. This time the target points are not the Berries but potential Pedicel_Origins, which were computed in a previous step. After that the procedure is the same as before. Once the twigs are constructed, Pedicels are inserted between Berries and Pedicel_Origins. In a last substep a voxel grid is constructed around the grape cluster and for each voxel a value is computed that states whether it is possible that a new Berry might be inserted there. Berries are then inserted as long as there are no intersections with other parts of the cluster. The final result of this second step is then a first model-driven reconstruction hypothesis.

As is done in this thesis, the model-driven hypothesis is used as an initialization for an optimization in Jelden [2014], as well. The difference is that there is only one jump for local refinements. This jump randomly selects one of the nodes in the stem system and uses the algorithms of the model-driven reconstruction to build the part below the selected node anew. For this, a different set of parameter values is chosen, so that each iteration results in a different stem system. As is the case here, this procedure is adapted by Simulated Annealing to terminate the search.

5.2.2 Reconstruction of Grape Clusters in BBCH73

In Lenz [2014]the development stage BBCH73 ”groat-sized berries” was investigated. Figure 5.9 exemplifies a result. In this stage the berries are so small that the stem system is almost completely visible. This gives the opportunity for a larger emphasis on the data-driven hypothesis. This data-driven process already yields a complete reconstruction hypothesis, therefore the second and third reconstruction steps were left out.

On the one hand the data-driven hypothesis generation relies on the same berry detection as this thesis. On the other hand the idea is to actually grow a reconstruction hypothesis into the point cloud. This is done in a bi-directional manner. One direction is from the peduncle

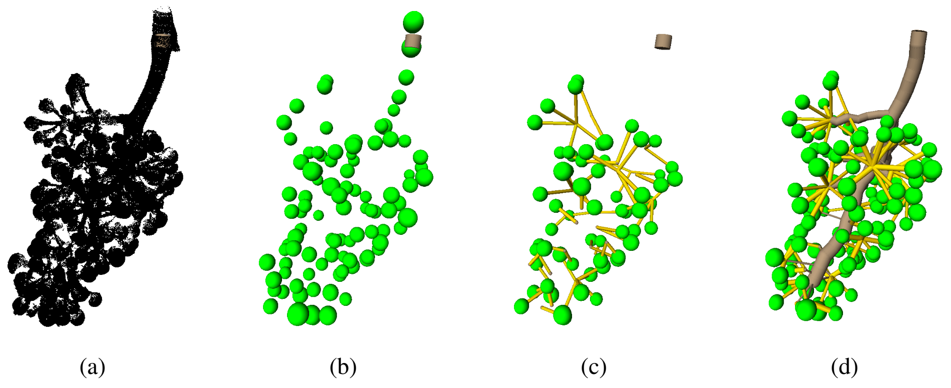


Figure 5.9: Result of the BBCH73 reconstruction on one example grape cluster. From the point cloud (a) the Berries are detected (b). Then the Berries are connected by Pedicels (some Berries are temporarily omitted) (c). In (d) the final reconstruction is shown.

along the rachis and twigs. The other direction is from the berries toward the twigs. In the first substep the detected Berries are used as the basis for finding the appropriate Pedicels. Potential Pedicels are found by running k-means on the points in the surroundings of each Berry and using the means as temporary end points of the Pedicels. These Pedicels are then grown by considering the points in a look-ahead cone whose tip is located at the end point of a Pedicel and that is opened in the direction of the Pedicel away from the Berry. The new end point for a Pedicel is the center of mass of the points in the cone. Pedicel hypotheses are discarded if they satisfy certain stop criteria. For final Pedicels whose end points are close to each other a common Pedicel_Origin is found as the point that minimizes the distance to all end points. In the next substeps these Pedicel_Origins will be used to guide the forward reconstruction of rachis and twigs. The reconstruction of the rachis is done in a similar way as the growth of the pedicels, namely by selecting the center of mass of a look-ahead cone at its tip as the next end point of the current frustum. The current Rachis frustum is split in two if a certain length has been reached, so that the curvature of the rachis can be handled. Along the rachis, branchings are detected by considering points in a look-ahead cylinder at the tip of the current Rachis frustum. Points in that cylinder are projected onto the cylinder base plane. If the maximum distance between those projected points is large enough a branching is detected. The reconstruction of twigs is again done by using the look-ahead cone to determine the next frustum end point. After having reconstructed the rachis and the twigs, remaining Pedicel_Origins are connected by short frustums.

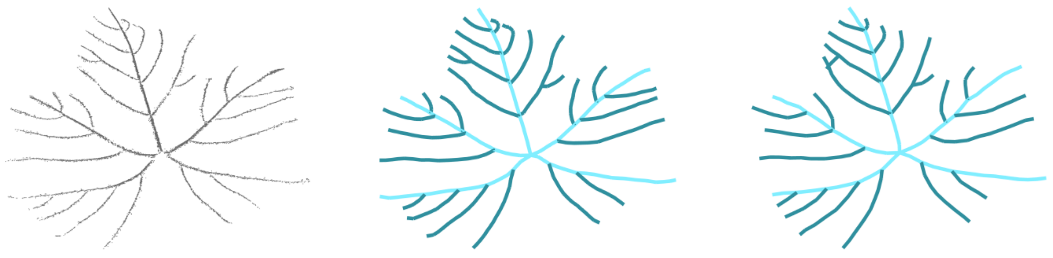


Figure 5.10: *This image is taken from Schneider [2012] and shows an example result of that work. In the left there is the pointcloud after erasing non-vein points. The middle image shows the manually created reference reconstruction. The right image shows the actual result.*

5.2.3 Reconstruction of Leaf Veines

In Schneider [2012] the reconstruction approach was applied to the venation of vine leaves. Figure 5.10 exemplifies a result. The branching structure of leaf veins is comparable to the one observed in grape clusters but, of course, described by different parameters and components and by a different structural model. This shows that the general formulation can be transferred to other branching structures. The sensor data are 3D point clouds of a total of 22 grapevine leaves of five different cultivars. They were generated with the same sensor set-up as in this thesis. In a pre-processing, the point clouds were cropped to the points of the actual veins by a method from Paulus [2014].

In the first step of reconstructing the visible components the end point of the petiole is detected – this is the point where the five main leaf veines have their common origin – and for each vein an initial frustum is inserted in the corresponding direction. For the second reconstruction step there is no purely model-driven generation of a reconstruction hypothesis. Instead an approach is used that on the one hand grows a hypothesis into the data and on the other hand alters the existing hypothesis. This is done by the selection of RJMCMC jumps. There are a total of five jumps: (1) Change the rotation and length of a frustum. (2/3) Add or remove a frustum. (4/5) Add or remove a branch. The first jump alters the appearance of an existing part of the hypothesis by changing its parameter values. Jumps 2 through 5 alter the appearance of the hypothesis by adding or deleting parts. These jumps make sure that the hypothesis can grow into the data and that it can be reduced in cases when the reconstruction is too far away from the data. The algorithm is also equipped with Simulated Annealing to make sure it terminates.

5.3 Summary

In this chapter the reconstruction approach was applied to grapevine grape clusters and leaves. At first, a detailed description of the application on grape clusters in full ripeness was given that encompasses details on all three reconstruction steps. In the further sections it was shown that the general approach can handle a different set of RJMCMC jumps in an otherwise mostly similar set-up, that if sensor data of the inner structure are visible a more data-driven emphasis can yield a complete hypothesis, and that the approach can be applied to a different branching structure. In all four cases satisfying results were presented. All the approaches were able to handle the challenges stated in Section 1.2, namely complexity and variability of plant architectures, and occlusions and disconnectedness of sensor data. The next chapter gives a detailed evaluation of the approach from Section 5.1 for the reconstruction of BBCH89 grape clusters.

Chapter 6

Evaluation

After the general presentation of the approach and a more detailed inspection of some applications, this chapter gives an in-depth evaluation of the results of the application on grape clusters in development stage BBCH89, as described in Section 5.1. This evaluation is divided into four sections: Numbers of components, positions of components, lengths of components, and phenotype. The aim is to separately evaluate the structure as well as the geometry of the results. The first section, numbers of components, tests if the correct numbers of components are reconstructed and is therefore concerned with the structure. It rates the quality of results by a comparison of the numbers of components in the automatically generated results to manually created ground truth reconstructions. The second section, positions of components, is more concerned with the geometry and tests if the components are found in the right places. It evaluates the distance of component root positions in comparison to the positions on the ground truth data. The distance function builds on assignments of components based on the Hungarian method (Kuhn [1955], Kuhn [1956], Munkres [1957]). The third section, lengths of components, is also geometry-centered and evaluates the differences in the sum of lengths of components to the ground truth data. The fourth section, phenotype, shows how a set of phenotypic traits can be computed from the results, some of them more precisely than was possible before by manual measurements.

For the following inspection four point clouds of grape clusters in development stage BBCH89 'ripe berries' were used. The four exemplars are presented in Figure 6.1, one per row. For each exemplar the figure shows the point cloud of the grape cluster with berries, the point cloud of the stem system after removing the berries, and the manually created ground truth. The chosen grape clusters have different sizes with lengths between 9.5 cm and 13 cm, and widths between 5.5 cm and 8.5 cm, they have differently curved rachises, different numbers of berries, and different outer forms.

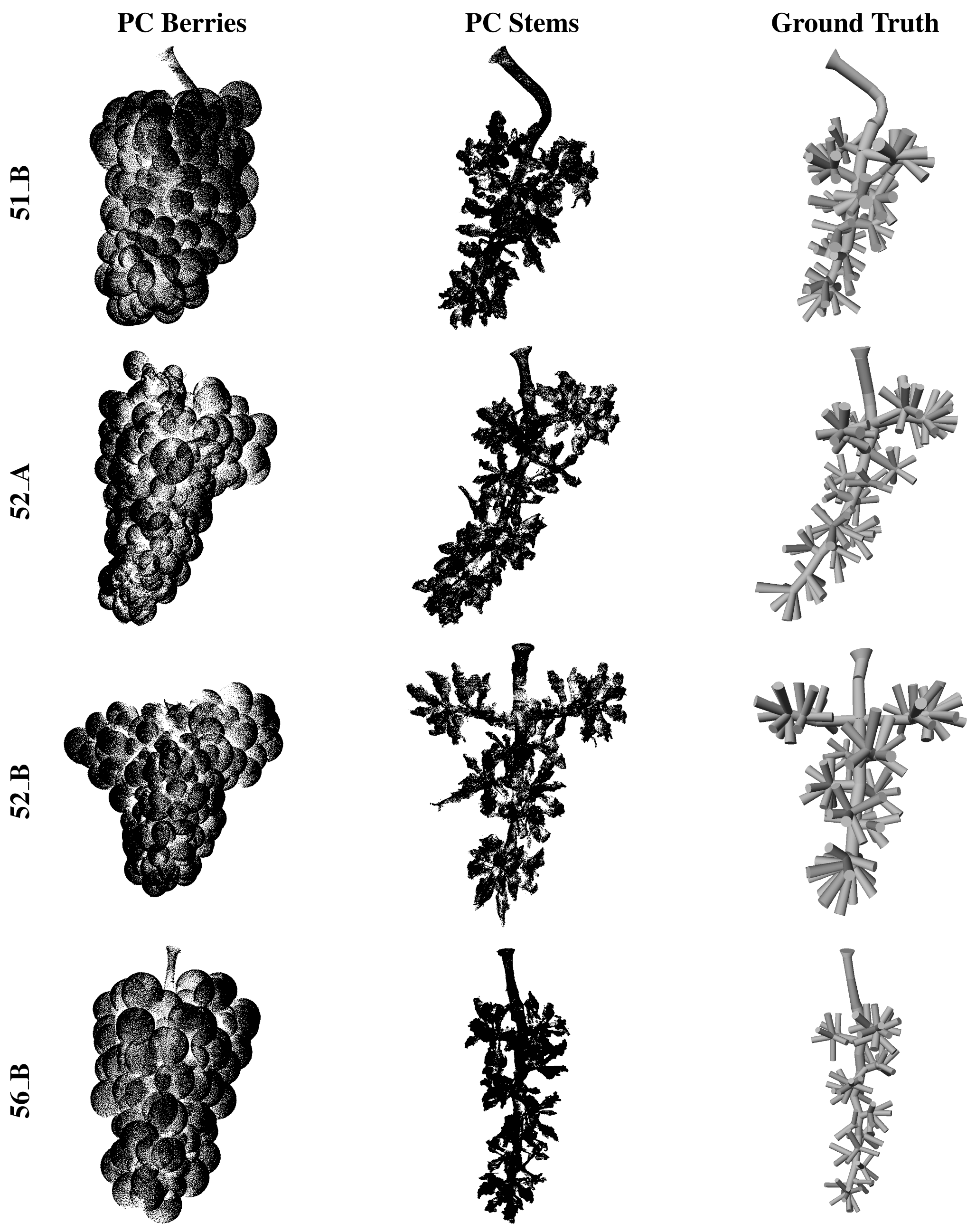


Figure 6.1: *Sensor data. The proposed method was evaluated on four exemplars, one per row. The first column shows the point clouds of the outer berries. In the second column the point clouds of the stem systems after removing the berries are shown. The third column depicts the manually created ground truth reconstructions of the stem systems.*

For all four exemplars manually created ground truth stem systems were generated. This was done in the RGG software GroIMP by manually placing frustums in the point clouds of the appropriate stem systems. Due to the complexity of the stem systems it is not always possible to clearly see which component a point of the point cloud belongs to. Therefore these ground truth reconstructions can only be an approximation to the truth and a certain tolerance has to be granted, especially for the very fine structures.

Since this is a probabilistic approach, with random choices of jumps and proposals, 100 results were computed from every point cloud. The start temperature for the Simulated Annealing was set to 1.0 and the cooling factor for the geometric annealing was 0.999, leading to about 7000 to 9000 RJMCMC iterations per exemplar. More iterations did not lead to significantly better results. The exemplars have names of the pattern 'number_letter' where the number is the number of the particular plant in the vineyard and the letter is the identifier for the specific cluster on that plant.

For the evaluation one could also think of using some measure to compress all investigated aspects into one evaluation function that outputs just one number. For example, in the work of Ferraro and Godin [2000] a distance measure between plant architectures is presented that is based on a result from graph theory for comparing rooted, unordered tree graphs. This method is able to handle topology, geometry, and component labels all at once. While this is a good method for summing all those aspects in just one number it is not what is aimed for in this evaluation. Here the focus is on a separate investigation of structure and geometry on the one hand and a detailed inspection on the different kinds of components on the other hand in order to point out particular strengths and weaknesses of the reconstruction approach. This can not be done with the mentioned method.

6.1 Numbers of Components

For an evaluation of the structure, the comparison to the ground truth data is done by comparing the numbers of the different components. The results are shown in tables, one for each investigated component. The entries for the tables were generated from the 100 results that were computed for each exemplar. Each table has three columns. The left one holds the values from the manually created ground truth data. The middle column shows the values from the initial reconstruction hypothesis. These are equal for all exemplars, except for the number of berries. They are still mentioned for a quick comparison. The right column depicts the values from the presented reconstruction approach and is divided into five subcolumns. The first two subcolumns show minimum and maximum values. The second two subcolumns hold average values, namely the median and the arithmetic mean. Finally, the last column depicts the standard deviation.

Number of Rachis Nodes

	manual	initial	automatic				
			min	max	median	mean	std
51_B	13	19	6	12	10	9.47	1.08
52_A	11	19	9	13	11	10.79	0.97
52_B	7	19	4	10	8	7.53	1.18
56_B	14	19	5	10	7	7.00	0.97

Table 6.1: *Results for the number of rachis nodes. In all cases the number starts at 19 and then decreases to a value closer to the truth. In cases 52_A and 52_B the median and the mean values are close to the original values. In the other two cases the difference is larger. The standard deviations are reasonably low.*

Number of Branchtwigs

	manual	initial	automatic				
			min	max	median	mean	std
51_B	3	0	2	4	2.5	2.52	0.54
52_A	3	0	3	6	4	4.13	0.58
52_B	4	0	2	5	3	3.06	0.56
56_B	3	0	2	3	2	2.18	0.38

Table 6.2: *Results for the number of branchtwigs. In all cases the number starts at 0 and then increases to a value closer to the truth. In case 51_B the median and mean values are closest to the ground truth value, but also the other three cases are satisfyingly close. The standard deviations are reasonably low in all cases.*

The first investigated trait is the number of nodes on the rachis, i.e., points on the rachis where a twig (branchtwig or tptwig) branches from, including the node at the end of the rachis where the terminal pedicels branch from. There is no distinction between nodes with one, two, or more branches. The results are shown in Table 6.1. Due to its model-driven nature, the number of nodes of the initial reconstruction hypothesis is an equal 19 for all exemplars. The ground truth values are lower in all four cases, so the automatic reconstruction has to correct the initial hypothesis in that aspect. In tendency this is done in all four cases. In the cases 52_A and 52_B the median and mean values are very close to the ground truth values and the difference between minimum and maximum values is also in an acceptable range. In the other two cases the values are not as close to the ground truth, especially in 56_B. On the other hand in all four cases the standard deviation is low, showing that the proposed method does not generate many outliers. All in all, for two out of the four exemplars, the algorithm is able to produce reliable results with good average values and standard deviations.

The second investigated trait is the number of branchtwigs, i.e., branches from the rachis that hold themselves not only terminal pedicels but also subtwigs. The results are shown in Table 6.2. The initial value for all exemplars is 0, the ground truth value is either 3 or 4.

Number of Subtwigs

	manual	initial	automatic				
			min	max	median	mean	std
51_B	6	0	2	4	3	2.77	0.66
52_A	8	0	3	6	4	4.51	0.69
52_B	6	0	2	5	3	3.29	0.65
56_B	6	0	2	4	2	2.31	0.50

Table 6.3: *Results for the number of subtwigs. In all cases the number starts at 0 and in all cases it does not increase enough to reach the ground truth values. On the other hand, the standard deviations are reasonably low.*

Number of Tptwigs

	manual	initial	automatic				
			min	max	median	mean	std
51_B	12	19	7	11	8	8.51	0.95
52_A	10	19	7	13	10	10.19	1.12
52_B	6	19	4	10	7	7.45	1.17
56_B	10	19	4	8	6	6.21	0.86

Table 6.4: *Results for the number of tptwigs. In all cases the number starts at 19 and decreases to values closer to the ground truth. In cases 52_A and 52_B the average values are very close to the ground truth, in the other two cases the difference is larger. The standard deviations are again reasonably low.*

Exemplar 51_B shows the best results in all aspects except the standard deviation. The other three are also satisfyingly close to the true values. As before, the minimum and maximum values, the average values, and the standard deviations show that the method is able to produce reliable results with only a few outliers.

As the third trait the number of subtwigs was investigated, i.e., the small twigs that branch from branchtwigs. The results are shown in Table 6.3. The value is initialized to 0 in all cases, the ground truth values are either 6 or 8. In no case the original value is reached, even the maximum values are lower than the ones from the ground truth. The algorithm has a tendency to produce only one subtwig for each branchtwig. On the other hand, the standard deviations are again reasonably low. For an improvement new jumps 'Split Subtwig' similar to 'Split Tptwig' and 'Combine Subtwigs' similar to 'Combine Tptwigs' might improve results.

Subject of the fourth investigation was the number of tptwigs, i.e., branches from the rachis that only hold terminal pedicels. The results are shown in Table 6.4. The initial value in all four cases is 19, while the ground truth values range between 6 and 12. The table reveals that on average the ground truth value is achieved almost exactly for exemplar 52_A and is also

Number of Berry Groups

	manual	initial	automatic				
			min	max	median	mean	std
51_B	22	20	13	17	15	14.80	0.91
52_A	23	20	18	20	20	19.83	0.40
52_B	17	20	13	18	15	14.80	1.06
56_B	20	20	11	13	12	11.70	0.64

Table 6.5: *Results for the number of berry groups. In all cases the number starts at 20. In cases 52_A and 52_B the average values are close to the ground truth, in the other two cases the difference is larger. The standard deviations are again reasonably low.*

very close for exemplar 52_B. In the other two cases the average values differ more. Again in all cases the standard deviation is low, compared to the ground truth values. All in all, for two out of the four exemplars, the algorithm is able to produce reliable results with good average values and standard deviations.

For the fifth investigation the number of berry groups was chosen. The results are shown in Table 6.5. The initial value in all four cases is 20, with ground truth values between 17 and 23. The best results are reached for exemplar 52_B and 52_A, but even for those exemplars there are not enough berry groups. In the other two cases the differences to the ground truth values are larger. Correlating these results with the ones from the last paragraphs for the numbers of the different twig types this is not surprising since in most cases there are too few twigs and therefore also too few berry groups.

The sixth and last trait for this evaluation is the number of berries. This includes the berries from the berry detector as well as the newly inserted ones during the local refinements. The results are shown in Table 6.6. Here the initial values are not the same for all cases, since they are not generated in the model-based step but in the data-driven step at the very beginning of the whole process. First of all, it can be seen that the initial number of berries from the berry detector is lower than the numbers from the ground truth. But that is obvious, since the point clouds only capture the outer berries of the grape clusters and therefore the berry detector can not detect all berries. Aside from that, the exemplar 56_B shows the best results, both in terms of average values and in terms of the standard deviation. In case 51_B the average values are too low and in the two remaining cases 52_A and 52_B the average values are too high. In all three cases also the standard deviations are quite high. This might be the case because of a lot of small berries that are inserted in the process or because intersections between the berries are not penalized enough.

In summary, this section showed that five out of six investigated traits can be reconstructed quite well. Of course, the ground truth values can not be achieved exactly. This has many

Number of Berries

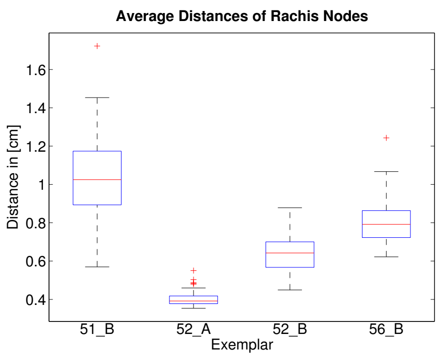
	manual	initial	automatic				
			min	max	median	mean	std
51_B	105	87	87	117	88	92.50	10.02
52_A	123	115	116	155	149	142.84	13.55
52_B	96	91	91	121	117	110.57	11.66
56_B	83	71	71	85	80	77.85	4.57

Table 6.6: *Results for the number of berries. The initial values differ due to the data-driven generation. For case 56_B the results are very good, while in the other cases too many or too few additonal berries are inserted.*

reasons. The first and most obvious one is that the interior is completely occluded, leaving no sensor data that could be used as evidence for or against certain hypotheses. Only the model knowledge can be used there. Which is also the second reason. The biological model is based on only a few example grape clusters. For a better model two options can be thought of. The first one is to increase the number of examples as the basis for the model. The second option is to investigate correlations between parameters, for example, if there is a connection between the number of berries and the number of berry groups or the number of subtwigs. The third reason is that the berry detector could be enhanced. It sometimes creates small berries that are not exactly located at the right positions and some berries are not detected at all. Since the process of creating the model-driven hypothesis and the optimization base on these results, an improvement in the berry detection might also improve final reconstruction results. Nevertheless, the presented results are already very promising and form a good basis to build upon. The next section evaluates if the components were not only found at all but if they were found at the right positions.

6.2 Positions of Components

For the second part of the evaluation the Hungarian method is used (Kuhn [1955], Kuhn [1956], Munkres [1957]). This is an algorithm for assignment problems between two sets A and B. The goal is to assign every element of set A to at most one element of set B, and vice versa, such that the assignments have either the least cost or the maximum benefit. An example is the assignment of employees to open working projects. To solve this assignment, a matrix M is generated that holds in each position m_{ij} a value that states how well the element of row i (an employee) is fit for an element of column j (a project). This matrix is transformed in several steps until a state of the matrix appears where a set of zeroes can be selected such that in every row and every column there is exactly one 0 selected. These zeroes encode the assignments; if an entry m_{ij} is 0, then employee i is assigned project j.



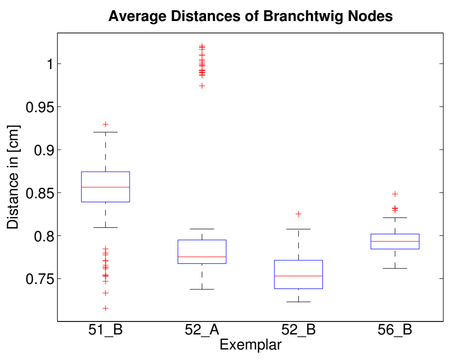


Figure 6.2: *Results for exemplar 52_A are clearly the best. All in all, the results are in similar ranges and for no exemplar are there any extreme results.*

Figure 6.3: *For all exemplars the range of values is small, even with the outliers. 51_B has the worst results, while the other three are comparably good.*

In this evaluation the Hungarian method is used as the basis for a distance function between reconstruction result and ground truth. Since the types of components are known, this can be used as additional information. For each component type a matrix is constructed that holds the euclidean distances between equal component types. Consider as an example the locations of the pedicel origins, i.e., the places from where the pedicels branch. For each such position in the ground truth, the euclidean distance to each pedicel origin in the reconstruction is computed. The same is done for the nodes on the rachis, for the nodes on branchtwigs, and for berries. For each component type the distance is computed as the sum of euclidean distances for their respective minimal assignment and finally these values are summed up to form the total distance. Again this is done for 100 reconstruction results for each of the four exemplars.

One characteristic has to be mentioned. When the berries are removed from the grape clusters, the stem system is more bent than before, since the weight of the berries no longer pulls the stems down. This has the effect that the reconstructions from the berry point clouds and the ground truth from the stem system point clouds are not directly comparable. Therefore, to generate more accurate results, the stem system ground truth was fit into the berry point cloud. This is, of course, only an approximation, but a better one than without this procedure. The structure is not altered.

The results are presented as boxplots. The red line depicts the median value of the 100 reconstruction results. The box contains 50% of all values and is limited at the top by the 75% quantile and at the base by the 25% quantile. The dashed lines – the whiskers – lead to the extreme points and their length is at most 1.5 times the height of the 50%-box. The red crosses mark outliers; values above or below the extreme points. For better comparability of the results, the average distance is used, since there can be different amounts of components. The distance only covers actual assignments. If there are more or less

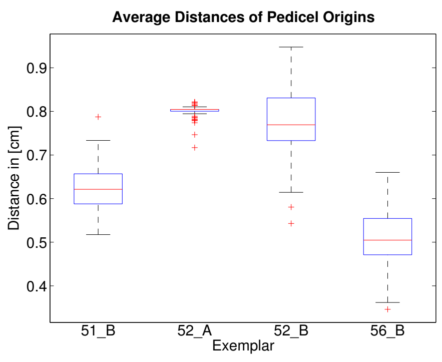


Figure 6.4: *51_B and 56_B are relatively close to the original positions, while 52_A and 52_B are more imprecise.*

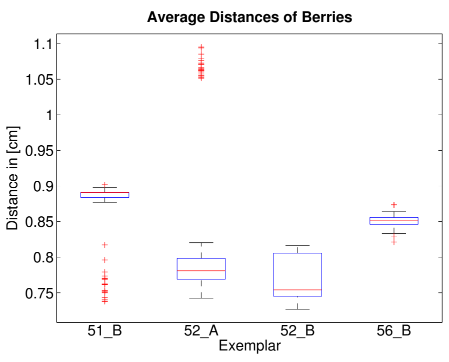


Figure 6.5: *All values are in comparable ranges. 51_B and 52_A produce some outliers, probably due to misplaced new berries.*

components reconstructed this is not covered. Instead an evaluation on this can be found in the previous section. In relation to the dimensions of the grape clusters and the slight deviations from the ground truth positions to the real positions, distances up to 0.5 cm are regarded as good, whereas values up to 1.0 cm are still regarded as satisfactory.

In Figure 6.2 the distribution of average distances of the rachis nodes are shown. The results for exemplar 52_A are clearly the best, since the box that holds 50% of all values is less than 1 mm in size and even the outlier values are below the best values of exemplar 51_B, which also has the broadest range of values. The other two exemplars achieve results between those two extremes. All in all, in most cases good or satisfactory results were achieved. At the moment, only the positions of branches on the rachis can be altered but not the positions of the frustums that build the rachis. For better results there should be local refinements to alter the rachis.

Figure 6.3 shows the distribution of average distances of branchtwig nodes. Again, 51_B has the worst results, while the other three are comparably good. On the other hand, for all exemplars the range of values is very small, even considering the outliers. For 51_B and 52_A there are a few outliers, where probably the sampling got stuck in a local optimum of the hypothesis space. For almost every reconstruction result the distances are in a satisfactory range. At the moment, only positions of branches on the branchtwigs can be moved, not the frustums that make the branchtwig. As was true for the rachis, this might help too.

Figure 6.4 depicts the distribution of average distances of positions of pedicel origins. On average, for 51_B and 56_B the results are closest to the original positions, but all results are in a satisfactory range. For 52_A the range of values is very thin, which means that the results are all very consistent. While 52_A and 52_B are farther away from the original positions than the other two, all results are in a satisfactory range.

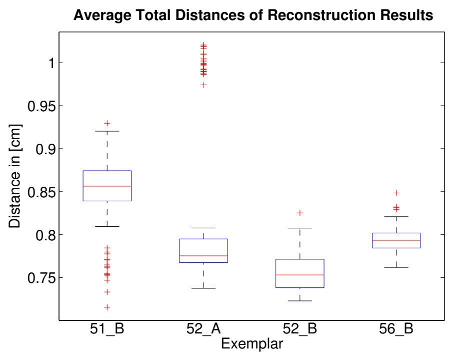


Figure 6.6: The best results are achieved for exemplars 52_A, 52_B, and 56_B. 51_B is only slightly inferior. 51_B and 52_A produce a few outliers.

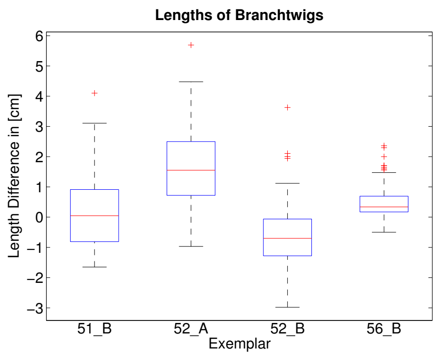


Figure 6.7: The best results are achieved for exemplar 56_B. But 51_B and 52_B are also in a satisfying range.

Figure 6.5 holds the distributions of average distances of berry positions. For this evaluation the berries' centers could not be used. The pedicels do not grow into the berries, therefore after removing the berries only the tips of the pedicels can be used for comparison. The figure shows that again all values are in comparable ranges, although 51_B and 52_A produce some outliers, probably due to misplaced new berries. The main factor to enhance results is an improvement of the berry detector but also the insertion of new berries in the interior of the clusters can be improved. The berry detector should be adapted to the more ellipsoid form of the berries, instead of the sphere approximation. For the insertion of new berries, at the moment one voxel of the voxel grid is chosen at random and then a radius for that berry is drawn at random, which might lead to the insertion of many small berries. It might be better to first draw a radius from the distribution of radii of the original berries from the grape clusters and then look for a voxel where this berry fits.

Figure 6.6 shows the combined results for all components. Some outlier results are in the range of 1 cm but most values range from 0.75 cm to 0.85 cm. The overall best results are achieved for exemplars 52_A, 52_B, and 56_B, while 51_B is only slightly inferior. 51_B and 52_A produce a few outliers.

In summary this section has shown that the components were reconstructed at places relatively close to the original positions. Possible improvements can be made at different places. The berry detector could be enhanced to detect more berries with a higher location accuracy. The insertion of new berries could be improved by first sampling a radius instead of sampling the location first. It should be possible to move branchtwig and rachis frustums. Still, the presented results are already very promising. The next section evaluates the lengths of components.

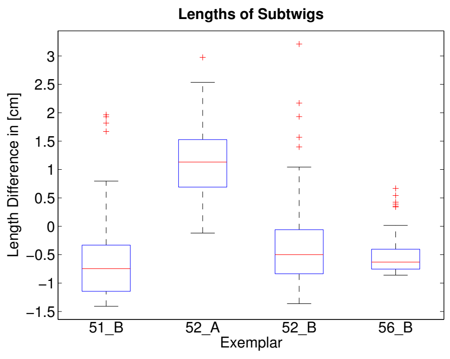


Figure 6.8: *51_A has the worst results, while the other three are comparably good, with 56_B having the best results.*

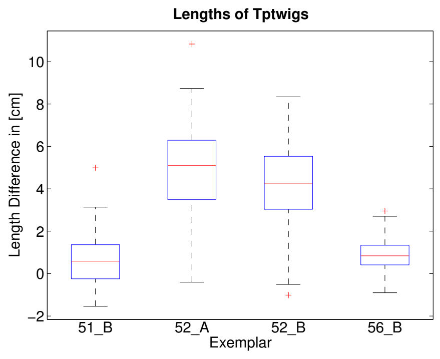


Figure 6.9: *51_B and 56_B clearly have the best results. 52_A and 52_B are more diverse. In general tptwigs are too long.*

6.3 Lengths of Components

This section evaluates the reconstruction results in terms of the lengths of branchtwigs, subtwigs, tptwigs, and pedicels. For each component type c, every exemplar and every reconstruction result a value

$$\psi = \sum_{i=1}^{n} len(c_{rec,i}) - \sum_{i=1}^{m} len(c_{gt,i})$$

is computed. This is the difference of the sum of lengths of the different components of the reconstruction with the sum of lengths of the ground truth data. The results are again shown as boxplots

Figure 6.7 shows the results for the branchtwigs. The best results are achieved for exemplar 56_B. The 50%-box covers around 0.5 cm and the median is close to 0. Even the extreme values are not much further. The results for 51_B and 52_B are not quite as good but still in a satisfying range. 52_A has a slightly broader range of values and a higher median value.

The results for subtwigs are presented in Figure 6.8. They are similar to the results of the branchtwigs in that 52_A has the worst results, while the other three are comparably good. The range of values is lower in all cases, which is due to the already small subtwigs. Again 56_B has the best results with a median close to 0 and a thin 50%-box.

For the length of tptwigs, shown in Figure 6.9, exemplars 51_B and 56_B clearly have the best results with the median closest to 0 and a small 50%-box. The results for 52_A and 52_B are more diverse and with a higher median. This correlates with the number of tptwigs (see Table 6.4). 51_B and 56_B have on average too few tptwigs, while 52_A and 52_B have

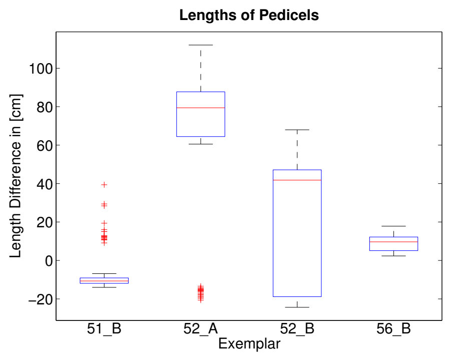


Figure 6.10: *51_B and 56_B are good and reliable, 52_A produces too long pedicels, and 52_B produces very diverse results.*

almost the correct number of tptwigs. It can be concluded that in general the reconstructed tptwigs are slightly too long.

The results for the lengths of pedicels, as depicted in Figure 6.10, are rather diverse. 51_B and 56_B achieve very good and reliable results. In 52_A the lengths are constantly estimated too long, in 52_B the 50%-box is very large. These are also the two exemplars, where there are too many berries in the reconstruction results, and therefore also pedicels. A better berry detector and a better insertion of new berries will improve these results.

In summary this section has shown that for the different twig types in most cases the lengths are estimated quite well. For pedicels the lengths are mostly overestimated. This is also due to the fact that often there are too many newly inserted berries. An improvement for the twig lengths could be achieved by enhancing the model and by allowing to change the structure of the rachis. For the pedicels, as before, an improved berry detector would help.

6.4 Phenotype

The goal of this thesis was to develop a reconstruction method of plant architecture for a fast and precise phenotyping. After evaluating the method itself in the last three sections, in this section a selection of phenotypic traits is presented and the results achieved for those traits on the reconstruction results. One part of this selection consists of known and established phenotypic traits from OIV [2009]. But where for those traits just coarse categories were used so far, with the presented approach these traits can be improved to quantified measures. Not for all traits the final reconstruction is needed. For example, for the length and width of the grape cluster the input point cloud would already suffice. As another example, for the total volume of berries the result from the berry detector would already yield an estimate, even without the inner berries that are only added later in the optimization. But this is not in contrast to the aim of this thesis, as it was already pointed out in Section 1.3 that the three reconstruction steps offer the opportunity for phenotyping in an increasing level of detail. Nevertheless, in the following all traits were computed on the final reconstruction results. As for the evaluation, the phenotyping bases on the 100 results per exemplar, to show the reliability.

The OIV descriptor list OIV [2009] gives an overview of established phenotypic traits of grapevine plants in general and grape clusters in particular. Some of those traits for grape clusters need further sensor data and extensions to the presented method before they can be regarded here, for example, the lignification of the stems, the color of the berry skin, or the juiciness of the berry flesh. More importantly, others can now be computed more precisely, for example, the density of the cluster. There can also be new traits that were not considered before, because they were too time-consuming or would destroy the plant. For example, all the values computed in Section 6.1 and Section 6.3 can be seen as phenotypic traits. Before, one would have had to take the berries off the grape cluster and count the numbers of components or measure their lengths, but now this can be done automatically on a much larger scale. Also this enables the possibility to track these values over time, since the cluster does not have to be destroyed. Besides those, the following phenotypic traits are presented here, ordered from coarse to fine:

- OIV 202: Length of Grape Clusters
- OIV 203: Width of Grape Clusters
- OIV 222: Uniformity of Berry Sizes
- OIV 238: Average Length of Pedicels
- Total Volume of Berries
- OIV 204: Density of Grape Clusters
- Berries per Berry Group
- All the values computed in Section 6.1 and Section 6.3, namely the numbers and lengths of arbitrary components

The OIV descriptors 202 (length of the grape cluster) and 203 (width of the grape cluster) are the easiest to determine since the input point cloud is already sufficient. Figure 6.11 shows the definitions of length and width according to the OIV descriptors. The length is defined as the distance between the topmost berry and the lowest berry. The width is defined as the maximum distance of the lateral berries. Table 6.7 contains the measurements on the four example grape clusters. Both values do not change during the local refinements since the outer berries are not moved and new berries are only inserted inside the cluster. Therefore in these cases absolute values are shown instead of distributions of values. Also, to the results are added the according category from the OIV descriptor, which range from 1 (very short/narrow) to 9 (very long/wide).

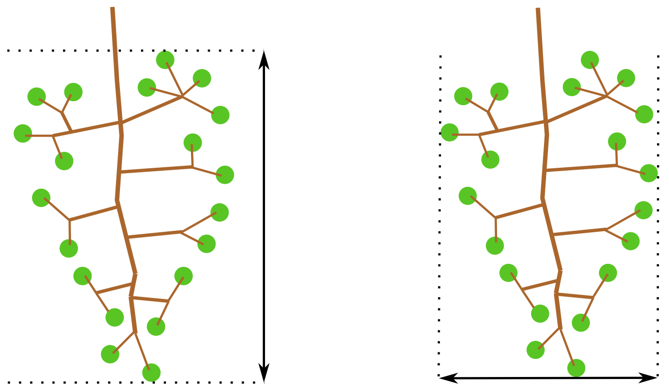


Figure 6.11: Definitions of length and width of grape cluster according to OIV descriptors 202 and 203. Left: The length is defined as the distance between the topmost and the lowest berry. Right: The width is defined as the maximum distance of the lateral berries

Length and Width of Grape Clusters

	Length in [cm]		Width in [cm]	
	Measured	OIV Category	Measured	OIV Category
51_B	10.49	3	5.94	1
52_A	12.94	3	8.31	3
52_B	10.88	3	7.74	3
56_B	9.56	1	5.79	1

Table 6.7: The lengths and widths of the four grape clusters along with the according categorizations from OIV descriptors 202 and 203.

For the OIV descriptor 222, uniformity of berry sizes, at least the results from the berry detector are needed as an estimate, but since the OIV descriptor also regards all the berries of a cluster, the additional berries from the optimization should be included. The OIV descriptor 222 only contains two categories: uniform and not uniform. With the presented approach it is possible to calculate a numeric value for the uniformity. In cases where a coarser categorization is needed, this is still possible by defining intervals of values. For simplicity and as a first approach, the uniformity is computed as $1/\sigma$, where σ is the standard deviation of berry radii. A larger standard deviation means a greater heterogeneity of radii and therefore a value for the uniformity closer to 0. In contrast, a low standard deviation means greater homogeneity of radii and therefore a greater value for the uniformity. Results are presented in Figure 6.12. For exemplars 51_B, 52_A, and 56_B the results are quite reliable (in the sense of a low standard deviation), albeit at different levels of uniformity. 52_B shows much more variability. For a comparison to the OIV descriptor, a threshold value has to be found for separating cultivars that are uniform from those, which are not.

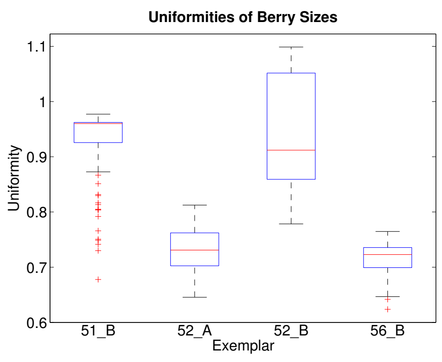


Figure 6.12: For exemplars 51_B, 52_A, and 56_B the results are quite reliable. 52_B shows much more variability.

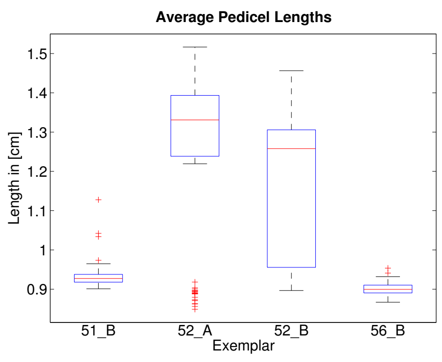


Figure 6.13: 51_B and 56_B produce the most reliable results, for 52_A and 52_B the variance is too high.

For the OIV descriptor 238, average pedicel length, the result from the model-driven hypothesis could be used, but the final reconstruction result will yield a more accurate picture. The length of pedicels has already been evaluated in the last section, albeit with a different focus. In Figure 6.10 the total sum of pedicel lengths was compared to the according ground truth value. In contrast, OIV descriptor 238 investigates the average lengths of pedicels. Results on this are presented in Figure 6.13. The OIV descriptor states five categories from 'very short' to 'very long.' On the basis of the median values, exemplars 51_B and 56_B would be categorized as '5, medium' and 52_A and 52_B as '7, long.'

A new phenotypic trait, that could not be determined before, at least not without destroying the grape cluster, is the total sum of volumes of berries. Figure 6.14 shows the results. The volumes are plotted in milliliters. As before, 52_B shows the most variability in the values, whereas for the other three the 50%-box is quite small. This trait resembles the OIV descriptor 233 'must yield,' but that one can only be computed by taking the cluster off the plant. The total volume of berries can be used to estimate the must yield without destroying grape clusters.

The OIV descriptor 204 for cluster density simply has five categories from 'very loose' to 'very dense,' but no formula is given to calculate a numeric value. With the presented method a quantified measure can be defined as

$$\rho = \frac{\sum_{i=1}^{n} \text{vol}(berry_i)}{100 * (\sum_{i_1}^{m} \text{len}(twig_i) + \text{len}(rachis))},$$

i.e., as the sum of volumes of all berries divided by the sum of the lengths of all twigs and the rachis. The density grows with more berry volume. It decreases with longer twigs because the berry volumes get more separated. The results are depicted in Figure 6.15. Since the

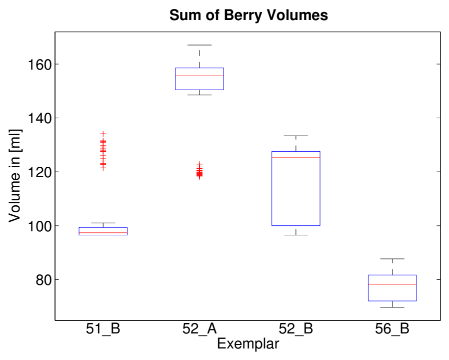


Figure 6.14: 52_B has the largest variability, whereas the others yield reliable result. This can be used as a non-invasive alternative to OIV 233 'must yield.'

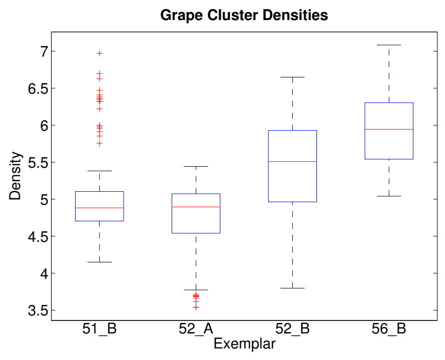


Figure 6.15: Densities are comparably high for all exemplars. A comparison to ground truth is not possible.

ground truth data does not contain berries, a comparison cannot be made. But for the same cultivar the values should be comparably high, which is the case here, although for 52_B and 56_B the variance of values is higher than for the other two exemplars. Again, a better berry detection might improve the results and the reliability of results. Also, for comparability with earlier manual measurements the numeric values have to be mapped to the categories from the OIV descriptor.

A further option for phenotyping is the combination of traits. For example, one could investigate the number of berries per berry group. For this, clearly the final reconstruction results are needed, since in the model-driven hypotheses there is always the same number of berry groups. Figure 6.16 shows results for this combination of traits. It shows for each of the four exemplars the total number of occurrences of berry group sizes, summed over the 100 reconstruction results. For exemplars 51_B and 56_B the distribution of berry group sizes approximately follows a Lognormal distribution, as was pointed out in Section 5.1.3, albeit with the modes between 4 and 6 instead of the earlier observed 3 and 4. For the other two exemplars, 52_A and 52_B, the berry groups are even larger. This is in agreement with the large number of additionally added berries, as observed in Section 6.1.

In summary this section has shown how phenotypic traits can be computed on different levels of detail. Some only rely on sensor data or the data-driven hypothesis, others need the final reconstruction results. For established phenotypic traits, summarized in OIV [2009] the coarse categorizations could be improved by introducing quantified measures, for example for the density of grape clusters. For those measures a mapping between numeric values and categories has to be found to be able to compare current and past manual measurements with the automatically generated ones. This section has also shown that new traits can now be considered that were not possible before, at least not without taking the grape clusters off the

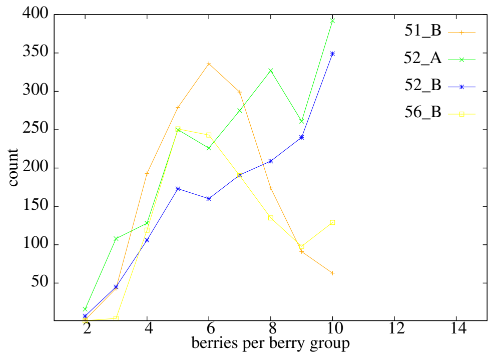


Figure 6.16: For exemplars 51_B and 56_B the distribution of berry group sizes approximately follows a Lognormal distribution. For the other two exemplars there are too many large berry groups.

plant and removing the berries. Also, this broad selection gives the opportunity to combine traits to more complex ones.

6.5 Summary

This chapter had two parts. In the first three sections a detailed evaluation was given by comparing results from the reconstruction approach with manually created ground truth reconstructions. In the fourth section a selection of phenotypic traits was presented.

The evaluation was done on four grapevine grape clusters in development stage BBCH89 'ripe berries,' and split into three parts: Numbers of components, positions of components, and lengths of components. The purpose was to separately evaluate the structure and the geometry of results. The evaluation showed that, in general, the results are already precise and reliable, i.e., for most evaluated measures the median values were close to the ground truth values and the standard deviations were low. But there are also some shortcomings. For example, in two cases there were too many newly inserted berries, berry groups are too large, and there are not enough subtwigs. To overcome these shortcomings there are several options. For the biological model it would be helpful to investigate more exemplars as the basis for a better capture of the plant's variability. This would also help in discovering correlations between parameters. The berry detection must be enhanced in the direction that more berries are found and that the radii and positions are estimated more precisely. The addition of berries in the occluded areas could be improved by first sampling a radius and then a position.

The phenotyping section showed how established descriptors can be computed and improved to quantified measures and how the reconstruction results can be used to investigate new traits and combinations of traits that could not be considered so far, without much manual labor and/or destroying the plant. Of course, the quality of the derivation of those traits relies on the quality of the reconstruction results. An improvement there will therefore also make the phenotypic traits more reliable.

Chapter 7

Summary and Outlook

The aim of this thesis was to develop a method for the reconstruction of plant architectures from sensor data as the basis for an automized phenotyping in order to widen the phenotyping bottleneck. In the context of the CROP.SENSe.net project the concentration was on grapevine grape clusters, and in particular on the development stage BBCH89 'ripe berries.' The main challenges were the complexity of plant architectures, the variability of architectures among cultivars and exemplars of the same cultivar, the disconnectedness of the sensor data, and the occlusions of parts of the plant.

To tackle these challenges a general approach for the reconstruction was presented that consisted of mainly two parts. (1) A model of the plant that consisted of a structural model that was enhanced with an automatic parameter value estimation to form the biological model. (2) The reconstruction that was based on the model and comprised of three steps that enabled a coarse-to-fine refinement of reconstruction results and phenotyping capabilities. This general method was implemented in two applications. One application was the reconstruction of grapevine grape clusters in three different development stages that posed different challenges in terms of complexity and occlusions. The concentration was on grape clusters in full ripeness, since they are the most challenging in terms of occlusions and a well-designed model. The second application was on the reconstruction of veines from grapevine leaves.

The application on fully ripe grape clusters was the basis for a detailed evaluation of the procedure. Investigated were the numbers of occurrences of the different component types by comparison to ground truth values, the placement of components in space by a distance function between a reconstruction and ground truth values, and the lengths of branches by comparison to ground truth values. In general, the evaluation showed that the presented method is able to reliably compute good reconstructions. Reliability was tested by generating 100 reconstruction results each for every of the four exemplars that were used for the evaluation. In a fourth investigation the focus was on the phenotyping. It was shown that established phenotypic traits can be computed and that they can even be made more precise by computing quantified measures instead of coarse categorizations. It was also shown that new traits can be found, which are not possible to determine manually.

The presented results are already very promising. On that basis there are several interesting aspects for future work on the approach and for further applications of the results.

Jump Selection: At the moment every available jump is executed with the same probability. There are at least two ways in which this can be enhanced. (1) Jumps can be grouped into categories that represent how much the structure is changed by the jumps. Then one can couple this with Simulated Annealing and mostly execute jumps that implement large changes at high temperatures and as the temperature decreases relay the probabilities toward the jumps that implement only small changes. This way the hypothesis space would be explored better and it would allow for a larger set of more specialized jumps. New jumps could be, for example, the splitting and combining of subtwigs. (2) Jumps can be selected on the basis of the available sensor data. Whenever there are parts of the reconstructed structure that already explain the sensor data quite well, jumps would tend to ignore these areas and concentrate more on the occluded parts.

Whole Plant / Other Plants: So far, the presented approach was applied to grape clusters and vine leaves. This can be used as the basis for an application of the approach to the whole grapevine plant. One can combine the existing implementations with a reconstruction of trunk and canes. As was done with grape clusters, the whole plant reconstruction can also be applied to different development stages. Furthermore it would be interesting to apply the approach to other branching plant structures, for example, to trees.

Sensors: In the presented implementation a laser rangefinder was used. On the one hand this has the advantage of highly dense and very precise data. But on the other hand the data lack, for example, color information. There are different parts of the plant with different colors and they can not be distinguished by range information alone. Therefore it would be interesting to apply the approach to combined range and color data. There are several possibilities to do so. One could fuse the existing sensor setup with data from a single camera, use a stereo camera setup and compute point clouds with color information or use an RGBD sensor like Microsoft's Kinect. These additional data would be helpful in every case where differently colored components of the structure can be seen.

Evaluation of phenotypic traits: Reconstruction results can be archived. With manual phenotyping only the values of traits can be stored, because otherwise plants had to be conserved. This offers many applications: (1) On the stored results one can calculate and track phenotypic traits over a period of several years. (2) It is possible to derive new traits and for newly discovered traits it is possible to evaluate them on the results from past years. (3)

One can investigate correlations between phenotypic traits and find out if these correlations are specific to a certain cultivar or if they are valid for a set of cultivars.

Scale Space: The idea of a coarse-to-fine reconstruction can be transferred to different scales. In a vineyard, for example, the coarsest scale could be the vineyard as a whole, then the separate rows of vines, refining to single vines, single canes on the vines and, finally, the grape clusters or leaves of the finest scale. For each scale a certain set of phenotypic traits could be derived. One trait that is valid for all scales is the yield, which gets more and more accurately approximated as the scale gets finer.

Appendix A

Probability Theory

The method of Reversible Jump Markov Chain Monte Carlo has a probabilistic nature. For a better understanding of the method, this appendix introduces basic concepts and notations of probability theory. Let X a discrete or continuous *random variable*. A random variable can take a value in a certain range of values with certain probabilities. If x is a specific value for the random variable X, then

$$p(X = x) = p(x)$$

denotes the probability that variable X takes the value x, which is also called the *marginal distribution*. Furthermore for discrete random variables it holds that

$$0 \leq p(x) \leq 1 \ \forall x,$$

and that the integral (or the sum in case of discrete random variables) over all possible values of X is exactly 1:

$$\int p(x)\, dx = 1.$$

For two random variables X and Y the *joint distribution* $p(X = x,\ Y = y)$ gives the probability that X has the value x and at the same time Y has the value y. It holds that

$$\begin{aligned} p(X = x,\ Y = y) &= p(x,\ y) \\ &= p(x \mid y) \cdot p(y) \\ &= p(y \mid x) \cdot p(x), \end{aligned}$$

where $p(x \mid y)$ is the *conditional probability* that describes the probability that X takes a certain value x under the condition that Y has already taken a certain value y. If X and Y are *independent* it holds that

$$p(x, y) = p(x) \cdot p(y).$$

Furthermore, for $p(y) > 0$ it is true that

$$p(x \mid y) = \frac{p(x, y)}{p(y)}.$$

And if X and Y again are independent, then it is true that

$$p(x \mid y) = \frac{p(x) \cdot p(y)}{p(y)} = p(x),$$

so knowledge about the value of Y has no influence on the probability of X taking a specific value. Also important is the *law of total probability*:

$$p(x) = \int p(x \mid y) \cdot p(y)\, dy$$

So if one cannot infer $p(x)$ directly it can still be computed by calculating the conditional probability over all values of y. Now one can derive *Bayes' Theorem*

$$\begin{aligned} p(x \mid y) &= \frac{p(x, y)}{p(y)} \\ &= \frac{p(y \mid x) \cdot p(x)}{p(y)}, \end{aligned}$$

where $p(y \mid x)$ is often called the *likelihood*, $p(x)$ the prior, and $p(y)$ the *normalizer*. The normalizer often can not be calculated directly, but indirectly by the law of total probability.

List of Figures

List of Tables

List of Algorithms

Glossary

Acropetal Description of the order of organs on a stem from the base to the tip. For example, in an acropetal development of organs the youngest ones are located at the tip. (Compare basipetal)
Ampelography Identification and classification of grapevine cultivars.
Anthesis The state of opened flowers.

Basipetal Description of the order of organs on a stem from the tip to the base. (Compare acropetal)
Berry The fruit of the grapevine plant. (Synonym: grape)
Berry group A set of berries and pedicels that branch from the same node.
Blade The laminar part of a leaf.
Bud Part of the plant that develops into a flower, a leaf, or a new branch.
Bunch of grapes The panicle of the grapevine. (Synonym: grape cluster)

Cane Mature shoot of the grapevine plant, that holds the grape clusters, leaves, and tendrils.
Cordon Stem that holds the fruiting canes.
Cultivar A group of cultivated plants, that stands out due to some features (mostly morphological, but also physiological, chemical, etc.) and keeps these cultivar-specific features during reproduction.

Enology The art and science of wine making.

Flower Part of the plant from which the seed or fruit develops.

Grape The fruit of the grapevine plant. (Synonym: berry)
Grape cluster The panicle of the grapevine. (Synonym: bunch of grapes)

Inflorescence Definable part of a plant, that carries the flowers and whose terminal meristems are consumed for the building of flowers.

Internode Part of the stem between nodes that does not carry any leaves or inflorescences.

Meristem Tissue that permanently is in readiness for cell division. (Opposite: permanent tissue)

Node Area of the stem, where one or several leaves or inflorescences grow.

Panicle Inflorescence where the main stem and the lateral branches end in flowers (terminal flowers) and the branching degree increases in downward direction more or less regularly. In its typical form the panicle therefore has a cylindrical outline.

Pedicel A small stem that attaches a berry to the stem system.

Peduncle A stem that connects the grape cluster to the cane.

Permanent tissue Tissue of cells that usually do not divide any more.

Raceme Simple inflorescence with a continuous main stem and pediculated lateral flowers. In ancient times the latin word racemus was mostly used for the inflorescence of the grapevine, which in present botanical linguistic usage is a panicle.

Rachis The main stem of a grape cluster.

Rudiment The first recognizable state of development of an organ.

Shoot A leafy sprout of a plant, i.e., a stem including its leaves (rarely used for the stem only). From the primary shoot lateral shoots emerge by branching.

Stem One of the basic organs of a plant, marked by the ability of foliation and branching. Forms and names of the stem are diverse. For example, when distinguishing between the main stem and its branches, one uses stalk and branch, also the very thick main stem of a tree is called trunk, an unbranched, hollow stem, like that of the grasses, culm.

Stem system A panicle without any grapes.

Tendril Plant component that holds the plant onto a bearer.

Training system The form a viticulturist wants a grapevine to grow.

Trellis system A system of posts and wires a grapevine is grown into.

Trunk The basic above-ground structure that holds the remaining components.

Vein Part of the leaf responsible for the transport of water and assimilates and often also for mechanical strengthening.

Veraison The change of color of the grapes. Goes along with the ripening of the grapes.

Viticulture The science, production, and study of grapes.

Bibliography

Al-awadhi, F., Hurn, M., and Jennison, C. (2004). Improving the Acceptance Rate of Reversible Jump MCMC Proposals. *Statistics and Probability Letters*, 69(2):189–198.

Al-awadhi, F., Hurn, M., and Jennison, C. (2011). Three-dimensional Bayesian image analysis and confocal microscopy. *Journal of Applied Statistics*, 38(1):29–46.

Alberch, P. (1991). From genes to phenotype: dynamical systems and evolvability. *Genetica*, 84(1):5–11.

Alenya, G., Dellen, B., and Torras, C. (2011). 3D modelling of leaves from color and ToF data for robotized plant measuring. In *IEEE International Conference on Robotics and Automation*, pages 3408–3414.

Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I. (2003). An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1-2):5–43.

Balfer, J. (2012). 3D Skeletonization for the Analysis of Grapevine Structure. Master's thesis, Rheinische Friedrich-Wilhelms-Universität Bonn.

Balfer, J., Schöler, F., and Steinhage, V. (2013). Semantic Skeletonization for Structural Plant Analysis. In *International Conference on Functional-Structural Plant Models*, pages 42–44.

Barthélémy, D. and Caraglio, Y. (2007). Plant Architecture: A Dynamic, Multilevel and Comprehensive Approach to Plant Form, Structure and Ontogeny. *Annals of Botany*, 99(3):375–407.

Binney, J. and Sukhatme, G. S. (2009). 3D Tree Reconstruction From Laser Range Data. In *IEEE International Conference on Robotics and Automation*, pages 1321–1326.

Bishop, C. M. (2007). *Pattern Recognition and Machine Learning*. Springer.

Boudon, F., Preuksakarn, C., Ferraro, P., Diener, J., Nikinmaa, E., and Godin, C. (2013). Quantitative assessment of automatic reconstructions of branching systems. In *International Conference on Functional-Structural Plant Models*, pages 64–66.

Brenner, C. and Ripperda, N. (2006). Extraction of Facades using RJMCMC and Constraint Equations. *ISPRS Commission III Symposium*, 36(3):155–160.

Brooks, S. P., Giudici, P., and Roberts, G. O. (2003). Efficient construction of reversible jump Markov chain Monte Carlo proposal distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(1):3–39.

Buck-Sorlin, G., Hemmerling, R., Kniemeyer, O., Burema, B., and Kurth, W. (2008). A Rule-based Model of Barley Morphogenesis, with Special Respect to Shading and Gibberellic Acid Signal Transduction. *Annals of Botany*, 101(8):1109–1123.

Buck-Sorlin, G., Kniemeyer, O., and Kurth, W. (2006). Physiologie und Morphologie der Pappel (Populus sp.) modelliert mit Relationalen Wachstumsgrammatiken. In *DVFFA Sektion Forstliche Biometrie und Informatik*, pages 1–11.

Bucksch, A. K. (2011). Revealing the Skeleton from Imperfect Point Clouds. PhD thesis, Delft University of Technology.

Bucksch, A. K., Lindenbergh, R. C., and Menenti, M. (2009). SkelTre - fast skeletonisation for imperfect point cloud data of botanic trees. In *Eurographics / 3D Object Retrieval Workshop*, pages 13–20.

Busemeyer, L., Mentrup, D., Möller, K., Wunder, E., Alheit, K., Hahn, V., Maurer, H. P., Reif, J. C., Würschum, T., Müller, J., Rahe, F., and Ruckelshausen, A. (2013). BreedVision - A Multi-Sensor Platform for Non-Destructive Field-Based Phenotyping in Plant Breeding. *Sensors*, 13(3):2830–2847.

Cao, J., Tagliasacchi, A., Olson, M., Zhang, H., and Su, Z. (2010). Point cloud skeletons via Laplacian-based contraction. In *Shape Modeling International*, pages 187–197.

Cerny, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(l):41–51.

Chen, Y.-L., Chen, B.-Y., Lai, S.-H., and Nishita, T. (2010). Binary Orientation Trees for Volume and Surface Reconstruction from Unoriented Point Clouds. *Computer Graphics Forum*, 29(7):2011–2019.

Chen, Y.-P. P. and Colomb, R. M. (2003). Database Technologies for L-System Simulations in Virtual Plant Applications on Bioinformatics. *Knowledge and Information Systems*, 5(3):288–314.

Chéné, Y., Rousseau, D., Lucidarme, P., Bertheloot, J., Caffier, V., Morel, P., Belin, E., and Chapeau-Blondeau, F. (2012). On the use of depth camera for 3D phenotyping of entire plants. *Computers and Electronics in Agriculture*, 82:122–127.

Chib, S. and Greenberg, E. (1995). Understanding the Metropolis-Hastings Algorithm. *The American Statistician*, 49(4):327–335.

Colin, F. and Houllier, F. (1991). Branchiness of Norway spruce in north-eastern France: modelling vertical trends in maximum nodal branch size. *Annals of Forest Science*, 48(6):679–693.

Cornea, N. D., Silver, D., and P., M. (2007). Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548.

Dick, A., Torr, P., and Cipolla, R. (2004). Modelling and Interpretation of Architecture from Several Images. *International Journal of Computer Vision*, 60(2):111–134.

Dornbusch, T., Wernecke, P., and Diepenbrock, W. (2007). Description and visualization of graminaceous plants with an organ-based 3D architectural model, exemplified for spring barley (Hordeum vulgare L.). *The Visual Computer*, 23(8):569–581.

Fan, Y. and Brooks, S. P. (2000). Bayesian Modelling of Prehistoric Corbelled Domes. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 49(3):339–354.

Ferraro, P. and Godin, C. (2000). A distance measure between plant architectures. *Annals of Forest Science*, 57(5-6):445–461.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Fourcaud, T., Dupuy, L., Sellier, D., Ancelin, P., and Lac, P. (2004). Analysis of the Relationship Between Tree Structure and Biomechanical Functions. In *International Workshop on Functional-Structural Plant Models*, pages 302–306.

Fourcaud, T., Zhang, X., Stokes, A., Lambers, H., and Körner, C. (2008). Plant Growth Modelling and Applications: The Increasing Importance of Plant Architecture in Growth Models. *Annals of Botany*, 101(8):1053–1063.

Frijters, D. (1978). Principles of Simulation of Inflorescence Development. *Annals of Botany*, 42(3):549–560.

Ganster, B. and Klein, R. (2008). 1-2-tree: Semantic Modeling and Editing of Trees. In *Vision, Modeling, and Visualization*, pages 51–60.

Godin, C. (2000). Representing and encoding plant architecture: A review. *Annals of Forest Science*, 57(5-6):413–438.

Godin, C. and Caraglio, Y. (1998). A Multiscale Model of Plant Topological Structures. *Journal of Theoretical Biology*, 191(1):1–46.

Godin, C., Costes, E., and Sinoquet, H. (1999). A Method for Describing Plant Architecture which Integrates Topology and Geometry. *Annals of Botany*, 84(3):343–357.

Godin, C. and Sinoquet, H. (2005). Functional-structural plant modelling. *New Phytologist*, 166(3):705–708.

Goel, N. and Rozehnal, I. (1991). Some non-biological Applications of L-Systems. *International Journal of General Systems*, 18(4):321–405.

Green, J. M., Appel, H., Rehrig, E. M., Harnsomburana, J., Chang, J.-F., Balint-Kurti, P., and Shyu, C.-R. (2012). PhenoPhyte: a flexible affordable method to quantify 2D phenotypes from imagery. *Plant Methods*, 8:45.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732.

Green, P. J. (2003). Trans-dimensional Markov chain Monte Carlo. In Green, P. J., Hjort, N. L., and Richardson, S., editors, *Highly Structured Stochastic Systems*, pages 179–198. Oxford University Press.

Guédon, Y., Barthélémy, D., Caraglio, Y., and Costes, E. (2001). Pattern Analysis in Branching and Axillary Flowering Sequences. *Journal of Theoretical Biology*, 212(4):481–520.

Hajek, B. (1988). Cooling Schedules for Optimal Annealing. *Mathematics of Operations Research*, 13(2):311–330.

Hall, A., Louis, J., and Lamb, D. (2003). Characterising and mapping vineyard canopy using high-spatial-resolution aerial multispectral images. *Computers & Geosciences*, 29(7):813–822.

Hastie, D. I. (2005). Towards Automatic Reversible Jump Markov Chain Monte Carlo. PhD thesis, University of Bristol.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Hellman, E. W. (2003). Grapevine Structure and Function. In Hellman, E. W., editor, *Oregon Viticulture*, pages 5–19. Oregon State University Press.

Hemmerling, R., Evers, J. B., Smoleňová, K., Buck-Sorlin, G., and Kurth, W. (2013). Extension of the GroIMP modelling platform to allow easy specification of differential equations describing biological processes within plant models. *Computers and Electronics in Agriculture*, 92:1–8.

Hemmerling, R., Kniemeyer, O., Lanwert, D., Kurth, W., and Buck-Sorlin, G. (2008). The Rule-Based Language XL and the Modelling Environment GroIMP Illustrated with Simulated Tree Competition. *Functional Plant Biology*, 35(10):739–743.

Hijazi, Y., Bechmann, D., Cazier, D., Kern, C., and Thery, S. (2010). Fully-automatic Branching Reconstruction Algorithm: Application to Vascular Trees. In *Shape Modeling International*, pages 221–225.

Honda, H. (1971). Description of the form of trees by the parameters of the tree-like body: Effects of the branching angle and the branch length on the shape of the tree-like body. *Journal of Theoretical Biology*, 31(2):331–338.

Huang, C., Yang, W., Duan, L., Jiang, N., Chen, G., Xiong, L., and Liu, Q. (2013a). Rice panicle length measuring system based on dual-camera imaging. *Computers and Electronics in Agriculture*, 98:158–165.

Huang, C.-Y., Jheng, W.-T., Tai, W.-K., Chang, C.-C., and Way, D.-L. (2013b). Procedural grape bunch modeling. *Computers & Graphics*, 37(4):225–237.

Ijiri, T., Owada, S., and Igarashi, T. (2006). The Sketch L-System: Global Control of Tree Modeling Using Free-Form Strokes. In *International Symposium on Smart Graphics*, pages 138–146.

IPGRI, UPOV, and OIV (1997). Descriptors for Grapevine (Vitis ssp.). International Union for the Protection of New Varieties of Plants, Geneva, Switzerland/Office International de la Vigne et du Vin, Paris, France/International Plant Genetic Resources Institute, Rome, Italy.

Jaeger, M. and Reffye, P. H. (1992). Basic concepts of computer simulation of plant growth. *Journal of Biosciences*, 17(3):275–291.

Jelden, P. (2014). 3D Rekonstruktion von Trauben im Stadium des Reifebeginns durch modellbasierte Analyse von hochaufgelösten Laserabstandsdaten. Diplomarbeit, Rheinische Friedrich-Wilhelms-Universität Bonn.

Jirasek, C., Prusinkiewicz, P., and Moulia, B. (2000). Integrating biomechanics into developmental plant models expressed using L-systems. In *Plant Biomechanics Conference*, pages 615–624.

Khan, Z., Balch, T., and Dellaert, F. (2005). MCMC-based particle filtering for tracking a variable number of interacting targets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1805–1819.

Kicherer, A., Roscher, R., Herzog, K., Šimon, S., Förstner, W., and Töpfer, R. (2013). BAT (Berry Analysis Tool): A high-throughput image interpretation tool to acquire the number, diameter, and volume of grapevine berries. *Vitis*, 52(3):129–135.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220(4598):671–680.

Kniemeyer, O. (2008). Design and Implementation of a Graph Grammar Based Language for Functional-Structural Plant Modelling. PhD thesis, Technical University Cottbus.

Kniemeyer, O., Barczik, G., Hemmerling, R., and Kurth, W. (2008). Relational Growth Grammars – A Parallel Graph Transformation Approach with Applications in Biology and Architecture. In Gasteratos, A., Vincze, M., and Tsotsos, J. K., editors, *Computer Vision Systems*, volume 5008 of *Lecture Notes in Computer Science*, pages 152–167. Springer.

Kolev, K., Klodt, M., Brox, T., and Cremers, D. (2009). Continuous Global Optimization in Multiview 3D Reconstruction. *International Journal of Computer Vision*, 84(1):80–96.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

Kuhn, H. W. (1956). Variants of the hungarian method for assignment problems. *Naval Research Logistics Quarterly*, 3(4):253–258.

Lafarge, F., Descombes, X., Zerubia, J., and Pierrot-Deseilligny, M. (2008). Building reconstruction from a single DEM. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Lebon, E., Pellegrino, A., Tardieu, F., and Lecoeur, J. (2004). Shoot Development in Grapevine (Vitis vinifera) is Affected by the Modular Branching Pattern of the Stem and Intra- and Inter-shoot Trophic Competition. *Annals of Botany*, 93(3):263–274.

Lenz, C. (2014). 3D-Rekonstruktion von Trauben im Schrotkornstadium durch modellbasierte Analyse von hochaufgelösten Laserabstandsdaten. Bachelor thesis, Rheinische Friedrich-Wilhelms-Universität Bonn.

Li, Y., Wu, X., Chrysatou, Y., Sharf, A., Cohen-or, D., and Mitra, N. J. (2011). GlobFit: consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics*, 30(4):52.

Lindenmayer, A. (1968a). Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology*, 18(3):280–299.

Lindenmayer, A. (1968b). Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology*, 18(3):300–315.

Lipp, M., Wonka, P., and Wimmer, M. (2009). Parallel Generation of L-Systems. In *Vision, Modeling, and Visualization*, pages 205–214.

Livny, Y., Yan, F., Olson, M., Chen, B., Zhang, H., and El-sana, J. (2010). Automatic reconstruction of tree skeletal structures from point clouds. *ACM Transactions on Graphics*, 29(6):151.

Lorenz, D. H., Eichhorn, K. W., Bleiholder, H., Klose, R., Meier, U., and Weber, E. (1994). Phänologische Entwicklungsstadien der Weinrebe (Vitis vinifera L. ssp. vinifera). – Codierung und Beschreibung nach der erweiterten BBCH-Skala. *Die Weinwissenschaft - Viticultural and Enological Sciences*, 49(2):66–70.

Lorenz, D. H., Eichhorn, K. W., Bleiholder, H., Klose, R., Meier, U., and Weber, E. (1995). Growth Stages of the Grapevine: Phenological growth stages of the grapevine (Vitis vinifera L. ssp. vinifera) – Codes and descriptions according to the extended BBCH scale. *Australian Journal of Grape and Wine Research*, 1(2):100–103.

Louarn, G., Dauzat, J., Lecoeur, J., and Lebon, E. (2008). Influence of trellis system and shoot positioning on light interception and distribution in two grapevine cultivars with different architectures: an original approach based on 3D canopy modelling. *Australian Journal of Grape and Wine Research*, 14(3):143–152.

Louarn, G., Guedon, Y., Lecoeur, J., and Lebon, E. (2007). Quantitative Analysis of the Phenotypic Variability of Shoot Architecture in Two Grapevine (Vitis vinifera) Cultivars. *Annals of Botany*, 99(3):425–437.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., and Teller, A. H. (1953). Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21(6):1087–1092.

Mullins, M. G., Bouquet, A., and Williams, L. E. (1992). *Biology of the Grapevine*. Cambridge University Press.

Munkres, J. (1957). Algorithms for the Assignment and Transportation Problems. *SIAM Journal on Applied Mathematics*, 5(1):32–38.

Musialski, P., Wonka, P., Aliaga, D. G., Wimmer, M., Gool, L. V., and Purgathofer, W. (2012). A Survey of Urban Reconstruction. *Computer Graphics Forum*, 32(6):146–177.

Měch, R. and Prusinkiewicz, P. (1996). Visual Models of Plants Interacting with their Environment. In *Conference on Computer Graphics and Interactive Techniques*, pages 397–410.

Ning, X., Zhang, X., and Wang, Y. (2009). Tree Segmentation from Scanned Scene Data. In *International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications*, pages 360–367.

Nuske, S., Achar, S., Bates, T., Narasimhan, S., and Singh, S. (2011). Yield estimation in vineyards by visual grape detection. In *International Conference on Intelligent Robots and Systems*, pages 2352–2358.

Ochmann, S., Vock, R., Wessel, R., Tamke, M., and Klein, R. (2014). Automatic Generation of Structural Building Descriptions from 3D Point Cloud Scans. In *International Conference on Computer Graphics Theory and Applications*.

OIV (2009). OIV descriptor list for grape varieties and vitis species (2nd edition).

Pallas, B., Loi, C., Christophe, A., Cournède, P.-H., and Lecoeur, J. (2009). A Stochastic Growth Model of Grapevine with Full Interaction Between Environment, Trophic Competition and Plant Development. In *International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications*, pages 95–102.

Paulus, S. (work in progress, 2014). Nutzung von Laserscannern zur Phänotypisierung von Pflanzen als Vorbereitung für den Einsatz im Hochdurchsatz. PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn.

Paulus, S., Dupuis, J., Mahlein, A.-K., and Kuhlmann, H. (2013). Surface feature based classification of plant organs from 3D laserscanned point clouds for plant phenotyping. *BMC Bioinformatics*, 14:238.

Pirk, S., Stava, O., Kratt, J., Said, M. A. M., Neubert, B., Měch, R., Benes, B., and Deussen, O. (2012). Plastic trees: interactive self-adapting botanical tree models. *ACM Transactions on Graphics*, 31(4):50.

Power, J. L., Bernheim Brush, A. J., Prusinkiewicz, P., and Salesin, D. H. (1999). Interactive Arrangement of Botanical L – System Models. In *Symposium on Interactive 3D Graphics*, pages 175–182.

Prusinkiewicz, P. (1986). Graphical Applications of L-Systems. In *Graphics Interface*, pages 247–253.

Prusinkiewicz, P. (1998). Modeling of spatial structure and development of plants: a review. *Scientia Horticulturae*, 74(1-2):113–149.

Prusinkiewicz, P., James, M., and Měch, R. (1994). Synthetic Topiary. In *Conference on Computer Graphics and Interactive Techniques*, pages 351–358.

Prusinkiewicz, P., Karwowski, R., Měch, R., and Hanan, J. (2000). L-Studio/cpfg: A Software System for Modeling Plants. In Nagl, M., Schürr, A., and Münch, M., editors, *Applications of Graph Transformations with Industrial Relevance*, volume 1779 of *Lecture Notes in Computer Science*, pages 457–464. Springer.

Prusinkiewicz, P. and Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. Springer.

Prusinkiewicz, P., Mündermann, L., Karwowski, R., and Lane, B. (2001). The Use of Positional Information in the Modeling of Plants. In *Conference on Computer Graphics and Interactive Techniques*, pages 289–300.

Raumonen, P., Casella, E., Disney, M., Akerblom, M., and Kaasalainen, M. (2013). Fast automatic method for constructing topologically and geometrically precise tree models from TLS Data. In *International Conference on Functional-Structural Plant Models*, pages 89–91.

Reisner-Kollmann, I., Luksch, C., and Schwärzler, M. (2011). Reconstructing Buildings as Textured Low Poly Meshes from Point Clouds and Images. In *Eurographics*, pages 17–20.

Richardson, S. and Green, P. J. (1997). On Bayesian Analysis of Mixtures with an Unknown Number of Components. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59(4):731–792.

Ripperda, N. (2008). Grammar Based Facade Reconstruction using RjMCMC. *Photogrammetrie, Fernerkundung, Geoinformation*, 2:83–92.

Ripperda, N. and Brenner, C. (2009). Application of a Formal Grammar to Facade Reconstruction in Semiautomatic and Automatic Environments. In *AGILE Conference on GIScience*.

Roscher, R., Herzog, K., Kunkel, A., Kicherer, A., Töpfer, R., and Förstner, W. (2014). Automated image analysis framework for high-throughput determination of grapevine berry sizes using conditional random fields. *Computers and Electronics in Agriculture*, 100:148–158.

Roth-Nebelsick, A., Uhl, D., Mosbrugger, V., and Kerp, H. (2001). Evolution and Function of Leaf Venation Architecture: A Review. *Annals of Botany*, 87(5):553–566.

Sapp, J. (1983). The struggle for authority in the field of heredity, 1900-1932: New perspectives on the rise of genetics. *Journal of the History of Biology*, 16(3):311–342.

Schlecht, J. W., Barnard, K., Spriggs, E., and Pryor, B. (2007). Inferring Grammar-based Structure Models from 3D Microscopy Data. In *Conference on Computer Vision and Pattern Recognition*, pages 469–476.

Schnabel, R., Degener, P., and Klein, R. (2009). Completion and Reconstruction with Primitive Shapes. *Computer Graphics Forum*, 28(2):503–512.

Schnabel, R., Wahl, R., and Klein, R. (2007). Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, 26(2):214–226.

Schneider, D. (2012). Rekonstruktion von 3D-Blattstrukturen über ein Markov-Chain-Monte-Carlo Verfahren. Diplomarbeit, Rheinische Friedrich-Wilhelms-Universität Bonn.

Schöler, F., Balfer, J., and Steinhage, V. (2013). Automated Parameter Estimation for a Plant Architecture Model. In *International Conference on Functional-Structural Plant Models*, pages 22–24.

Schöler, F. and Steinhage, V. (2012). Towards an Automated 3D Reconstruction of Plant Architecture. In Schürr, A., Varró, D., and Varró, G., editors, *Applications of Graph Transformations with Industrial Relevance*, volume 7233 of *Lecture Notes in Computer Science*, pages 51–64. Springer.

Shavrukov, Y. N., Dry, I. B., and Thomas, M. R. (2004). Inflorescence and bunch architecture development in Vitis vinifera L. *Australian Journal of Grape and Wine Research*, 10(2):116–124.

Shlyakhter, I., Rozenoer, M., Dorsey, J., and Teller, S. (2001). Reconstructing 3D Tree Models from Instrumented Photographs. *IEEE Computer Graphics and Applications*, 21(3):53–61.

Steinhage, V., Schöler, F., and Balfer, J. (2012). A Model-Based Approach to High Performance Phenotyping. In *International Conference on Informatics for Environmental Protection*, pages 303–310.

Vail, W. E. and Marois, J. J. (1991). Grape Cluster Architecture and the Susceptibility of Berries to Botrytis cinerea. *Phytopathology*, 81(2):188–191.

Vanegas, C. A., Aliaga, D. G., and Benevs, B. (2010). Building Reconstruction using Manhattan-World Grammars. In *Conference on Computer Vision and Pattern Recognition*, pages 358–365.

Waagepetersen, R. and Sorensen, D. (2001). A Tutorial on Reversible Jump MCMC with a View toward Applications in QTL-mapping. *International Statistical Review*, 69(1):49–61.

Wahl, R., Schnabel, R., and Klein, R. (2008). From Detailed Digital Surface Models to City Models Using Constrained Simplification. *Photogrammetrie Fernerkundung GeoInformation*, 3:207–215.

Watanabe, T., Hanan, J. S., Room, P. M., Hasegawa, T., Nakagawa, H., and Takahashi, W. (2005). Rice Morphogenesis and Plant Architecture: Measurement, Specification and the Reconstruction of Structural Development by 3D Architectural Modelling. *Annals of Botany*, 95(7):1131–1143.

Weinmann, M., Ruiters, R., Osep, A., Schwartz, C., and Klein, R. (2012). Fusing Structured Light Consistency and Helmholtz Normals for 3D Reconstruction. In *British Machine Vision Conference*, pages 108.1–108.12.

Worth, P. and Stepney, S. (2005). Growing Music: Musical Interpretations of L-systems. In Rothlauf, F., Branke, J., Cagnoni, S., Corne, D. W., Drechsler, R., Machado, Y. J. P., Marchiori, E., Romero, J., Smith, G. D., and Squillero, G., editors, *Applications on Evolutionary Computing*, volume 3449 of *Lecture Notes in Computer Science*, pages 545–550. Springer.

Xu, L., Henke, M., Zhu, J., Kurth, W., and Buck-Sorlin, G. (2009). A Rule-Based Functional-Structural Model of Rice Considering Source and Sink Functions. In *International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications*, pages 245–252.

Yan, D.-M., Wintz, J., Mourrain, B., Wang, W., Boudon, F., and Godin, C. (2009). Efficient and robust reconstruction of botanical branching structure from laser scanned points. In *IEEE International Conference on Computer-Aided Design and Computer Graphics*, pages 572–575.

Zaniewski, T. and Bangay, S. (2003). Simulation and Visualization of Fire using Extended Lindenmayer Systems. In *International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, pages 39–48.