I N A U G U R A L - D I S S E R T A T I O N

zur

Erlangung der Doktorwürde

der

Naturwissenschaftlich-Mathematischen Gesamtfakultät

der

Ruprecht-Karls-Universität

Heidelberg

vorgelegt von

M.Sc. - Somporn Chuai-Aree

aus Nakhon Si Thammarat, Thailand

Tag der mündlichen Prüfung: 08. Juni 2009

Modeling, Simulation and Visualization of Plant Growth

Gutachter: Prof. Dr. Dr. h.c. mult. Willi Jäger

Prof. Dr. Dr. h.c. Hans Georg Bock

Zusammmenfassung

Pflanzenmodellierung ist ein interessantes und herausforderndes Thema für die wissenschaftlich interdisziplinäre Forschung im Bereich der Mathematik, Biologie, Botanik, Agrarwirtschaft und Informatik. Im Rahmen dieser Dissertation wird die auf *Lindenmayer Systeme* (L-Systeme) und *Partikel Systeme* (PT-Systeme) basierende Modellierung, Simulation und Visualisierung von Pflanzenwachstum präsentiert und anhand von zwei Methoden zur Erzeugung von Pflanzenstruktur vorgestellt. Die erste Methode basiert auf *Geklammerten*, *Stochastischen* und *Parametrischen L-Systemen*. Sie ist für eine präzise Modellierung von bereits bekannten Pflanzenstrukturen geeignet und bietet auch die Möglichkeit, die komplexe Struktur in kleine Bestandteile bezüglich der Produktionsregeln zu zerlegen. In der zweiten Methode wird das PT-System für die Simulation grober Struktur und schneller Produktionsvorgänge eingesetzt, die auf vordefinierter Form und Volumen von Spross und Wurzel der Pflanze basiert. Beide Methoden können für die Modellierung von Pflanzenspross, Wurzel und Blattader eingesetzt werden.

Der Prototyp dieser beiden Methoden ist in einer Weise konstruiert, die die physiologischen Daten der Maße realer Pflanzen berücksichtigt wie beispielsweise Länge und Durchmesser des Internodiums, Länge und Durchmesser der Zweige, Länge und Breite des Blattes, Länge und Breite der Wurzel. Diese Daten werden durch Parameterschätzung mit der Anwendung der *Levenberg-Marquardt Methode* bestimmt, die auf einer N-Puls sigmoidalen Funktion basiert. Alle angepassten Parameter können im Prototyp für die Simulation von Wachstumsverhalten einer Pflanze verwendet werden.

Beide vorgeschlagenen Methoden werden für die künstliche Erzeugung bestimmter Pflanzenarten eingesetzt, die mit L-Systemen vertraute Experten von der Natur ablesen und in ein künstliches Modell konvertieren. Auch schlagen wir hier eine Methode für das Umwandeln der erhobenen Daten in ein künstliches Verzweigungsnetzwerk vor, das sogenannte ,,inverse Problem vom L-System". Dieses inverse Problem vom L-System bietet die Möglichkeit, die Struktur eines Verzweigungsnetzwerks mithilfe von Eingabebildern oder Volumendaten der komplexen Struktur zu rekonstruieren. Die tatsächlich wachsende Wurzel im Bodenvolumen kann mit *Computer Tomography* (CT) gescannt und die Wurzelstruktur aus dem Volumen segmentiert werden. Die endgültige rekonstruierte Struktur wird in L-Systemen basierend auf *Geklammerten* und *Parametrischen L-Systemen* für die Weiterverwendung beschrieben.

Die Struktur und das Wachstum der Wurzelsysteme sind stark von Umgebungsfaktoren im Boden abhängig. Die Diffusionsgleichung und *Richardsgleichung* werden verwendet, um die Diffusion der Nährstoffe und den Fluss des Wassers zu beschreiben. Das Wurzelstystem wächst gleichzeitig und abhängig davon, wie die Diffusion der Nährstoffe und der Fluss des Wassers verläuft. Nährstoff- und Wasseraufnahme werden zu jedem Zeitpunkt des Wachstumsprozesses berechnet.

Diese Dissertation fördert letztendlich neue Methoden für die Modelierung und Simulierung von Pflanzenwachstum aufgrund von Klimafaktoren, die mit einem von uns neu entwickelten Software Tool durchgeführt werden kann. Ergebnisse, die in dieser Dissertation erreicht werden, können in vielen verwandten Gebieten angewendet werden wie zum Beispiel in der Landwirtschaft, Pflanzenmodellierung, Agrarmanagement, Ökonomie, etc. Die Visualisierung des virtuellen Pflanzenwachstums, das mit L-Systemen, PT-System, inversem Problem, Wasserfluss und Nährstoffdiffusion modelliert wird, kann durch die von uns entwickelte Software *PlantVR* (*Plant Virtual Reality*) dargestellt werden.

Schlagworte : Lindenmayer Systeme (L-Systeme), Partikeltransportsysteme (PT-Systeme), modellierendes Pflanzenwachstum, inverses Problem von L-Systemen, Grundwasserströmung und Nährstoffdiffusion (Zerstäubung).

Summary

Plant modeling is an interesting and challenging field in the interdisciplinary scientific researches for integrating the knowledge in mathematics, biology, botany, agriculture and computer science. In this dissertation on modeling, simulation and visualization of plant growth based on *Lindenmayer Systems* (L-systems) and *Particle Transportation Systems* (PT-systems), we propose two possibilities for generating plant structures. The first method is using L-systems based on *Bracketed*, *Stochastic* and *Parametric L-systems.* It is suitable for generating a known plant structure, which is easy to model and decomposes the whole plant structure to small simple components such as production rules. In the second method PT-system is used for simulating the rough structure and fast generating of plant structure by giving predefined plant shape or volume of shoot and root domain. Both methods can be used to generate plant shoots, roots, leaf veins and other branching structures as well.

The prototypes of these two methods are designed for incorporation of measured qualitative and quantitative data from actual plants such as internode length and diameter, branch length and diameter, leaf width and length, root length and diameter. These quantitative data are approximated by parameter estimation using *Levenberg-Marquardt method* based on N-pulses sigmoidal function. All fitted parameters can be used in each prototype to simulate plant growth behavior.

Both systems are used to create artificial plant models from natural plants by expertise in L-systems. In order to describe plant structure in a systematic way, we also propose the method for transforming acquisition data to artificial branching network, the so-called "inverse problem of L-systems". The inverse problem of L-systems provides the way to reconstruct the branching structure from input images or volume data of the complicated network structures. The actual growing root in soil volume can be scanned by *Computed Tomography* (CT) scanners and the root structure can be segmented from the volume. The final reconstructed structure is represented in L-systems description based on *Bracketed* and *Parametric L-systems* for further use.

For the development of root systems and the root growth, the environmental factors in soil profile play an important role. The diffusion equation and *Richards equation* are used to describe nutrient diffusion and water flow in soil volume, respectively. The root system is growing simultaneously during nutrient diffusion and water flow. Nutrient and water uptake are computed at each time step for the next iteration of growth process.

Finally, this dissertation promotes new approaches for modeling and simulation of plant growth depending on environmental factors by using the computer simulation tools. The results obtained in this dissertation can be applied in many disciplinary fields; e.g. agriculture, plant modeling, crop management, economy, etc. The simulation and visualization of plant growth based on L-systems, PT-systems, inverse problem, water flow and nutrient diffusion are presented by our self-developed software tool so-called *PlantVR* (*Plant Virtual Reality*).

Keywords : Lindenmayer systems (L-systems), particle transportation systems (PT-systems), plant modeling, inverse problem of L-systems, water flow and nutrient diffusion.

Acknowledgements

I would like to express my sincere thanks and gratitude to my supervisors, Prof. Dr. Dr. h.c. mult. Willi Jäger, Prof. Dr. Dr. h.c. Hans Georg Bock, *Interdisziplinaräres Zentrum für Wissenschaftliches Rechnen* (IWR), University of Heidelberg, and also Prof. Suchada Siripant, *Advanced Virtual and Intelligent Computing* (AVIC) Research Center, Chulalongkorn University, Bangkok, for their excellent support over the years of my PhD study. I am greatly indebted to these people for their contributions to major ideas in my research, for many useful and inspiring discussions and especially for the excellent supervision which they gave me all the time. I also thank Dr. Susanne Krömker and Dr. Michael Winckler for the interesting discussion about computer graphics and image processing.

Special thanks also go to Prof. Dr. Urich Schurr, Director at ICG-III, *Forschungszentrum Jülich* and Professor of Phytosphere Research, University of Düsseldorf for valuable discussion about plant, leaf and root growth both in Heidelberg and Jülich, Prof. Dr. Rüdiger Hell from *Heidelberg Institute of Plant Sciences* (HIP) at the University of Heidelberg for his suggestions and helps for growing sample plants at HIP together with Michael Schilbach, Adelheid Fuxa at *German Cancer Research Center* (DKFZ), who allowed me to use the *Computed Tomography* (CT) scanner for scanning many pots of plants, Dr. Mariya Ptashnyk for many times discussion on root growth processes, Prof. Dr. Peter Bastian and Dr. Olaf Ippisch for their supports and suggestions on water flow and solute transport calculation, Dr. Peter Kolesik from University of Adelaide for his excellent support on volume data of soil.

I heartily thank all secretaries Ina Scheid, Margret Rothfuß, Gabriela Schocke and Elke Pietschmann for their contributions to the friendly environment in the working groups, and our system administrators for the great jobs they did for me and for our groups. I would like to extend my thanks to Dr. Elfriede Friedmann, Dr. Sanae Rujivan, Dr. Pheakdey Nguonphan, Le Thi Thanh An and all my colleagues in the research groups of *Applied Analysis*, *Visualization and Numerical Geometry* and *Simulation and Optimization* for their friendship, their assistance and for creating the open and constructive atmosphere that made my time there very pleasant.

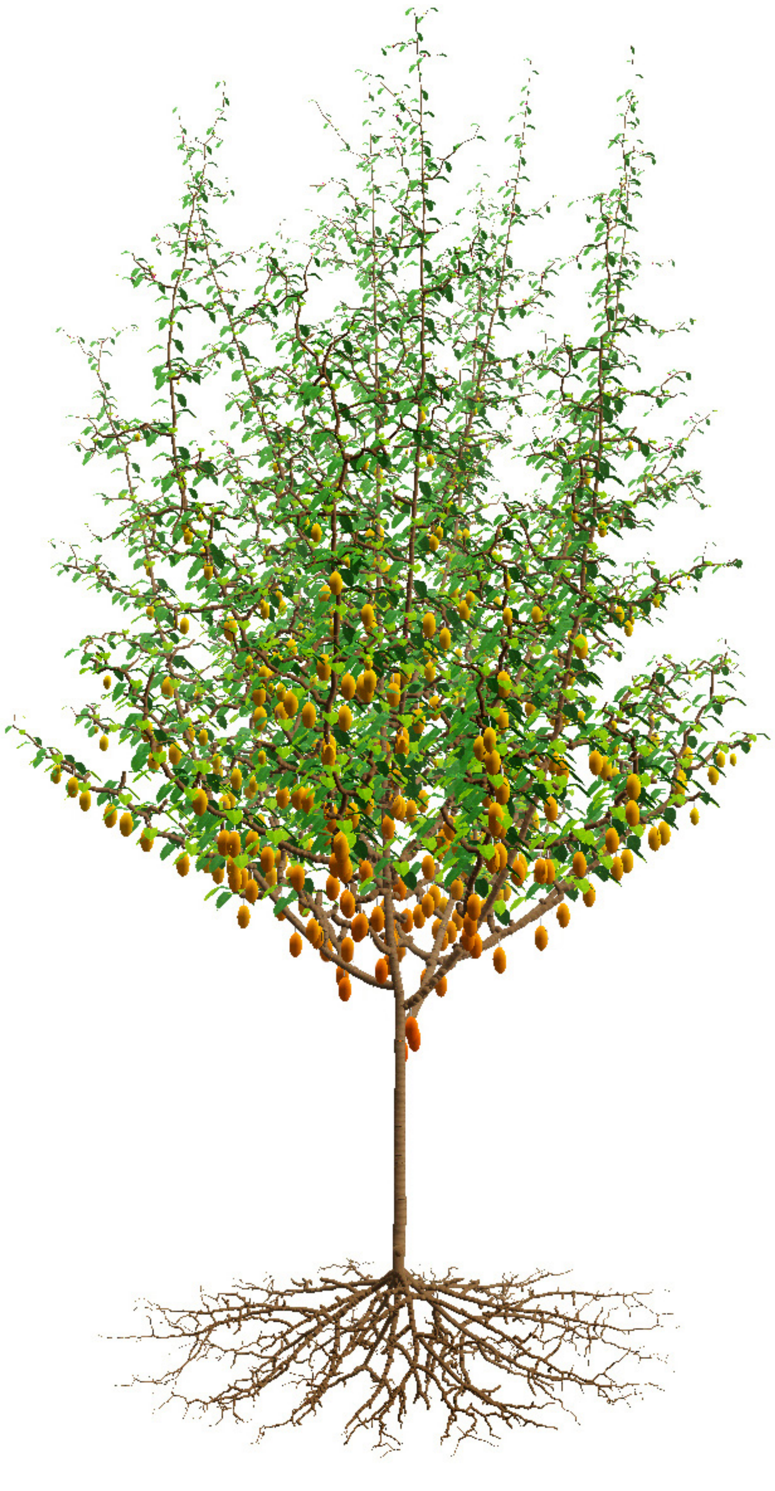
Here I like to express my gratitude to the directors of IWR for creating a great environment for doing interdisciplinary research. I have been particularly lucky to be a member of the IWR's graduate program (*The International Graduiertenkolleg*, IGK 710) "Complex Processes: Modeling, Simulation and Optimization", which provided many inspiring interdisciplinary contacts.

Financial support by the *Ministry of Science and Technology* (MOST), Thai government for studying, living for a whole period of study, financial support by the IWR's graduate program for participating in many conferences during research at IWR, and also financial support by the IWR, University of Heidelberg at the final part of research are gratefully acknowledged. I also would like to express my sincere thanks to the *Office of Educational Affairs* (OEA) both in London and Bangkok, and the *Thai Students Consulting* (TSC) in Berlin, Germany, for their kind supports based on all official documents and all annual meetings for improving the quality of Thai students. I am deeply indebted to my colleagues at the *Department of Mathematics and Computer Science*, Faculty of Science and Technology, Prince of Songkla University (PSU), Pattani Campus for their time and moral support.

I wish to thank the Chuai-Arees, my parents, sister and brothers, for their endless love and moral support. I also would like to express my sincere appreciation to all teachers, without them I cannot have reached this experience. Last but not least I wish to thank all of my friends and their unlimited friendship. Finally, special thanks go to Parilak Klinchang, Kittiporn Plaimas and Apichat Suratanee for their kind support.

Somporn Chuai-Aree
Heidelberg, Germany
April, 2009

Contents



“The root of knowledge is in nature; without root, it is fruitless.”

Somporn Chuai-Aree
Heidelberg, Germany
September 30, 2007

Chapter 1

Introduction

"God always acts in the simplest way."
— Galileo Galilei (1564-1642)

In this dissertation, we study modeling, simulation and visualization of plant growth. At first we introduce the motivation and problems of our research. Section 1.1 gives a literature review. Section 1.2 presents the aims and contributions of our research. Finally, we finish this chapter with the description of the contents in this paper.

1.1 Motivation

Plant modeling is an interesting and challenging problem integrating many disciplinary fields, e.g. mathematical modeling, computer graphics, computer science, agriculture, botany, biology, soil science. Plant structure consists of the shoot and the root part. Light, carbon dioxide (CO_2), oxygen (O_2), temperature, pH, water and nutrient play a big role for the plant growth and development.

For plant modeling, we have not only to consider geometric structure of plant but also processes and functions in plants. To describe plant structure (see Figure 1.1), the biologist *Aristid Lindenmayer (1925-1989)* introduced a code representing the geometry, known as *Lindenmayer systems* or *L-systems* [Prusinkiewicz and Lindenmayer, 1990]. In L-systems the structure is decomposed into plant modules and submodules using a graph theoretical concept and characterized by a code. L-systems are also a useful tool in other research areas, i.e. to describe networks in computer science, anatomy, etc.

Several methods are used to generate plant structure. Here we restrict ourselves to the following approaches:

- L-systems method, and
- Particle transportation method (PTM).

In L-systems, the plant is decomposed into different modules (shoot and root) which are characterized by graphs. Each module and its submodules are generated iteratively.

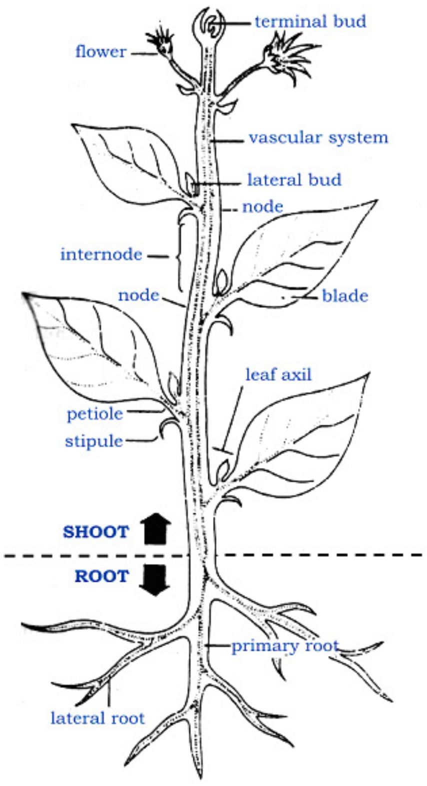


Figure 1.1: Structure 's diagram of flowering plant.

We obtain the development in discrete steps, not continuously (see Figure 1.2). In Chuai-Aree *et al.*, [Chuai-Aree, 2000], [Chuai-Aree et al., 2002] and [Rodkeaw et al., 2004], a continuous evolution in time based on physiological laws (see Figure 1.3) was used to obtain the plant growth process. To generate the structure it is necessary to know the plant specific growth rules.

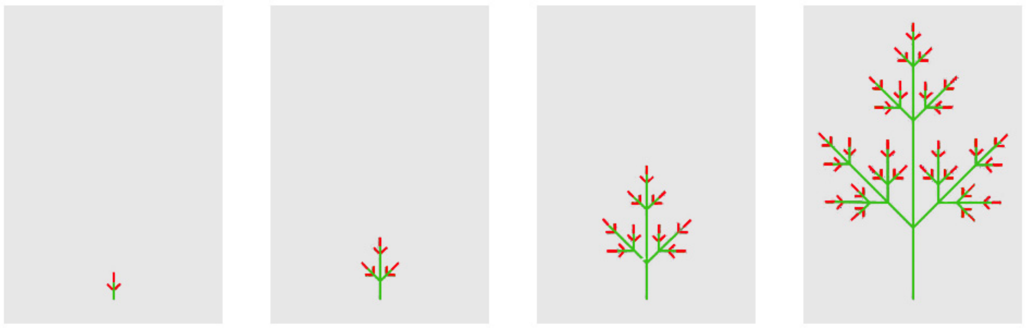


Figure 1.2: Deterministic L-systems animated by iteration step n = 1, 2, 3, 4.

PTM follows the concept that plant structures are generated by flows which can be approximated by trajectories of interacting particles, given by specified laws. The boundary of the geometry and target points have to be prescribed (see Chapter 3). PTM can

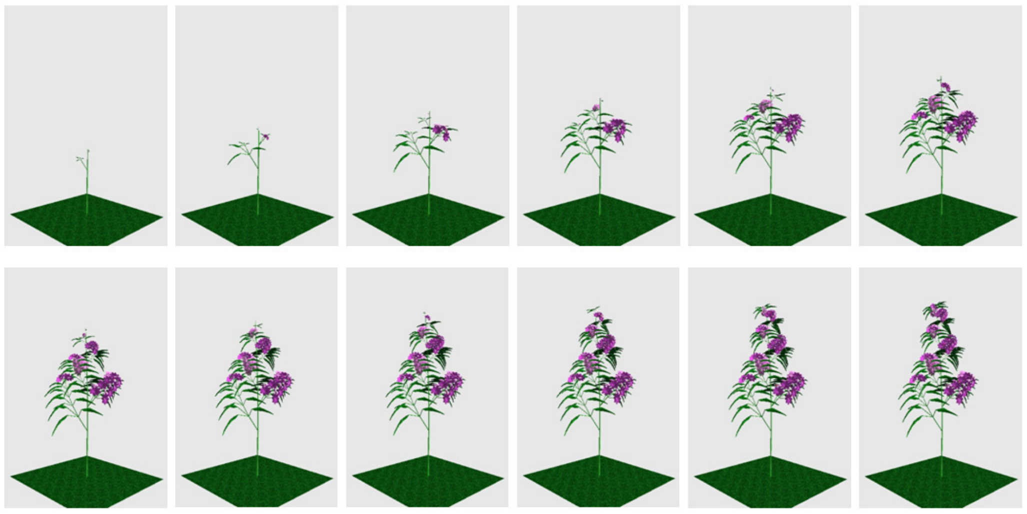


Figure 1.3: Deterministic L-systems animated by time step dt.

generate structure very fast by moving particles in a well-defined vector field. However, it is not easy to control the total particle flow. The algorithm may generate structures in an ordering which does not represent real plants.

PTM has been applied to model grass structure by Reeves [Reeves, 1983]. Rodkaew *et al.* introduced a PTM describing leaf veins in [Rodkaew et al., 2002], plant shoot and root structures in [Rodkaew et al., 2003] and [Rodkaew, 2004], using random and controlled particle flows.

Fractal geometry was introduced by Mandelbrot [Mandelbrot, 1983] as a new conceptual framework to analyze and model the nature of complex shapes. It has also been applied to generate plant structures with help of cellular automata. Iterative fractal methods were used to illustrate how complex vegetal-like structures could become (see Smith [Smith, 1984], Barnsley [Barnsley, 1988], Prusinkiewicz and Hanan [Prusinkiewicz and Hanan, 1989]). Such models were used to generate artificial plants in modeling applications, e.g. Chen et al. [Chen et al., 1994], Prusinkiewicz et al. [Prusinkiewicz et al., 2001]. Fractal geometry was used also to analyze the complexity in plant shapes by determining their fractal dimension.

A main problem is the identification of the geometry of a plant given real geometric data. An algorithm has to be developed deriving from image data the L-string, which characterizes the geometry of the plant. We call this problem the *inverse problem of L-systems*. The system can be used to describe tree-like structure based on bracketed and parametric L-system. This method allows us to observe the development of a plant in the soil. Using this non-destructive method, retrieving data from *Computed Tomography* (CT) or *Magnetic Resonance Imaging* (MRI), the 3D geometry of the plant is determined.

Plain geometry description of plant structure is not sufficient for plant modeling. Functional processes including interaction with environment are very important for growth modeling. For functional processes in this research, we firstly start the root growth modeling in soil solution. The problem of root growth in the 2-dimensional soil had been studied by Lungley [Lungley, 1973].

Modeling the root system and its interaction with the environment plays an crucial role in our investigation. The plant root system is a complex network of section of roots. Physiologically it plays an important role. Roots absorb water and nutrients from the soil. The efficiency water and nutrient uptake influent the crop yield. The circulation of water through the soil, the plant, and atmosphere, is an important hydrological subprocess in an ecological system. Furthermore, the architecture of root system and its mechanics is basic for the stability of the plant and the surrounding soil. Plant roots help to prevent erosion of the soil surface layer and landslides. The root system development is affected by its surrounding environment by gravity, water, temperature, nutrients, and soil structure. As a result of these effects, the root system changes its morphology and adapts to changes in the surrounding environment. Understanding the development of the root system is fundamental to understand its physiological and morphological functions. Since root systems are hard to be accessed, they are not studied to an extend as shoots and leaves are. Even if the root system could be excavated from the ground, it is almost impossible to observe continuous root system development. In this sense, modeling and simulation are providing proper tools to investigate the root systems. Solving the inverse problem of L-systems may help to reconstruct their growth dynamics (see Section 2.7).

1.2 Purpose and Contribution of our Research

The objectives of this study are

1. to develop a new code for plant structure, which can generate plant structures using L-systems and PTM,
2. to investigate the inverse problem of L-systems by given images and volume data of branching structure as input data,
3. to understand the mechanism of plant root systems development and the effect of soil water and nutrient conditions on root system development,
4. to combine structural and functional plant modeling as a computer simulation and visualization tool, and
5. to apply these methods to improve the optimal growth of some plants for the farmers.

In the following, we summarize the main contribution of our research.

A prototype of plant growth (shoot, root, and leaf structure) from experiments including physiological growth factors is developed. L-systems and PTM are used to generated plant structures. The algorithm can produce deterministic and stochastic structures. The geometrical structures and the functional processes are combined in simulating the plant growth. Phenomenological production rules take into account growth factors which are not modelled explicitly. However, nutrients and water uptake by roots are treated with model equations.

We improved the L-systems by proposing a continuous evolution in time based on physiological laws (see [Chuai-Aree, 2000] and [Chuai-Aree et al., 2002]). In case of PTM, we improved the work of Rodkaew *et al.* [Rodkeaw et al., 2004, Rodkaew, 2004] by generating in the natural ordering the structure of plant: we replaced the given target point by the given starting point and considered a growing volume. The particle distribution may be randomly initially chosen and evolves influenced by environmental factors. The direction of the trajectories is chosen in each time step by an averaging rule. Plant structure is growing smoothly as natural behavior. PTM is used to describe leaf veins in a predefined leaf shape, synthetic shoot and root structures in a predefined volume. We also provide the method for generating the L-string (see Definition from input image data using the *inverse problem of L-systems*.

In the functional and biological processes, we study how to generate the root systems. In this case several algorithms are proposed, e.g. L-systems and PTM. Water flow and nutrient diffusion in soil solution are combined with root growth systems. Nutrient diffusion in soil solution is described by diffusion equation in 3-dimensional space. Water flow in soil solution is represented by *Richards equation*.

There are now both plant structures and plant functions. Then we combine both geometrical structures and the biological processes. The geometrical structures are generated by L-systems, inverse problem of L-systems, and PTM. Since the root systems are growing based on water and nutrient in soil. We propose the simple algorithm for root growth based on gradient field of nutrient concentration. The root tips are growing to the direction of maximum nutrient concentration. We later design the algorithm for PTM to cooperate with environmental factors by considering water content and nutrient concentration in each voxel space as the source of root growth direction.

Finally, we obtain a computer simulation tool, *PlantVR* software. There are four softwares working related to *PlantVR*: *BranchRecon* (Inverse problem of L-systems), *LeafDS* (Leaf surface design), *LeafVein* (Leaf vein generation), and *PlantVR-MuPhi* communication.

Finally, the simulation and visualization of structural and functional plant modeling are implemented in order to improve and evaluate the modeling.

In this dissertation, new methods are developed for doing research in agriculture. We bring different disciplines, e.g. plant sciences, mathematics and computer science, together for a better understanding of the plant structures and their functions.

1.3 Description of the Content

This dissertation is organized into six chapters and three appendices. The first part of this thesis is composed of one review chapter which provides the fundamental for the work developed.

Chapter 2 is devoted to describe network systems and some examples. The classes of L-systems and new code for plant structure are presented. The inverse problem of L-systems is explained and network structures are reconstructed from two or three dimensional input data.

Chapter 3 contains the proposed method of PTM and its improvements. The PTM is introduced for generating leaf veins, plant shoot and root structures. The detail of algorithms is given and results of simulation are presented.

In Chapter 4, the interaction of root systems and the environment is treated. We model and simulate the water flow in soil by Richards equation and nutrient flow by diffusion and transport equations. The result simulation are visualized in 3D using transparent technique.

In Chapter 5, the following software tools are introduced and discussed:

- *PlantVR* (Plant Virtual Reality),
- *BranchRecon* (Inverse problem of L-systems),
- *LeafDS* (Leaf surface design),
- *LeafVein* (Leaf vein generation),
- *PlantVR-MuPhi* Communication (*PlantVR* with *MuPhi* library).

Chapter 6 summarizes the conclusions and discusses the results and future work to be done.

The appendices contain the rewriting algorithm in Pascal-code, data structure and variables of plant prototype, L-system interpretation algorithm, and natural selection algorithm, explicit formula needed for the parameter estimation in Chapter 2, and some gallery results.

Finally, all softwares and some animations obtained in this dissertation are provided in CD-ROM at the end of dissertation.

Chapter 2

Networks, L-systems and Plant Prototype

"Nature is pleased with simplicity and affects not the pomp of superfluous."
— Isaac Newton (1642-1727)

This chapter describes general network structures and methods for representing network structures using Lindenmayer systems. The classes of Lindenmayer Systems [see Prusinkiewicz and Lindenmayer, 1990] are described. Lindenmayer systems (L-systems) were conceived as a mathematical theory of plant development. Originally, they did not include enough detail to allow for comprehensive modeling of higher plants. The emphasis was on plant topology, that is, the neighborhood relations between cells or larger plant modules. Their geometric aspects were beyond the scope of the theory. Subsequently, several geometric interpretations of L-systems were proposed with a view to turning them into a versatile tool for plant modeling.

This chapter is organized into three parts. First part, the general network structures such as branching structures or plant structure is described. The methods for describing the structure namely "classes of L-systems" are given in the second part (see Section 2.2). The new improvement of L-systems used in this research is explained in Section 2.3, 2.4, 2.5 and 2.6. Finally, the third part describes the inverse problem of L-systems since the input images can be retrieved from end user. It is fully explained in Section 2.7.

2.1 Networks

A network is a system of nodes and edges connecting nodes. Networks can be considered embedded in space and dynamically evolving in time. Dealing with plant growth we are considering dynamical networks embedded in R^2 respectively in R^3. A root and a shoot system can be considered as a dynamical network in time, however, since it is not sufficient, to consider only its topology, geometric properties have to be added and coded. In fact, concerned with biological growth, we have to formulate laws describing the evolution of such a network. Since the physiological processes in the plant itself are not well understood, we have to rely on phenomenological laws for the growth processes.

We remark here, that it is one of the challenges to Mathematics to tread processes on dynamically evolving networks, stochastic or deterministic in its nature. Our investigation is focussed on networks in plant growth. However, some of our results will be useful also for networks in other situation. Here, we refer to some networks, where e.g. our contributions to coding with L-systems and the corresponding inverse problem might be useful for the networks of blood vessels, bronchia, neurons, bones, etc.

In plants, the transports of substrates and water are always in the network via xylem and phloem. The connectivity of these networks have been built in different structures based on their genetics, environment and interactions between genetics and environment.

This section describes the topology of graphs, directed graphs, networks and connectivity, which will be used in the further sections.

2.1.1 Examples of Networks in Plants

This section shows some examples of plant networks, e.g. branch shoot networks of trees in Figure 2.1, root networks in Figure 2.2, hairy roots in Figure 2.3, and leaf veins in Figure 2.4.

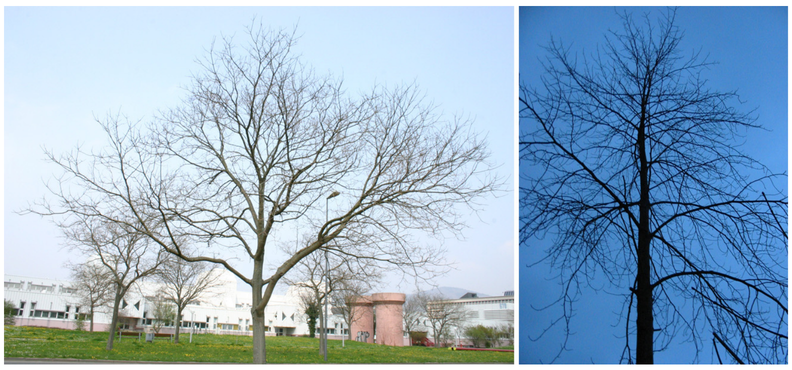


Figure 2.1: Examples of tree networks.

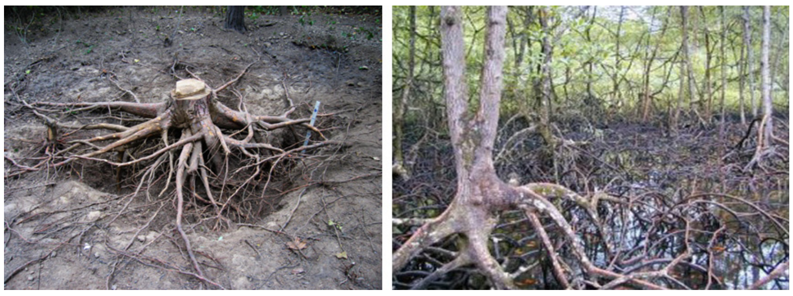


Figure 2.2: Examples of root networks (left image: tree root system [Cermak, Retrived 2007], right image : mangrove roots from wikipedia website).

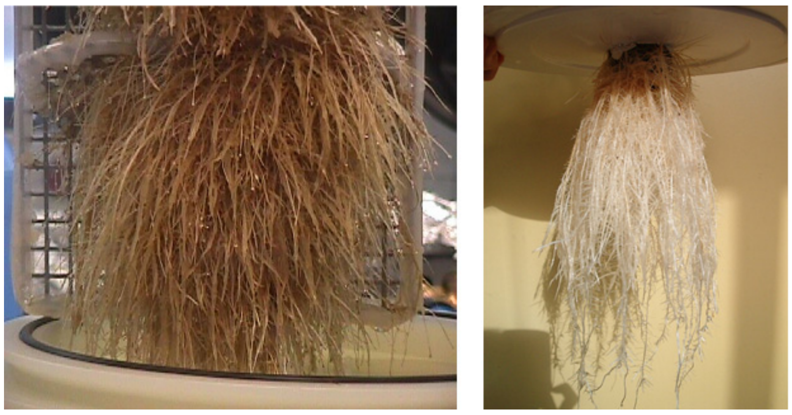


Figure 2.3: Examples of hairy roots (left image : root reactor [RootTec, Germany], right image : from wikipedia website).

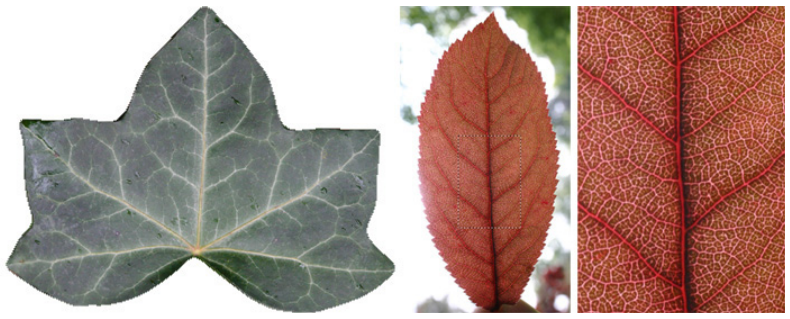


Figure 2.4: Examples of leaf vein networks.

2.1.2 Graph, Directed Graph, Network and Connectivity

This section describes the definition of graph, directed graph, and network and connectivity. Graphs and networks are later used for connecting all plant components.

Graph

A graph for which the relations between pairs of vertices are symmetric, so that each edge has no directional character (as opposed to a directed graph). Unless otherwise indicated by context, the term “graph” can usually be taken to mean “undirected graph”. The definition of graph is defined in Definition 2.1.

Definition 2.1 *(**Graph**) A graph $G = (V, E)$ consists of a set of vertices, V, and a set of edges, E. Each edge is a pair (v, w), where $v, w \in V$. Edges are sometimes referred to as arcs. If the pair is ordered, then the graph is directed. Directed graphs are sometimes referred to as digraphs. Vertex w is adjacent to v if and only if $(v, w) \in E$. In an undirected graph with edge (v, w), and hence (w, v), w is adjacent to v and v is adjacent to w. Sometimes an edge has a third component, known as either a weight or a cost.*

Directed Graph

A graph in which each graph edge is replaced by a directed graph edge, is also called a digraph. A directed graph having no multiple edges or loops (corresponding to a binary adjacency matrix with 0s on the diagonal) is called a simple directed graph. A complete graph in which each edge is bi-directed is called a complete directed graph. A directed graph having no symmetric pair of directed edges (i.e., no bi-directed edges) is called an oriented graph. A complete oriented graph (i.e., a directed graph in which each pair of nodes is joined by a single edge having a unique direction) is called a tournament. The definition of digraph is defined in Definition 2.2.

Definition 2.2 ***(Digraph)*** *A digraph is an ordered pair of sets $G = (V, A)$, where V is a set of vertices and A is a set of ordered pairs (called arcs) of vertices of V.*

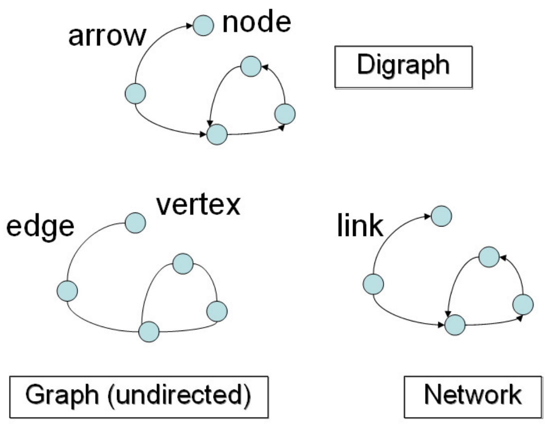


Figure 2.5: Graphs, digraphs and networks (modified from [Bossomaier and Green, 2000]).

Figure 2.5 shows examples of undirected graph, directed graph and network. Undirected graph consists of set of vertices and edges. There are no direction specifying between vertex as directed graph. In case of directed graph, the connection between node is represented by *arrow* while the network represented by *link* from the start node to the target node.

Network and Connectivity

Network is the combination of graph and directed graph. It consists of nodes and branches. The connection of nodes is linked by branches. There are nine different network topologies given in Figure 2.6, e.g. fully connected topology, bus topology, star topology, ring topology, tree topology, mesh topology, hybrid topology, dual ring topology, and linear topology.

In plant structures, bus, tree, linear and hybrid topology can be used to represent plant shoots and roots while the combination of star and ring topology can be applied for leaf veins and combination of petals in flowers.

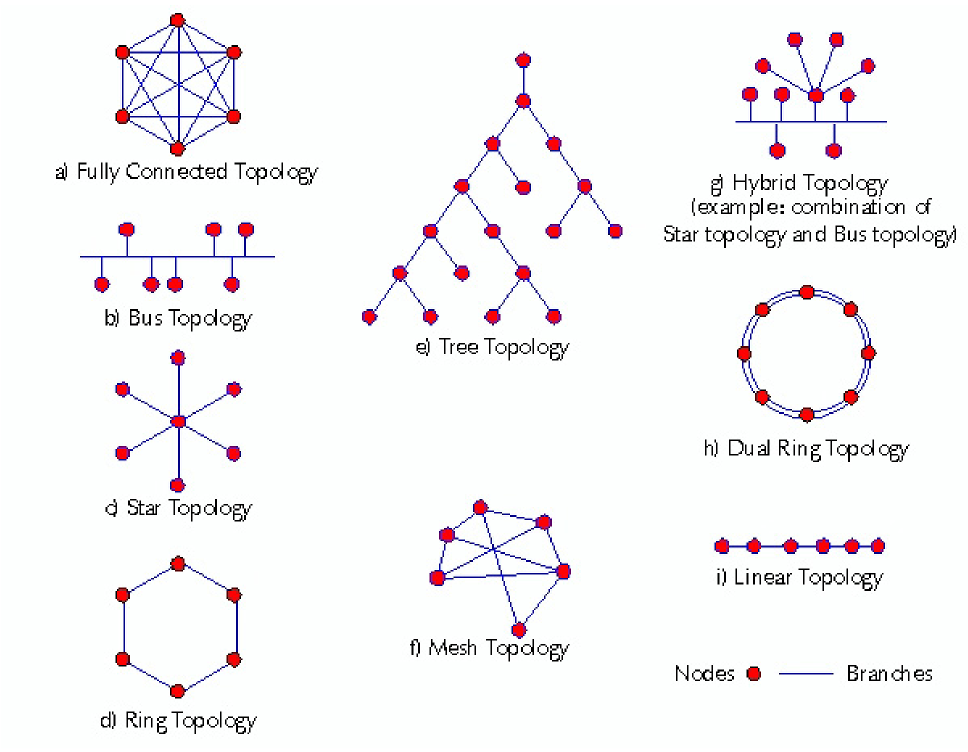


Figure 2.6: Examples of network topology and nodes connectivity.

2.2 Lindenmayer Systems

In 1968, the theoretical biologist from the university of Utrecht, Aristid Lindenmayer (1925-1989), introduced a new type of string-rewriting mechanism, called L-systems [see Prusinkiewicz and Lindenmayer, 1990] (see Figure 2.7). The essential difference between Chomsky grammars and L-systems lies in the method of applying productions. In Chomsky grammars production rules are applied sequentially, whereas in L-systems they are applied in parallel and simultaneously replace all letters in an initial string. This difference reflects the biological motivation of L-systems. Production rules are intended to capture cell divisions in multi-cellular organisms, where many cell divisions may occur at the same time. Parallel production application has an essential impact on the formal properties of rewriting systems.

This section describes the classes of L-systems from the rewriting systems, deterministic and context-free L-systems (OL-systems), turtle interpretation of strings, modeling in three dimensions, bracketed OL-systems, stochastic L-systems, edge and node rewriting, parametric L-systems, and turtle interpretation of parametric words.

2.2.1 Rewriting Systems

The main concept of L-systems is rewriting method. The rewriting method is a technique for defining complex objects by successively replacing parts of a simple initial object using a set of *rewriting rules* or *production rules* [Prusinkiewicz and Lindenmayer, 1990]. The

Figure 2.7: Aristid Lindenmayer, 1925-1989 (image from [Prusinkiewicz and Lindenmayer, 1990]).

classic example of graphical object defined in terms of rewriting rules is the snowflake curve shown in Figure 2.8, proposed in 1905 by von Koch [Prusinkiewicz and Lindenmayer, 1990]. Mandelbrot restates this construction as follows:

One begins with *two shapes*, an *initiator* and a *generator*. The latter is an oriented broken line made up of N equal sides of length r. Thus each stage of the construction begins with a broken line and consists in replacing each straight interval with a copy of the generator (see Figure 2.9), reduced and displaced so as to have the same end points as those of the interval being replaced (from [Prusinkiewicz and Lindenmayer, 1990]).

An L-system is *context-free* if each production rule refers only to an individual symbol and not to its neighbours. If a rule depends not only on a single symbol but also on its neighbours, it is called a *context-sensitive L-system.* If there is exactly one production rule for each symbol, then the L-system is said to be deterministic (a *deterministic context-free L-system* is popularly called a D0L-system). If there are several production rules, and each production rule is chosen with a certain *probability* during each iteration, then it is a *stochastic L-system.*

2.2.2 Deterministic and Context-Free L-systems

Deterministic and context-free L-systems (DOL-systems) are the simplest class of L-systems. We starts with an example that introduces the main concept in intuitive terms.

From the example in [Prusinkiewicz and Lindenmayer, 1990], we consider strings or words built of two symbols a and b, which may occur many times in a string. Each symbol is associated with a rewriting rule. The rule $a \rightarrow ab$ means that the letter a is to be replaced by the string ab, and the rule $b \rightarrow a$ means that the letter b is to be

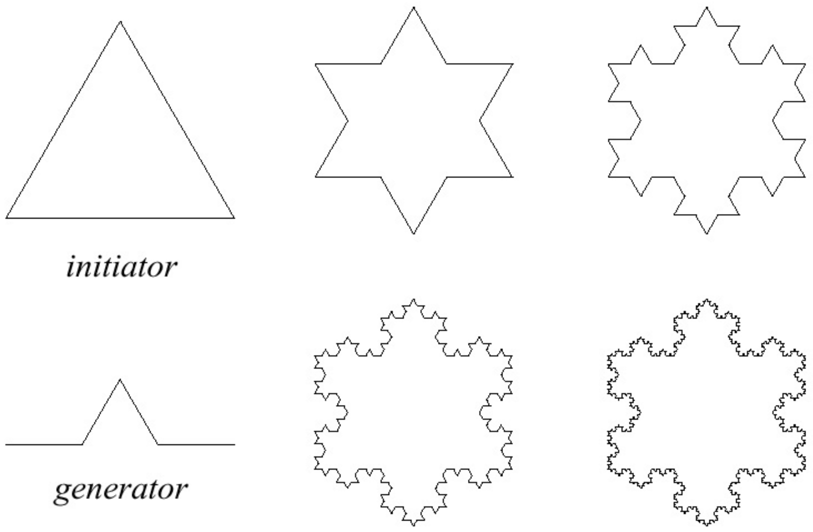


Figure 2.8: Construction of the snowflake curve (image from [Prusinkiewicz and Lindenmayer, 1990]).

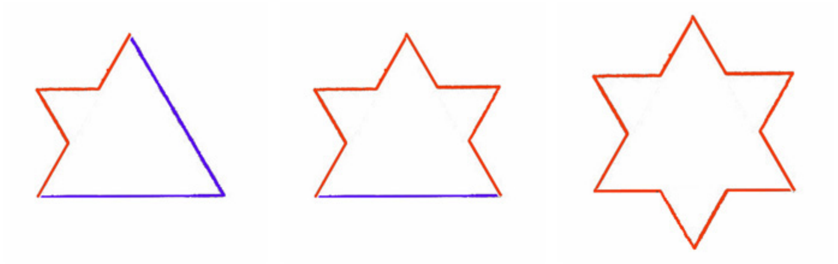


Figure 2.9: A step by step of the construction of the snowflake curve from Figure 2.8 in the first iteration.

replaced by letter a. The rewriting method starts from a distinguished string called the *axiom*. Assume that the axiom consists of a single letter b. In the first derivation step (the first step of rewriting, n=1), the axiom the letter b is replaced by the letter a using the production $b \rightarrow a$. In the second step (n=2) a is replaced by ab using production $a \rightarrow ab$. The word ab consists of two letters. Both of them are simultaneously replaced in the next derivation step. Thus, a is replaced by ab, b is replaced by a, and the string aba results. In a similar way, the string aba yields $abaab$ which in turn yields $abaababa$, then $abaababaabaab$, and so on as illustrated in Figure 2.10. The definition of DOL-systems is defined in Definition 2.3.

$$\begin{cases} n & = & 5 & (iterations) \\ V & : & \{a, b\} & (variables) \\ \omega & : & b & (axiom) \\ p_1 & : & a \rightarrow ab & (1^{st}\, production\, rule) \\ p_2 & : & b \rightarrow a & (2^{nd}\, production\, rule) \end{cases} \tag{2.1}$$

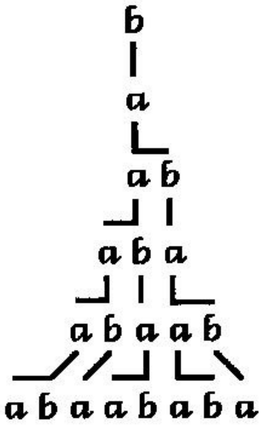


Figure 2.10: Example of a derivation in a DOL-system (image from [Prusinkiewicz and Lindenmayer, 1990]).

Definition 2.3 *(**DOL-systems**) Let* V *denote an alphabet,* V^* *the set of all words over* V*.* V^+ *the set of all nonempty words over* V*. A* string OL-system *is an ordered triplet* $G =< V, \omega, P >$*, where*

> V *is the* alphabet *of the system,*
> $\omega \in V^+$ *is a nonempty word called the* axiom*, and*
> $P \subset V \times V^*$ *is a finite* set of productions*.*

A production $(a, \chi) \in P$ is written as $a \to \chi$. The letter a and the word χ are called the *predecessor* and the *successor* of this production, respectively. It is assumed that for any letter $a \in V$, there is at least one word $\chi \in V^*$ such that $a \to \chi$. If no production is explicitly specified for a given predecessor $a \in V$, the *identity production* $a \to a$ is assumed to belong to the set of productions P. An OL-system is deterministic (noted DOL-system) if and only if for each $a \in V$ there is exactly one $\chi \in V^*$ such that $a \to \chi$.

Let $\mu = a_1 \dots a_m$ be an arbitrary word over V. The word $\nu = \chi_1 \cdots \chi_m \in V^*$ is *directly generated* by μ, noted $\mu \Rightarrow \nu$, if and only if $a_i \to \chi_i$ for all $i = 1, \dots, m$. A word ν is generated by G in a derivation of *length* n if there exists a *developmental sequence* of words $\mu_0, \mu_1, \dots, \mu_n$ such that $\mu_0 = \omega, \mu_n = \nu$ and $\mu_0 \Rightarrow \mu_1 \Rightarrow \dots \Rightarrow \mu_n$.

The following example provides another illustration of the operation of DOL-systems. The formalism is used to simulate the development of a fragment of a multi-cellular filament such as that found in the blue-green bacteria *Anabaena catenula* and various algae [Prusinkiewicz and Lindenmayer, 1990]. The symbols $\boldsymbol{a}$ and $\boldsymbol{b}$ represent cytological states of the cells (their size and readiness to divide). The subscripts $\boldsymbol{l}$ and $\boldsymbol{r}$ indicate cell polarity, specifying the positions in which daughter cells of type $\boldsymbol{a}$ and $\boldsymbol{b}$ will be produced. The development is described by the following L-system:

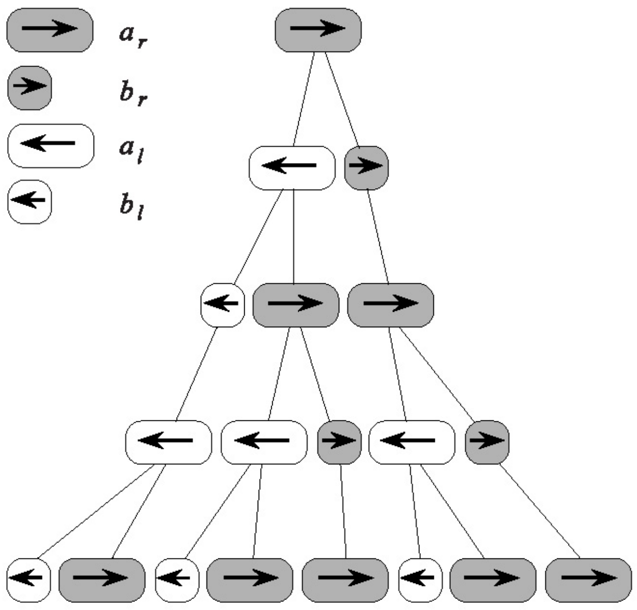


Figure 2.11: Development of a filament (*Anabaena catenula*) simulated using a DOL-system (image from [Prusinkiewicz and Lindenmayer, 1990]).

$$\begin{cases} n & = & 4 & (\textit{iterations}) \\ V & : & \{a_l, a_r, b_l, b_r\} & (\textit{variables}) \\ \omega & : & a_r & (\textit{axiom}) \\ p_1 & : & a_r \rightarrow a_l b_r & (1^{st}\, \textit{production rule}) \\ p_2 & : & a_l \rightarrow b_l a_r & (2^{nd}\, \textit{production rule}) \\ p_3 & : & b_r \rightarrow a_r & (3^{rd}\, \textit{production rule}) \\ p_4 & : & b_l \rightarrow a_l & (4^{th}\, \textit{production rule}) \end{cases} \tag{2.2}$$

Starting from a single cell a_r (the axiom), the following sequence of words is generated in each iteration:

$$\begin{cases} n = 0, & a_r \\ n = 1, & a_l b_r \\ n = 2, & b_l a_r a_r \\ n = 3, & a_l a_l b_r a_l b_r \\ n = 4, & b_l a_r b_l a_r a_r b_l a_r a_r \\ n = \ldots, & \ldots \end{cases} \tag{2.3}$$

Under a microscope, the filaments appear as a sequence of cylinders of various lengths, with a-type cells longer than b-type cells. The corresponding schematic image of filament development is shown in Figure 2.11. Note that due to the discrete nature of L-systems, the continuous growth of cells between subdivisions is not captures by this model.

For more general L-systems, an L-systems grammar is defined as a tuple G in Definition 2.4 and production rule is expressed in Definition 2.5.

Definition 2.4 *(**L-systems grammar**) L-system grammars are defined as a tuple $G = \{V, S, \omega, P\}$, where*

V	*(the alphabet or variables) is a set of symbols containing elements that can be replaced,*
S	*(constants) is a set of symbols containing elements that remain fixed constants,*
ω	*(start, axiom or initiator) is a string of symbols from V defining the initial state of the system, and*
P	*is a set of rules or productions defining the way variables can be replaced with combinations of constants and other variables. A production consists of two strings -the predecessor and the successor.*

Definition 2.5 *(**Production rule**) Production rule (a, χ) in L-systems grammar can be written as $a \to \chi$, where a is a predecessor which is replaced by a successor χ.*

2.2.3 Turtle Interpretation of Strings

This section explains the interpretation of strings and shows some examples. In order to model the higher plants, a graphical interpretation of L-systems is required. The first results in this direction were published in 1974 by Frijters and Lindenmayer, and Hogeweg and Hesper (see [Prusinkiewicz and Lindenmayer, 1990]). In both cases, L-systems were used primarily to determine the branching topology of the modeled plants. The geometric aspects, such as the lengths of line segments and the angle values, were added in a post-processing phase. The results of Hogeweg and Hesper were subsequently extended by Smith [Smith, 1984], who demonstrated the potential of L-systems for realistic image synthesis.

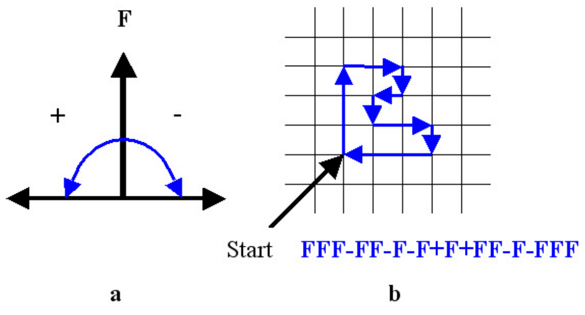


Figure 2.12: Turtle interpretation of string (modified from [Prusinkiewicz and Lindenmayer, 1990]).

The basic idea of turtle interpretation is shown in Figure 2.12. There are three directions of turtle movement, e.g. left, forward, and right. The rotation angle is controlled by symbol $+$ for turning counter-clockwise and symbol $-$ for turning clockwise. The symbol F is walking forward with the constant length. Figure 2.12 shows the turtle movement of string $FFF - FF - F - F + F + FF - F - FFF$. A *state* of the *turtle* is defined as a triplet (x, y, α), where the Cartesian coordinates (x, y) represent the turtle's *position*, and

Table 2.1: The two-dimensional turtle interpretation.

Symbols	Meaning
F	Move forward a step of length d. The state of the turtle changes to (x', y', α), where $x' = x + d\,cos(\alpha)$ and $y' = y + d\,sin(\alpha)$. A line segment between points (x, y) and (x', y') is drawn.
f	Move forward a step of length d without drawing a line.
$+$	Turn left by angle δ. The next state of the turtle is $(x, y, \alpha + \delta)$. The position orientation of angles is counter-clockwise.
$-$	Turn right by angle δ. The next state of the turtle is $(x, y, \alpha - \delta)$. The position orientation of angles is clockwise.

the angle α, call the *heading*, is interpreted as the direction in which the turtle is facing. Given the *step size* d and the *angle increment* δ, the turtle can respond to commands represented by the following symbols in Table 2.1.

Given a string ν, the initial position of the turtle (x_0, y_0, α_0), fixed parameters d and δ , the *turtle interpretation* of ν is the figure (set of lines) drawn by the turtle in response to the string ν in Figure 2.12(b). This method can be applied to interpret strings which are generated by L-systems. For example, Figure 2.13 presents four approximations of the *quadratic Koch island*. These figures were obtained by interpreting strings generated by the following L-system:

$$\begin{cases} n & : & 3 & (Iterations) \\ V & : & \{F\} & (Variable\,or\,alphabets) \\ S & : & \{-, +\} & (Constants) \\ \omega & : & F - F - F - F & (Axiom) \\ P & : & F \rightarrow F - F + F + FF - F - F + F & (Production\,rule) \end{cases} \tag{2.4}$$

The images correspond to the strings obtained in derivations of length 0 to 3. The angle increment δ is equal to 90. The step length d is decreased four times between subsequent images, making the distance between the endpoints of the successor polygon equal to the length of the predecessor segment (see [Prusinkiewicz and Lindenmayer, 1990]).

2.2.4 Modeling in Three Dimensions

Turtle interpretation of L-systems can be extended to three dimensions following the ideas of Abelson and diSessa [Prusinkiewicz and Lindenmayer, 1990]. The idea is to represent the current *orientation* of the turtle in space by three vectors $\vec{X}, \vec{Y}, \vec{Z}$ indicating the direction in X-axis, Y-axis, Z-axis, respectively. These vectors have orthogonal unit vector that satisfy the equation (2.5)

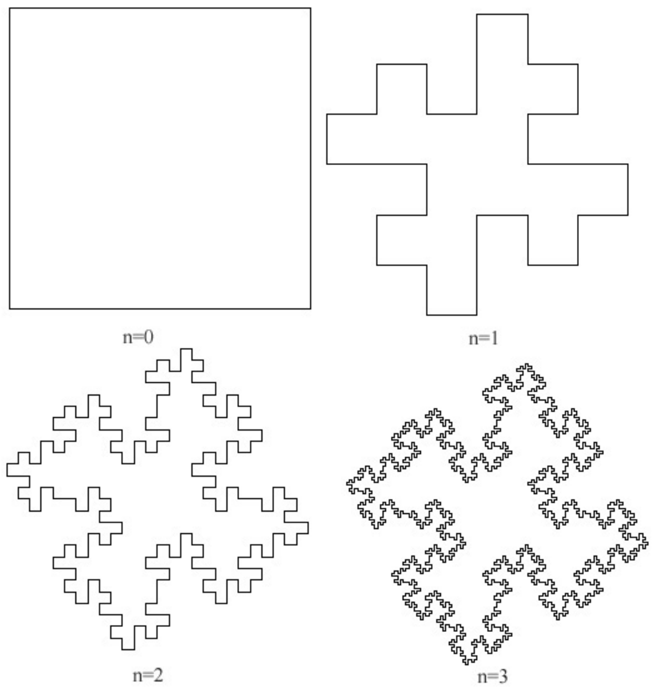


Figure 2.13: Generating a quadratic Koch island (modified from [Prusinkiewicz and Lindenmayer, 1990]).

$$\vec{Z} \times \vec{X} \;=\; \vec{Y}. \tag{2.5}$$

Rotations of the turtle are expressed by the equation (2.6)

$$[\vec{X'} \quad \vec{Y'} \quad \vec{Z'}] \;=\; [\vec{X} \quad \vec{Y} \quad \vec{Z}] \quad R \tag{2.6}$$

where $\boldsymbol{R}$ is a 3×3 rotation matrix in equation (2.7). The rotations by angle θ about vectors $\vec{X}$, $\vec{Y}$ and $\vec{Z}$ are represented by the following matrices:

$$\begin{aligned}
R_x(\theta) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}, \\
R_y(\theta) &= \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \\
R_z(\theta) &= \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{2.7}$$

Table 2.2: The symbol of three-dimensional turtle interpretation.

Symbols	Meaning
$+$	Roll counterclockwise by angle δ_z, using rotation matrix $R_z(\delta_z)$
$-$	Roll clockwise by angle δ_z, using rotation matrix $R_z(-\delta_z)$
&	Roll counterclockwise by angle δ_y, using rotation matrix $R_y(\delta_y)$
ˆ	Roll clockwise by angle δ_y, using rotation matrix $R_y(-\delta_y)$
\	Roll counterclockwise by angle δ_x, using rotation matrix $R_x(\delta_x)$
/	Roll clockwise by angle δ_x, using rotation matrix $R_x(-\delta_x)$
|	Roll back, using rotation matrix $R_y(180)$

The following symbols in set S control turtle orientation in three-dimensional space as Figure 2.14.

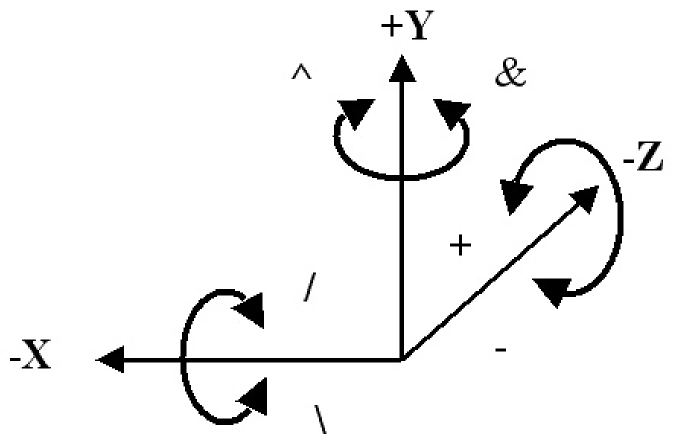


Figure 2.14: Controlling the turtle in three-dimensions [Chuai-Aree, 2000].

In this case, set of constants S is defined as $S = \{+, -, \&, \hat{}, \backslash, /, |\}$. The meaning of each constant in S is shown in Table 2.2.

2.2.5 Bracketed OL-systems

The definition of tree L-systems does not specify the data structure for representing axial trees. One possibility is to use a list representation with a tree topology. Alternatively, axial tree can be represented using *strings with brackets* [Prusinkiewicz and Lindenmayer, 1990]. A similar distinction can be observed in Koch constructions, which can be implemented either by rewriting edges and polygons or their string representations. An extension of turtle interpretation to strings with brackets and the operation of bracketed L-systems [Prusinkiewicz and Lindenmayer, 1990] are described below.

Two new symbols are introduced to represent a branch. They are interpreted by the turtle in Table 2.3. A set of constants S is defined as $S = \{+, -, \&, \hat{}, \backslash, /, |, [,]\}$. The meaning of bracketed symbols in S is described in Table 2.3.

Table 2.3: The bracketed symbols.

Symbols	Meaning
[	Push the current state of the turtle onto a pushdown stack. The information saved on the stack contains the turtles position and orientation, and possibly other attributes such as the color and width of lines being drawn
]	Pop a state from the stack and make it the current state of the turtle. No line is drawn, although in general the position of the turtle changes

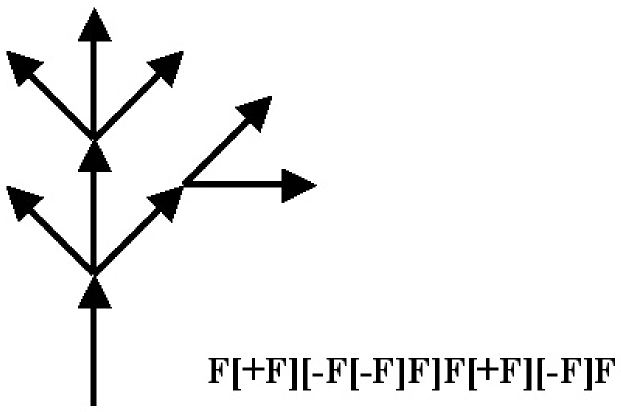


Figure 2.15: Bracketed string representation of an axial tree structure (modified from [Prusinkiewicz and Lindenmayer, 1990]).

An example of an axial tree and its string representation are shown in Figure 2.15. This example can be easily describe step by step for L-system interpretation (see Figure 2.16).

The interpretation starts with the arrow up with the symbol F (see Figure 2.16 in the second image from left in the first row), starts a new branch with the symbol [, turns left 45 degrees with +, draws the arrow with length F, moves back to the start point of its branch with symbol], and now the interpretation string becomes $F[-F]$ (see Figure 2.16 in the third image from left in the first row). The new right branch starts with the symbol [for opening the new branch , symbol $-$ for turning right 45 degrees, symbol F for drawing the arrow with length F. The interpretation is proceeded until all symbols reached.

Derivations in bracketed OL-systems proceed as in OL-systems with out brackets. The brackets replace themselves. Examples of two-dimensional branching structures generated by bracketed OL-systems are shown in Figure 2.17.

Another example of a three-dimensional plat is shown in Figure 2.18. The L-systems can be described and analyzed in a similar manner. Figure 2.18 illustrates some examples of a three-dimensional bush-like structure generated by a bracketed L-system

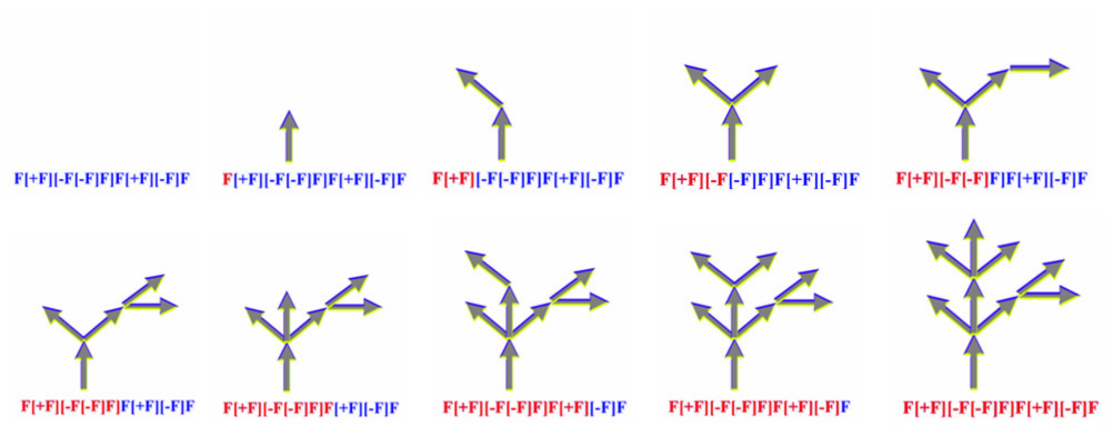


Figure 2.16: Step by step bracketed string representation of an axial tree structure, red string illustrates the structure in each step from left to right.

[Prusinkiewicz and Lindenmayer, 1990]. Production p_1 creates three new branches from an apex of the old branch. A branch consists of an edge F forming the initial internode, a leaf L and an apex A (which will subsequently create three new branches). Productions p_2 and p_3 specify internode growth. In subsequent derivation steps, the internode gets longer and acquires new leaves. This violates a biological rule of *subapical growth*, but produces an acceptable visual effect in a still picture. Production p_4 specifies the leaf as a filled polygon with six edges. Its boundary is formed from the edges f enclosed between the braces { and }. The symbols ! and ' are used to decrement the diameter of segments and increment the current index to the color table, respectively.

Examples of branching structures generated by this L-system with derivations of length 7 are shown in Figure 2.18. Note that these structures look like different specimens of the same plant species.

2.2.6 Stochastic L-systems

All plants generated by the same deterministic L-system are identical. An attempt to combine them in the same picture would produce a striking, artificial regularity. In order to prevent this effect, it is necessary to introduce specimen-to-specimen variations that will preserve the general aspects of a plant but will modify its details.

Variation can be achieved by randomizing the turtle interpretation, the L-system, or both. Randomization of the interpretation alone has a limited effect. While the geometric aspects of a plant such as the stem lengths and branching angles are modified, the underlying topology remains unchanged. In contrast, stochastic application of productions may affect both the topology and the geometry of the plant. The definition of stochastic OL-system is defined in Definition 2.6.

Definition 2.6 ***(stochastic OL-system)*** *A stochastic OL-system is an ordered quadruplet $G_p = < V, \omega, P, \pi >$. The alphabet V, the axiom ω and the set of productions P are defined as in an OL-system. π is set of production probabilities mapped one to one into P.*

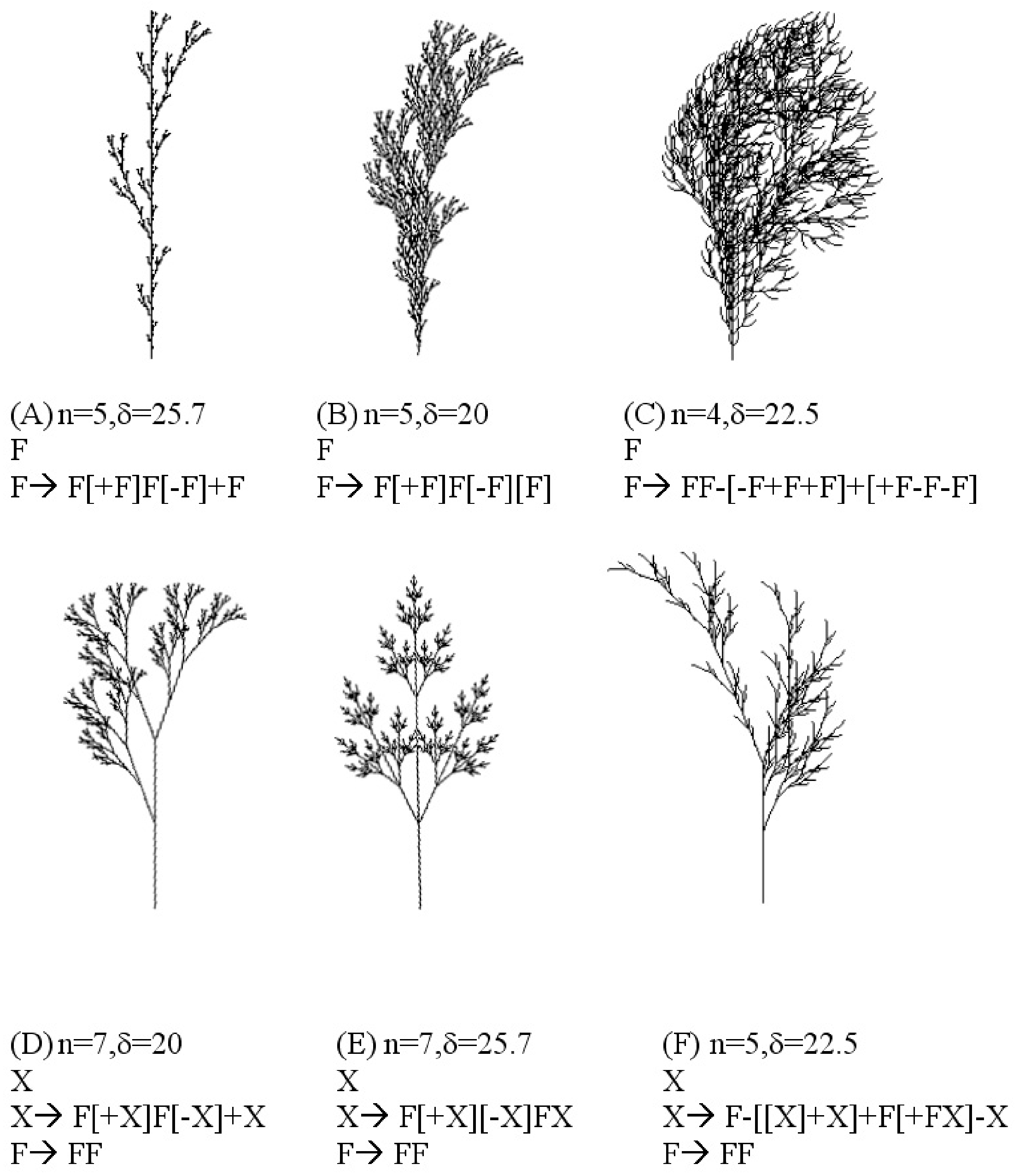
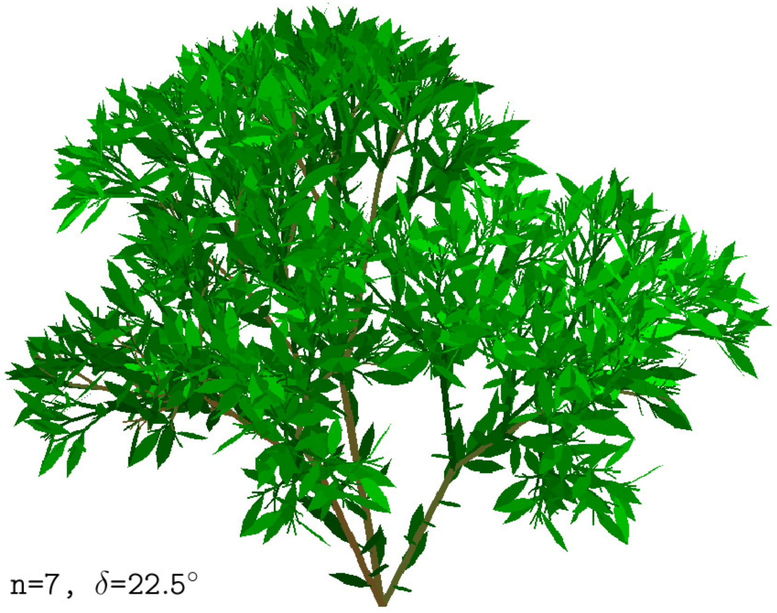


Figure 2.17: Examples of plant-like structures generated by bracketed OL-systems (image from [Prusinkiewicz and Lindenmayer, 1990]).

Function $\pi : P \rightarrow (0, 1]$, called the *probability distribution*, maps the set of productions into the set of *production probabilities*. It is assumed that for any letter $a \in V$, the sum of probabilities of all productions with the predecessor a is equal to 1.

We will call the derivation $\mu \Rightarrow \nu$ a *stochastic derivation* in G_π if for each *occurrence* of the letter a in the word μ the probability of applying production p with predecessor a is equal to $\pi(p)$. Thus, different productions with the same predecessor can be applied to



n=7, δ=22.5°

ω : `A`
p_1 : `A` $\rightarrow$ `[&FL!A]/////’[&FL!A]///////’[&FL!A]`
p_2 : `F` $\rightarrow$ `S ///// F`
p_3 : `S` $\rightarrow$ `F L`
p_4 : `L` $\rightarrow$ `[’’’∧∧{-f+f+f-|-f+f+f}]`

Figure 2.18: A three-dimensional busk-like structure (image from [Prusinkiewicz and Lindenmayer, 1990]).

various occurrences of the same letter in one derivation step.

A simple example of a stochastic L-system is given below.

$$
\left\{
\begin{array}{lcll}
V & : & \{F\} & (Alphabet) \\
S & : & \{+,-,[,]\} & (Constants) \\
\omega & : & F & (Axiom) \\
P & : & \{p_1,p_2,p_3\} & (Production\,rules) \\
\pi & : & \{0.33,0.33,0.34\} & (Production\,probabilities) \\
p_1 & : & F \rightarrow^{0.33} F[+F]F[-F]F & (1^{st}\,Production\,rule) \\
p_2 & : & F \rightarrow^{0.33} F[+F]F & (2^{nd}\,Production\,rule) \\
p_3 & : & F \rightarrow^{0.34} F[-F]F & (3^{rd}\,Production\,rule)
\end{array}
\right. \tag{2.8}
$$

The three cases of the stochastic L-system in L-systems (2.8) can be visualized in the graphical form as Figure 2.19. The production probabilities are listed after the derivation symbol $\rightarrow$. The production rules p_1, p_2 and p_3 can be selected with approximately the same probability of 0.33, 0.33 and 0.34, respectively.

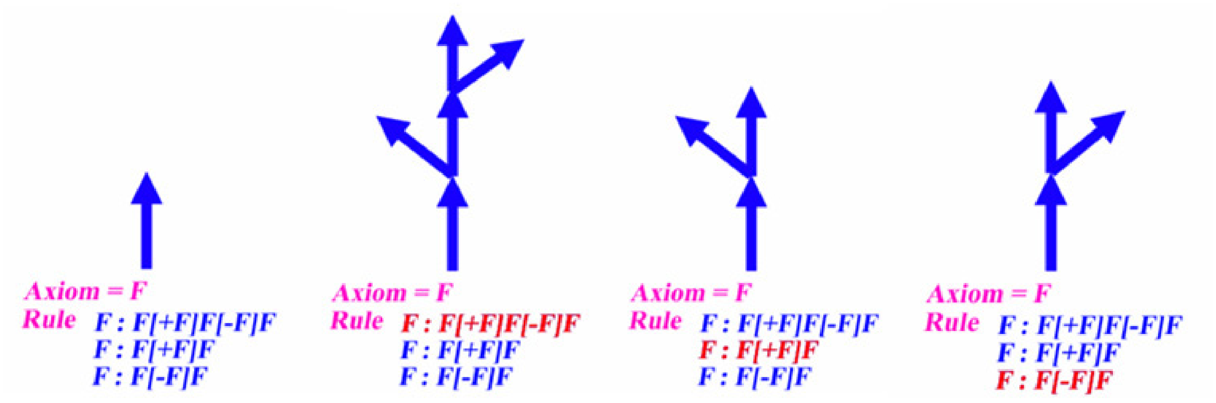


Figure 2.19: A stochastic L-system and its visualization of each production rule in L-system (2.8), an axiom F, p_1, p_2, and p_3 are arranged from left to right, respectively.

2.2.7 Edge and Node Rewriting

Random modification of productions gives little insight into the relationship between L-systems and figures they generate. However, we often wish to construct an L-systems which captures a given structure or sequence of structures representing a developmental process. This is called the *inference problem* in the theory of L-systems. Although some algorithms for solving it were reported in the literature [Prusinkiewicz and Lindenmayer, 1990], the are still too limited to be of practical value in the modeling of higher plants. Consequently, the methods introduced below are more intuitive in nature. They exploit two modes of operation for L-systems with turtle interpretation, called *edge rewriting* and *node rewriting* using terminology borrowed from graph grammars [Prusinkiewicz and Lindenmayer, 1990]. In the case of edge rewriting, productions substitute figures for polygon edges, while node rewriting, productions operate on polygon vertices. Both approaches rely on capturing the recursive structure of figures and relating it to a tiling of a plane. Although the concepts are illustrated using abstract curves, they apply to branching structures found in plants as well.

Edge rewriting identifies edges as specific types of edges, which the turtle does not interpret, but the different types of edges affect rewriting rules.

Example:

Node rewriting substitutes new polygons for nodes of the predecessor curve.

The Hilbert curve can be expressed by a rewrite system (L-system). From L-system 2.4 we can write Hilbert curve in the form of $G = \{V, S, \omega, P\}$ as follows:

$$\begin{cases} V & : & \{L, R\} & (Alphabet) \\ S & : & \{F, +, -\} & (Constants) \\ \omega & : & L & (Axiom) \\ P & : & \{p_1, p_2\} & (Production\, rules) \\ p_1 & : & L \rightarrow +RF - LFL - FR+ & (1^{st}\, production\, rule) \\ p_2 & : & R \rightarrow -LF + RFR + FL- & (2^{nd}\, production\, rule) \end{cases} \tag{2.9}$$

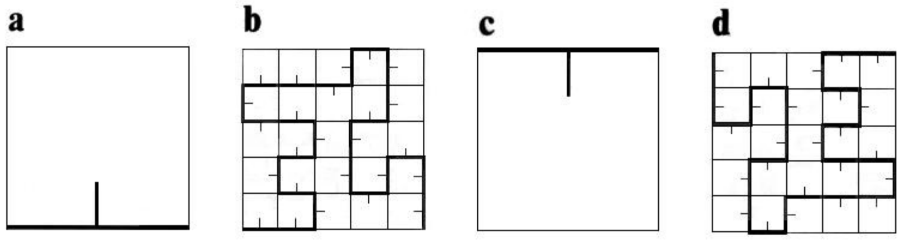


Figure 2.20: Construction of the E-curve on the square grid, **a)**F_l, **b)**$F_l \rightarrow F_lF_l+F_r+F_r-F_l-F_l+F_r+F_rF_l-F_r-F_lF_lF_r+F_l-F_r-F_lF_l-F_r+F_lF_r+F_r+F_l-F_l-F_rF_r+$, **c)**$F_r$, **d)**$F_r \rightarrow -F_lF_l+F_r+F_r-F_l-F_lF_r-F_l+F_rF_r+F_l+Fr-F_lF_rF_r+F_l+F_rF_l-F_l-F_r+F_r+F_l-F_l-F_rF_r$ (image from [Prusinkiewicz and Lindenmayer, 1990]).

In this case, F means "draw forward", + means "turn left 90 degrees", and − means "turn right 90 degrees". In order to write Hilbert curve in general rules, we can simply write as follows (see L-system 2.10).

$$\begin{cases} L_{n+1} = +R_nF - L_nFL_n - FR_n+ & (1^{st}\, production\, rule) \\ R_{n+1} = -L_nF + R_nFR_n + FL_n- & (2^{nd}\, production\, rule) \end{cases} \tag{2.10}$$

$$\begin{aligned} L_{n+1} &= +R_nF - L_nFL_n - FR_n+ \\ R_{n+1} &= -L_nF + R_nFR_n + FL_n- \end{aligned}$$

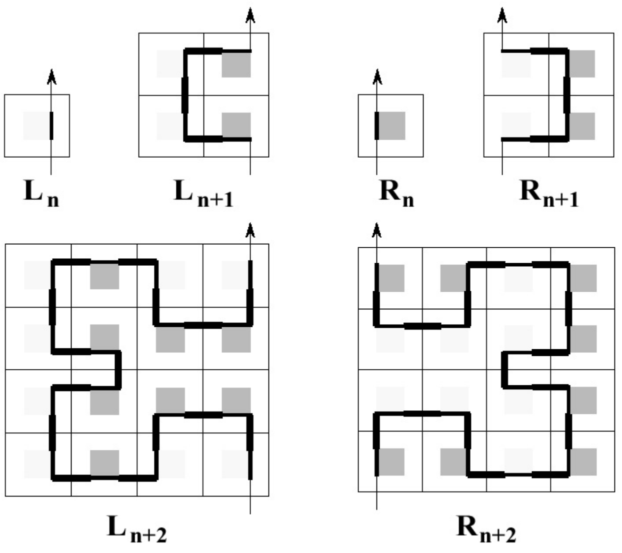


Figure 2.21: Recursive construction of the Hilbert curve in term of replacement (image from [Prusinkiewicz and Lindenmayer, 1990]).

Figure 2.21 illustrates the recursive construction of the Hilbert curve in term of replacement by node rewriting method based on L_n and R_n and two production rules in L-system 2.10.

The curves are generated by either edge or node rewriting method. As in the case of edge rewriting, the relationship between node rewriting and tiling of the plane extends to branching structures. It offers a method for synthesizing L-systems that generate objects with a given recursive structure, and links methods for plant generation based on L-systems.

2.2.8 Parametric L-systems

Parametric L-systems operate on *parametric words*, which are strings of *modules* consisting of *letters* with associated *parameters*. The letters belong to an *alphabet* V, and the parameters belong to the set of *real numbers* $\Re$. A module with letter $A \in V$ and parameters $a_1, a_2, \ldots, a_n \in \Re$ is denoted by $A(a_1, a_2, \ldots, a_n)$. Every module belongs to the set $M = V \times \Re^*$, where $\Re^*$ is the set of all finite sequences of parameters. The set of all strings of modules and the set of all nonempty strings are denoted by $M^* = (V \times \Re^*)^*$ and $M^+ = (V \times \Re^*)^+$, respectively (see [Prusinkiewicz and Lindenmayer, 1990]).

The real-valued *actual* parameters appearing in the words correspond with *formal* parameters used in the specification of L-systems productions. If Σ is a set of formal parameters, then $C(\Sigma)$ denotes a *logical expression* with parameters from Σ, and $E(\Sigma)$ is an *arithmetic expression* with parameters from the same set. Both types of expressions consist of formal parameters and numeric constants, combined using the arithmetic operators $+, -, *, /$; the exponentiation operator $\hat{}$, the relational operators $<, >, =$; the logical operator $!, \&, |$ (not, and, or); and parentheses ( ). Standard rules for constructing syntactically correct expressions and for operator precedence are observed. Relational and logical expressions evaluate to zero for false and one for true. A logical statement specified as the empty string is assumed to have value one. The sets of all correctly constructed logical and arithmetic expressions with parameters from Σ are noted $C(\Sigma)$ and $E(\Sigma)$. The definition of parametric OL-system is defined in Definition 2.7.

Definition 2.7 ***(A parametric OL-system)*** *A parametric OL-system is defined as an ordered quadruplet $G =< V, \Sigma, \omega, P >$, where*

- V is the *alphabet* of the system,
- Σ is the *set of formal parameters*,
- $\omega \in (V \times \Re^*)^+$ is a nonempty parametric word called the *axiom*,
- $P \subset (V \times \Sigma^*) \times C(\Sigma) \times (V \times E(\Sigma))^*$ is a finite *set of productions.*

The symbols : and $\rightarrow$ are used to separate the three components of a production : the *predecessor*, the *condition* and the *successor*. For example, a production with predecessor $A(t)$, condition time $t > 5$ and successor $B(t + 1)CD(t\hat{}0.5, t - 2)$ is written as

$$A(t) \quad : \quad t > 5 \quad \rightarrow \quad B(t + 1)CD(t\hat{}0.5, t - 2). \tag{2.11}$$

A production *matches* a modules in a parametric word if the following conditions are met:

- the letter in the module and the letter in the production predecessor are the same,
- the number of actual parameters in the module is equal to the number of formal parameters in the production predecessor, and
- the condition evaluates to *true* if the actual parameter values are substituted for the formal parameters in the production.

A matching production can be *applied* to the module, creating a string of modules specified by the production successor. The actual parameter values are substituted for the formal parameters according to their positions. For example, production (L-system (2.11)) matches a module $A(9)$, since the letter A in the module is the same as in the production predecessor, there is one actual parameter in the module $A(9)$ and one formal parameter in the predecessor $A(t)$, and the logical expression $t > 5$ is true for $t = 9$. The result of the application of this production is a parametric word $B(10)CD(3,7)$.

If a module a produces a parametric word χ as the result of a production application in an L-system G, we write $a \to \chi$. Given a parametric word $\mu = a_1a_2 \dots a_m$, we say that the word $\nu = \chi_1\chi_2 \dots \chi_m$ is *directly derived* from (or generated by) μ and write $\mu \Rightarrow \nu$ if and only if $a_i \to \chi_i$ for all $i = 1, 2, \dots, m$. A parametric word ν is generated by G in a *derivation of length n* if there exists a sequence of words $\mu_0, \mu_1, \dots, \mu_n$ such that $\mu_0 = \omega$, $\mu_n = \nu$ and $\mu_0 \Rightarrow \mu_1 \Rightarrow \dots \Rightarrow \mu_n$.

An example of a parametric L-system is given below in (2.12).

$$\left\{\begin{array}{lllllll} V & : & \{A, B\} \\ \Sigma & : & \{x, y\} \\ \omega & : & B(2)A(4,4) \\ P & : & \{p_1, p_2, p_3, p_4\} \\ p1 & : & A(x,y) & : & y <= 3 & \rightarrow & A(x*2, x+y) \\ p2 & : & A(x,y) & : & y > 3 & \rightarrow & B(x)A(x/y, 0) \\ p3 & : & B(x) & : & x < 1 & \rightarrow & C \\ p4 & : & B(x) & : & x >= 1 & \rightarrow & B(x-1) \end{array}\right. \tag{2.12}$$

As in the case of non-parametric L-systems, it is assumed that a module replaces itself if no matching production is found in the set P. The words obtained in the first few derivation steps are show in Figure 2.22.

2.2.9 Turtle Interpretation of Parametric Words

If one or more parameters are associated with a symbol interpreted by the turtle, the value of the first parameter controls the turtles state. If the symbols are not followed by any parameter, default values specified outside the L-system are used as in the non-parametric case. The basic set of symbols affected by the introduction of parameters is listed below in Table 2.4.

It should be noted that symbols +, &, and / are used both as letters of the alphabet V and as operators in logical and arithmetic expressions. Their meaning depends on the context.

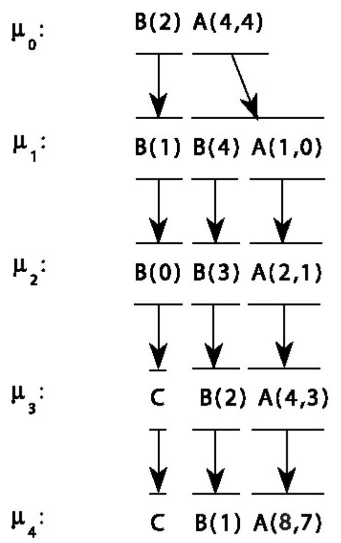


Figure 2.22: The initial sequence of strings generated by the parametric L-system specified in prototype (2.12) (image from [Prusinkiewicz and Lindenmayer, 1990]).

Table 2.4: The turtle interpretation of parametric words.

Symbols	Meaning
$F(a)$	Move forward a step of length $a > 0$. The position of the turtle changes to (x', y', z'), where $x' = x + a\vec{X}_x$, $y' = y + a\vec{X}_y$, $z' = z + a\vec{X}_z$. A line segment is drawn between points (x, y, z) and (x', y', z').
$f(a)$	Move forward a step of length a without drawing a line.
$+(a)$	Rotate about Z-axis by an angle of a degrees. If a is positive, the turtle is turned to the left and if a is negative, the turn is to the right.
$\&(a)$	Rotate about Y-axis by angle of a degrees. If a is positive, the turtle is pitched down and if a is negative, the turtle is pitched up.
$/(a)$	Rotate about X-axis by an angle of a degrees. If a is positive, the turtle is rolled to the right and if a is negative, it is rolled to the left.

2.3 Plant Structures and Their Functions

This section roughly describes about plant structures; i.e. shoot, root, leaf, flower, and their functions. Plants were classified to be a member in domain of *Eukaryota* and kingdom of *Plantae* by Haeckel (1866). Plants are a main group of living things including familiar organisms, e.g. trees, flowers, ferns, herbs, and mosses. There are about 350,000 species of plant defined. Figure 2.23 shows plant components and some functions of leaves, shoot, root components (see [Bell and Bryan, 1993a,b]).

Plants are living things that make their own food using energy from the sun by photosynthesis processes. There are thousands of different plants on earth, some living on land and others in water. Each type of plant has different structural features to help it survive in each environmental condition. The most common type of plant is the flowering plant; they are all made up of roots, stems, leaves and flowers.

Roots, which are most often found in the ground, are important for plants water, nutrient uptake and starch storage. Roots also help stabilize the plant as they hold it firmly in the ground. Roots provide anchorage in the soil. Root hairs provide huge surface area for water and nutrient absorption. Root tip is the area of cell division. Root cap protects and lubricates the growing root.

The stem of a plant transports water from roots to leaves so that the plant can grow, leaves can develop, and sunlight can be found for the formation of food through photosynthesis. Stems also support the leaves, flowers and fruits in the plant life cycle. Leaves grow from the stem of the plant and can have many different shapes and sizes. The transformation of the suns energy into food, photosynthesis, generally takes place in the leaves, so all actual plants have leaves.

Leaves are also important in regulation of water. Transpiration, or release of water, takes place in the leaves so that the roots can draw in fresh water. There are two types of leaves, simple and compound; simple leaves are those with only one leaf blade on a stalk while compound leaves are those with more than one leaf blade on a single stalk. Leaves are a plant's food factory. They are the main site of photosynthesis, where sugars are made from water and carbon dioxide, using sunlight energy.

The illustrations of leaf structures and leaf connection to its stem are given in Figure 2.24 and Figure 2.25. The examples of leaf margins are shown in Figure 2.26. They show many different types of leaf margins, e.g. undulate, sinuate, serrate, dentale, lobate, scalloped, palmate, digitate, bipinnatisect, tripinnatisect, pinnatisect, palmatisect, pedate, palmatilobate, bipartite, tripartite, palmatipartite, pinnatipartite, and pinnatifid.

Flowers are responsible for reproduction, production of seeds that can grow into new plants. Many parts of the flower are designed for reproduction. Flowers begin as buds, with sepals surrounding them for protection. The sepals are forced apart to reveal the petals as the flower starts to bloom. The petals surround the pistol and stamen of the flower that interact for reproduction. In the center of the flower is the pistil, the female part that contains the ovary, which houses the egg cells, called ovules. Stamens, the male parts of the flower, surround the pistil. Each stamen has a stalk, called the filament, that holds up the anther where pollen is produced. When the pollen is ripe it is released from the anther into the air so it can come into contact with the pistil. The pollen fertilizes

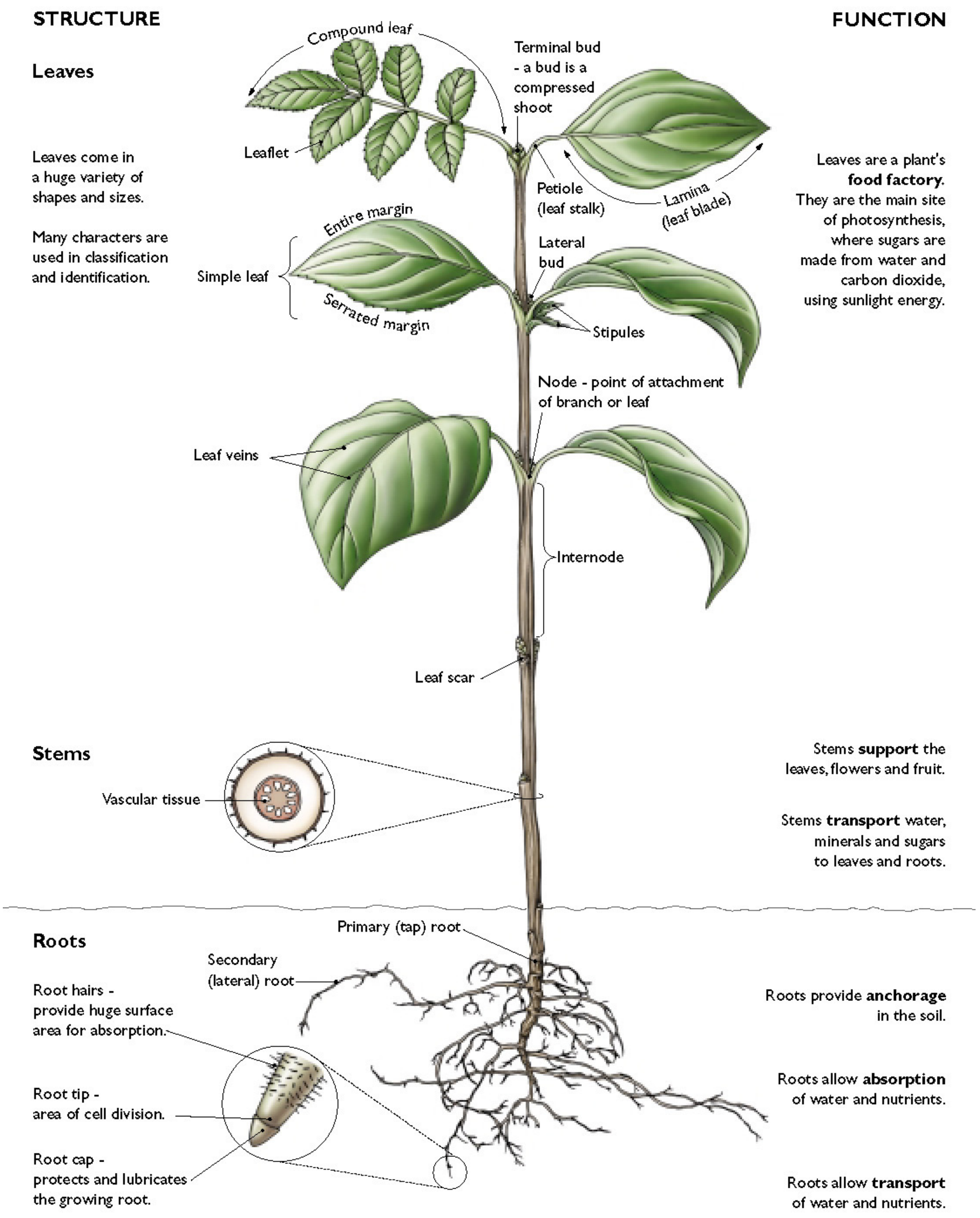

STRUCTURE
FUNCTION
Leaves
Compound leaf
Terminal bud - a bud is a compressed shoot
Leaflet
Leaves come in a huge variety of shapes and sizes.
Many characters are used in classification and identification.
Petiole (leaf stalk)
Lamina (leaf blade)
Leaves are a plant's **food factory**. They are the main site of photosynthesis, where sugars are made from water and carbon dioxide, using sunlight energy.
Entire margin
Simple leaf
Serrated margin
Lateral bud
Stipules
Node - point of attachment of branch or leaf
Leaf veins
Internode
Leaf scar
Stems
Vascular tissue
Stems **support** the leaves, flowers and fruit.
Stems **transport** water, minerals and sugars to leaves and roots.
Roots
Primary (tap) root
Secondary (lateral) root
Root hairs - provide huge surface area for absorption.
Roots provide **anchorage** in the soil.
Root tip - area of cell division.
Roots allow **absorption** of water and nutrients.
Root cap - protects and lubricates the growing root.
Roots allow **transport** of water and nutrients.

Figure 2.23: Plant structure, plant components and their functions (image from Royal Botanic Gardens Kew (England) [Bell and Bryan, 1993a,b]).

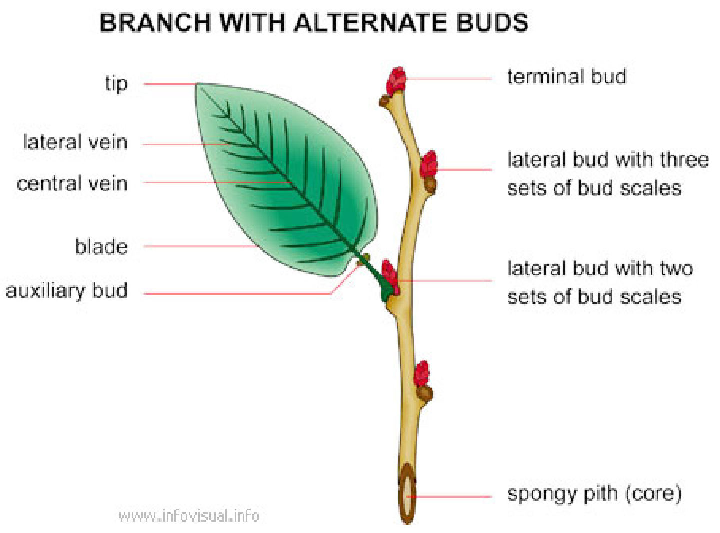

BRANCH WITH ALTERNATE BUDS
tip
lateral vein
central vein
blade
auxiliary bud
terminal bud
lateral bud with three sets of bud scales
lateral bud with two sets of bud scales
spongy pith (core)
www.infovisual.info

Figure 2.24: Example of leaf structure with single leaf and branch with alternative buds, leaf structure consists of leaf tip, leaf blade, lateral vein, central vein and auxiliary bud. There is lateral bud at each node and terminal bud at the branch tip. (image from http://www.infovisual.info)

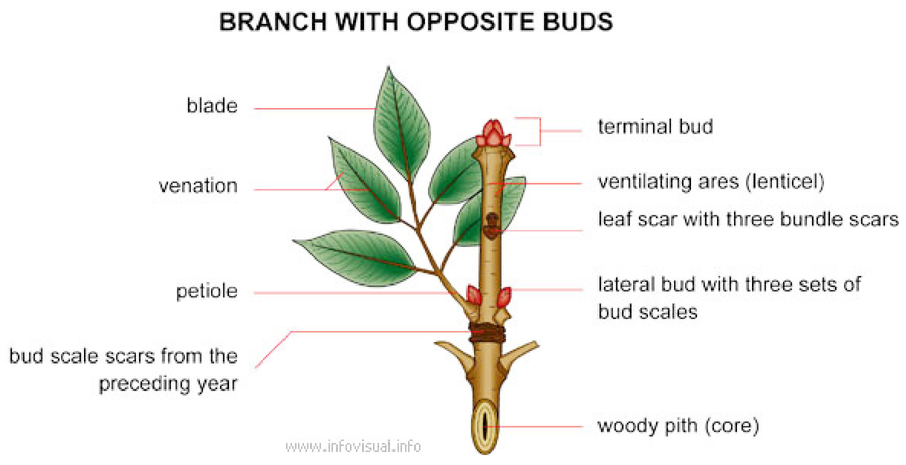

BRANCH WITH OPPOSITE BUDS
blade
venation
petiole
bud scale scars from the preceding year
terminal bud
ventilating ares (lenticel)
leaf scar with three bundle scars
lateral bud with three sets of bud scales
woody pith (core)
www.infovisual.info

Figure 2.25: Example of leaf structure with compound leaves and branch with opposite buds, this compound leaves consist of five leaves. The petiole is the connection of all leaves with the main stem. (image from http://www.infovisual.info).

the ovules in the ovary and seeds begin to form. Flowers' beautiful petals are designed to attract pollinators, such as bees and butterflies. Some petals even have markings that point in the direction of the nectar the bees have come to collect.

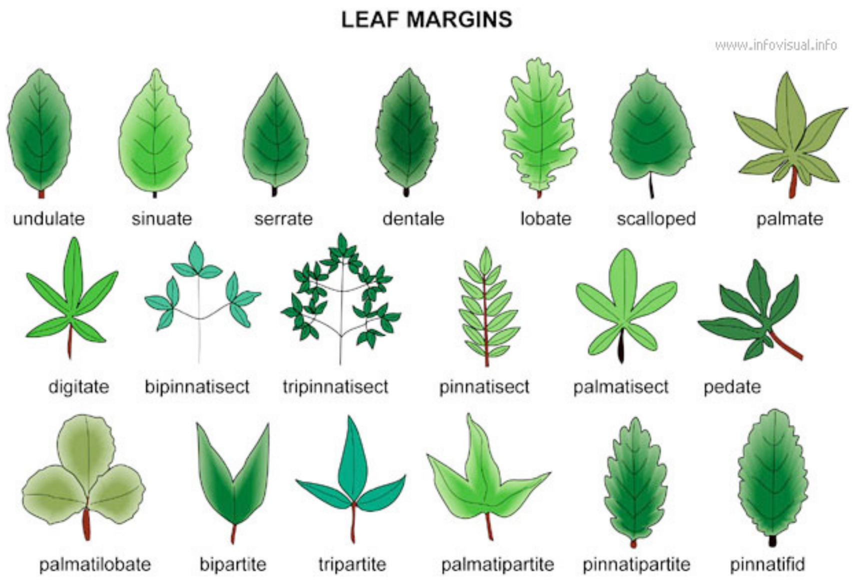


Figure 2.26: Example of leaf margins, e.g. undulate, sinuate, serrate, dentale, lobate, scalloped, palmate, digitate, bipinnatisect, tripinnatisect, pinnatisect, palmatisect, pedate, palmatilobate, bipartite, tripartite, palmatipartite, pinnatipartite, and pinnatifid. (image from http://www.infovisual.info).

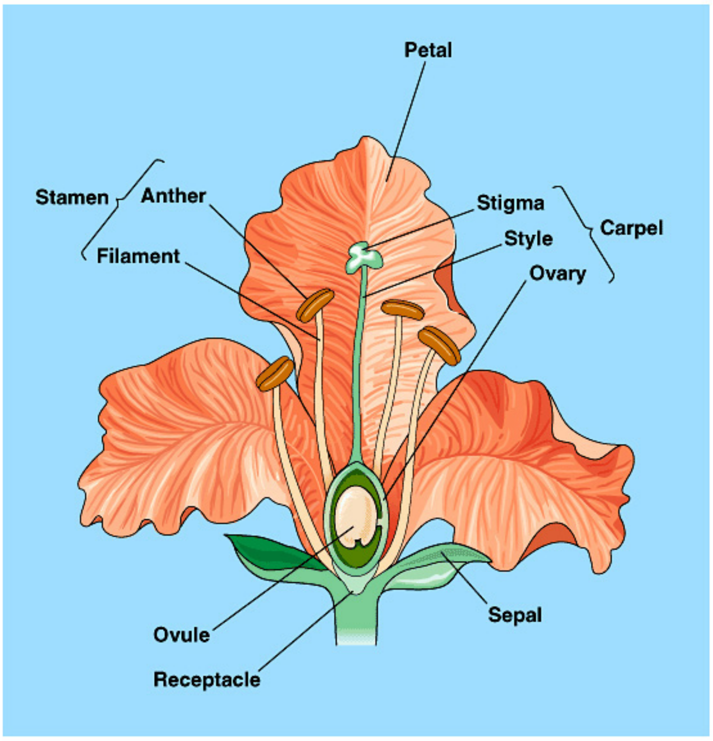


Figure 2.27: Example of flower structure, the flower components consist of petal, stamen, anther, filament, carpel, stigma, style, ovary, ovule, receptacle and sepal. (image from http://hawaii.hawaii.edu).

2.3.1 Plant Components and Their Representations

Problem 2.1 *How do we represent plant structures?*

This sections describes plant components and component representations using L-systems. In Section 2.2, the classes of L-systems are explained in detail. They have attracted the attention of the computer scientists who investigated them through formal language theory. Specialists in computer graphics, particularly Prusinkiewicz have used the L-systems to produce realistic images of trees, bushes, flowers and some images are well illustrated in [Prusinkiewicz and Lindenmayer, 1990].

For a three dimensional movement, a component is free to move in any X, Y, or Z direction. Hence, the directional angles in this case are three directions. The initial three directional angles, a_x, a_y, a_z of the first unit movement are set with respect to X, Y and Z axes, respectively. The directional angles of the other unit movements are computed in a similar fashion to that of the two dimensional case. Three constants, d_x, d_y, d_z, are used to adjust the direction of the unit movements. In addition, the physical location of the unit movement must be represented by XYZ coordinates. Therefore, a unit movement described in the Cartesian coordinate system is denoted by a hexaplet (X, Y, Z, a_x, a_y, a_z). After adding/subtracting d_x, d_y, d_z, the new XYZ coordinates of the movement is computed by multiplying the coordinates of the current movement with the rotation matrices R_x, R_y, R_z shown in equation (2.7). The rotation of a unit movement and its direction are captured in a symbolic form similar to two dimensional case by using these symbols /, \, &, ˆ , +, -, and |. The meaning of each symbol is explained in Table 2.5. We are now define the terms of "L-symbols" in the definition 2.8.

Definition 2.8 ***(L-symbols)*** *L-symbols are letters in set $V \cup S$ representing set of plant components and set of plant orientations S in plant prototype, where $V = \{$ I, i, P, p, A, L, F$\}$ and $S = \{$ /, \, &, ˆ , +, -, |, [,] $\}$.*

The meaning of each symbol is described in Table 2.5.

2.3.2 L-system Prototype

This section presents L-string definition and format used in L-system prototype. Some examples of L-system prototype are shown in simple L-system and Stochastic L-system. The definition of L-string is defined in Definition 2.9.

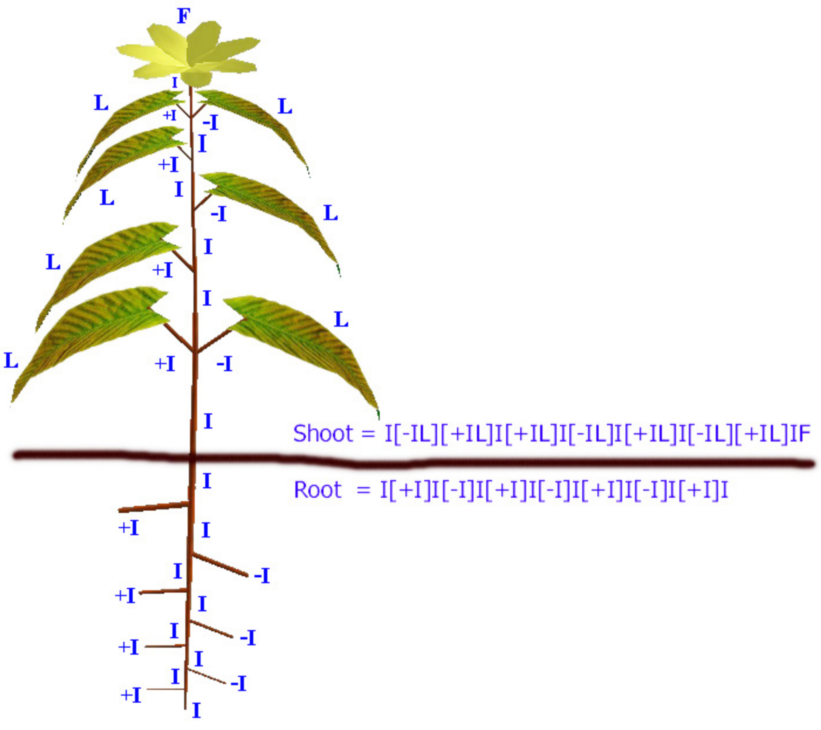
Definition 2.9 ***(L-string)*** *An L-string is a sequence of the combination of letters in set $V \cup S$ representing the final string after rewriting the L-system at the n^{th} iteration.*

An example of a simple turtle in a 3-dimensional space is given in production rule (2.13). $(d_x = d_y = d_z = 45, a_x = a_y = a_z = 0)$

$$\begin{array}{lcl} Shoot & = & I[-IL][+IL]I[+IL]I[-IL]I[+IL]I[-IL][+IL]IF \\ Root & = & I[+I]I[-I]I[+I]I[-I]I[+I]I[-I]I[+I]I \end{array} \tag{2.13}$$

Table 2.5: The symbol of three-dimensional turtle interpretation of plant structure.

Symbols	Meaning
I	Generate the plant shoot/root internode
i	Generate the plant shoot/root short internode
P	Generate the plant shoot/root petiole
p	Generate the plant shoot/root short petiole
A	Generate the plant shoot apex or root tip
L	Generate the plant shoot leaf
F	Generate the plant shoot flower
$+(\delta_z)$	Roll counterclockwise by angle δ_z, using rotation matrix $R_z(\delta_z)$
$-(\delta_z)$	Roll clockwise by angle δ_z, using rotation matrix $R_z(-\delta_z)$
$\&(\delta_y)$	Roll counterclockwise by angle δ_y, using rotation matrix $R_y(\delta_y)$
$\hat{}(\delta_y)$	Roll clockwise by angle δ_y, using rotation matrix $R_y(-\delta_y)$
$\backslash(\delta_x)$	Roll counterclockwise by angle δ_x, using rotation matrix $R_x(\delta_x)$
$/(\delta_x)$	Roll clockwise by angle δ_x, using rotation matrix $R_x(-\delta_x)$
$\vert$	Roll back, using rotation matrix $R_y(180)$
[	Push the current state of the turtle onto a pushdown stack to create a new branch
]	Pop a state from the stack and make it the current state of the turtle to close the branch



Shoot = I[-IL][+IL]I[+IL]I[-IL]I[+IL]I[-IL][+IL]IF

Root = I[+I]I[-I]I[+I]I[-I]I[+I]I[-I]I[+I]I

Figure 2.28: A simple L-system interpretation of simple plant.

The simple script of L-systems in production rule (2.13) can be visualized in Figure 2.28.

We define the symbol "I" for an internode, "L" for a leaf and "F" for a flower of the plant. From the example in shoot there are 13 internodes - six internodes for the main stem, seven internodes for petioles and their leaves in each direction - and one flower. In root part, there are 15 internodes - eight internodes for taproot, seven internodes for lateral roots. The visualized image is shown in Figure 2.28.

2.3.3 Stochastic L-systems

To implement variation of plant structures, the probability distribution of all possible cases for all species can be observed and categorized in m different possible cases with probability p_j where $j = 1, 2, 3, ..., m$. The production rule can be represented by the equation (2.14).

$$\begin{cases} a_i & \rightarrow \quad p_1\mu_1, p_2\mu_2, p_3\mu_3, ..., p_m\mu_m, \\ \displaystyle\sum_{j=1}^{m} p_j & = \quad 1, \mathit{and} \quad 0 < p_j < 1 \end{cases} \tag{2.14}$$

where a_i, p_j, μ_j are predecessor of i^{th} production rule, probability value of j^{th} species, and successor of j^{th} species, respectively. The summation of probability values of all possible successors m in i^{th} production rule is equal to one.

The syntax of the production rule is defined as the following format

$$Pred \quad = \quad (Prob_1)Succ_1, (Prob_2)Succ_2, (Prob_3)Succ_3, \ldots, (Prob_n)Succ_n, \tag{2.15}$$

where $Pred$ is the predecessor of successor $Succ_i$ with probability value $Prob_i$ when $i = 1, 2, 3, ..., n$. The sum of probabilities of all successors with the same predecessor $Pred$ is equal to one.

For example, a simple stochastic L-system is given in production rule (2.16).

$$I \quad = \quad (0.40)I[+iL]I, (0.30)I[-iL]I, (0.30)I[-iL][+iL]I \tag{2.16}$$

In the above production rule, the predecessor I is to be replaced by one of three possible cases in a set of successors with probability values 0.40, 0.30, 0.30, respectively. It can be visualized in Figure 2.29.

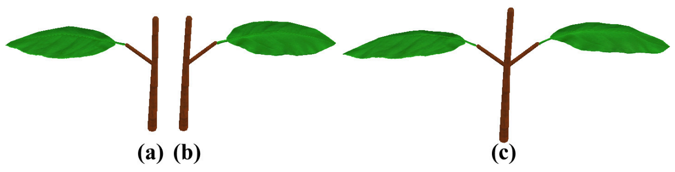


Figure 2.29: An example of stochastic L-System, (a) is $(0.40)I[+iL]I$, (b) is $(0.30)I[-iL]I$, and (c) is $(0.30)I[-iL][+iL]I$.

2.4 Plant Module Design

Problem 2.2 *How can we design the plant module based on L-systems?*

The L-system description of a plant is defined in form of a set of iterations, a set of directional and a sizing parameters, an initial string, a set of production rules and a set of terminating productions. This description has the following format in Table 2.6.

Table 2.6: The L-system prototype for plant representation.

Plant_Name {	Root {
Shoot {	Iterations=N_r
Iterations=N_s	Angle=δ_r
Angle=δ_s	Diameter=D_r
Diameter=D_s	Axiom=ω_r
Axiom=ω_s	Production 1
Production 1	Production 2
Production 2	...
...	Production n_r
Production n_s	Endrule
Endrule	Endproduction 1
Endproduction 1	Endproduction 2
Endproduction 2	...
...	Endproduction m_r
Endproduction m_s	}
}	}

The meaning of each keyword in Table 2.6 is given as follows:

Plant_Name

Plant_Name is the name of the plant module. There are two modules of a plant prototype. They are “Shoot” and “Root”, respectively. The prototype of the shoot module is similar to the root module and uses the same keyword.

Shoot, Root

Shoot and *Root* are used to separate the shoot (see subscript with s) and root (see subscript with r) module, respectively.

Iterations=N

This input sets the number of *iterations* N for selecting and rewriting the production rules where $N \geq 0$. Each production rule is selected according to the appearance of the symbols in the current string.

Angle=δ

This *angle* δ is used to set the angle of the branch where $-360 \leq \delta \leq 360$. For example, symbol "-" is to turn right by an angle δ , symbol " ˆ " is to pitch up by an angle δ and symbol "/" is to roll right by an angle δ.

Diameter=D

This *diameter* D syntax is used to set the diameter of the first internode where $D > 0$. The other internodes are controlled by the given function related to the first internode.

Axiom=ω

This *axiom* ω string is used to set the initial state of the plant structure. The first internode is located at the origin (0,0,0) and pointed towards the positive Y axis. The three angles for a 3-dimensional space (a_x, a_y, a_z) are set to zero for the first internode.

Production 1 . . . Production n

Each *production* consists of a set of predecessor and successor(s). The format of production is given in production rule (2.15). The probability of normal production rule is equal to 1. The predecessor is a set of single character and the successor is a set of string.

Endrule

To terminate the rewriting of a production rule, a terminating symbol must be substituted for the corresponding symbol used by the *endproduction rule* . The substitution rules are defined in the *endproduction* 1 to *endproduction m*. The endproduction rules are only called at the last N^{th} iteration.

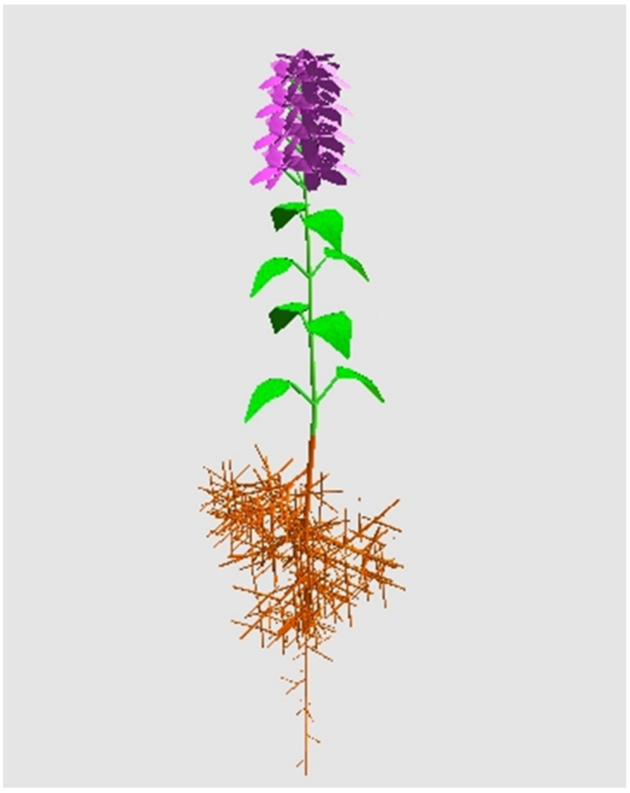
Endproduction 1 . . . Endproduction m

The format of endproduction rule is the same as the production rule. The endproduction is used to prevent some symbols which are not in { I, i, P, p, L, F, A }.

Character "{" and "}" The character "{" and "}" are the beginning and the ending of L-systems structure, respectively.

Figure 2.30 shows an example of 3D plant both in shoot and root part from the plant prototype in the left side. In the L-system code, there are *Shoot* and *Root* part. Each part consists of its own sub L-system. There are three iterations, rotating angle of each branch and sub-branch for 45 degrees, initial diameter for 1.5 units, one production rule, and three end production rules in the shoot part. In the root part, it consists of two iterations, 20 degrees for rotating angle, one unit for initial root diameter, two production rules and two end production rules, respectively. After rewriting method in both shoot and root part, the transformation of L-string in graphical form is visualized in the plant structure as Figure 2.30.

We now have the plant prototype for generating the plant structure, but the growth processes are needed to embed into the plant symbols. In this new method, the number



```
Plant{
    Shoot{
                Iterations=3
                Angle=45
                Diameter=1.5
                Axiom=iiii[-iL][+iL]I[/iL][\iL]BA
                A=I[-IIF][+IIF][/IIF][\IIF]AK
                ENDRULE
                A=I[-IIF][+IIF][/IIF][\IIF]K
                K=IIF
                B=I[-iL][+iL]I[/iL][\iL]i
    }
    Root{
                Iterations=2
                Angle=20
                Diameter=1
                Axiom=iii[-A]I[+A]Ii[-i]A[+i]BiIIiii
                A=I[/A][\A]B
                B=I[-B]I[+B]A
                Endrule
                A=Ii[-i][+i]iI
                B=Ii[-i][+i]iI
    }
}
```

Figure 2.30: An example L-system prototype and 3D plant structure both in shoot and root module.

of iteration is not used to animate the plant development, but the sigmoidal function of each plant component is combined during the growth processes. Next section describes more detail in growth function.

2.5 Growth Function

Problem 2.3 *How can we grow the plant using L-systems?*

This section explains the growth function used in this dissertation. The collected data from experiment can be used to calculate the appropriate parameter values. The raw data are approximated as a growth function in Figure 2.31 using a sigmoidal curve approximation.

The raw data are converted to growth function $G(t)$ of length or width at time t and is given below,

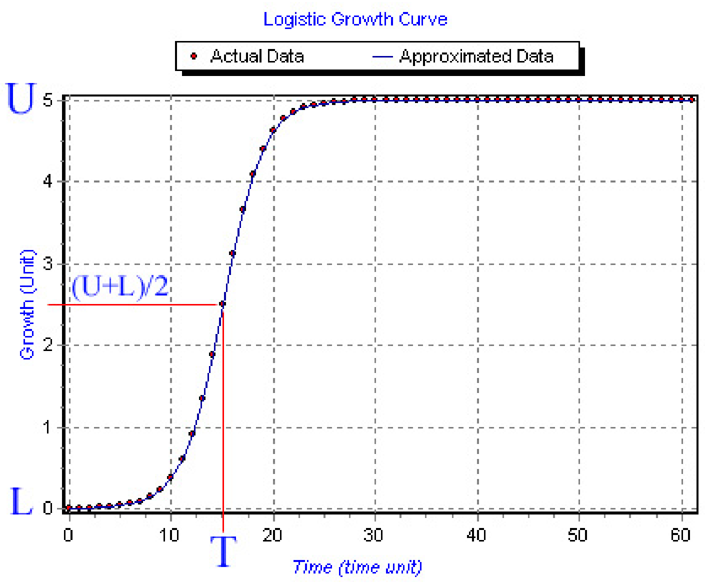


Figure 2.31: Sigmoidal curve approximation of $G(t)$.

$$G(t) = L + \frac{U - L}{1 + e^{m(T-t)}} \tag{2.17}$$

where

L	: the minimum value of length, width or diameter,
U	: the maximum value of length, width or diameter,
m	: the approximated slope of raw data,
T	: the time at $G(t) = (U + L)/2$, and
t	: the independent time variable.

Besides the growth function, there is another function that we use to control all the components of the plant topology, such as, the length of each internode from the first to the last internode. The function is

$$Y_i = c(a)^{n_i} \tag{2.18}$$

where Y_i is the length of internode i, c is a constant, a is a real value greater than zero and n_i is the level of internode i. We also use this equation to control the size of petiole, leaf, flower from the first to the last level. The initial time of each component is specified by the following linear equation.

$$B_i = \beta n_i + b \tag{2.19}$$

where B_i is the initial time of component i, β is the acceleration rate of B_i, n_i is the level of component i and b is a constant. Every component of plant is controlled by its growth function with either the same or different slope m, time T, the maximum U and the minimum L.

2.5.1 Growth Functions in Plant Components

This section describes growth function of growing plant based on logistic equation. The collected data can be approximated by parameter estimation using non-linear regression. The Levenberg-Marquardt method is applied for minimizing total square error E.

Growth Model

The exponential growth function had been widely used at the beginning of growth modeling. Let $G(t)$ be growth value or biomass of organism at time t. The exponential growth rate $\frac{dG(t)}{dt}$ at time t is proportional to the growth value $G(t)$, which is written in equation (2.20).

$$\frac{dG(t)}{dt} \propto G(t) \tag{2.20}$$

The exponential growth function is given in the ordinary differential equation form as follows.

$$\frac{dG(t)}{dt} = \alpha G(t) \tag{2.21}$$

The solution of equation (2.21) is

$$G(t) = \beta e^{\alpha t} \tag{2.22}$$

where α is growth rate constant of $G(t)$ and β is initial biomass $G(0)$. Since the exponential growth function has no bounded value, which cannot be described for real biomass for large time. The equation (2.22) is modified by adding the negative feedback term $(1 - \frac{G(t)}{U})$. The growth rate of biomass converges to the limited upper bounded value U.

Figure 2.32 shows the comparison of exponential and logistic growth with the same value of $\alpha = 0.2$. The growth value of both curves has the same growth behavior until $t \leq 35$, then exponential curve is exploding fast while logistic curve is converging to carrying capacity U.

$$\frac{dG(t)}{dt} = \alpha G(t)(1 - \frac{G(t)}{U}) \tag{2.23}$$

From equation (2.23), the feedback term $(1 - \frac{G(t)}{U})$ converges to 1 when $G(t) \ll U$ and converge to zero when $G(t)$ reaches the limited growth value U as $G(t) \to U$. Therefore the growth rate of the first stage is based on exponential growth and decreasing to be zero when $G(t)$ converges to limited growth value U. The solution of equation (2.23) is

$$G(t) = \frac{U}{1 + e^{-\alpha(t-\beta)}} \tag{2.24}$$

where $G(t)$ is growth value or biomass, t is time, U, α and β are parameters of growth function.

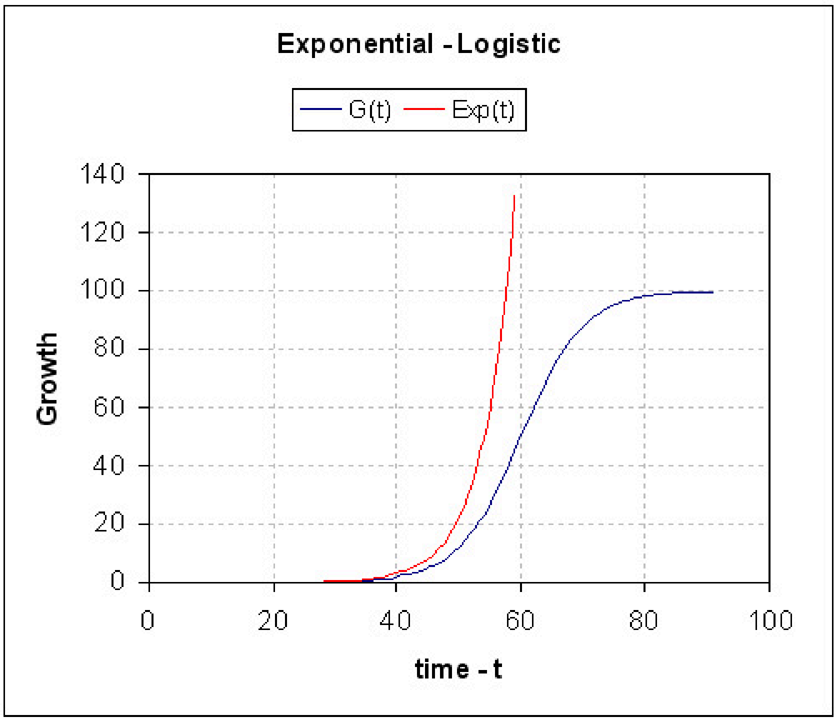


Figure 2.32: Comparison of exponential growth ($\beta = 0.001, \alpha = 0.2$), logistic growth ($U = 100, \alpha = 0.2, \beta = 60$).

The growth rate parameter α represents the width or steepness of sigmoidal curve in Figure 2.33(c), re-labelled m. The limit value parameter U specifies the different upper bounded value of logistic growth. The parameter β specifies the time when the growth reaches $\frac{U}{2}$, or the time midpoint of the growth behavior, re-labelled T. The equation (2.24) is rewritten as *Sigmoidal Growth* equation (2.25).

$$G(t) = \frac{U}{1 + e^{m(T-t)}} \tag{2.25}$$

where $G(t)$ is growth value or biomass, t is independent time variable, U is limit growth parameter, m is growth rate parameter and T is midpoint time parameter of growth at $G(t) = \frac{U}{2}$.

Equation (2.24) can be rewritten by changing parameter α in the form of Δt which means the growth period from 10% to 90%. It is visualized in Figure 2.35.

Let $t_{U10\%}$ be the growth value at 10%. So that $G(t_{U10\%})$ is defined in equation (2.26),

$$G(t_{U10\%}) = \frac{10}{100}U \tag{2.26}$$

and growth value at $t_{U90\%}$ is

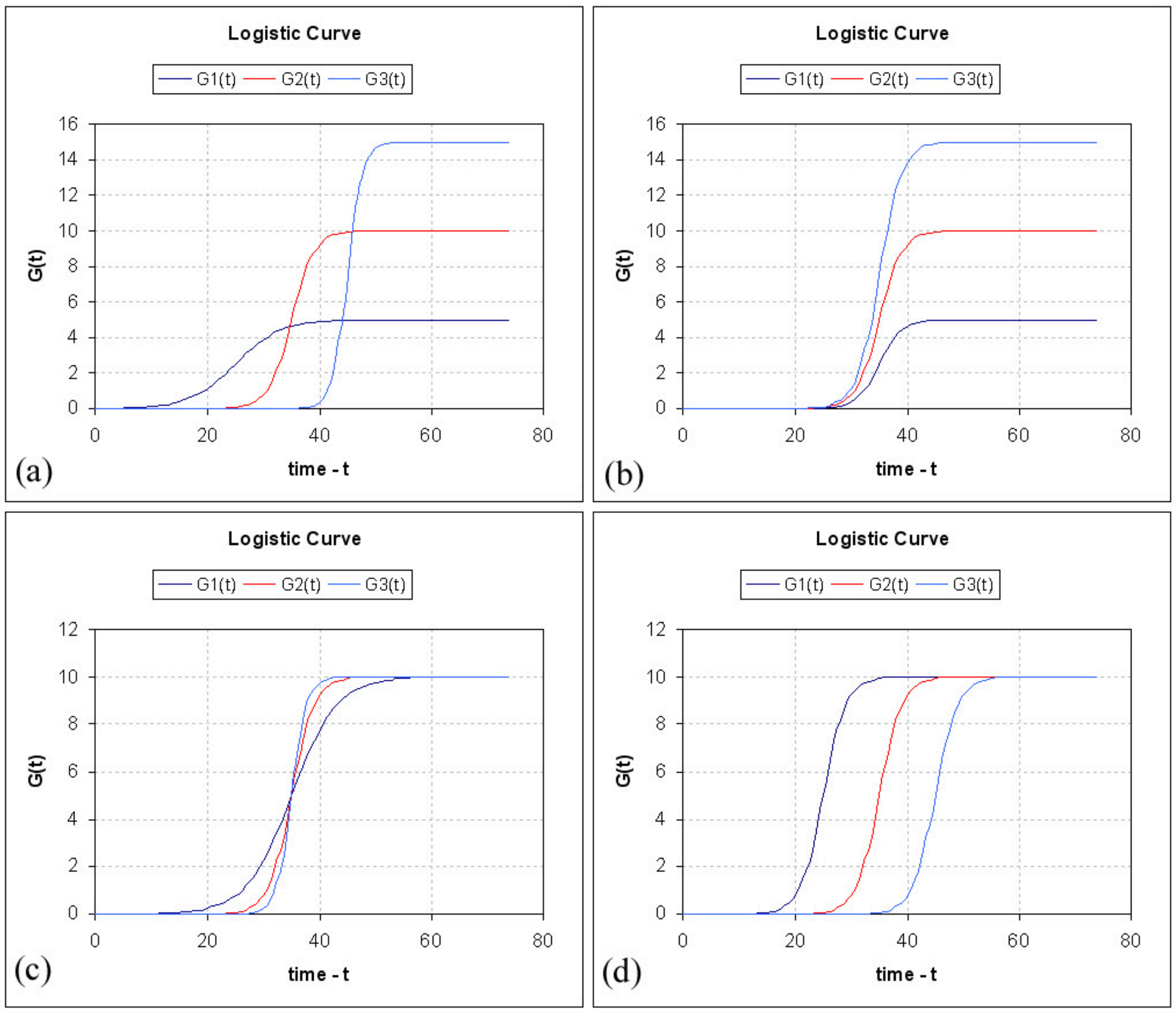


Figure 2.33: Logistic curves from equation (2.24), (a) different U, α, β, (b) different U, (c) different α, (d) different β.

Table 2.7: Three parameters of logistic curves in Figure 2.33.

Curves	$G_1(t)$			$G_2(t)$			$G_3(t)$		
Parameters	U	α	β	U	α	β	U	α	β
Figure (a)	5.00	0.25	25	10.0	0.50	35	15.0	0.75	45
Figure (b)	5.00	0.50	35	10.0	0.50	35	15.0	0.50	35
Figure (c)	10.0	0.25	35	10.0	0.50	35	10.0	0.75	35
Figure (d)	10.0	0.50	25	10.0	0.50	35	10.0	0.50	45

$$G(t_{U90\%}) = \frac{90}{100}U \tag{2.27}$$

Substitute equation (2.26) into equation (2.24), we get

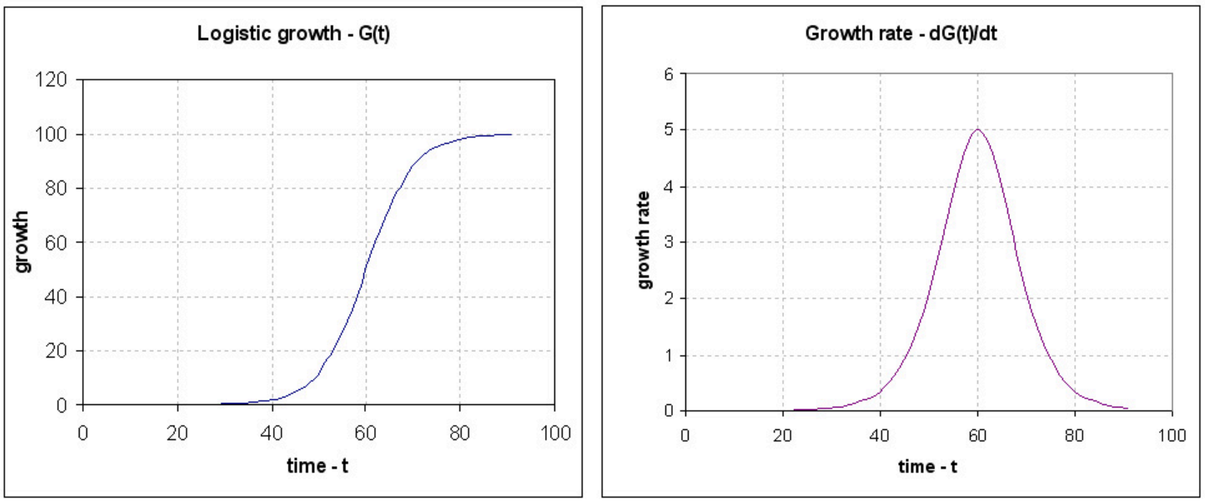


Figure 2.34: Comparison of logistic growth (left) and its growth rate (right) with parameters $(U = 100, m = 0.2, T = 60)$.

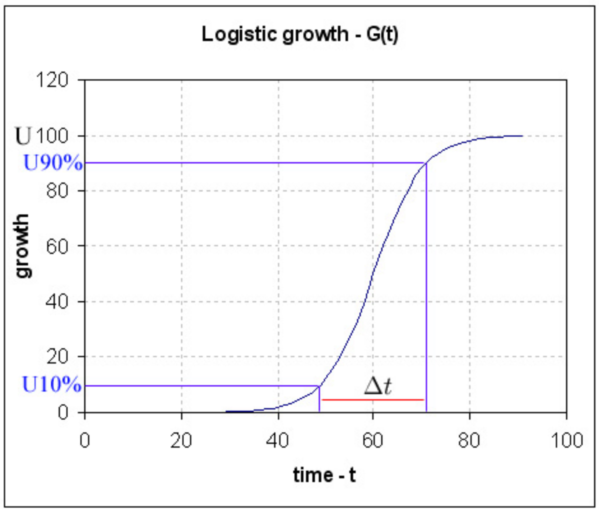


Figure 2.35: Growth period at 10% to 90%.

$$\begin{aligned} \frac{10}{100}U &= \frac{U}{1+e^{-\alpha(t_{U10\%}-\beta)}} \\ 9 &= e^{-\alpha(t_{U10\%}-\beta)} \\ ln(9) &= -\alpha(t_{U10\%}-\beta) \end{aligned} \tag{2.28}$$

and substitute equation (2.27) into equation (2.24),

$$\begin{aligned}
\frac{90}{100}U &= \frac{U}{1+e^{-\alpha(t_{U90\%}-\beta)}} \\
\frac{1}{9} &= e^{-\alpha(t_{U90\%}-\beta)} \\
-ln(9) &= -\alpha(t_{U90\%}-\beta).
\end{aligned} \tag{2.29}$$

Subtract equation (2.28) by (2.29), we have

$$ln(81) = \alpha(t_{U90\%}-t_{U10\%}). \tag{2.30}$$

Let Δt be the time of growth period from 10% to 90%, then $ln(81) = \alpha(\Delta t)$ and $\alpha = \frac{ln(81)}{\Delta t}$. Finally, substitute α into the equation (2.24), we get

$$G(t) = \frac{U}{1+e^{-\frac{ln(81)}{\Delta t}(t-\beta)}}. \tag{2.31}$$

Parameter β controls the growth period at $\frac{1}{2}U$, we change parameter β in the form of T which means the time at maximum growth rate.

Let T be $G(T) = \frac{U}{2}$ and substitute $G(T)$ into equation (2.31), we get

$$\begin{aligned}
\frac{U}{2} &= \frac{U}{1+e^{-\frac{ln(81)}{\Delta t}(T-\beta)}} \\
1 &= e^{-\frac{ln(81)}{\Delta t}(T-\beta)} \\
0 &= -\frac{ln(81)}{\Delta t}(T-\beta) \\
T &= \beta
\end{aligned} \tag{2.32}$$

Rearrange equation (2.31) and use $T = \beta$, it can be written as

$$G(t) = \frac{U}{1+e^{\frac{ln(81)}{\Delta t}(T-t)}}. \tag{2.33}$$

The equation (2.33) is comparable to equation (2.25) which means $m = \frac{ln(81)}{\Delta t}$.

We now have the derivation of growth function used in this work. In reality, plants are growing from the seed to be the actual plant with different growth behaviors: growth period, stable growth period, decreasing growth period, stable growth after decreasing stage, and death stage. In order to implement these behaviors, next section explains more detail in mathematical functions.

2.5.2 Plant Growth Process with Five Stages

For actual plant growth process, the growth function consists of five stages (see Figure 2.36). There are growth stage $G_1(t)$, stable stage $G_2(t)$, decreasing stage $G_3(t)$, stable after decreasing stage $G_4(t)$ and death stage $G_5(t)$. Each stage is shown in equations (2.34).

In equation (2.34) and Figure 2.36, there are eight time intervals. These functions, $G_1(t)$, $G_2(t)$, $G_3(t)$, $G_4(t)$ and $G_5(t)$, are used in $0 \leq t < T_2$, $T_2 \leq t < T_3$, $T_3 \leq t < T_5$, $T_5 \leq t < T_6$ and $T_6 \leq t \leq T_8$, respectively. The growth process with five stages are written in the following equations (2.34),

$$\begin{aligned}
G_1(t) &= L + \frac{U-L}{1+e^{m_1(T_1-t)}} && for \quad 0 \leq t < T_2, \\
G_2(t) &= U && for \quad T_2 \leq t < T_3, \\
G_3(t) &= L_1 + \frac{U-L_1}{1+e^{-m_2(T_4-t)}} && for \quad T_3 \leq t < T_5, \\
G_4(t) &= L_1 && for \quad T_5 \leq t < T_6, \\
G_5(t) &= L + \frac{L_1-L}{1+e^{-m_3(T_7-t)}} && for \quad T_6 \leq t \leq T_8,
\end{aligned} \tag{2.34}$$

where	L	: the minimum value of length, width or diameter,
	L_1	: the minimum value of length, width or diameter in the decreasing stage,
	U	: the maximum value of length, width or diameter,
	m_1	: the approximated slope of raw data in the growth stage,
	m_2	: the approximated slope of raw data in the decreasing stage,
	m_3	: the approximated slope of raw data in the death stage,
	T_1	: the time at $G(t) = (U + L)/2$,
	T_2	: the starting time at $G(t) = U$ in the stable stage after growth stage,
	T_3	: the ending time at $G(t) = U$ in the stable stage before decreasing,
	T_4	: the time at $G(t) = (U + L_1)/2$,
	T_5	: the starting time at $G(t) = L_1$ in the stable stage after decreasing,
	T_6	: the ending time at $G(t) = L_1$ in the stable stage after decreasing,
	T_7	: the time at $G(t) = (L_1 + L)/2$,
	T_8	: the ending time at $G(t) = L$ in the death stage, and
	t	: the independent time variable.

Function $G_3(t)$, $G_4(t)$ and $G_5(t)$ are designed for optional processes in biological behaviors.

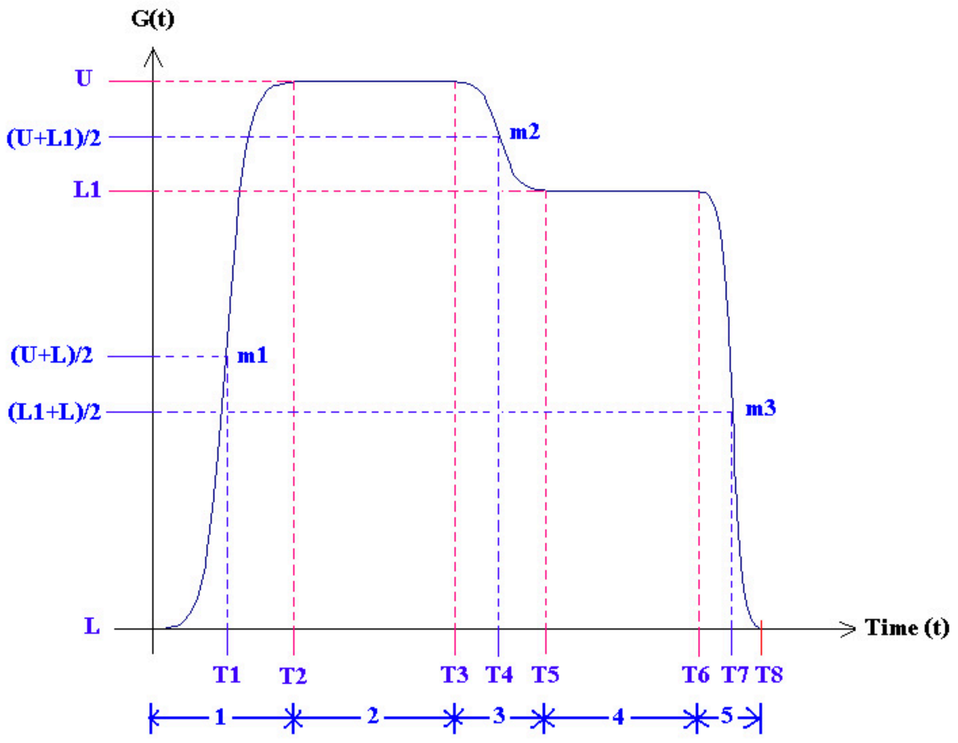


Figure 2.36: Prototype of five stages of growth function.

We now have the sigmoidal functions describing in different growth behaviors of plant growth. From the general experiments, the collected data can be approximated for solving the optimal parameter values. The next section provides the parameter estimation based on sigmoidal function.

2.5.3 Parameter Estimation Based on Sigmoidal Function

Problem 2.4 *How can we estimate all parameters in N-pulses sigmoidal function?*

This section describes a procedure of parameter estimation for approximation the optimal parameters of sigmoidal growth function in equation (2.25). The model function can be modified in N-pulses of the general sigmoidal growth function. It is given in equation (2.35).

$$\begin{aligned} G(t) &= \frac{U_1}{1+e^{m_1(T_1-t)}} + \frac{U_2}{1+e^{m_2(T_2-t)}} + ... + \frac{U_N}{1+e^{m_N(T_N-t)}} \\ &= \sum_{k=1}^{N}(\frac{U_k}{1+e^{m_k(T_k-t)}}) \end{aligned} \tag{2.35}$$

To simplify more general parameters, all parameters in equation (2.35) can be rewritten to

$$\begin{aligned} G(t) &= \frac{a_1}{1+e^{a_2(a_3-t)}} + \frac{a_4}{1+e^{a_5(a_6-t)}} + ... + \frac{a_{3N-2}}{1+e^{a_{3N-1}(a_{3N}-t)}} or \\ &= \sum_{k=1}^{N} \left(\frac{a_{3k-2}}{1+e^{a_{3k-1}(a_{3k}-t)}} \right) \qquad (2.36) \\ &= G(t;a) \end{aligned}$$

where	G	: the growth function of nonlinear regression equation or dependent variable,
	t	: the time variable or independent variable of nonlinear regression equation,
	a	: the parameters of nonlinear regression equation.

The Levenberg-Marquardt (LM) method is used for calculating the set of parameter values of nonlinear regression equation. Let n be the number of measured data points (t_i, G_i) ; $i = 1, 2, 3, \ldots, n$ with m parameters; a_j; $j = 1, 2, 3, \ldots, m$.

Sum square error (SSE) is calculated from observed data G_t and approximated value $G(t;a)$ at time t in equation (2.37).

$$\begin{aligned} E(a) &= \sum_{i=1}^{n} (G_i - G(t_i;a))^2 \\ &= \sum_{i=1}^{n} \left(G_i - \sum_{k=1}^{N} \left(\frac{U_k}{1+e^{m_k(T_k-t_i)}} \right) \right)^2 \qquad (2.37) \\ &= \sum_{i=1}^{n} \left(G_i - \sum_{k=1}^{N} \left(\frac{a_{3k-2}}{1+e^{a_{3k-1}(a_{3k}-t_i)}} \right) \right)^2 \end{aligned}$$

where	$E(a)$	: the sum square error of nonlinear regression equation,
	G_i	: the growth value of measured data at i^{th} data point,
	G	: the growth function of nonlinear regression equation or dependent variable,
	t_i	: the time variable or independent variable at i^{th} data point, and
	a	: the parameters of nonlinear regression equation.

$E(a)$ can be written in second order of Taylor's series in equation (2.38).

$$E(a_{t+1}) \approx E(a_t) + \nabla E(a_t)^T \cdot (a_{t+1} - a_t) + \tfrac{1}{2}(a_{t+1} - a_t)^T \cdot \nabla^2 E(a_t) \cdot (a_{t+1} - a_t) \qquad (2.38)$$

where

$$\nabla E(a_t) = \begin{bmatrix} \dfrac{\partial E(a_t)}{\partial a_{1t}} \\ \dfrac{\partial E(a_t)}{\partial a_{2t}} \\ \vdots \\ \dfrac{\partial E(a_t)}{\partial a_{mt}} \end{bmatrix}, \qquad (2.39)$$

and $\nabla^2 E(a_t)$ can be computed from equation (2.40).

$$
\begin{aligned}
\nabla^2 E(a_t) &= \begin{bmatrix}
\frac{\partial^2 E(a_t)}{\partial a_{1t}\partial a_{1t}} & \frac{\partial^2 E(a_t)}{\partial a_{1t}\partial a_{2t}} & \frac{\partial^2 E(a_t)}{\partial a_{1t}\partial a_{3t}} & \cdots & \frac{\partial^2 E(a_t)}{\partial a_{1t}\partial a_{mt}} \\
\frac{\partial^2 E(a_t)}{\partial a_{2t}\partial a_{1t}} & \frac{\partial^2 E(a_t)}{\partial a_{2t}\partial a_{2t}} & \frac{\partial^2 E(a_t)}{\partial a_{2t}\partial a_{3t}} & \cdots & \frac{\partial^2 E(a_t)}{\partial a_{2t}\partial a_{mt}} \\
\frac{\partial^2 E(a_t)}{\partial a_{3t}\partial a_{1t}} & \frac{\partial^2 E(a_t)}{\partial a_{3t}\partial a_{2t}} & \frac{\partial^2 E(a_t)}{\partial a_{3t}\partial a_{3t}} & \cdots & \frac{\partial^2 E(a_t)}{\partial a_{3t}\partial a_{mt}} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial^2 E(a_t)}{\partial a_{mt}\partial a_{1t}} & \frac{\partial E(a_t)}{\partial a_{mt}\partial a_{2t}} & \frac{\partial E(a_t)}{\partial a_{mt}\partial a_{3t}} & \cdots & \frac{\partial^2 E(a_t)}{\partial a_{mt}\partial a_{mt}}
\end{bmatrix} \\
&= [h_{i,j}] \\
&= \left[\frac{\partial^2 E(a_t)}{\partial a_{it}\partial a_{jt}}\right] \quad \forall i,j; \qquad \textit{where } i,j = 1,2,3,...,m, \textit{ and } m = 3N.
\end{aligned} \tag{2.40}
$$

Equation (2.38) can be rewritten in gradient vector g and Hessian matrix H in equation (2.41).

$$E(a_{t+1}) = E(a_t) + g^T \cdot (a_{t+1} - a_t) + \frac{1}{2}(a_{t+1} - a_t)^T \cdot H \cdot (a_{t+1} - a_t) \tag{2.41}$$

where	g	: the vector of first order partial derivative of $E(a_t)$,
	H	: the vector of second order partial derivative of $E(a_t)$,

To compute parameter a for minimizing an error $E(a)$, the partial derivative with respect to a_n is computed as follows:

$$
\begin{aligned}
\frac{\partial E(a_{t+1})}{\partial a_t} &= \frac{\partial E(a_t)}{\partial a_t} + \frac{\partial g^T \cdot (a_{t+1} - a_t)}{\partial a_t} + \frac{1}{2}\frac{\partial (a_{t+1} - a_t)^T \cdot H \cdot (a_{t+1} - a_t)}{\partial a_t} \\
0 &= \frac{\partial E(a_t)}{\partial a_t} + g^T(-1) + H(a_{t+1} - a_t)(-1)
\end{aligned} \tag{2.42}
$$

consider the minimum point by given $\frac{\partial E(a_t)}{\partial a_t} = 0$, then

$$
\begin{aligned}
0 &= g^T + H(a_{t+1} - a_t) \\
a_{t+1} &= a_t - H^{-1}g,
\end{aligned} \tag{2.43}
$$

The term $H^{-1}g$ in equation (2.43) is step size of changing of parameter a in each iteration. Levenberg-Marquardt proposed to change the term $H^{-1}g$ to $(H + \lambda I)^{-1}g$ in order to reduce the number of iteration. It means the step size is small when λ is big and step size is big when λ is small. This method results in the reduction of the number of iteration by equation (2.44). The λ value depends on $E(a)$ in each iteration.

$$a_{t+1} = a_t - (H + \lambda I)^{-1} g, \tag{2.44}$$

To reduce the complexity for calculation the matrix H, we can derive H in term of Jacobian matrix (see [Suratgar et al., 2005]). The equation (2.44) can be rewritten in the following equation

$$a_{t+1} = a_t - (2J \cdot J^T + \lambda I)^{-1} J, \tag{2.45}$$

where $H \cong 2J \cdot J^T + S$, $S = \sum_{i=1}^{n}(E(a_i) \cdot \nabla^2 E(a_i))$, but S is small value, then $H \cong 2J \cdot J^T$.

Levenberg-Marquardt's Algorithm for Parameter Estimation

The Levenberg-Marquardt's Algorithm is a well-known method for optimizing error [Press et al., 1992]. The algorithm is given in the following consecutive steps in Algorithm 2.1.

Algorithm 2.1 ***LevMar***

1. Given an initial guess for the set of fitted parameters $a_0, (t = 0)$ and tolerance value σ (i.e. 10^{-10}),
2. Compute gradient vector g and Hessian matrix H,
3. Take a modest value for step size λ, let $\lambda = 0.001$,
4. Calculate $\delta a_{t+1} = (H + \lambda I)^{-1} g$ and consider the following conditions,
   - If $E(a_{t+1}) \geq E(a_t)$ increase λ by a factor of 10 then compute (4),
   - If $E(a_{t+1}) < E(a_t)$ decrease λ by a factor of 10, update the trial solution $a_{t+1} = a_t + \delta a_{t+1}$,
   - If $E(a_{t+1}) \leq \sigma$ then stop the process, else compute (4), where σ is minimum error,
5. Solution of the set of fitted parameters is a_{t+1}.

Finally, we get all fitted parameter values from the parameter estimation method. All fitted parameter values can be used in the model to simulate the plant growth processes. The next section describes the procedure of plant simulation and shows some results.

2.6 Procedure of Plant Simulation

Problem 2.5 *How does the plant simulation work?*

2.6.1 L-system Transformation

After measuring the plant, we have the plant structure as L-system code. There are two parts of L-system code. The first part is shoot module and the second part is root module. The L-strings of shoot and root parts are generated by rewriting of shoot and root modules (No. 1, No.3) and these L-strings represent the shoot and root structures (No. 2, No. 4) of the plant, respectively. These processes are given in Figure 2.37.

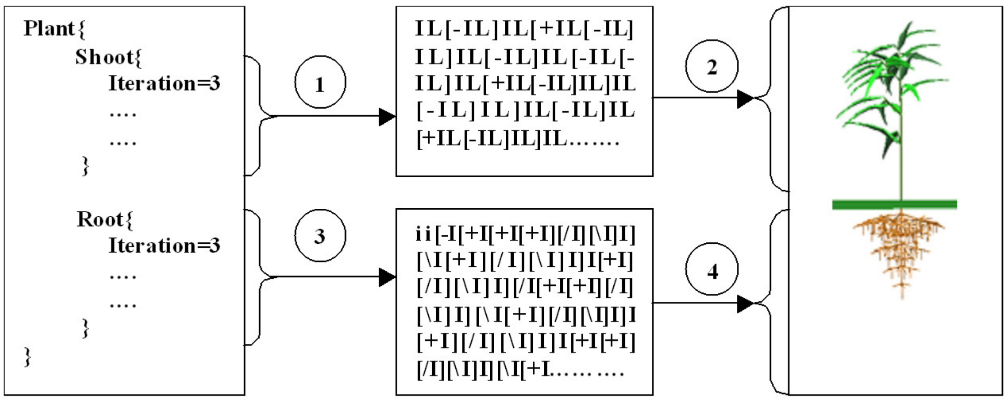

Plant{
Shoot{
Iteration=3
....
....
}
Root{
Iteration=3
....
....
}
}
1
IL[-IL]IL[+IL[-IL]IL]IL[-IL]IL[-IL[-IL]IL[+IL[-IL]IL]IL[-IL]IL]IL[-IL]IL[+IL[-IL]IL]IL.......
2
3
ii[-I[+I[+I[+I][/I][\I]I][\I[+I][/I][\I]I]I[+I][/I][\I]I][/I[+I[+I][/I][\I]I][\I[+I][/I][\I]I]I[+I][/I][\I]I]I[+I[+I][/I][\I]I][\I[+I.........
4

Figure 2.37: L-system transformation (see [Chuai-Aree et al., 2002]).

The structure of simulation starts from data collection of the actual plant and also captures the structure of the plant to be the L-system code. The measured data are approximated by the growth function based on a sigmoidal curve. The L-system code is compiled by the rewriting method to be an L-string at the last iteration. The L-string and the growth function of each symbol are combined together and represent the structure of the plant and its development. The system generates the 3D-plant development by visualization method. We have to evaluate the plant growth and adjust some parameters, such as, the size of each internode, to make the plant model and plant growth to look more realistic. The flow diagram of our simulation is shown in Figure 2.38.

In order to understand the plant structure using the stochastic and bracketed L-systems, an example of plant structure is given below. There are three iterations of shoot part and root part. The shoot part has only one production rule with three cases of stochastic L-system and one *endproduction* rule $A{=}IL$. For the root part there are four production rules using the stochastic L-system, one production rule using normal rule and five *endproduction* rules A, B, C, D and E.

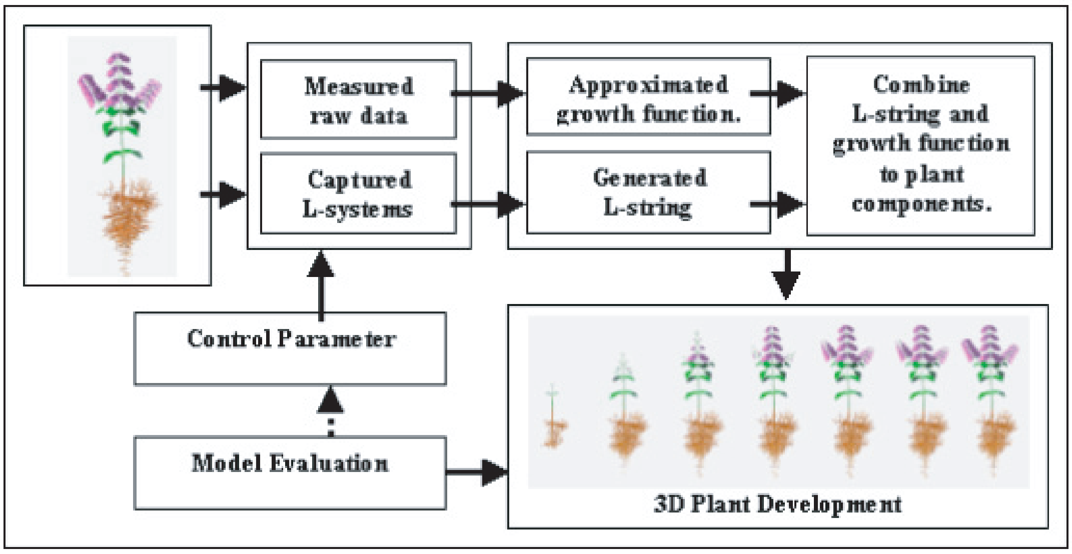


Figure 2.38: Flow diagram for simulating and visualizing of plant growth [Chuai-Aree et al., 2002].

```
Plant{
  Shoot{
    $Iterations = 3$
    $Angle = 45$
    $Diameter = 0.7$
    $Axiom = A$
    $A = (0.33)A[+A]A[-A]A, (0.33)A[+A]A, (0.34)A[-A]A$
    Endrule
    $A = IL$
  }
  Root{
    $Iterations = 3$
    $Angle = 45$
    $Diameter = 0.5$
    $Axiom = ii[-A][+B][/C][\backslash D]E$
    $A = (0.5)I[+A][/A][\backslash A]A, (0.25)I[+A][/A]A, (0.25)I[+A][\backslash A]A$
    $B = (0.5)I[-B][/B][\backslash B]B, (0.25)I[-B][\backslash B]B, (0.25)I[-B][/B]B$
    $C = (0.5)I[-C][+C][\backslash C]C, (0.25)I[-C][\backslash C]C, (0.25)I[+C][\backslash C]C$
    $D = (0.5)I[-D][+D][/D]D, (0.25)I[+D][/D]D, (0.25)I[-D][/D]D$
    $E = I[-A][+B][/C][\backslash D]E$
    Endrule
    $A = I$
    $B = I$
    $C = I$
    $D = I$
    $E = I$
  }
}
```

The given L-system code is compiled and then we get the different sequences of the production rules. From the example there are 27 different structures (33 different structures, 3 cases of stochastic rule for 3 iterations) of shoot part and 312 different structures (81 cases of stochastic rule for 3 iterations) of root part.

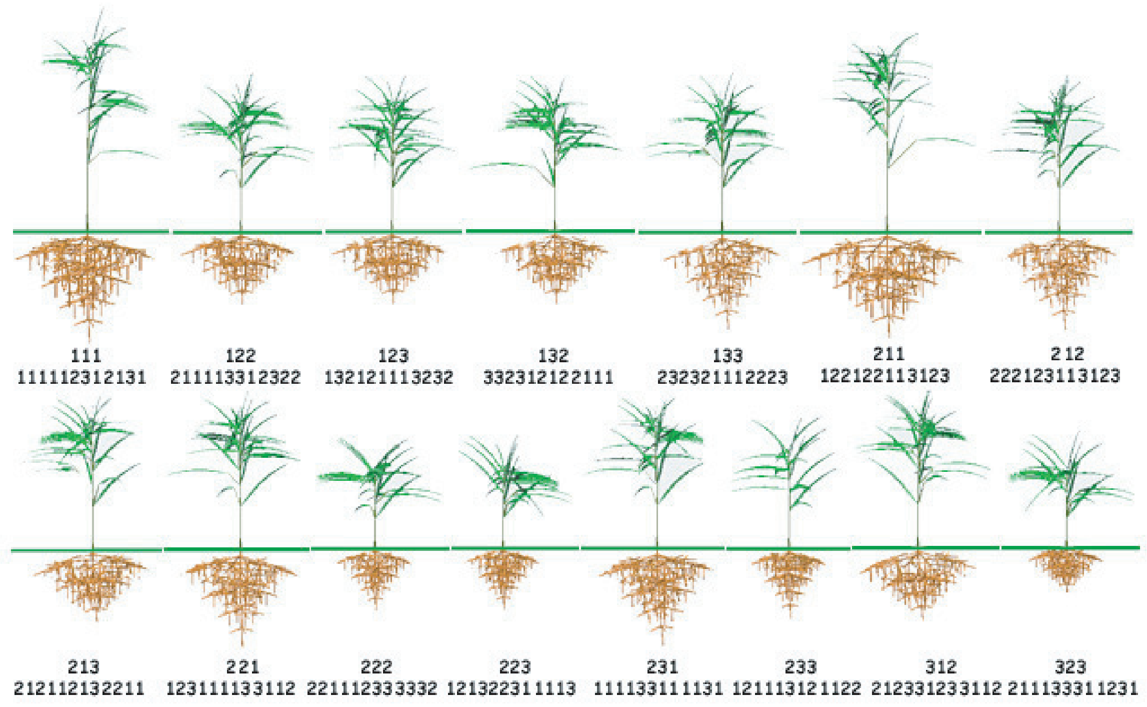


Figure 2.39: The different plant structure generated by Stochastic L-system [Chuai-Aree et al., 2002].

Fifteen of twenty-seven structures of this plant prototype are given in Figure 2.39. There are two lines of the number of each plant structure. The first line is the sequence of shoot structure and the root structure is given in the second line. The visualized images of plant growth from time t=22 to t=80 are shown in Figure 2.40.

The visualization of the example is shown in vegetative state. Cylinders are used to represent internodes and petioles segments. Spheres are used to represent jointed internodes. Triangular polygons are used to represent leaves and flowers.

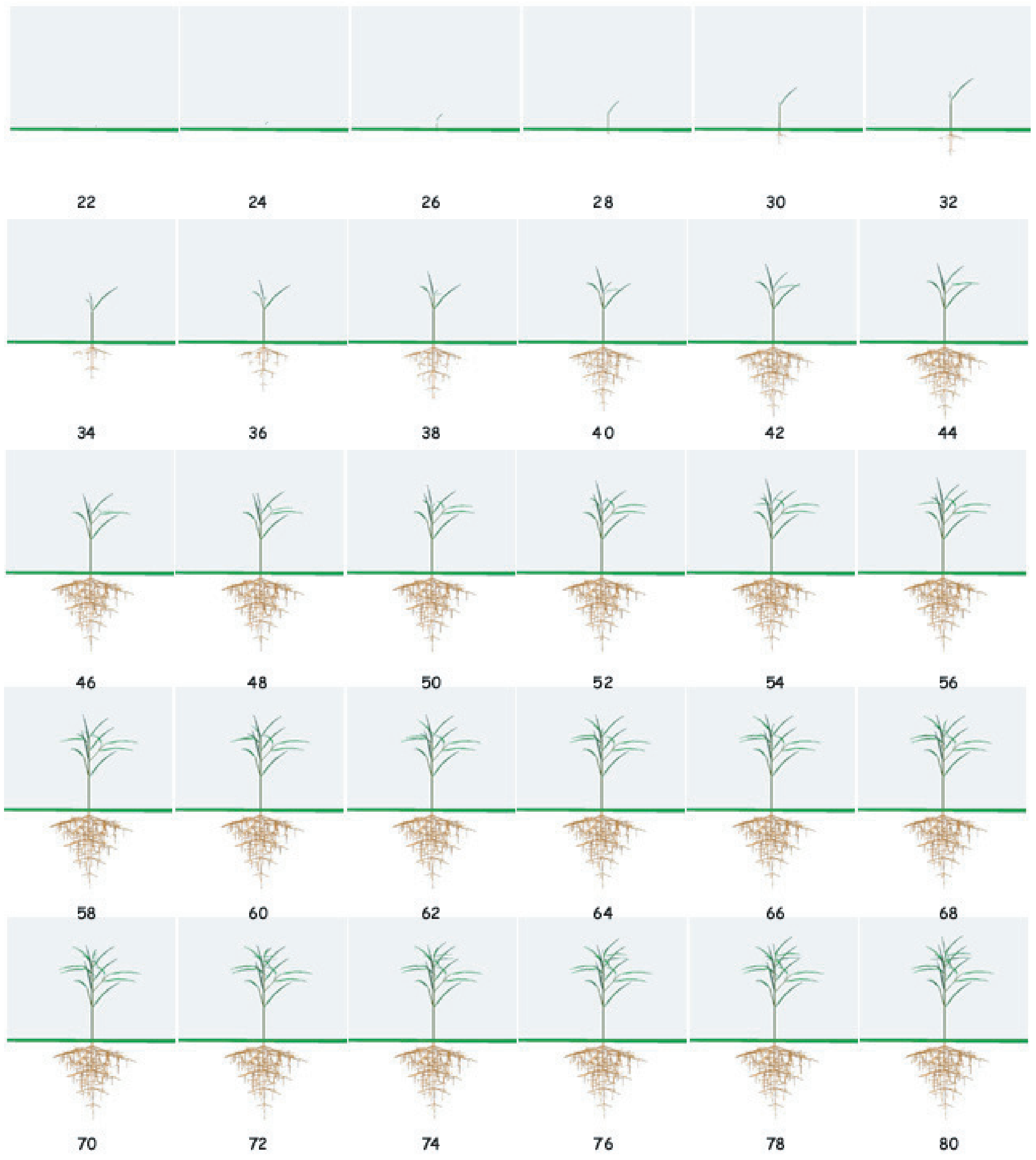


Figure 2.40: The development of plant growth from time t=22 to t=80 [Chuai-Aree et al., 2002].

2.6.2 Discussion and Conclusion

A prototype program called *PlantVR* has presented to create the continuous development of plant shoot and root models by parametric functional symbols based on the *bracketed L-system* and *stochastic L-system*. The optional *Endrule* keywords are added to the L-system in order to prevent some symbols that are not defined for plant definition in Table 2.5. The visualization technique makes the plant looking more realistic and every component can be controlled by the growth functions. The development of plant growth is smoother and more natural. This prototype can be used to generate a realistic model of the plant.

The prototype can be improved to observe the plant growth from specific experiments, e.g. the experiment of plant growth based on environmental factors; light, water, nutrient, pH, and salinity. The prototype can be shown in Figure 2.41. The growth functions approximated from parameter estimation are based on the specific experiments.

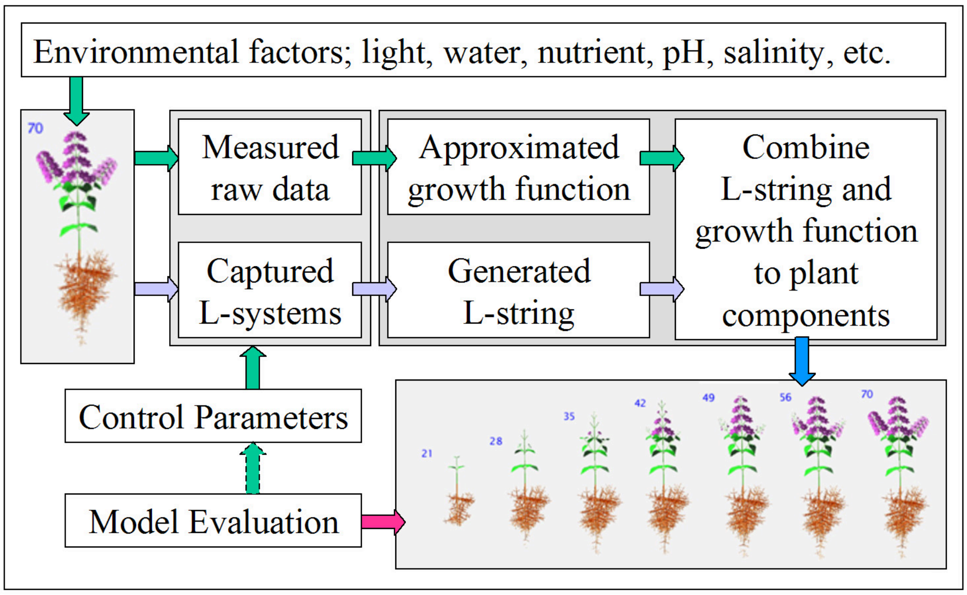

Environmental factors; light, water, nutrient, pH, salinity, etc.
70
Measured raw data
Approximated growth function
Combine L-string and growth function to plant components
Captured L-systems
Generated L-string
Control Parameters
Model Evaluation
21
28
35
42
49
56
70

Figure 2.41: Diagram of plant simulation and visualization in *PlantVR*.

2.7 Inverse Problem of L-Systems

Problem 2.6 *How can we reconstruct the L-systems code from input image or volume?*

This section describes the inverse problem of L-Systems which representing the branching structures, e.g. root, shoot, leaf structures in 2D and 3D structure. Not only the plant structure, but also the other branching structures for example the vein, lung or other structures in human body can be applied.

L-systems have been used to generate and describe the geometrical structures for example, branch structures, graph structures, both in biology and medicine. The L-systems consist of a number of iteration n, an initial string ω and a set of production rules P. The production rules are a set of predecessors a and successors χ. They are written as the form $a \rightarrow \chi$. The production rules have been defined and analyzed from the real structure by a structure decomposition manually. The rules are compiled and transformed to represent 2D and 3D structure. However, the complicated structures are not easy to decompose and time consuming to get such production rules. In this section, we propose an algorithm to solve this problem automatically from 2D input images by given initial *pixels* (picture elements) or *voxels* (volume elements). The data acquisition can be retrieved from 2D image scanner, camera, CT-scanner or MRI. The methods namely *Region and Volume Growing Methods* are applied to bound the target object. The skeletonizing method is an important part in our reconstruction. The L-systems are reconstructed for representing the structure from 2D input image or sliced images of the volume data.

In recent years there has been significant interest in using graph-based abstraction of skeleton for qualitative shape recognition. The application of such methods has affected very much in biology, medicine, image recognition, etc. There are many different methods to overcome the reconstruction of branching structures. Some methods are very sensitive to noisy disturbance. Methods based on *Voronoi techniques* in [Russ, 1995] preserve topology, but heuristic pruning measures are introduced to remove unwanted edges. Methods based on Euclidean distance functions in [Russ, 1995] can localize skeletal points accurately, but often at the cost of altering the object's topology. In this dissertation we introduce a region growing method, which is represented by the method of finite difference method. It is easy to implement in order to calculate the target object.

Figure 2.42 shows the diagram of L-systems for describing the plant structure in Chuai-Aree *et al.* [Chuai-Aree et al., 2005a]. The inverse problem of L-systems is the method to reconstruct the branch structures from 2D input image or sliced images of the volume data to a set of production rules.

This section is divided into 11 parts as follows: data acquisition, computed tomography, the flow diagram for reconstruction method of branching structures, anisotropic diffusion filtering, region and volume growing method, skeletonizing method, construction of branching structures, resolution reduction, L-string construction algorithm, experiment and results, and discussion and conclusion, respectively.

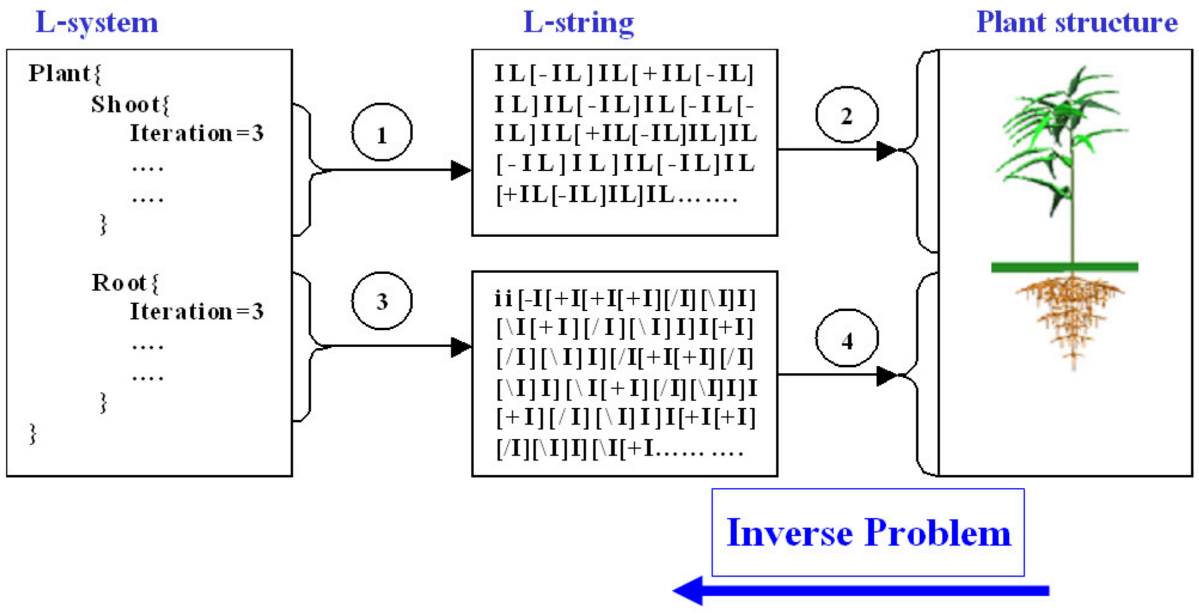

L-system
L-string
Plant structure
Plant{
Shoot{
Iteration=3
....
....
}
Root{
Iteration=3
....
....
}
}
1
2
3
4
IL[-IL]IL[+IL[-IL]IL]IL[-IL]IL[-IL[-IL]IL[+IL[-IL]IL]IL[-IL]IL]IL[-IL]IL[+IL[-IL]IL]IL.......
ii[-I[+I[+I[+I][/I][\I]I][\I[+I][/I][\I]I]I[+I][/I][\I]I][/I[+I[+I][/I][\I]I][\I[+I][/I][\I]I]I[+I][/I][\I]I]I[+I[+I][/I][\I]I][\I[+I.........
Inverse Problem

Figure 2.42: The transformation of L-systems, L-string, plant structure and its inverse problem.

2.7.1 Data Acquisition

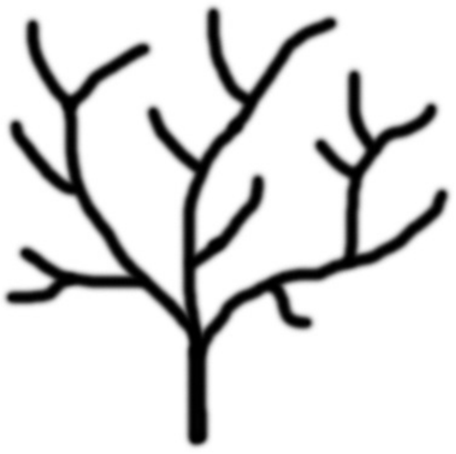
Problem 2.7 *How can we retrieve the input data both in 2D and 3D?*

The data acquisition can be retrieved from 2D image scanner, camera, *CT-Scanner* or *MRI* by scanning the source of data. 2D image containing the tree-like network, e.g. photographs, 3D volume data of soil i.e pot of root, human body having network structures; i.e. lung structure, blood vessels, and other sources of data can be used as input data for the process of inverse problem of L-systems. Figure 2.43 shows some 2D examples of branch structures of plant, rice root, leaf vein, and neuron. An example of 3D input data using CT-scanner is presented in Figure 2.44. We scanned the pot of plant using CT-scanner for observing root structures.

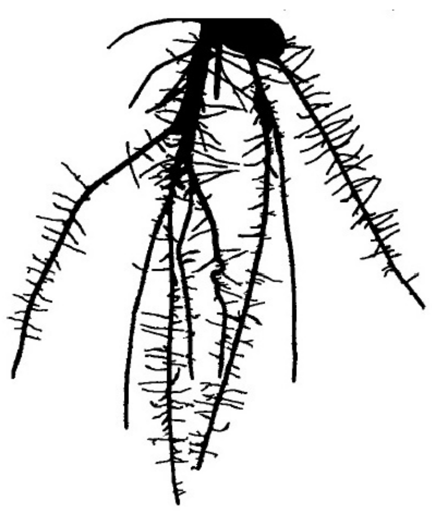
2.7.2 Computed Tomography (CT) Scanner

This section presents an example of CT data from soil volume in cylindrical pipe. The volume size is $512 \times 512 \times 1278$ voxels $(W \times H \times L)$. The gray level of each voxel in cylinder volume represent the X-ray absorbance. Plant roots have a value between -100 and +300. Air is -1000, water is 0, clay is between +900 and +1200, sand is between +500 and +800. Organic matter has the same absorbance as root. It can be distinguished from the roots only on morphology - roots are "wormlike" structures. We are working with organic deposits build up 2-mm-thick horizontal layers in the soils. The range of X-ray absorbance of plant is presented in Figure 2.45.

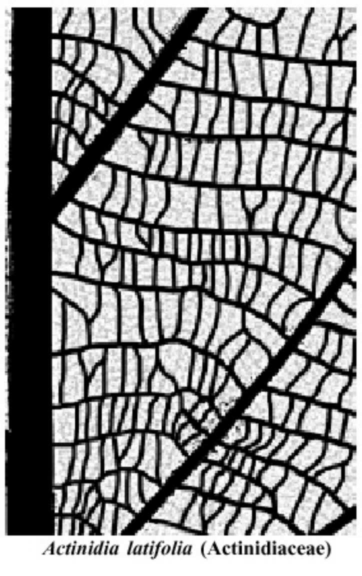
Figure 2.46 shows the cylindrical pot of *canola* root. The cross-section and distribution of soil particles before planting the canola plant was coded by using *Amira* software and shown in Figure 2.47.



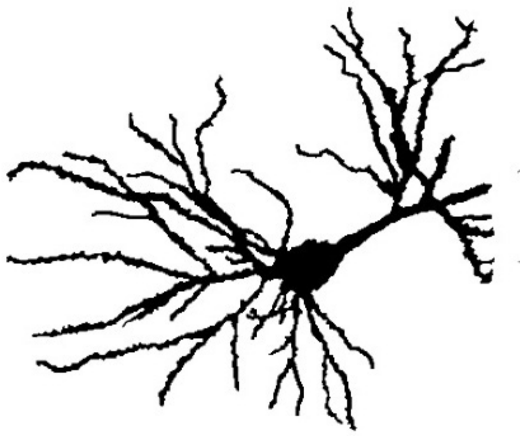
(a) brach image



(b) rice root



(c) leaf vein



(d) neuron image

Figure 2.43: Example of 2D input images containing network structure, (a) branch image, (b) rice root, (c) leaf vein, (d) neuron image.

Figure 2.48, Figure 2.49, and Figure 2.50 illustrate the canola pot with vertical cross-section, soil particles, segmented canola root, tunnels used for fertilizer injection, and organic matters.

Figure 2.51 shows the segmented canola root with organic deposit layers in the pot. The root are growing to the organic layers which are in the same range of X-ray absorbance as the roots.

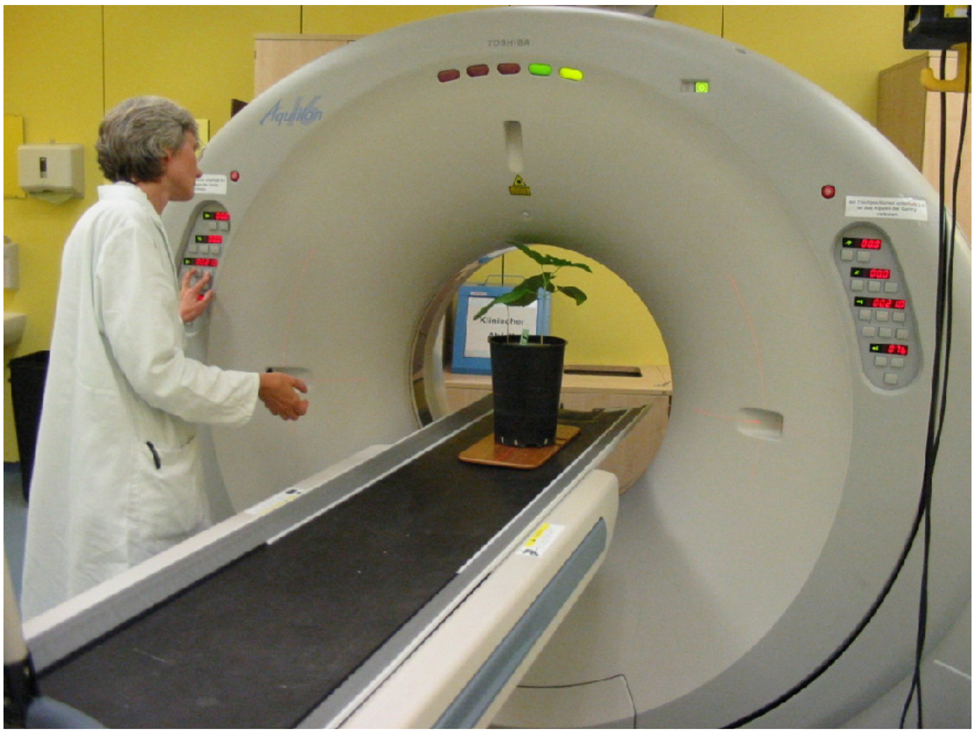
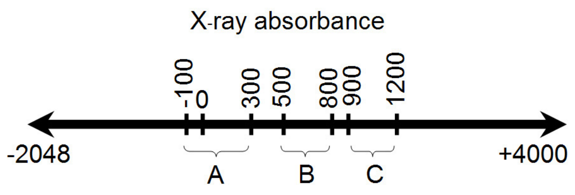
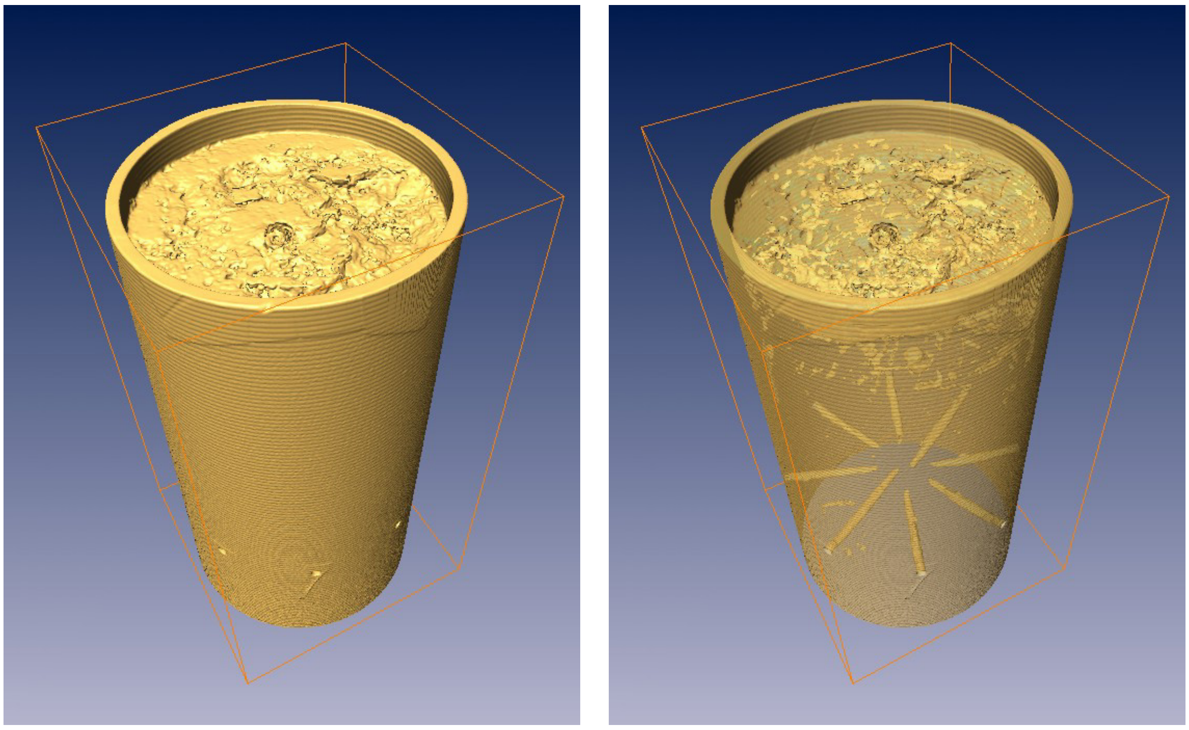


Figure 2.44: Example of 3D input data using CT (Computed Tomography) scanner (at CT Abteilung, DKFZ, Germany).



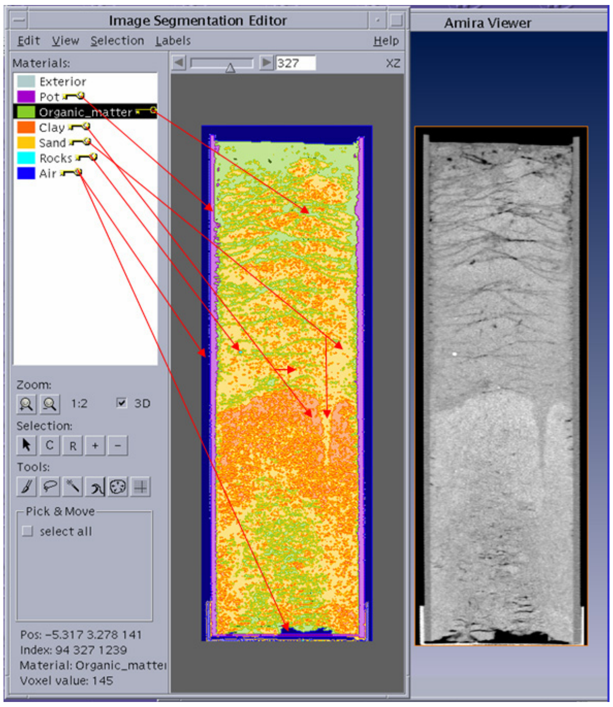
Object	Absorbance Value
Air	< -1000
Water	0
Root/Organic Matter	[-100,300]
Sand	[500,800]
Clay	[900,1200]

Figure 2.45: The range of X-ray absorbance of plant pot using CT (Computed Tomography) scanner.



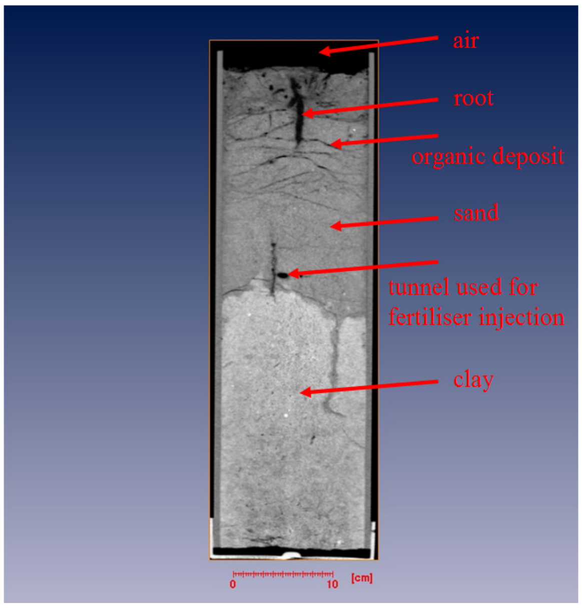
(a) (b)

Figure 2.46: Example of canola root in cylindrical pot.

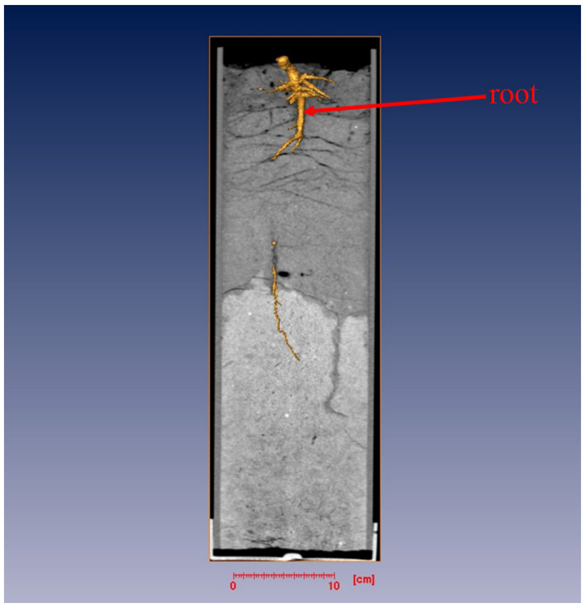


Pos: −5.317 3.278 141
Index: 94 327 1239
Material: Organic_matter
Voxel value: 145

Figure 2.47: The range of X-ray absorbance of cylindrical pot before planting canola plant using CT scanner. It was coded and viewed in *Amira* software.

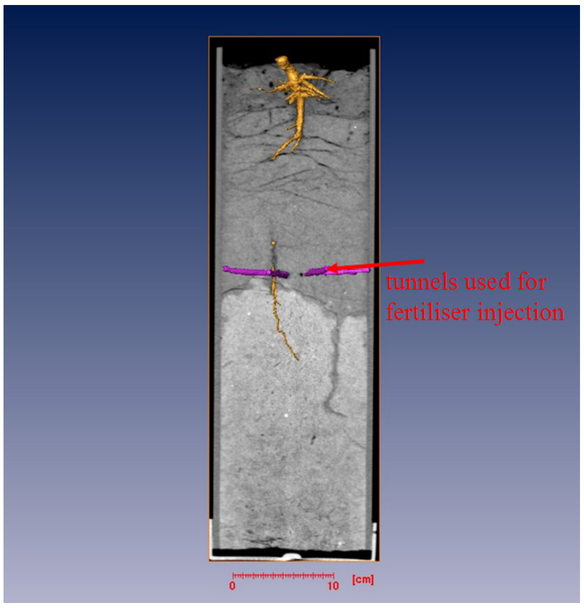


(a)

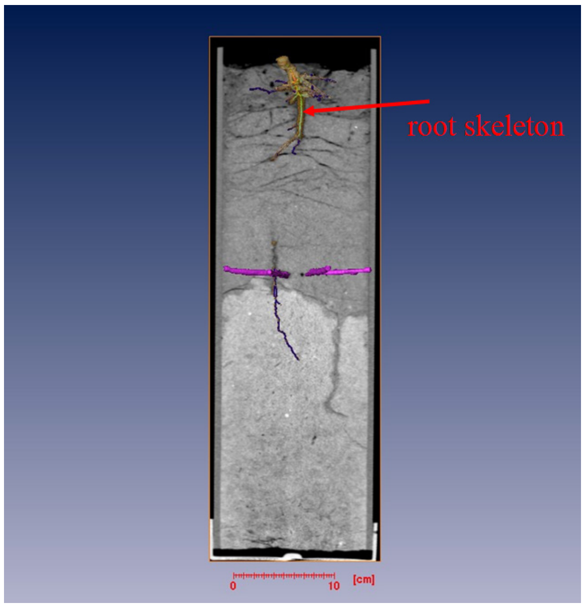


(b)

Figure 2.48: Example of canola pot with vertical cross-section, (a) they consist of air, root system, organic matter, sand, clay and tunnel for fertilizer injection, (b) the segmented canola root system was reconstructed [Kolesik, 2004] using Amira software.

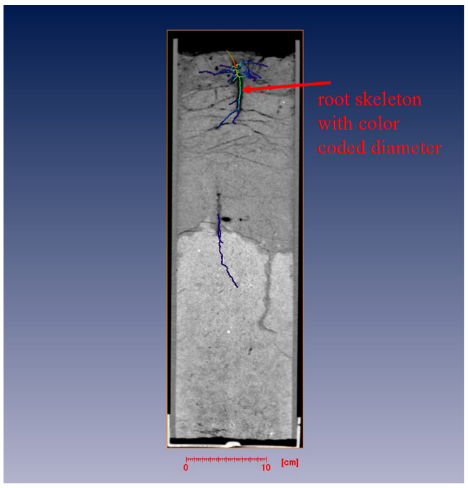


(a)

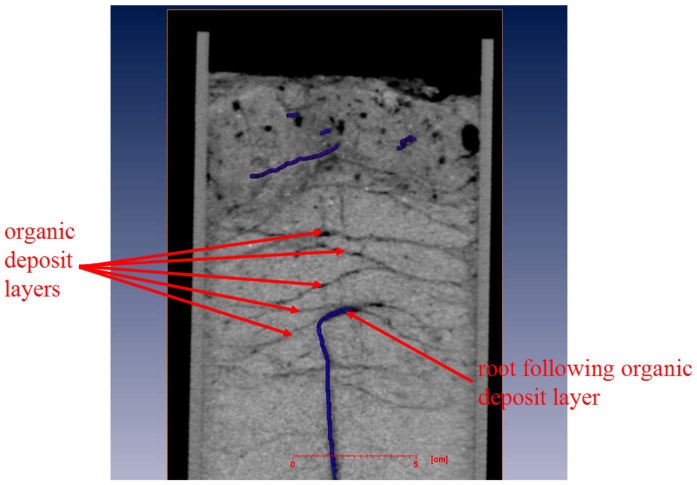


(b)

Figure 2.49: Example of canola pot with vertical cross-section, (a) the segmented canola root and tunnels used for fertilizer injection, (b) the segmented canola root system with root skeleton.

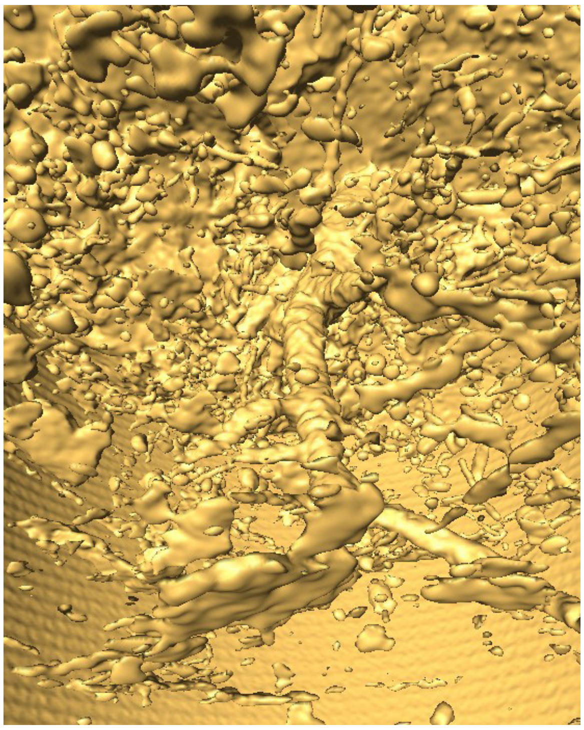


(a)

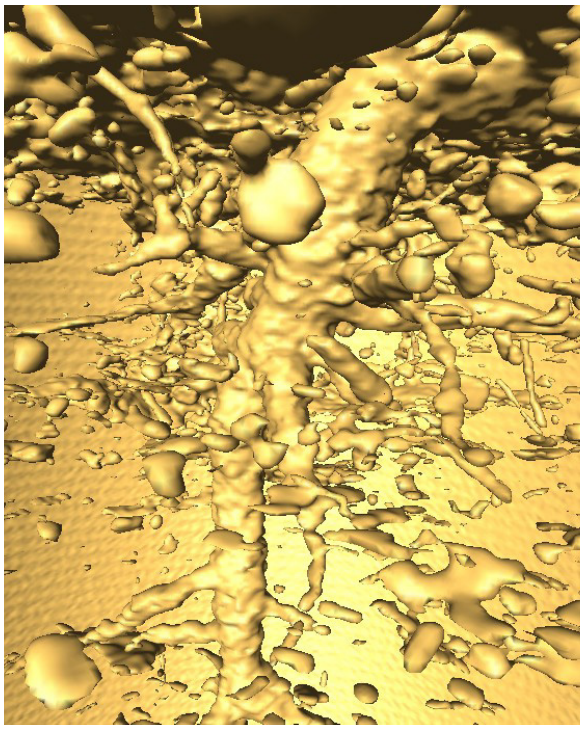


(b)

Figure 2.50: Example of canola pot with vertical cross-section, (a) the segmented canola root skeleton with color coded diameter, (b) the root path following organic deposit layer.



(a)



(b)

Figure 2.51: Example of canola pot, (a) the segmented canola root with organic deposit layer in the pot, (b) another view of segmented root in the pot.

Problem 2.8 *If there are some branches or network structures in a given input image or volume, how can we reconstruct the network structure?*

To solve the Problem 2.8, we have to segment the network structures from the given image or volume (see Section 2.7.3).

2.7.3 Flow Diagram for Reconstruction of the Branching Structures

This section proposes the method for reconstructing the branching structure from an input image and 3D volume data. There are seven consecutive steps given in the following algorithm:

Algorithm 2.2 ***ReconProc***

1. Read all sliced input images with the width W, height H, and Length L ($L = 1$ for 2D, $L > 1$ for 3D),
2. Do preprocessing the input image or volume by smoothing, de-noising, edge/surface preservation by anisotropic diffusion filtering,
3. Proceed with the region (2D) or volume (3D) growing method by given pixels $p_i(x, y)$ or voxels $p_i(x, y, z)$,
4. Do thinning and skeletonizing method,
5. Generate the network of branching structure and resolution reduction.
6. Generate L-system production rules or L-string.
7. Print L-system production rules or L-string to the output file.

There are seven steps of each procedure in Figure 2.52. The flow diagram starts by reading the input image for 2D and sliced images for 3D. The preprocessing step is applied by using anisotropic diffusion filtering. Either region or volume growing method is started by the set of given initial starting points. After the growing process stopped, thinning and skeletonizing method are applied, respectively. The network is reconstructed and network resolution is reduced. The L-system production rules are generated and ready for further uses.

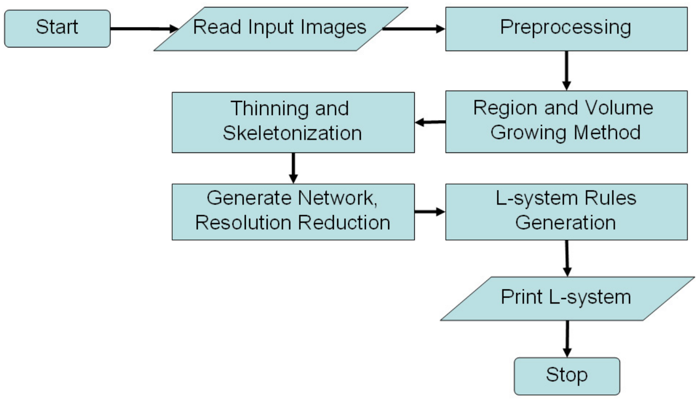

Start
Read Input Images
Preprocessing
Region and Volume Growing Method
Thinning and Skeletonization
Generate Network, Resolution Reduction
L-system Rules Generation
Print L-system
Stop

Figure 2.52: Flow diagram of reconstruction process of branching structure described in Algorithm 2.2 [Chuai-Aree et al., 2007].

The example input images and volume sliced images are shown in Figure 2.53.

2.7.4 Anisotropic Diffusion Filtering

Anisotropic diffusion filtering is the method for smoothing, edge preservation, and denoising process [Perona and Malik, 1990] and [Weickert, 1998]. In this section we use the technique of the Perona-Malik (PM) model in [Perona and Malik, 1990] to process the smoothing, edge preservation and de-noising before doing region growing method, since this method can be grown very easily if the input image was smoothed by anisotropic diffusion process.

The nonlinear diffusion filtering of PM-model is based on the equation

$$\partial_t u = div(g(|\nabla u|^2)\nabla u), \tag{2.46}$$

and it uses diffusivity such as

$$g(s^2) = \frac{1}{1 + s^2/\lambda_u^2}. \tag{2.47}$$

The λ_u is conduction coefficient, where $20 \leq \lambda \leq 100$, and $s = |\nabla u|$ is absolute gradient of gray intensity u.

One advantage of nonlinear diffusion filtering is the combination of disconnected lines. The example of disconnected lines of branching structure is shown in Figure 2.54. The result after diffusion filtering is useful for region growing method. The equation (2.46) is solved by using finite difference method.

2.7.5 Region and Volume Growing Methods

This method is called “Seeded Region Growing” and it was introduced by Rolf Adams and Leanne Bischof [Adams and Bischof, 1994]. They presented a new method for segmen-

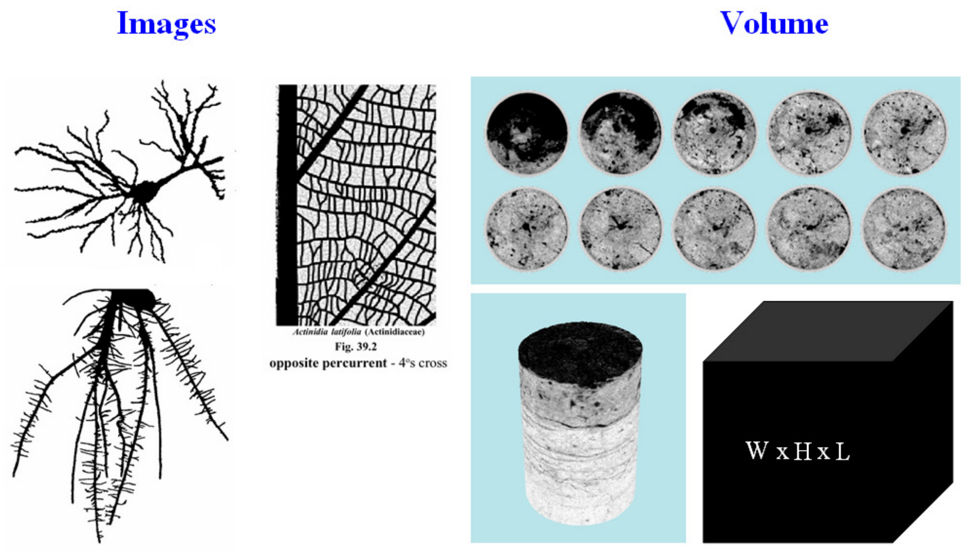

Images
Volume
Actinidia latifolia (Actinidiaceae)
Fig. 39.2
opposite percurrent - 4°s cross
W x H x L

Figure 2.53: Example of input images and volume sliced images of volume data [Chuai-Aree et al., 2007], neuron and rice root images from [Strzodka and Telea, 2004], leaf network from [Ash et al., 1999b], and actual soil volume data from P. Kolesik [Kolesik, 2004]

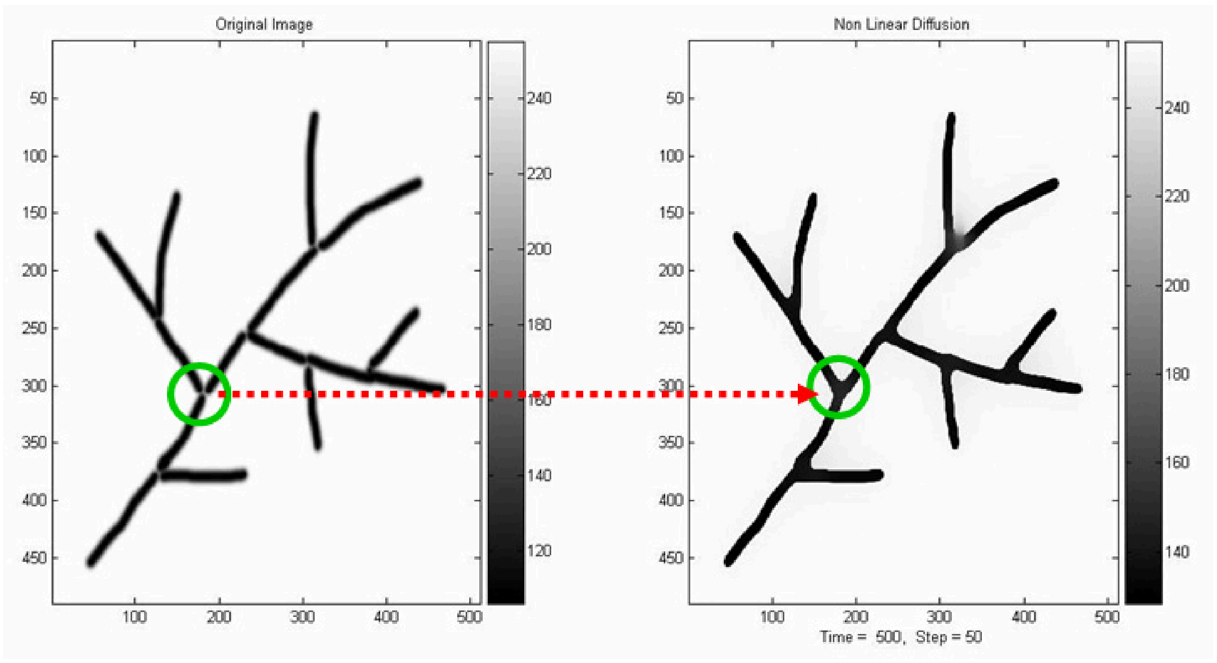


Figure 2.54: Example of nonlinear diffusion filtering for combination of disconnected lines [Chuai-Aree et al., 2007].

tation of intensity images, which is robust, rapid and free of tuning parameters. These characteristics allow implementation of one very good algorithm which could be applied to a large variety of images. This method, however, requires selection of seed regions, what has to be done manually and it classifies this approach to the class of semiautomated algorithms. The algorithm grows these seed regions until all image pixels have been processed. The region growing method is kind of a Level-Set Method [Sethian, 1999].

The input images of this thesis are assumed to contain the branch structures. The system is started by giving the initial starting pixels on the branching structure. The initial pixels will be growing around the given points and diffuse under the condition of gray intensity of each pixel. If the considered pixel satisfies the condition in branching set, that pixel will be taken into the set of branching structure. The development of growing process is computed by the finite difference method. In 3D case, the pixel is considered as the voxel.

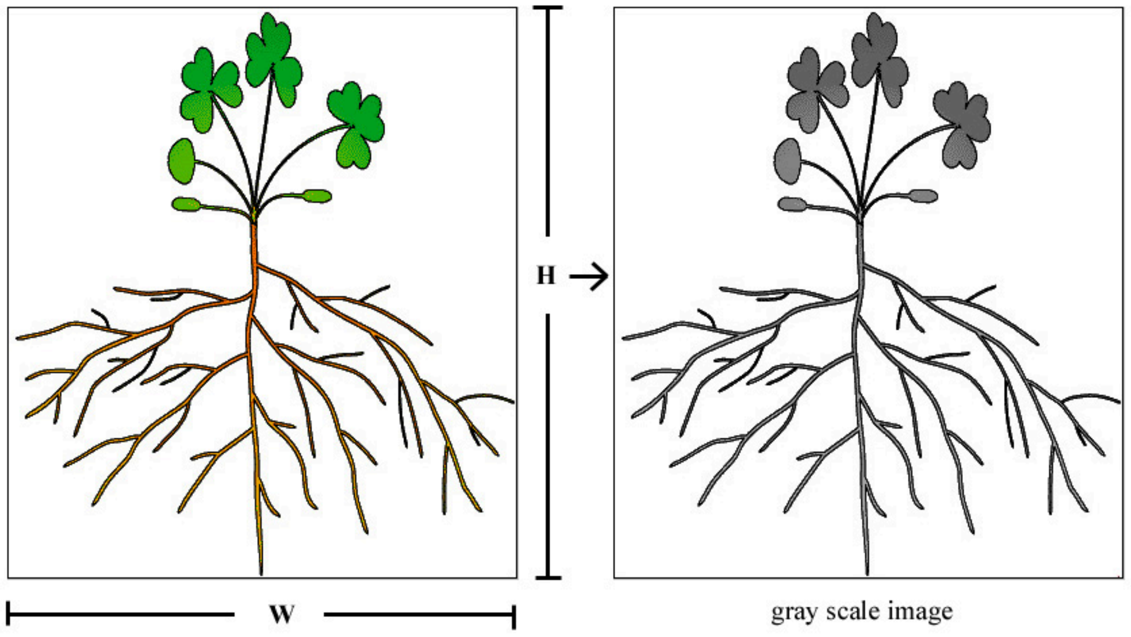


Figure 2.55: Example of input image, the color image (left) converted to the gray-scale image (right) [Chuai-Aree et al., 2007].

The pixel $p_{i,j}$ consists of four elements namely red ($RR_{i,j}$), green ($GG_{i,j}$), blue ($BB_{i,j}$) for color image and gray ($YY_{i,j}$). The color input image is converted to gray-scale image as Figure 2.55 by using the following equation (2.48).

$$\begin{aligned} YY_{i,j} &= Round(0.299 * RR_{i,j} + 0.587 * GG_{i,j} + 0.114 * BB_{i,j}); \\ &\quad \forall YY_{i,j}, \forall RR_{i,j}, \forall GG_{i,j}, \forall BB_{i,j} \in \{0, 1, 2, ..., 255\} \end{aligned} \tag{2.48}$$

Where $YY_{i,j}$, $RR_{i,j}$, $GG_{i,j}$, and $BB_{i,j}$ are gray intensity of gray image, red, green, and blue channel of color image, respectively. $YY_{i,j}$ is the integer value from 0 (black) to 255 (white) value. The *Round* function returns the integer value for $YY_{i,j}$ intensity. The color intensities $RR_{i,j}$, $GG_{i,j}$, and $BB_{i,j}$ are also integer value from color image. The gray intensity bar is shown in Figure 2.56.

We define I, B, G as a set of all pixels in input gray-scale image for 2D or set of all voxels in input gray-scale volume image for 3D, set of pixels which is branching structure,

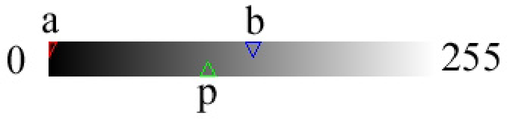


Figure 2.56: The gray-scale intensity bar from zero intensity (black) to 255 intensity (white) with the range $[a, b]$ and current point p.

and set of given initial pixels, respectively. So then we have $G \subseteq B$ and $B \subseteq I$. The description of each set is given below.

$$\begin{aligned}
I &= \begin{cases} \{p_{i,j} \mid (0 \le p_{i,j} \le 255) \wedge (1 \le i \le W) \wedge \\ \qquad (1 \le j \le H)\} & for \quad 2D \\ \{p_{i,j,k} \mid (0 \le p_{i,j,k} \le 255) \wedge (1 \le i \le W) \wedge \\ \qquad (1 \le j \le H) \wedge (1 \le k \le L)\} & for \quad 3D \end{cases} \\
B &= \{p \in I \mid (a \le Int(p) \le b) \wedge (0 \le a \le 255) \wedge (0 \le b \le 255) \wedge (a \le b)\} \\
G &= \{\exists p \mid p \in B\}
\end{aligned} \tag{2.49}$$

Where W, H, L, a and b are the width, height, depth of the image, the minimum and maximum intensity value of branching structure set, respectively. The value of a, b, and p are illustrated in the Figure 2.56. The function *Int(p)* is the intensity of pixel p.

Figure 2.57 shows the setup of an input image and shows the eight directions of region growing method around the initial given pixel. The step by step of region growing method is represented in Figure 2.58.

Region growing algorithm for 2D image

This section describes a step by step of the region growing method. From the given input image to the target branch region, there are eight consecutive steps as follows.

Algorithm 2.3 ***RGM2D***

1. Load all pixels $P_{i,j}$ from input image, convert to gray intensity value and store in an array M as original gray-scale array. The size of array M is $H\times W$.
2. Initialize zero array M^{Old} and M^{New} for calculating the growing method. The size of array M^{Old} and M^{New} is $W\times\ H$.
3. Define the set of given initial pixels G by user and set the value of the initial points in array M^{Old} to the medium intensity, e.g. 128.
4. Define the condition of a and b in equation (2.49) for region growing method by user. The considering interval of branching structure will be in [InitialValue - a, InitialValue + b]. The InitialValue is gray intensity of the last given pixel.

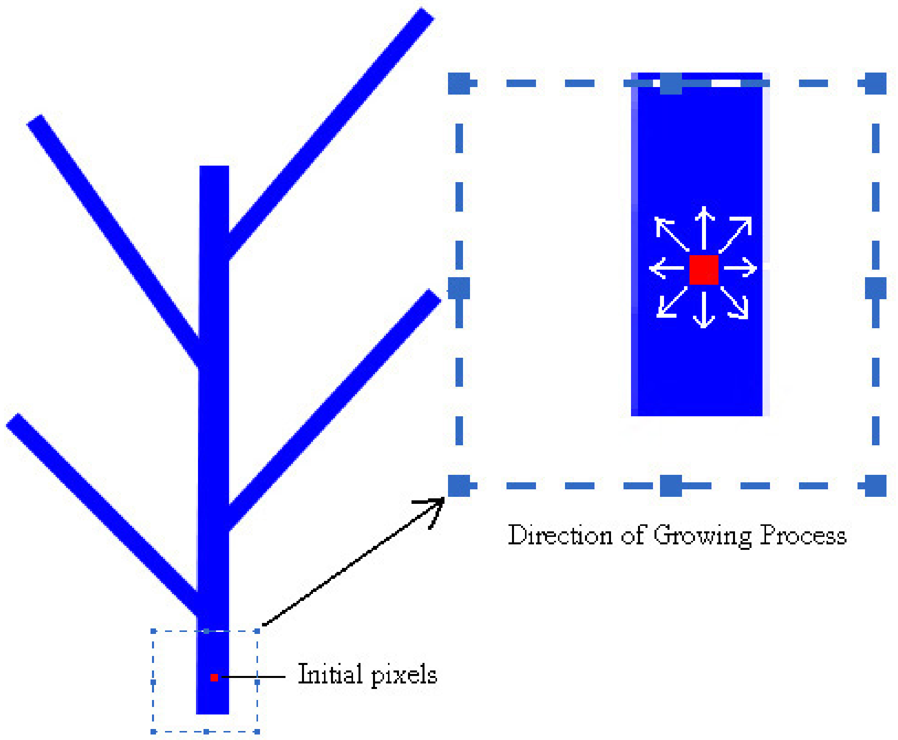


Figure 2.57: Setup of the input image and given initial pixel.

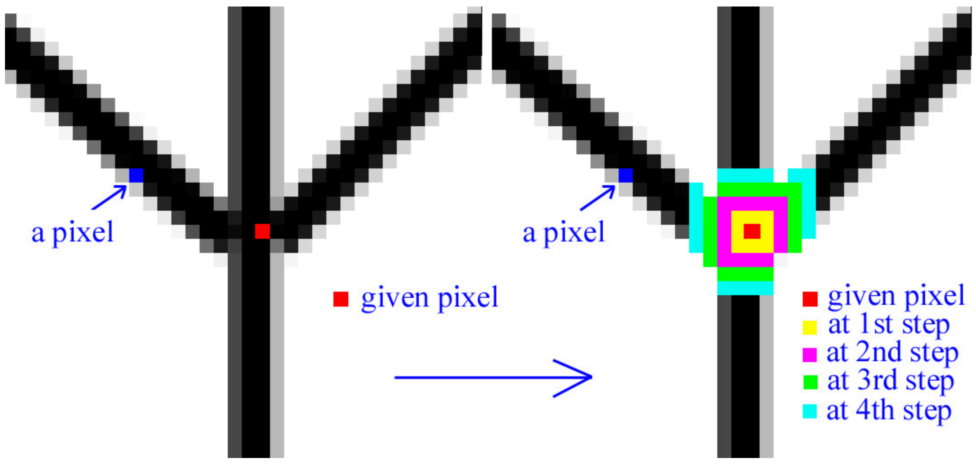


Figure 2.58: Growing method of each step.

5. Start the region growing method.

6. Set *Oldpoint* = *0*, *Newpoint* = *1*, *Iteration*= *0*

```
While ($Oldpoint \neq Newpoint$) do {
Set $Oldpoint$=$Newpoint$ , and $Iteration = Iteration + 1$
For all elements in $M$ do {
If ($M_{i,j} \geq M_{Gx,Gy} - a$) and ($M_{i,j} \leq M_{Gx,Gy} + b$) Then {
```

- Region growing of each pixel $p_{i,j}$ in G by considering the 8-neighboring pixels, we use the following equation to proceed the growing method.

$$\begin{aligned} M_{i,j}^{New} = {} & M_{i,j}^{Old} + Round(0.99 * (M_{i-1,j}^{Old} + M_{i+1,j}^{Old} \\ & + M_{i,j-1}^{Old} + M_{i,j+1}^{Old} + M_{i-1,j-1}^{Old} + M_{i+1,j-1}^{Old} \\ & + M_{i-1,j+1}^{Old} + M_{i+1,j+1}^{Old} - 8 * M_{i,j}^{Old})) \end{aligned} \tag{2.50}$$

- Take all new neighboring points in to G, which satisfy the condition in B. If $M_{i,j}^{New} \neq 0$, it means the pixel $p_{i,j}$ satisfies the condition.
- Count all growing points in G and set the value to *Newpoint*.

```
} // If
} // For
} // While
```

7. Update $B = G$ and set B is stored information of branching structure satisfying the condition in B.

8. The array M^{New} and B will be used for skeletonizing method.

After executing the Algorithm 2.3(*RGM2D*), we use a clover plant as a case study. Six initial pixels (red label) are given in Figure 2.59. The region growing method is running until reaching all clover pixels.

Volume growing algorithm for 3D volume data

Figure 2.60 shows an example of soil volume with canola root scanned by CT scanner. The given size of input volume is $512 \times 512 \times 1313$.

The volume growing method is given in the following steps in Algorithm 2.4. There are eight steps for processing the volume growing method in the 3D volume data.

Algorithm 2.4 *VGM3D*

1. Load all voxels $P_{i,j,k}$ from sliced input images, convert to gray intensity value and store to an array M as original gray-scale array. The size of array M is $W{\times}H{\times}L$.

2. Initial zero array M^{Old} and M^{New} for calculating the growing method. The size of array M^{Old} and M^{New} is $W{\times}H{\times}L$.

3. Define the set of given initial voxels G by user and set the value of the initial points in array M^{Old} to the medium intensity, e.g. 128.

4. Define the condition of a and b in equation (2.49) for volume growing method by user. The considered interval of branching structure will be in $[InitialValue - a, InitialValue + b]$. The $InitialValue$ is gray intensity of the last given voxel.

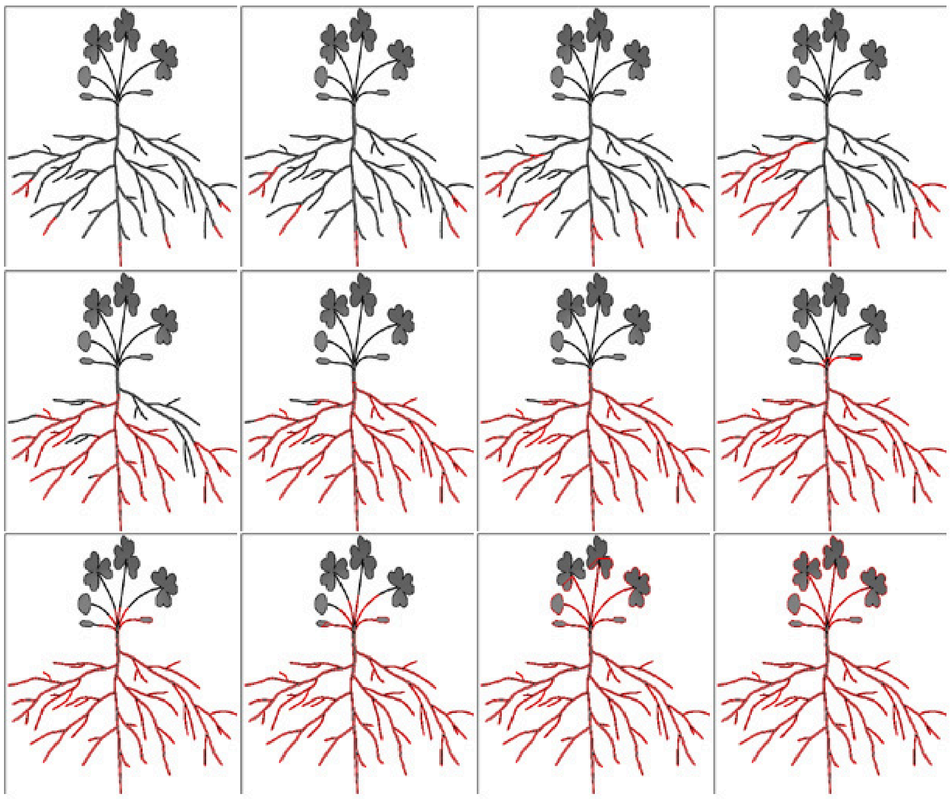


Figure 2.59: Region growing in branching structure of the clover plant.

5. Start the volume growing method.

6. Set *Oldpoint = 0, Newpoint = 1, Iteration = 0*
   While (*Oldpoint* $\neq$ *Newpoint*) do {
   Set *Oldpoint=Newpoint* , and $Iteration = Iteration + 1$
   For all elements in M do {
   If $(M_{i,j,k} \geq M_{Gx,Gy,Gz} - a)$ and $(M_{i,j,k} \leq M_{Gx,Gy,Gz} + b)$ Then {

   - Region growing of each voxel $p_{i,j,k}$ in G by considering the 26-neighboring voxels, we use the following equation (2.51) to proceed the growing method.

$$\begin{aligned} M^{New}_{i,j,k} \quad = \quad & M^{Old}_{i,j,k} + Round(0.99 * (M^{Old}_{i-1,j,k-1} + M^{Old}_{i+1,j,k-1} \\ & + M^{Old}_{i,j-1,k-1} + M^{Old}_{i,j+1,k-1} + M^{Old}_{i-1,j-1,k-1} \\ & + M^{Old}_{i+1,j-1,k-1} + M^{Old}_{i-1,j+1,k-1} + M^{Old}_{i+1,j+1,k-1} \\ & + M^{Old}_{i,j,k-1} + M^{Old}_{i-1,j,k} + M^{Old}_{i+1,j,k} + M^{Old}_{i,j-1,k} \\ & + M^{Old}_{i,j+1,k} + M^{Old}_{i-1,j-1,k} + M^{Old}_{i+1,j-1,k} + M^{Old}_{i-1,j+1,k} \\ & + M^{Old}_{i+1,j+1,k} + M^{Old}_{i-1,j,k} + M^{Old}_{i+1,j,k} + M^{Old}_{i,j-1,k} \\ & + M^{Old}_{i,j+1,k} + M^{Old}_{i-1,j-1,k} + M^{Old}_{i+1,j-1,k} + M^{Old}_{i-1,j+1,k} \\ & + M^{Old}_{i+1,j+1,k} + M^{Old}_{i+1,j+1,k+1} - 26 * M^{Old}_{i,j,k})) \end{aligned} \tag{2.51}$$

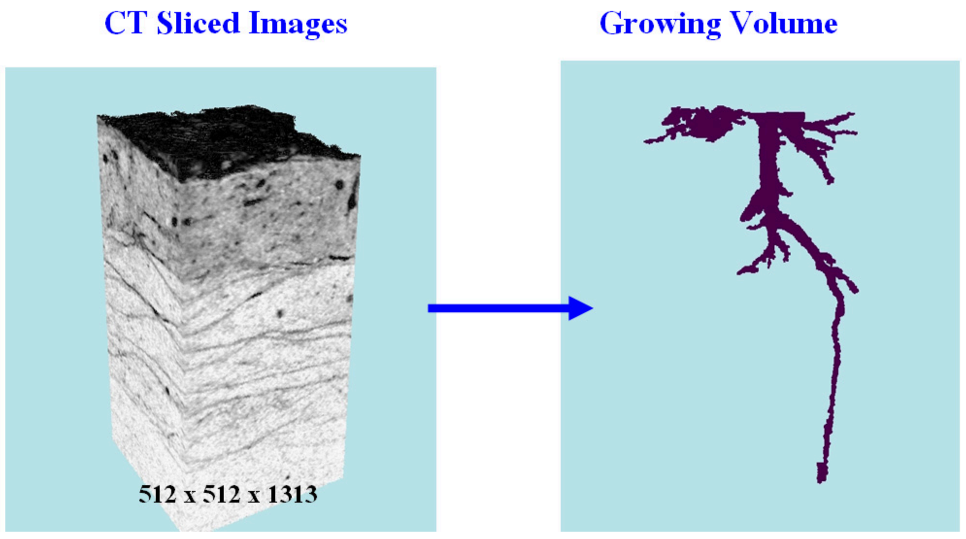


Figure 2.60: Example of CT volume data of plant root.

- Take all new neighboring voxels into G, which satisfy the condition in B. If $M^{New}_{i,j,k} \neq 0$, it means the voxel $p_{i,j,k}$ satisfies the condition.
- Count all growing voxels in G and set the value to *Newpoint*.

```
} // If
} // For
} // While
```

7. Update $B = G$ and set B is stored information of branching structure satisfying the condition in B.

8. The array M^{New} and B will be used for skeletonizing method.

Figure 2.61 shows the sequence of volume growing method of root data in several steps after applying Algorithm 2.4(*VGM3D*). The remark for a very fast computation of region and volume growing method is simply calculated by using only M^{Old} instead of M^{New} on the left hand side in equation (2.50) and (2.51). The region and volume growing methods can be accelerated for fast computation.

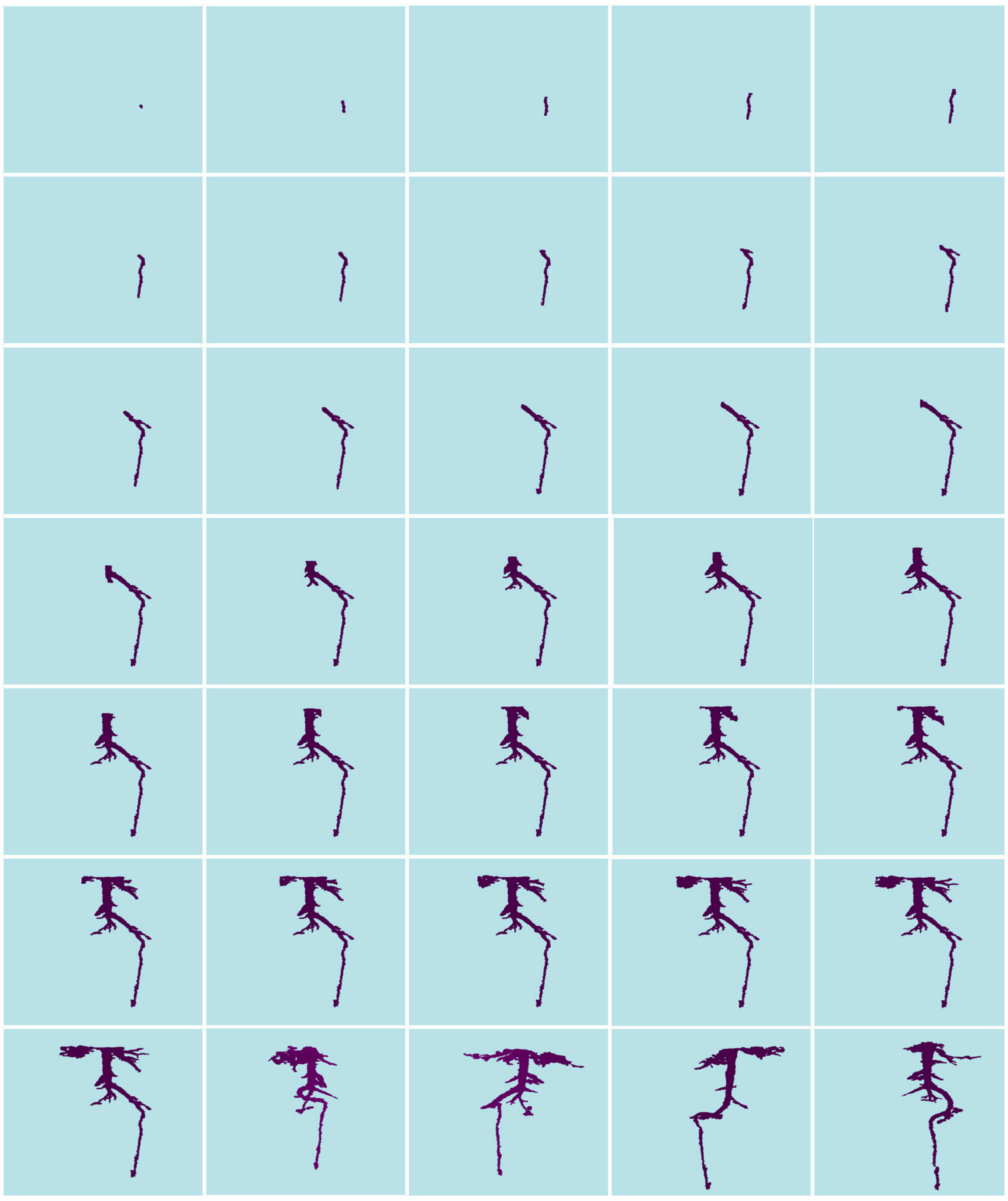


Figure 2.61: Development of volume growing method of root data in several steps after giving the initial root voxel in the pot data. The processing will stop when there is no increasing of the number of root voxels [Chuai-Aree et al., 2007].

2.7.6 Skeletonizing Methods

Problem 2.9 *How can we calculate the skeleton of reconstructed network?*

This section presents the method how to proceed the thinning process from B from the previous section. After the thinning process was done, the skeleton of branching structure can be found.

Skeletonization is the method for peeling off of a pattern as many pixels as possible without affecting the general shape of the pattern. It means after pixels have been peeled off, the pattern is still connected with one pixel line width. The skeleton hence obtained must have the following properties; as thin as possible, connected, centered. In this dissertation, we have improved the so-called Hilditch's Algorithm for skeletonization.

Hilditch's Algorithm and Improvement

There are two versions for Hilditch's algorithm [Azar and Toussaint, 1997], one using a 4x4 window and the another one using a 3x3 window. Here we are concerned with the 3x3 window version. Hilditch's algorithm consists of performing multiple passes on the pattern and on each pass.

Definition 2.10 ***(Hilditch's Window Setup)*** *A 3×3 window is arranged by the sequence (p2,p3), (p3,p4), (p4,p5), (p5,p6), (p6,p7), (p7,p8), (p8,p9), (p9,p2). The p1 is the center of the window. It is shown in Figure 2.62(a).*

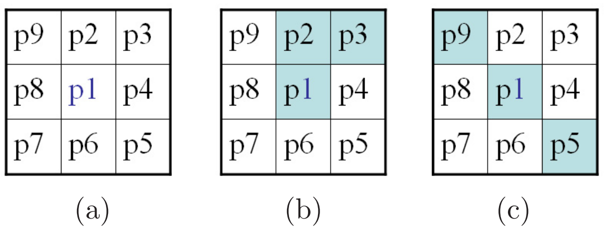
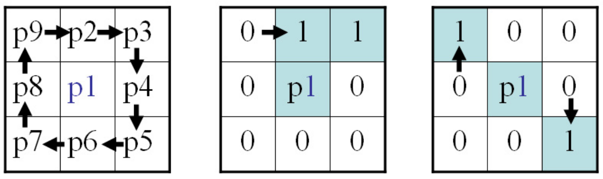


Figure 2.62: A setup of Hilditch's algorithm, (a) point labels from $p1$ to $p9$, (b) $F_a(p1) = 1$, $F_b(p1) = 2$, (c) $F_a(p1) = 2$, $F_b(p1) = 2$.

Consider the following 8-neighborhood of a pixel *p1* from Figure 2.62, we want to decide whether to peel off *p1* or keep it as part of the resulting skeleton. For this purpose we arrange the eight neighbors of *p1* in a clock-wise order and we define two functions: $F_a(p1)$ and $F_b(p1)$. The function $F_a(p1)$ counts the number of (0,1) pattern corresponding to pixel $p1$. It means the two neighboring pixels are arranged from white pixel (zero) to black pixel (one). The function $F_b(p1)$ counts only the number of non-zero neighbors of pixel $p1$. Each pixel p_i will be set to one if it is black pixel, otherwise it will be set to zero since we need the multiplication (*) in the third and fourth conditions of the algorithm.

Definition 2.11 *(**Hilditch's Functions**) The Hilditch's algorithm consists of two functions; $F_a(p1)$ and $F_b(p1)$. They are defined as follows:*

1. *Function $F_a(p1)$ returns the number of (0,1) patterns (or number of arrow in Figure 2.63) in the sequence* (p2,p3), (p3,p4), (p4,p5), (p5,p6), (p6,p7), (p7,p8), (p8,p9), (p9,p2), *and*
2. *Function $F_b(p1)$ gives a number of non-zero neighbors of pixel* p1.



(a) (b) (c)

Figure 2.63: Function $F_a(p1)$ and $F_b(p1)$ from Figure 2.62, (a) cycle of pattern from $p2$ to $p9$, (b) $F_a(p1) = 1$ (number of arrows represents (0,1) pattern), $F_b(p1) = 2$, (c) $F_a(p1) = 2$ (two arrows for two (0,1) patterns), $F_b(p1) = 2$.

From Figure 2.62(b) and 2.63(b), we have $F_a(p1) = 1$ since there is only one (0,1) pattern from pixel $(p9, p2)$ and $F_b(p1) = 2$ because of the non-zero neighbors of pixel $p2$ and $p3$. The same procedure is applied to Figure 2.62(c) and 2.63(c). It shows $F_a(p1) = 2$ and $F_b(p1) = 2$.

The Hilditch's algorithm checks all the pixels and decides to change a pixel from black to white (or change the value of 1 to 0) if it satisfies the following definition:

Definition 2.12 *(**Hilditch's Algorithm**) The pixel p1 is removed from the image I if it satisfies the following four conditions:*

1. $2 \leq F_b(p1) \leq 6$,
2. $F_a(p1) = 1$,
3. $p2 * p4 * p8 = 0$ *or* $F_a(p2) \neq 1$, *and*
4. $p2 * p4 * p6 = 0$ *or* $F_a(p4) \neq 1$, *stop when nothing changes (no more pixels can be removed).*

Before calculating the skeleton of an given object, the height field map has to be proceeded as the following algorithm.

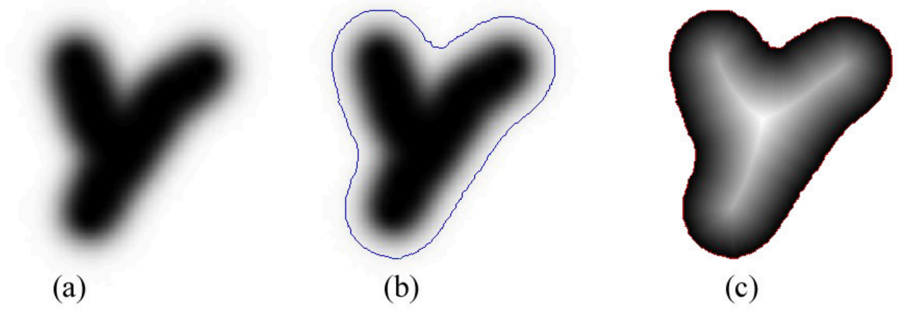


Figure 2.64: The height field map for the thickness of branching structure, (a) given object, (b) edge detection, (c) height field map (black is low, white is high distance).

Height field map algorithm for 2D images

This section explains the height field map algorithm for an input image. There are two steps shown in Algorithm 2.5(*HFM2D*):

Algorithm 2.5 *HFM2D*

```
1. Track the edge detection (in Figure 2.64)(b)) from array $M$ and put to the set of
   boundary points $E$ and other points to the set $O$ when $O = B - E$.

2. For all points $j$ in $O$ do {
      • initial $dmin_j = W + H$
      • For all points $i$ in $E$ do {
           – Calculate the distance $d_{i,j} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$
           – if $dmin_j > d_{i,j}$ then $dmin_j = d_{i,j}$
        }
   }
```

The input image I in gray-scale is given in Figure 2.64(a). The edge detection algorithm is applied in Figure 2.64(b). Finally, the result after processing this algorithm is shown in Figure 2.64(c).

Height field volume algorithm for 3D volume data

In case of 3D volume data, we expand the calculation in the previous section to volume data to label the value of distance from the boundary of the found object after region and volume growing method. There are two consecutive steps as follow:

Algorithm 2.6 ***HFV3D***

```
1. Track the edge/surface detection from array $M$ and put to the set of boundary
   voxels $E$ and other voxels to the set $O$ when $O = B - E$.
2. For all voxels $j$ in $O$ do {
     • initial $dmin_j = W + H + L$
     • For all voxels $i$ in $E$ do {
         – Calculate the distance $d_{i,j} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2}$
         – if $dmin_j > d_{i,j}$ then $dmin_j = d_{i,j}$
       }
   }
```

Skeletonizing algorithm for 2D images

In 2D input image, we apply the skeletonizing method after using region growing method. This section shows six consecutive steps in Algorithm 2.7(*Skel2D*) for calculating the skeleton from the given image.

Algorithm 2.7 ***SKel2D***

```
1. Take the array $M^{New}$ from the region growing method.
2. For all elements in $M^{New}$ {
     • Set the value of each element in array $M$ to 1 if $M^{New}_{i,j} \neq 0$.
   }
3. Calculate height field map (see Figure 2.64 for 2D).
4. Start the thinning process
     • For all elements in $M^{New}$ do {
         – If $M^{New}_{(i,j)} \neq 0$ then $M^{Old}_{i,j} = 1$ else $M^{Old}_{i,j} = 0$
       }
     • Set $Oldpoint = 0$, $Newpoint = 1$, and $Iteration = 0$
     • While ($Oldpoint \neq Newpoint$) do {
         – $Oldpoint = Newpoint$, $Iteration = Iteration + 1$
         – Set zero array $C$ with size $H \times W$.
```

```
    – For all elements in $M^{New}$ {
        ∗ Compute the Hilditch's algorithm
        ∗ Count $Newpoint$
      } // For
  } // While.
```

5. Remove some jagged points along the skeleton of branching structure and update B. Array M^{New} and set B are stored the information of branching structure.
6. Record the thickness information of the branching structure from calculating height field map.

After having the result from Algorithm 2.3 in Figure 2.59, then we apply the Algorithm 2.7. We are now having the result from skeletonizing method in Figure 2.65.

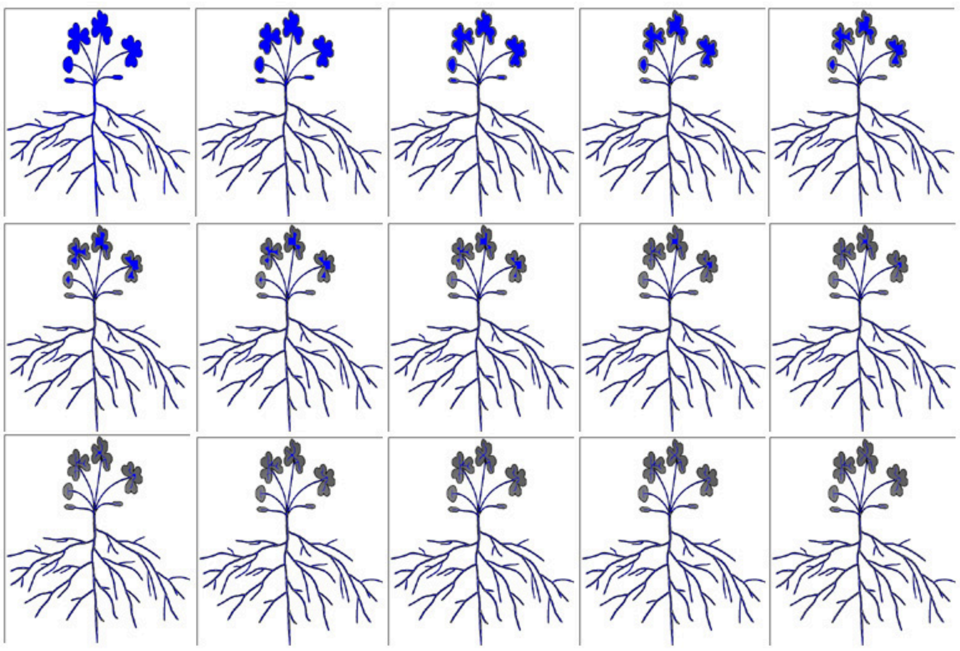


Figure 2.65: Skeletonization in branching structure of the clover plant.

Skeletonizing algorithm for 3D volume data

In 3D volume data, we calculate skeleton by applying the peeling off process from outside and label the number of each layer. The *Skel3D* is presented in the Algorithm 2.8.

Algorithm 2.8 ***SKel3D***

```
1. Take the array $M^{New}$ from the volume growing method.

2. For all elements in $M^{New}$ do
   - Set the value of each element in array $M$ to 1 if $M^{New}_{i,j,k} \neq 0$ for 3D.

3. Calculate depth field value (DFV) of each voxel by peeling off process.
   For all elements in $M^{New}$ do {
   - Set the depth field value at position $DFV_{i,j,k} = 1$
   - Count $Np$ if depth field value $DFV_{i,j,k} = 1$
   }

   $Iteration = 1$

   While ($Np > 0$) do {
      • $Iteration = Iteration + 1$
      • For all elements in $M^{New}$ {
        - Count the number of its 26-neighbors ($Cp$) if $DFV_{i,j,k} = 1$,
        - If $Cp < 20$ then set the depth field value of current voxel $DFV_{i,j,k} = Iteration$,
        } // For

      • Count $Np$ if depth field value $DFV_{i,j,k} = 1$
   } // While.

4. Construct the skeleton from the maximum $DFV_{i,j,k}$ to the smaller value by replacing
   the sphere and removing the smaller spheres inside the bigger sphere and connect
   them to its neighbors,

5. Remove some jagged points along the skeleton of branching structure and updating
   $B$. Array $M^{New}$ and set $B$ are stored the information of branch structure,

6. Record the thickness information of the branch structure from calculating depth
   field value.
```

2.7.7 Construction of the Branching Structures

Problem 2.10 *How can we construct the branching structure of reconstructed network?*

After skeletonizing method, the skeleton of branching structure is stored in array M^{new} and set B. Since all the skeleton points are not connected to each other by the line connection yet, this section is described the algorithm to generate the network and remove some unnecessary points, which are on the same straight line.

Algorithm for constructing the branching network

This section shows the algorithm to construct the branching network after using the skeletonizing method. The seven consecutive steps are shown as follows.

Algorithm 2.9 ***CBN***

```
1. Initialize a stack $S$ and start with a user supplied point for generating the network,
2. Calculate the nearest given starting point $R$ in array $M^{New}$ and push point $R$ into
   Stack $S$,
3. Set point $R$ as a root node of the branching structure $T$,
4. Set $M^{old} = M^{New}$ for marking the path that was discovered,
5. While (Stack $S$ is not empty) do {
     • Pop a point $P_{i,j}$ from the top of stack $S$ and set the value of $M^{Old}_{i,j} = 0$,
     • Look for all neighboring points of $P_{i,j}$ with a value is equal to 1, and push them
       to stack $S$ (If there are no neighboring point of $P_{i,j}$, the point $P_{i,j}$ will be set
       as a terminated point.),
     • Set all neighboring points of $P_{i,j}$ as children of $P_{i,j}$ and mark the value of that
       point in array $M^{Old}$ to be 0,
   } // While
6. Finally, the network of branching structure $T$ is constructed.
```

The next step, resolution reduction is needed to remove some unnecessary points.

2.7.8 Resolution Reduction

Problem 2.11 *How can we reduce the resolution of the reconstructed network?*

Up to now the network of branching structure T has been reconstructed, but it is still having a high resolution. In this section, we propose the algorithm to reduce the number of point in the network in Algorithm 2.10 (see Figure 2.66). The L-string construction is given in Algorithm 2.10.

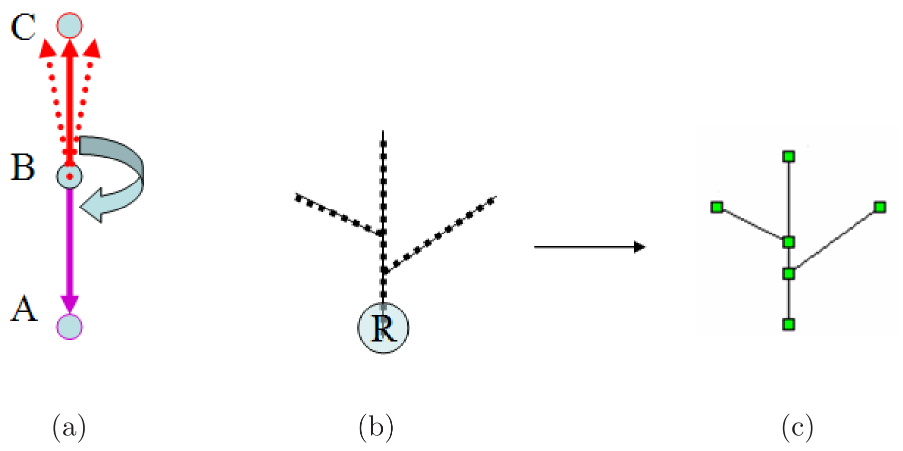


Figure 2.66: Resolution reduction process, (a) structure of considered point B for removing, (b) network before reduction, and (c) network after reduction.

Resolution reduction algorithm

This section explains the resolution reduction after executing the Algorithm 2.9(CBN). There are five steps in the following algorithm.

Algorithm 2.10 *ResRed*

```
1. Consider the network of branching structure $T$,
2. Start the process at the root node $R$ of branching structure $T$,
3. For all nodes in $T$ do {
     • For every 3 nodes, and each node has only one child, $A$ is the parent of $B$ and
       $B$ is the parent of $C$,
     • If node $A$, $B$, and $C$ are on the same line, then calculate the angle between
       $\overrightarrow{BA}$ and $\overrightarrow{BC}$,
     • If the angle between $\overrightarrow{BA}$ and $\overrightarrow{BC} \leq (180 \pm \delta)$, then remove node $B$ from $T$,
       where $\delta$ is the resolution angle for removal,
   } // For
4. Finally, the network of branching structure $T$ is regenerated.
```

2.7.9 Algorithm for L-string Construction

Problem 2.12 *How can we construct the L-string description from the reconstructed network?*

This section is the final section for constructing the L-string from the branch structure. The script of L-string can be constructed with six consecutive steps as follows.

Algorithm 2.11 ***LCon***

```
1. Read the network of branching structure $T$ after reducing its resolution,
2. Start the process at the root node $T$ of branching structure $T$,
3. Let $A$ be the root node and $B$ be the first child of root node $A$,
4. Calculate the angle $\delta_R$ between unit vector $\overrightarrow{j}$ and $\overrightarrow{AB}$,
5. For all nodes in $T$ which have parent node and at least one child do {
    • Let $B$ be the current node and $A$ be the parent of node $B$,
    • Calculate the vector $\overrightarrow{AB}$ and its length $D_{AB}$,
    • If node $B$ has more than one child then
        – Rearrange all children of node $B$ in the order of left branches "+($\delta$)I",
          right branches "-($\delta$)I", and middle branches "I" (has no angle) for L-string
          preparation and exchange all children of node $B$ in the same order left,
          right and middle, respectively,
    • For all children of node $B$ do {
        – Let $C$ be the current child of node $B$,
        – Calculate the vector $\overrightarrow{BC}$ and its length $D_{BC}$,
        – Calculate the angle $\delta_B$ between $\overrightarrow{AB}$ and $\overrightarrow{BC}$,
        – If the angle $\delta_B > \epsilon$ and node $B$ has more than one child then print "[" for
          adding a new branch,
        – Calculate the angle $\delta_N$ between unit perpendicular vector $\overrightarrow{n}$ of $\overrightarrow{AB}$, where
          $\overrightarrow{n}$ is rotated 90 degrees clockwise,
        – If $\delta_N > 90 + \epsilon$ degrees then print "+($\delta_B$)" else if $\delta_N < 90 - \epsilon$ print "-($\delta_B$)",
          where $\epsilon$ is a small angle,
        – Print the segment of $\overrightarrow{BC}$ in the form of "I($D_{BC}$)",
        – If node $C$ has no child and the angle $\delta_B > \epsilon$ then print "]" for ending its
          branch
    } // For
```

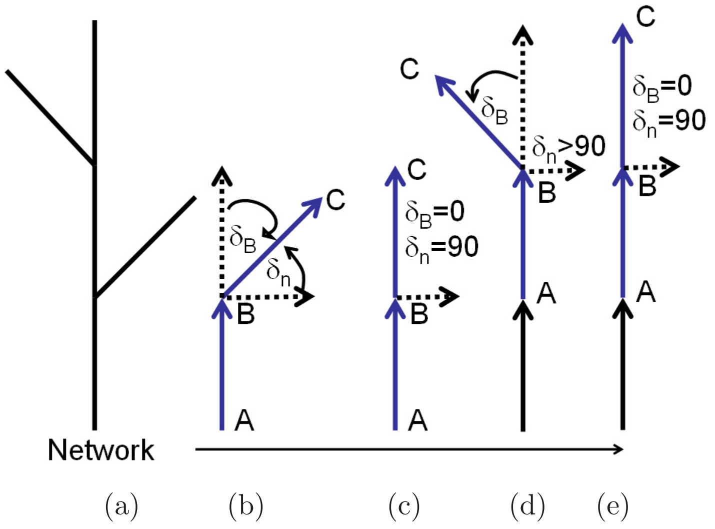


Figure 2.67: L-string construction: (a) input network from Algorithm 2.10, (b), (c), (d) and (e) step by step of L-string construction of input network (a)

```
} // For
```

6. Finally, the L-string code of the network T is generated.

Figure 2.67 shows the L-string reconstruction from input network from Algorithm 2.10 by applying the Algorithm 2.11. The L-string of the input network after applying the algorithm 2.11 is

"$I(181.0)[-(45.5)I(185.2)]I(179.0)[+(43.68)I(162.6)]I(188)$".

The given L-string starts with an internode with 181.0 pixel unit length, then draws a new branch rotated 45.5 degrees clockwise with 185.2 pixel unit length and closes its branch, then continues a new main stem with 179.0 pixel unit length, then draws a new branch rotated 43.68 degrees counter-clockwise with 162.6 pixel unit length, and finally draws a new main stem with 188 pixel unit length.

2.7.10 Experiments and Results

In this section we show the result of some examples. Since Figure 2.59 illustrates the development of region growing of every 10 steps from left to right and top to bottom. The red color from different starting points is growing until all branching structures are discovered. The skeletonization of the clover plant is shown in Figure 2.65. We are now summarizing the result of the clover plant in Figure 2.68, which contains region growing method (see Figure 2.59) in the first and second row and skeletonizing method (see Figure 2.65) in the third row.

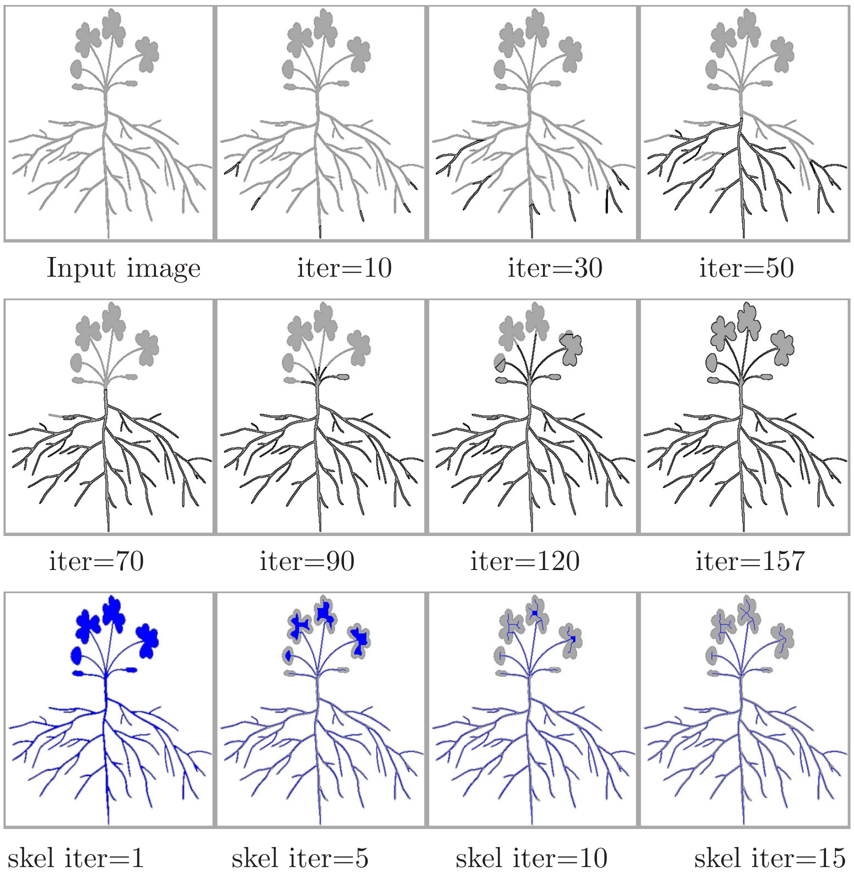


Figure 2.68: Region growing method and skeletonization in the branching structure of the clover plant.

The different resolution structures of the clover plant are shown in Figure 2.69. The user define δ value will reduce the structure resolution of the network.

The reconstruction procedure from an input image, region growing method, skeletonization, and 3D view after reconstruction is shown in Figure 2.70: (a) an input image, (b) result while applying region growing method, red pixels are labelled as a set of branching structure points, (c) result after applying skeletonizing algorithm, and (d) 3D view after reconstructing the branching network, respectively.

Figure 2.71 and 2.72 represent the branching structure and its reconstruction with

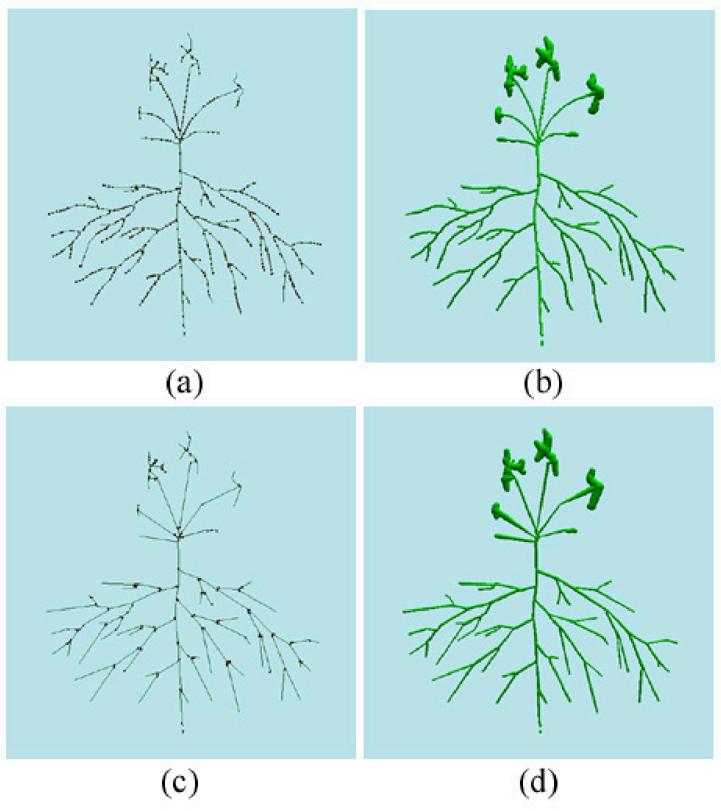


Figure 2.69: Some different resolution structures of the clover plant with the different δ values, (a) and (c) show the wire frame structure, (b) and (d) show 3D structure.

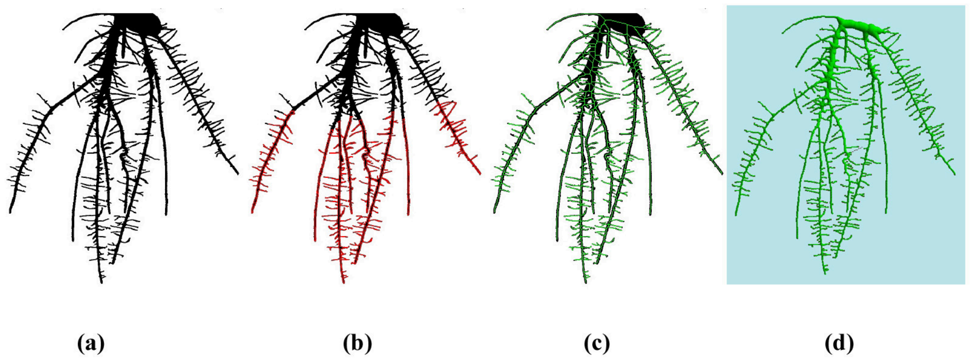


Figure 2.70: (a) rice root input image, (b) result while applying the region growing method (red pixels), (c) result after applying skeletonizing algorithm (green centerline), (d) result in 3D view after reconstruction.

different resolutions in 2D and 3D view, respectively.

Some Tree-like structures and their L-string codes are given in Figure 2.73 by applying the Algorithm 2.11. Each structure shows the number labels of the pixel node number.

Figure 2.74 shows some results of reconstructed canola roots reconstructed by P. Kolesik [Kolesik, 2004] in .MV3D file and a neuron structure of rat done by P. J. Broser

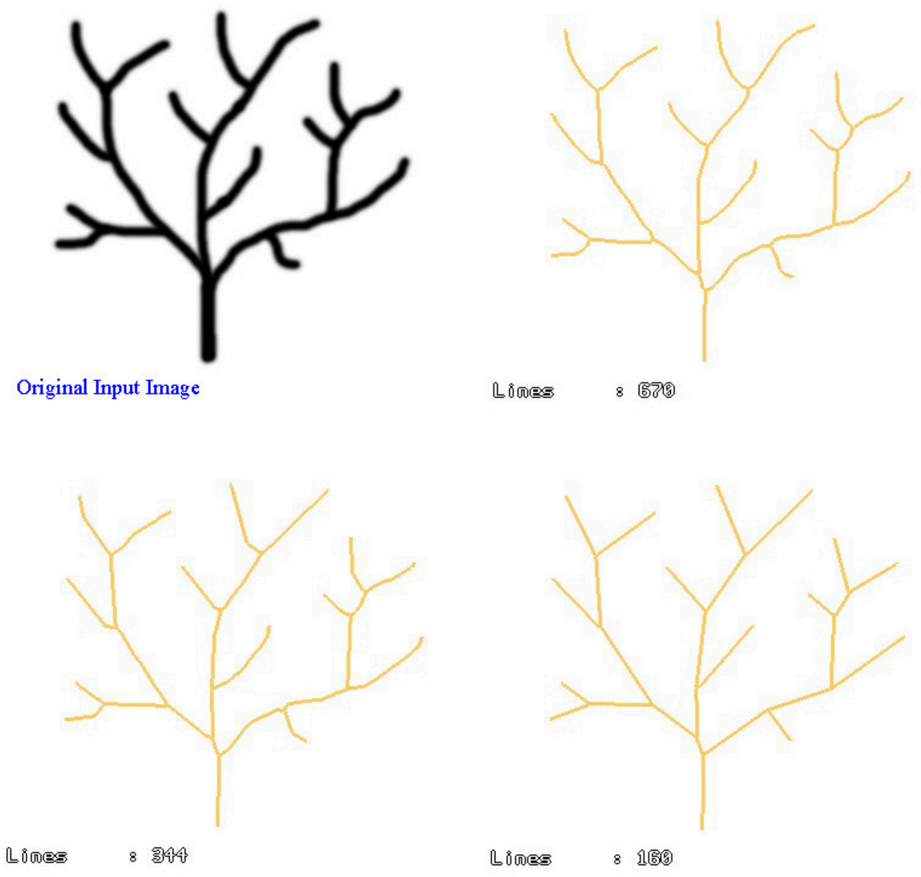


Figure 2.71: Branching structure and its reconstruction with different resolutions.

[Broser, 2006] saved in .HOC file format. They can be convert to L-string file format (.NLS).

In the Figure 2.76, we get the L-string from the algorithm as follows:

$$I(77.00)[+(23.12)I(18.38)[+(32.20)I(55.21)[+(34.47)I(61.00)$$
$$[+(22.74)I(43.08)][-(34.97)I(35.80)]][-(21.58)I(91.80)[+(12.00)$$
$$I(69.29)][-(28.96)I(71.17)[+(22.53)I(67.08)][-(58.40)I(73.81)]]]]$$
$$[-(23.57)I(48.01)[-(39.99)I(85.04)][-(4.662)I(78.40)[+(48.06)$$
$$I(58.05)][-(29.68)I(68.81)[+(58.33)I(74.84)][-(11.22)I(91.96)]]]]]$$
$$[-(54.56)I(79.05)[-(89.81)I(38.83)][-(16.81)I(65.14)[+(17.21)$$
$$I(90.44)][+(68.98)I(73.10)[+(55.56)I(32.80)][-(29.33)I(26.07)$$
$$[+(47.46)I(57.97)][-(26.05)I(57.45)]]]]]$$

Figure 2.77 shows the vascular aneurysm reconstruction with the different conditions of consideration during doing region growing method. Since the input image does not sharp enough, the smoothing process and anisotropic diffusion filtering can be applied for a preprocessing step before applying the region or volume growing method.

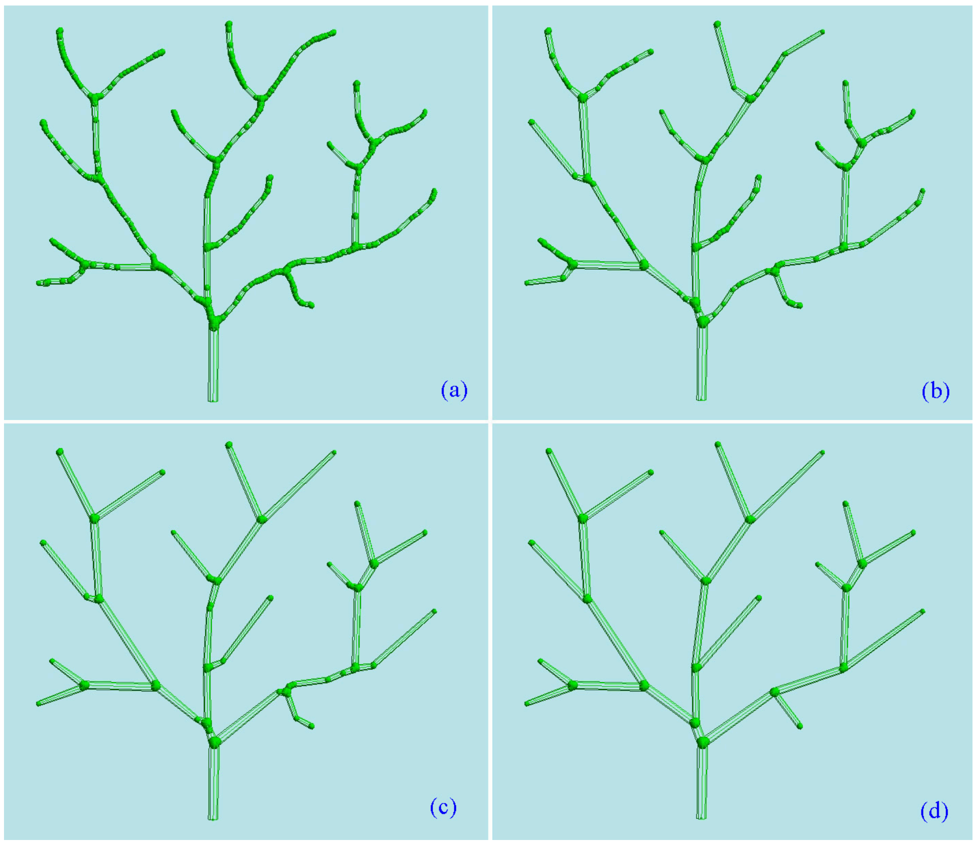


Figure 2.72: The different resolutions after reconstruction in 3D view.

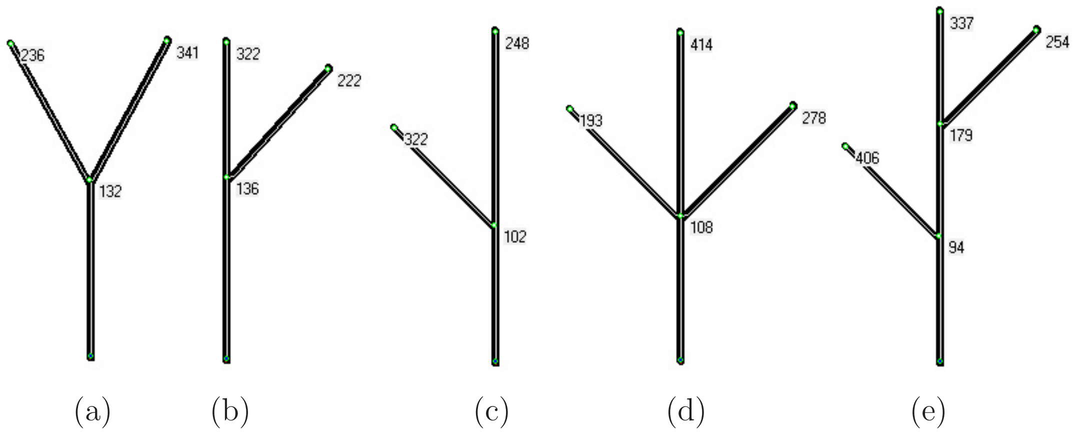


Figure 2.73: Some tree-like structures constructed in L-string codes: (a) I(132.0) [+(29.61) I(117.8)] [-(29.58) I(119.0)], (b) I(136) [-(43.1) I(111.0)] I(101), (c) I(102.0) [+(45.21) I(104.6)] I(145.0), (d) I(108)[+(46.05) I(115.2)] [-(45.6) I(117.3)] I(137), (e) I(94.00) [+(45.64) I(96.89)] [-(1.29) I(84.00)] [-(45.1) I(100.4)] I(84), the number labels in each network are the pixel node numbers.

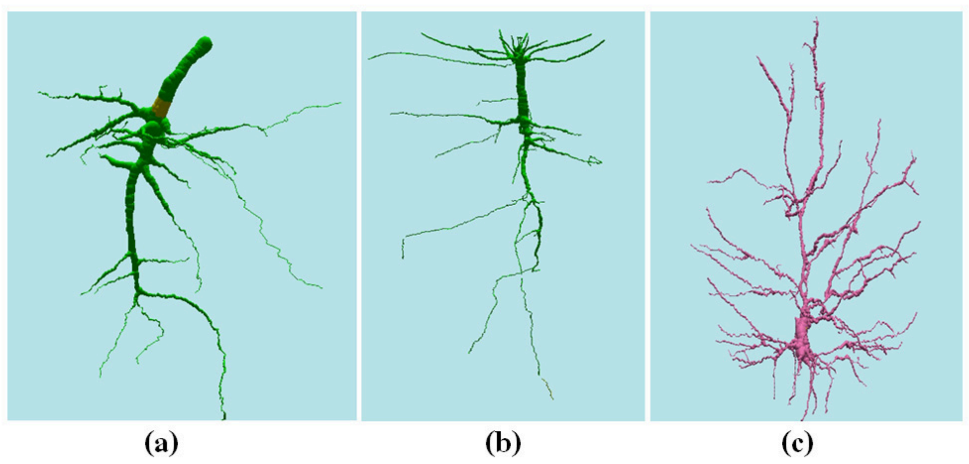
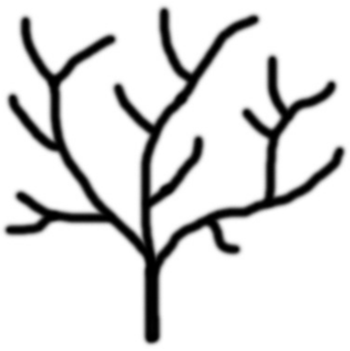
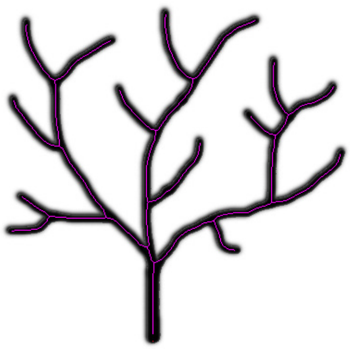


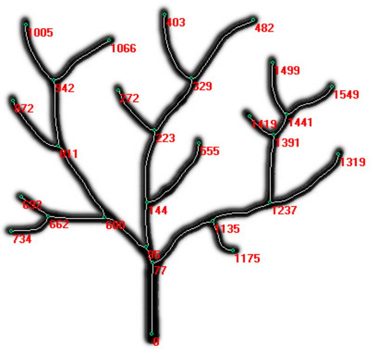
Figure 2.74: Some results of (a), (b) reconstructed roots, and (c) neuron structure in HOC file [Broser, 2006], [Chuai-Aree et al., 2007].



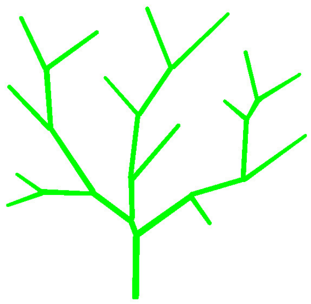
(a) input image



(b) skeletonization



(c) branch analysis



(d) reconstruction with resolution reduction

Figure 2.75: Example of L-string from input image, (a) input image, (b) skeletonization, (c) branch analysis, and (d) reconstruction with resolution reduction process.

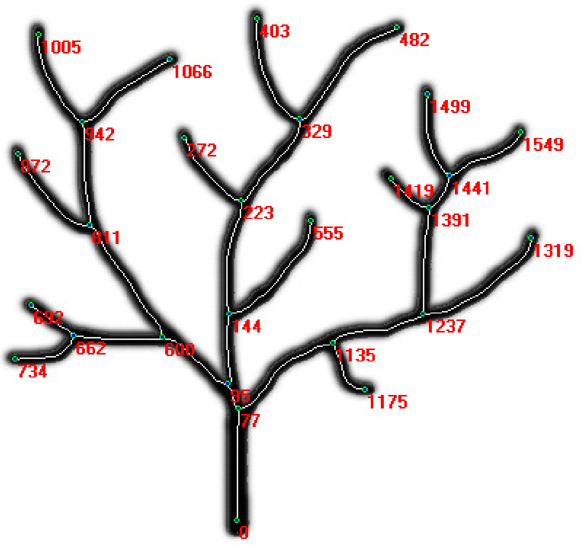


Figure 2.76: Branch analysis and node labelling.

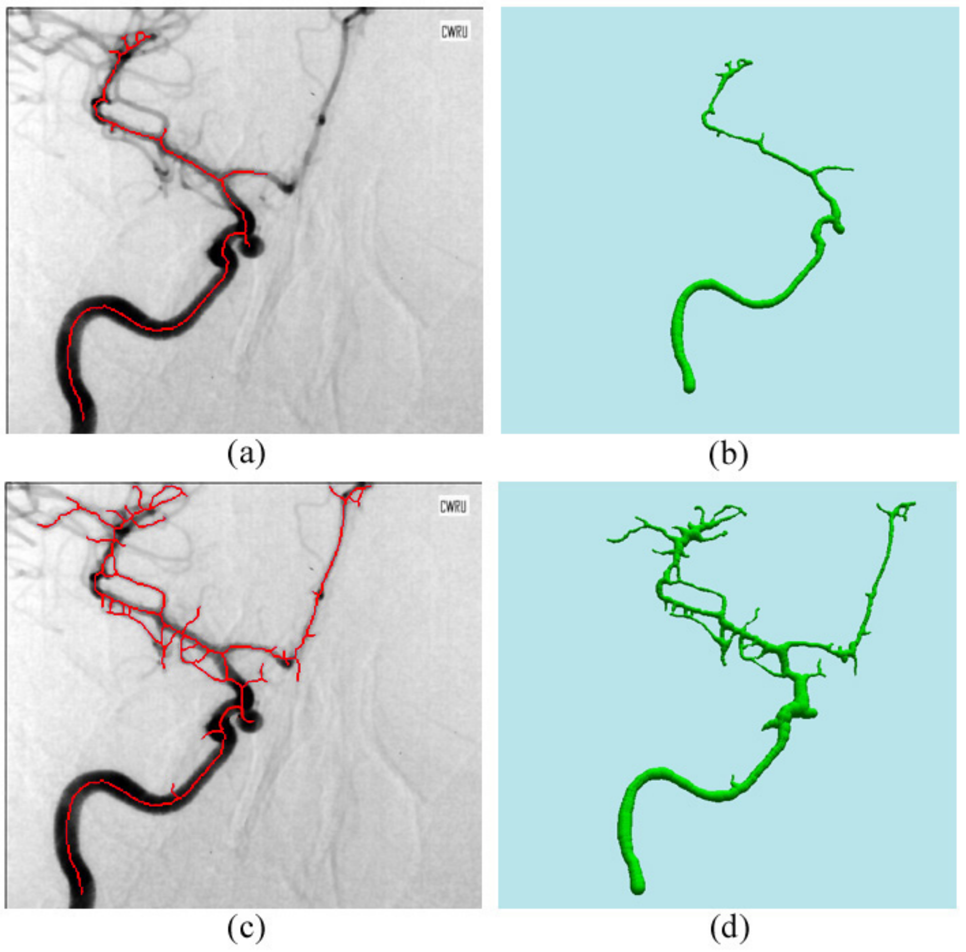


Figure 2.77: The vascular aneurysm and its reconstruction with different conditions and resolutions.

2.7.11 Discussion and Conclusion

The inverse problem of L-systems provides the methods to reconstruct the branching structure from the set of input images. It will be useful for the further uses in plant modeling, network modeling, bio-informatics and medical applications. One can use this method to observe the development of branching structures in biology, botany or medicine. The method allows the user to choose the resolution of the network since many cases the high resolution of network can be ignored. This method also provides to work with the sliced images from medical and biological applications. The volume of branching structure can be discovered and represented in a 3D object.

We are now summarizing the steps for reconstructing the network from input image. The region and volume growing methods are applied for recognizing the branching structure. The skeletonization is called for generating the pixel or voxel alignment. The resolution reduction is applied for removing unnecessary nodes. Finally, the network can be viewed in 3D object. The tree-like structure can be saved in L-string format for further uses. Figure 2.78 shows the steps of rice root reconstruction from the input image (a), growing region method (b), skeletonization (c), and 3D object (d).

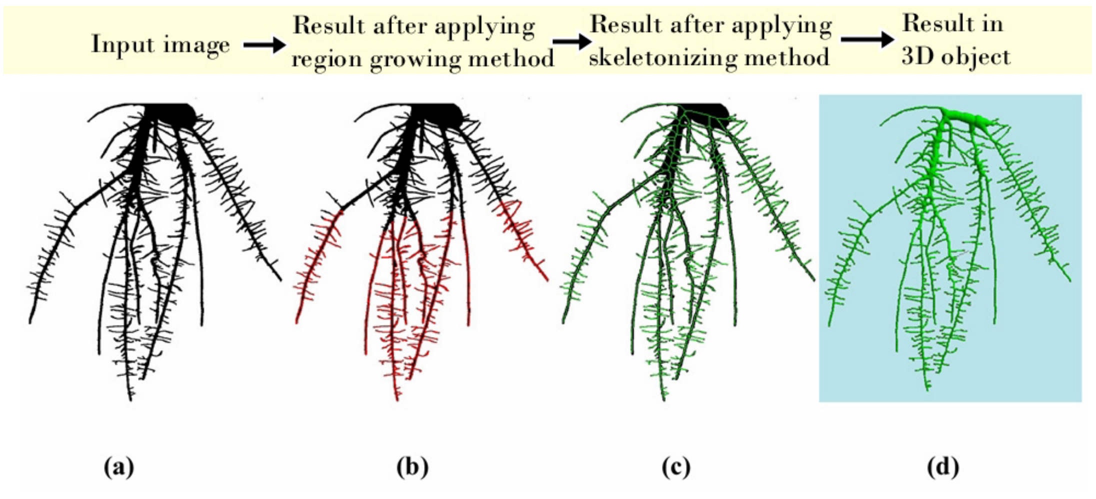

Input image
Result after applying region growing method
Result after applying skeletonizing method
Result in 3D object
(a)
(b)
(c)
(d)

Figure 2.78: The reconstruction process of rice root, (a) input image, (b) result after applying the region growing method, (c) result after applying skeletonizing method, and (d) result in 3D object.

For some input images contained some noise, the de-noising and smoothing algorithm are provided by using the anisotropic diffusion filtering algorithm. Then the normal processes: region or volume growing method, skeletonization, network construction, and 3D viewer, can be applied, respectively. Figure 2.79 shows the procedure of leaf vein reconstruction. The leaf vein cannot be now explained by our L-string reconstruction, but can be saved as graph format.

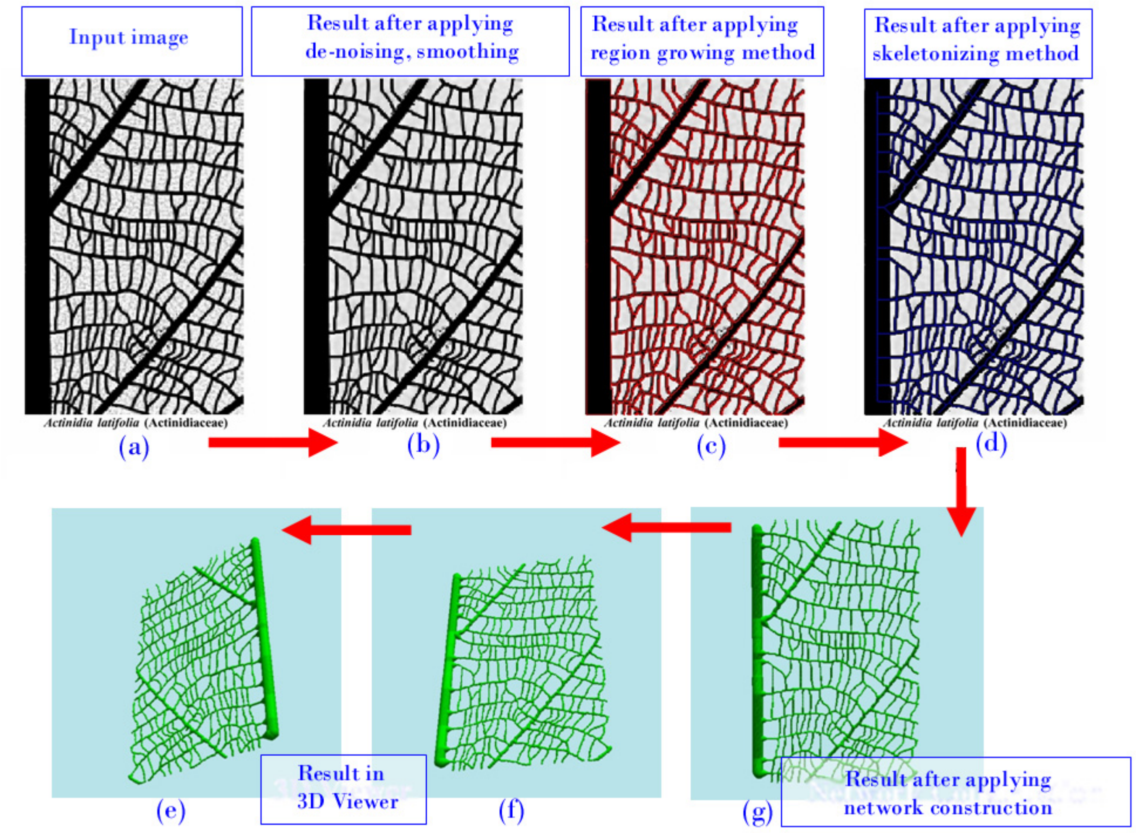

Input image
Result after applying de-noising, smoothing
Result after applying region growing method
Result after applying skeletonizing method
Actinidia latifolia (Actinidiaceae)
(a)
(b)
(c)
(d)
Result in 3D Viewer
Result after applying network construction
(e)
(f)
(g)

Figure 2.79: The reconstruction procedure, (a) input image, (b) result after applying de-noising, smoothing method, (c) result after applying region growing method, (d) result after applying skeletonizing method, (g) result after applying network construction algorithm, (f) and (e) result in 3D viewer.

In 3D soil volume data, the root reconstruction results are not good enough if there are a lot of organic deposit layers. Since the organic matter voxels are in the same interval as root voxels (see detail in Figure 2.45). The worm-like as morphological root structure is taken into account for considering the root path. The resolution of CT-scanner plays a very important role for retrieving the high resolution of input volume data. At least a few root voxels are recognized before segmentation. If the resolution of CT-scanner is less than the root tip then it is impossible to recognize in the given volume data.

Chapter 3

Particle Transportation Method

"Nature is the realization of the simplest conceivable mathematical ideas."
— Albert Einstein (1879-1955)

Particle systems is a modeling method used in a variety of applications in computer graphics. It has been used to model natural modeling such as water and fire based on many independent "particles" with independent characteristics and behaviors. In this dissertation, we would name the "particle transportation method" (PTM) instead of "particles systems" since we apply the different behavior of particle movement.

Branching system in plant is one of the challenging problem of a realistic models. The growth hormone namely auxin (IAA - indole-3-acetic acid) and water in plants play a major role of plant growth. Auxin is required for differentiation of the vascular cambium to develop the function of transportation. The auxin is important for the induction of leaf vein. The pathways of xylem and phloem are for water and food transportation through all parts of plant connected from root.

We propose a simulation algorithm for the branching of leaf venations caused by the movement of auxin in the leaf blade where there are light support. The branching of shoot which is the production of auxin and water transportation in plants together with light and available volume space. We consider the amount of water and nutrient in soil for the major roles of root growth. Light intensity and available space are mainly considered for the shoot growth.

This chapter is organized into six sections: introduction and problem setting, leaf vein generation, algorithm of growing plant generation, plant component arrangements, simulation and results, and discussion and conclusion.

We now motivate this chapter by a following problem and solve the problem by using PTM.

Problem 3.1 *How can we generate the leaf vein if a leaf margin and a source point P_0 are given?*

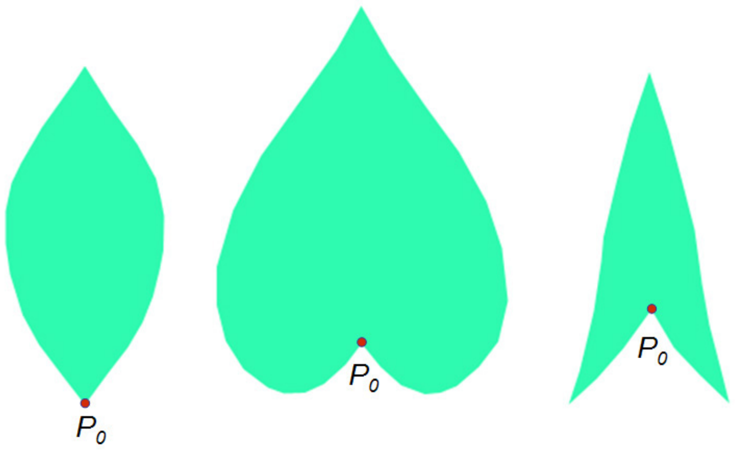


Figure 3.1: Examples of given leaf margin and source point P_0 (connection to petiole).

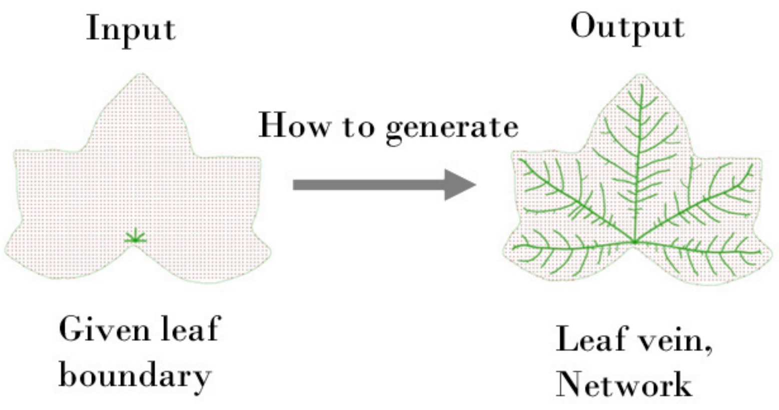


Figure 3.2: Problem and how to generate the leaf vein in the leaf margin.

3.1 Introduction and Problem Setting

Pattern formations of leaf veins and branching patterns were introduced by H. Meinhardt [Meinhardt, 1982]. In leaf veins, the midvein and the main lateral branches are formed before intercalary growth starts, which described why it is only higher order branches that usually form closed loops. For plant leaf venation, the veins are not only to remove the auxin from surrounding tissue but they transport auxin towards the roots. Figure 3.3 illustrates the simulation results proposed by H. Meinhardt [Meinhardt, 1982] based on the activator coupling with substances to control leaf vein differentiation.

In the leaf of Ginkgo tree in Figure 3.3(a), the activator maximum at the growing tip can only divide into two maxima for extending filament. The lateral branching is not allowed to differentiate. The simulation of dichotomous branching of this type is presented in Figure 3.3(b) based on only two controlling substances. The lateral branching of Fittonia leaf in Figure 3.3(c) is possible to differentiate for a new growing tip. The simulated vein in Figure 3.3(d) was started by one differentiated cell at the bottom-right corner of the domain grid. The first vein is growing towards the largest available space in the diagonal. It becomes a midvein of the leaf. Reconnections are possible between higher order branches. The higher order branches are more curved since they are often deflected by other growing tips. All growing tips can form closed loops.

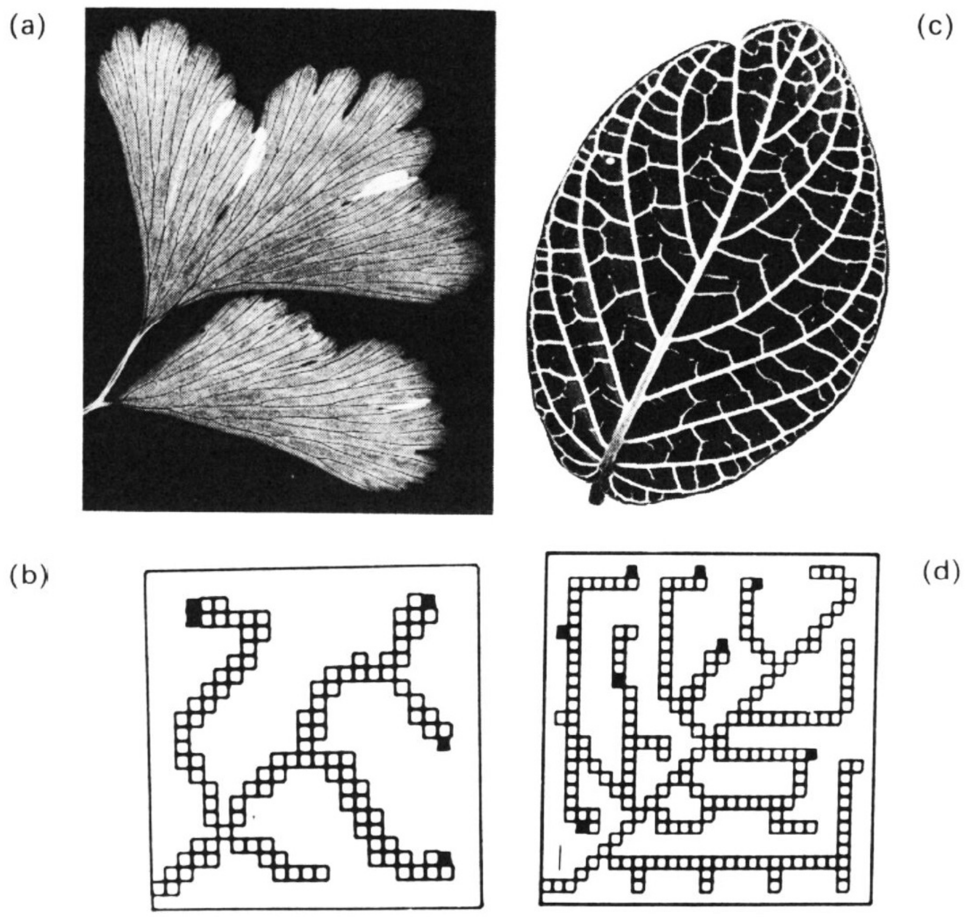


Figure 3.3: Comparison of branching patterns of leaves with their simulation (image from [Meinhardt, 1982]).

To implement the above leaf vein pattern formation, Rodkaew [Rodkaew, 2004] proposed the particle transportation method to provide the new way of leaf vein formation.

Particle transportation method (PTM) has been originally proposed by Rodkaew (see [Rodkaew, 2004]) for generating leaf vein structure of many different leaf shapes and tree-like structures. The idea is to distribute all auxin particles in the given leaf boundary using different types of distribution: square, hexagonal, jitter, uniform and edge distribution. Figure 3.4 shows the example of two types of the PTM simulation of leaf vein. There are two simple algorithms of PTM: Algorithm 3.1 and Algorithm 3.2.

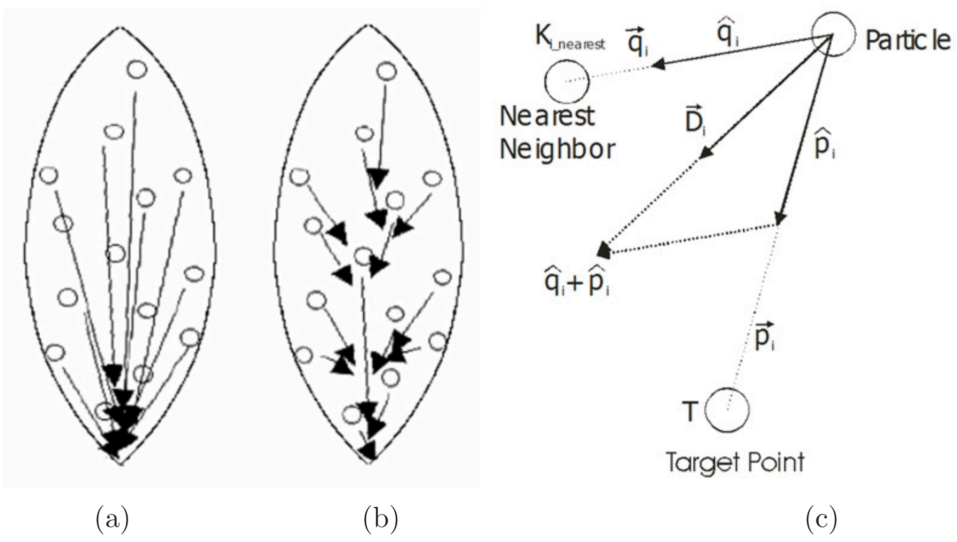


Figure 3.4: Simulated leaf vein by PTM, (a) algorithm 3.1: PTM I, (b) algorithm 3.2: PTM II, (c) moving direction of the particle (see Rodkaew *et al* [Rodkaew, 2004])

Algorithm 3.1 ***PTM-I***

1. Set the particles randomly in the domain of a leaf shape
2. Move the particles to the target (the petiole)
3. Repeat (2) until all particles reach the target

Algorithm 3.2 ***PTM-II***

1. Set the particles randomly in the boundary of a leaf shape
2. Try to use the same path:
    (a) Move current particle towards the nearest particle
    (b) Move current particle to the target together
3. Repeat the step 2 until all particles reach the target

Both algorithms were implemented to represent two types of leaf vein; monocotyledon (Algorithm 3.1: PTM-I) and dicotyledon plants (Algorithm 3.2: PTM-II). The given leaf boundary can be used from both in scanned actual leaf and synthetic leaf shape using spline curve (see Figure 3.5).

Both algorithms create veins from the trail of particles that are scattered within a leaf margin. Let S be the set of particles in leaf blade. At first, each particle i, p_i carries the energy e_i. The direction $\vec{D_i}$ of particle p_i is controlled by the following equation.

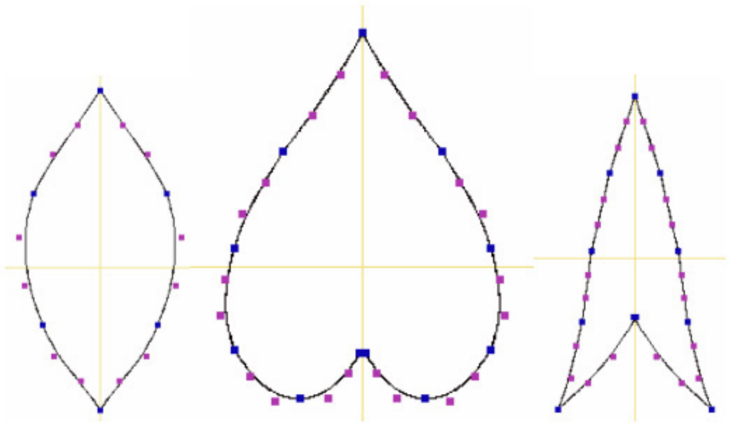


Figure 3.5: Synthetic leaf shapes using spline curve with control points (see Rodkaew *et al.* [Rodkaew, 2004]).

$$\vec{D_i} = \frac{\hat{p_i} + \hat{q_i}}{\|\hat{p_i} + \hat{q_i}\|} \tag{3.1}$$

where $\hat{p_i}$ is a unit vector denoting the direction from a particle to a presumed target point, $\hat{q_i}$ is a unit vector denoting the direction from a particle to its nearest neighbor $K_{i\_nearest}$ (see Figure 3.4(c)), and $\|\vec{x}\|$ is a length of vector x.The unit vectors $\vec{p_i}$ and $\vec{q_i}$ are calculated from the following equations,

$$\begin{aligned} \hat{p_i} &= \frac{\vec{p_i}}{\|\vec{p_i}\|}, \\ \hat{q_i} &= \frac{\vec{q_i}}{\|\vec{q_i}\|} \end{aligned} \tag{3.2}$$

The vector $\vec{p_i}$ and $\vec{q_i}$ can be calculated from the following equations,

$$\begin{aligned} \vec{p_i} &= (X_T - X_{P_i}, Y_T - Y_{P_i}), \\ \vec{q_i} &= (X_{P_{i\_nearest}} - X_{P_i}, Y_{P_{i\_nearest}} - Y_{P_i}) \end{aligned} \tag{3.3}$$

where (X_T, Y_T) and $(X_{P_{i\_nearest}}, Y_{P_{i\_nearest}})$ are the target position T and the nearest particle of P_i, respectively. The target position T in this case means the start point P_0 in Figure 3.1.

To make the direction vector $\vec{D_i}$ in equation (3.1) supporting both algorithms, the vector $\vec{D_i}$ can be calculated from the following equation with weighting parameters.

$$\vec{D_i} = \frac{w_p\hat{p_i} + w_q\hat{q_i}}{\|w_p\hat{p_i} + w_q\hat{q_i}\|} \tag{3.4}$$

where $w_p, w_q \in [0, 1]$, $w_p + w_q = 1.0$, and $w_p, w_q \in \mathbb{R}$.

Statement 3.1 *All particles in algorithm 3.1 and algorithm 3.2 can move to the target point by using the direction vector $\vec{D}_i$ in equation (3.4). The direction vector controls both algorithms as follows:*

1. *It supports the Algorithm 3.1 when $w_p = 1.0$ and $w_q = 0.0$,*
2. *It supports the Algorithm 3.2 when $w_p = 0.5$ and $w_q = 0.5$ or $w_p \cdot w_q \neq 0.0$.*

For each particle P_i at time $t+1$, the position of each particle $P_i(t+1)$ can be calculated from the following equation.

$$P_i(t+1) = P_i(t) + \vec{v}_i(t) \tag{3.5}$$

where $P_i(t+1)$ is the position of particle i at time $t+1$, $P_i(t)$ is the position of particle i at time t, $\vec{v}_i(t)$ is the velocity of particle i at time t, and $P_{t=0}$ is the initial position of particle i at time $t = 0$.

The velocity of the particle i at time t, $\vec{v}_i(t)$ is calculated from the following equation.

$$\vec{v}_i(t) = s \cdot \vec{D}_i(t) \tag{3.6}$$

where $\vec{D}_i(t)$ is the direction vector of particle i at time t, and s is the distance step size of particle in unit time.

When particle i is moving closer to the nearest particle j, then two particles are combined to be the new particles K_{i_new}. It is calculated as follows.

$$K_{i\_new} = K_i + K_{i\_nearest} \tag{3.7}$$

The equation (3.7) is called when the condition satisfies the following equation (3.8).

$$\|\vec{q}_i\| \leq R_i + R_{i\_nearest} \tag{3.8}$$

where $R_{i\_nearest}$ is the radius of the nearest particle of particle i, R_i is the radius of the particle i, and $\|\vec{q}_i\|$ is the length of the vector from the particle i to the nearest particle i.

After combining two particles, the position of a new particle is

$$P_{i\_new} = \frac{P_i \cdot R_i + P_{i\_nearest} \cdot R_{i\_nearest}}{R_i + R_{i\_nearest}}, \tag{3.9}$$

the energy of new particle is

$$E_{i\_new} = E_i + E_{i\_nearest}, \tag{3.10}$$

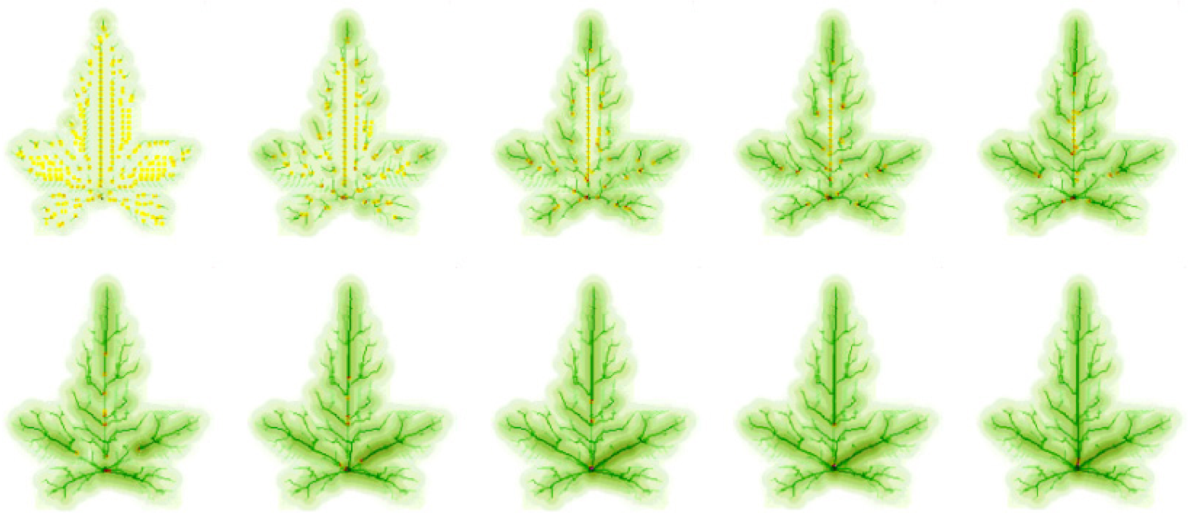


Figure 3.6: The development of vein structure using PTM from Rodkaew *et al* [Rodkaew, 2004].

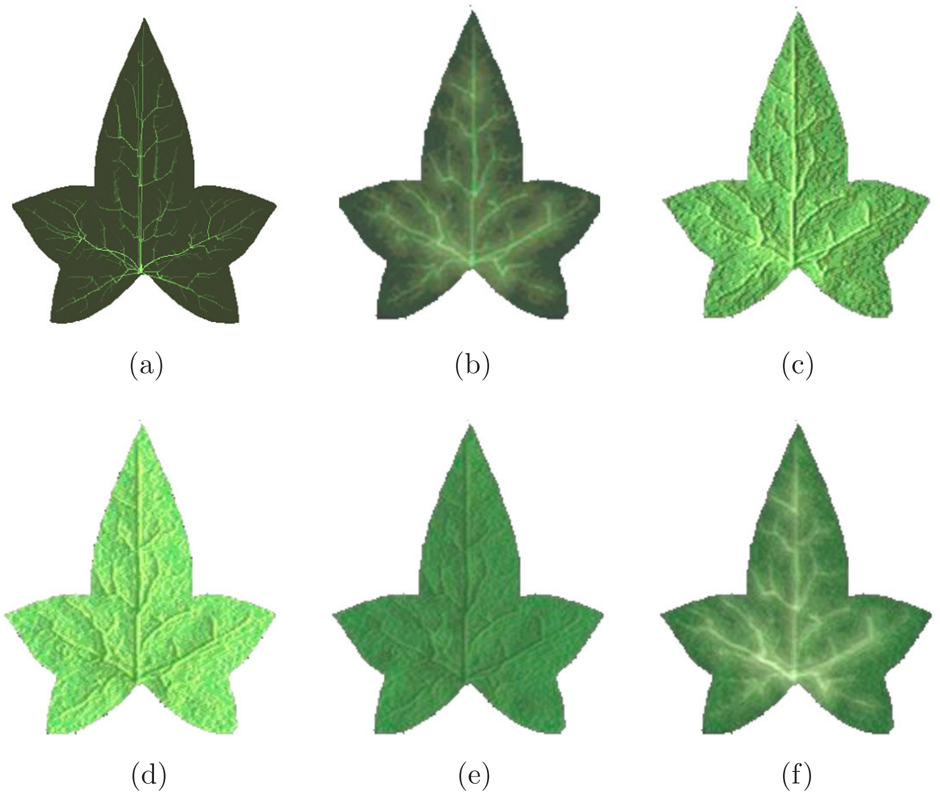


Figure 3.7: Synthetic leaf enhancement of the output from algorithm 3.2 in [Rodkaew, 2004].

and the radius of new particle is

$$R_{i\_new} = \frac{R_i + R_{i\_nearest}}{2}, \tag{3.11}$$

where P_{i_new}, E_{i_new}, and R_{i_new} are the properties of new particle K_{i_new}. The particle K_i and $K_{i\_nearest}$ are removed from the set of particles S.

Figure 3.6 shows development of synthetic leaf venation using PTM from Rodkaew *et al* [Rodkaew, 2004]). All particles from the starting time $t = 0$ are moving to the target point (the connection to petiole). Figure 3.7 shows the result of simulated ivy leaf from Figure 3.6. The image enhancement process is applied to improve the realism of the output. The effect for blurring (see Figure 3.7(b)) and bumping (see Figure 3.7(c) and Figure 3.7(d)) are used. The leaf color is changed to color sampling from the real leaf. The dual-tone color (see Figure 3.7(f)) is produced using the combination of prior images. The image in Figure 3.7(b) is changed to white color and added to Figure 3.7(e). This process is suitable for creating the leaf colors that depends on the vein structure.

Problem 3.2 *From algorithm 3.1(PTM-I) and algorithm 3.2(PTM-II), the set of particles S is initiated at the beginning, then all particles are moving to the target point T. All particles are combined to one particle at the target point. These algorithms simulate nice results of leaf veins, fast simulation, less control parameters, easy to implement, but there are some drawbacks; i.e. some particles across the existing network during particle movement, it does not support the concave boundary in the blade, the leaf margin is not growing. How can we improve these drawbacks from these proposed algorithms?*

We now start to solve this problem by changing the target point T to the source point P_0 which is the start point of all veins. The vein structure starts from P_0 and grows to take all particles to the vein. So, we are going to invert the previous methods in [Rodkaew, 2004]. At each time step of the simulation, we assume that the new product from photosynthesis is produced and needed to transport through the vein structure for producing other organisms in the plant body. The leaf vein generation is described in the next Section 3.2.

3.2 Leaf Vein Generation

This section explains a algorithm of leaf edge detection and leaf vein generation using new PTM. Before generating vein structure, a leaf margin is required as an input for the algorithm. The leaf margin can be retrieved from scanned real leaves or synthetic leaf margin using spline curve.

Some examples of scanned real leaves are given in Figure 3.8.

The input leaf margins can also be constructed by spline curves (see Figure 3.9).

Problem 3.3 *After having the leaf margin as an input, we need to get the leaf margin by edge detection process. How can we detect the leaf margin from a given leaf?*

The next Section 3.2.1 is about to solve this problem automatically after giving the leaf blade to the system.

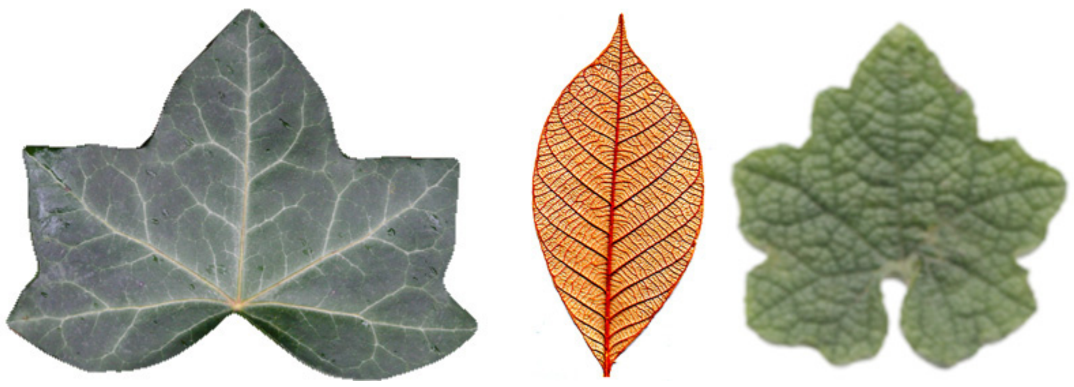

(a) Ivy leaf (b) Rubber leaf (c) Ivy gourd leaf

Figure 3.8: Some examples of scanned real leaves.

3.2.1 Leaf Edge Detection Algorithm

This section describes the algorithm for leaf detection. We firstly investigate the leaf edge detection from the given leaf margin. There are five consecutive steps in Algorithm 3.3 as follows.

Algorithm 3.3 ***LED***

1. Given an input leaf image or an initial leaf margin from spline curve,
2. Set a source point $P_0(x, y)$ in the input leaf image,
3. Check the leaf pixel by 3×3 window in counter-clockwise until the leaf pixel is found and set this pixel as a current pixel,
4. Calculate a polar coordinate of current pixel from equation (3.12),
5. Repeat (2) until the current pixel reaches the source point $P_0(x, y)$.

Leaf boundary is traced by the edge detection process given by the source point $P_0(x, y)$ from the input leaf. It is described by the polar equation in polar coordinate system (radius, angle) in equation (3.12). Figure 3.10 shows the process how the edge detection works in the Algorithm 3.3. At the current point, the consideration of edge detection starts from the last edge point (leaf pixel) in counter-clockwise until the new leaf pixel is found and it is set to the new current point for the next step. The leaf blade is bounded when the source point $P_0(x, y)$ is found as the next current point.

The list of leaf boundary pixels (points) is computed and stored in polar coordinate. The coordinate $(t, r(t))$ and $(t, \theta(t))$ are calculated corresponding to the source point $P_0(x, y)$ or (x_0, y_0) from the following equations $r(t)$ and $\theta(t)$ in equation (3.12).

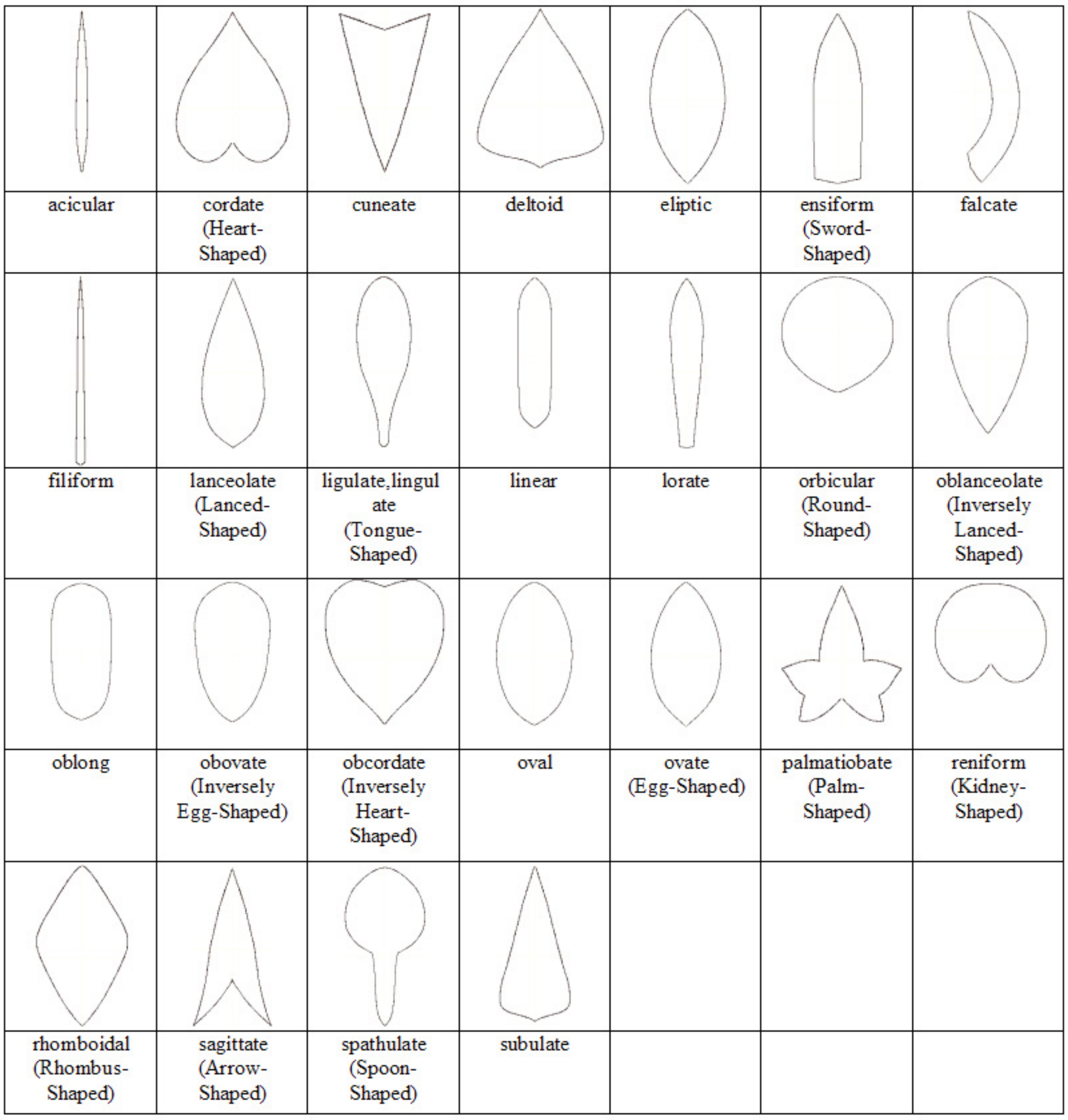

acicular
cordate (Heart-Shaped)
cuneate
deltoid
eliptic
ensiform (Sword-Shaped)
falcate
filiform
lanceolate (Lanced-Shaped)
ligulate,lingulate (Tongue-Shaped)
linear
lorate
orbicular (Round-Shaped)
oblanceolate (Inversely Lanced-Shaped)
oblong
obovate (Inversely Egg-Shaped)
obcordate (Inversely Heart-Shaped)
oval
ovate (Egg-Shaped)
palmatiobate (Palm-Shaped)
reniform (Kidney-Shaped)
rhomboidal (Rhombus-Shaped)
sagittate (Arrow-Shaped)
spathulate (Spoon-Shaped)
subulate

Figure 3.9: Leaf margins constructed by spline curve in [Rodkaew, 2004].

$$\begin{aligned} r(t) &= \sqrt{(x_t - x_0)^2 + (y_t - y_0)^2} \\ \theta(t) &= cos^{-1}\left(\frac{\overrightarrow{(x_t, y_t)} \cdot \overrightarrow{(1,0)}}{\sqrt{x_t^2 + y_t^2}}\right) \end{aligned} \tag{3.12}$$

where (x_t, y_t) is the current pixel of leaf boundary, $r(t)$ and (t) are radius or distance from source point P_0 to the current pixel P_t, and angle between the current vector $\overrightarrow{(x_t, y_t)}$ and unit vector $\overrightarrow{(1,0)}$, respectively. $\theta(t)$ is in [-90,240] in degree started from the first or forth quadrant.

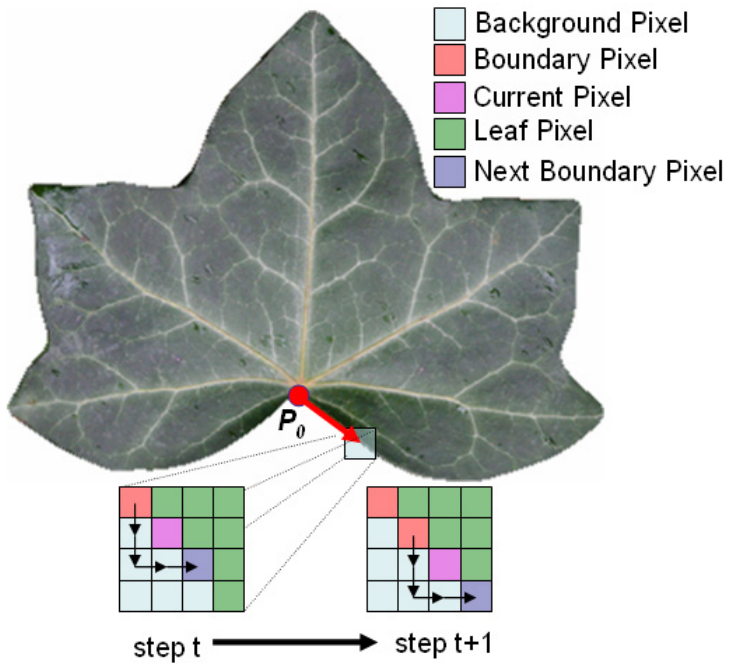


Figure 3.10: The process of leaf edge detection from step t to step $t+1$ tracing in counter-clockwise of the ivy leaf boundary.

The radius and angle of each boundary point is plotted against the boundary pixel number t. Figure 3.11 shows the development of leaf edge detection using Algorithm 3.3 at $t = 0$, 210, 404, 600, 810, and 975. Figure 3.12 presents the graph of radius $r(t)$ (green line) and angle $\theta(t)$ (red line) against the edge detection time step t. The tracing process is started from the source point in counter-clockwise until reaching the source point P_0 (red circle). Leaf boundary points are stored in the set of boundary points B (blue path).

In the graph of Figure 3.12, each point at time step t of edge detection can be also plotted to the original leaf margin based on $(r(t), \theta(t))$ coordinate.

We have now detected the leaf margin from the given leaf. The next step is to generated the leaf vein structure. The detail of leaf vein generation is presented in Section 3.2.2.

3.2.2 Leaf Vein Generation Algorithm

We now start a new algorithm to generate leaf venation network by the inverse direction of Rodkaew's algorithms in Rodkaew *et al* [Rodkaew, 2004]. The leaf margin detected in the Algorithm 3.3 is used by the following algorithm as the input set of boundary points B.

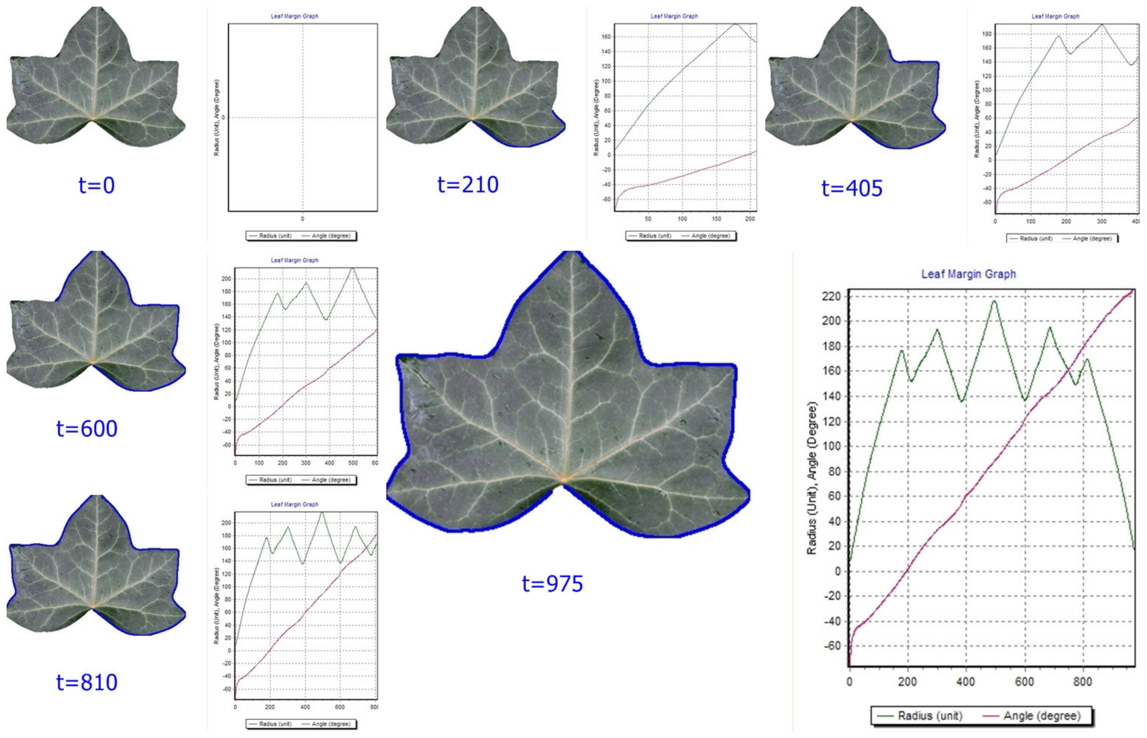


Figure 3.11: Development of leaf edge detection using algorithm 3.3.

Algorithm 3.4 ***PTM-LeafVein***

1. Given 2D initial boundary points of leaf margin from Algorithm 3.3,
2. Given source point $P_0(x, y)$ of petiole of leaf,
3. Random the set of particles S in the leaf blade for k particles, e.g. a set of auxin particles,
4. Generate the leaf vein to the available set of particles S based on product obtained by photosynthesis process in leaf blade,
5. Store the leaf vein network at the current state,
6. Update the leaf vein depending on leaf growth step d_l in time,
7. Do growing process $G_l(t)$ of leaf boundary and do the step 3, 4, 5 and 6 until reaching the appropriate time when the vein resolution is bounded.

In order to give more detail in the Algorithm 3.4, we now explain in the next Section 3.2.3.

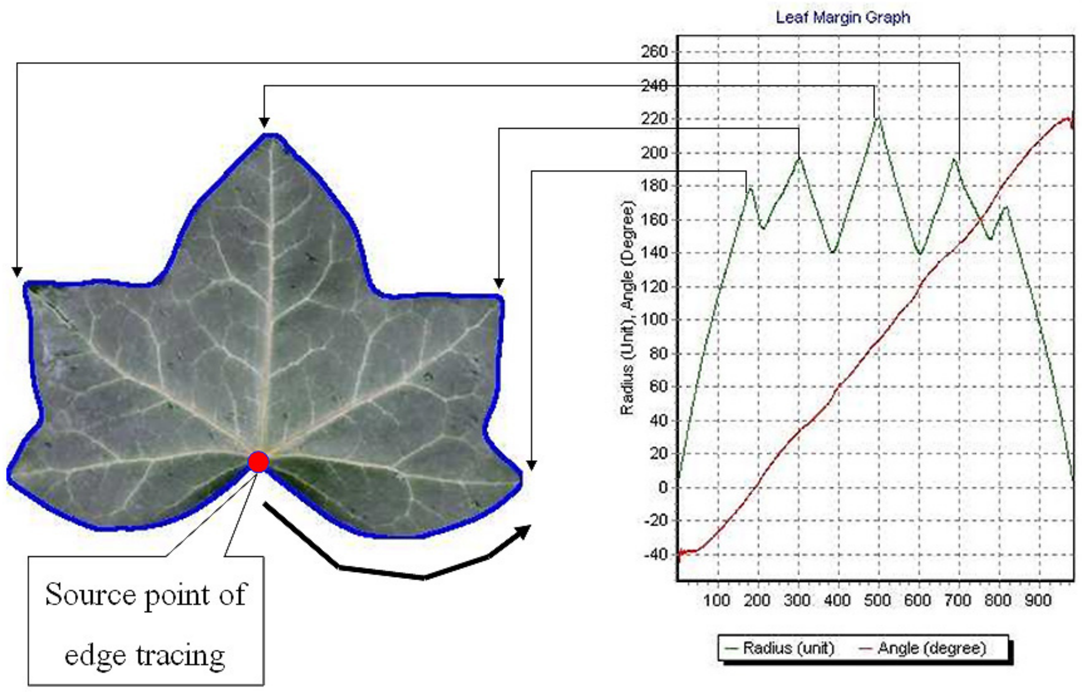


Figure 3.12: Tracing leaf boundary in polar coordinate system.

3.2.3 Leaf Vein Construction

This section describes leaf vein construction to explain in detail of the Algorithm 3.4.

After giving the initial leaf margin, the source point P_0 is defined for initial point of leaf vein. Let a point $n_0(x_0, y_0)$ be the given initial network point, d_l be the growing step size of leaf vein segment, respectively. At each time step t with step size d_t, all random particles are searching for nearest point in the set of network particles N, which its distance is less than R_c. Each point n_i in the set N has its own particles s_k in set S_i, which are respect to it as a nearest point. Each point n_i will grow as a new network point n_j in the next step if and only if it has more than one particle influence to it as a set of nearest points. The calculation of the absolute direction of n_j can be calculated from the average of all unit vector v_i with the growing step size d_l.

The new network point n_j is calculated from the following equations (3.13) and (3.15).

$$\begin{aligned} n_j &= n_i + d_s \frac{v_i}{\|v_i\|}, \\ v_i &= \sum_{k=1}^{M_i} \frac{(n_i - S_k)}{\|n_i - S_k\|}, \forall i \in N, \forall k \in S_i. \end{aligned} \tag{3.13}$$

From equation (3.13), M_i is the number of particles which are closet to network point

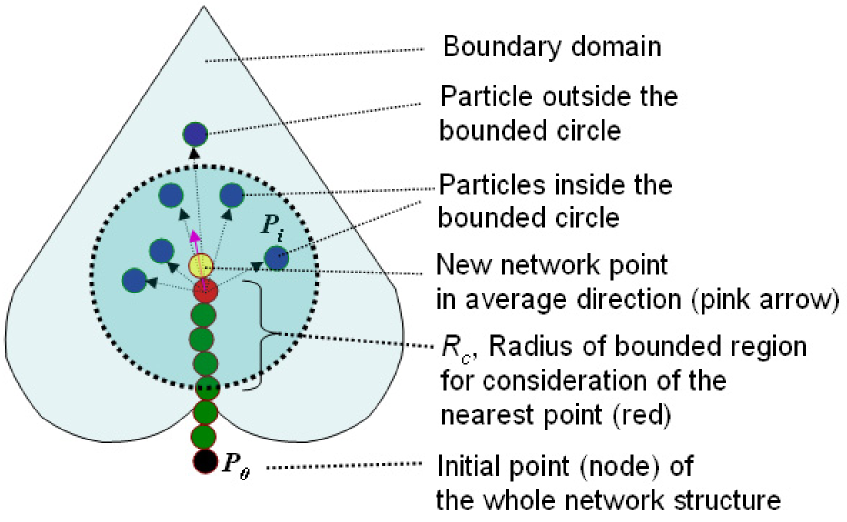

Boundary domain
Particle outside the bounded circle
Particles inside the bounded circle
P_i
New network point in average direction (pink arrow)
R_c, Radius of bounded region for consideration of the nearest point (red)
P_0
Initial point (node) of the whole network structure

Figure 3.13: Example of particle in leaf boundary.

n_i. The new segment of leaf network is created by drawing edge (n_i, n_j). The number of network point is increased every time step if there is at least a point, which is in S.

At each time step, the size of leaf margin can be fixed or can grow based on the following sigmoidal growth function in equation (3.14).

$$G_l(t) = L + \frac{U - L}{1 + e^{m(T-t)}} \tag{3.14}$$

where

$G_l(t)$	: the growth of leaf at time t,
L	: the minimum value of length from P_0 to $r(t_i)$ of the small leaf,
U	: the maximum value of length from P_0 to $r(t_i)$ of the big leaf,
m	: the approximated slope of raw data,
T	: the time at $G_l(t) = (U + L)/2$, and
t	: the independent time variable.

From the equation (3.14), the parameter L can be zero in case the leaf grows from a very small organism. The result after running the Algorithm 3.4 with fixed leaf margin at $t = 10$, 60, 85, and 120, is visualized in Figure 3.14.

Figure 3.15 shows the development of growing leaves from $t = t_0$ to $t = t_7$. The equation (3.14) is used to calculate the growth parameters, e.g. L, U, m, and T. The example of simulated leaf growth using the equation (3.14) with $L = 0$, $m = 0.20$, $T = 15$, and U value taken from the length of $r(t_i)$ from the source point P_0 to each boundary point i is illustrated in Figure 3.16.

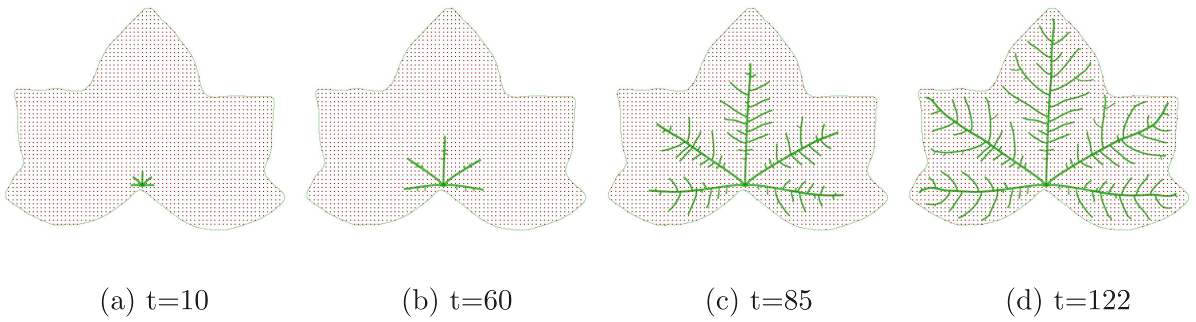

(a) t=10 (b) t=60 (c) t=85 (d) t=122

Figure 3.14: Result after running the Algorithm 3.4 with fixed leaf margin at (a) t=10, (b) t=60, (c) t=85, and (d) t=122.

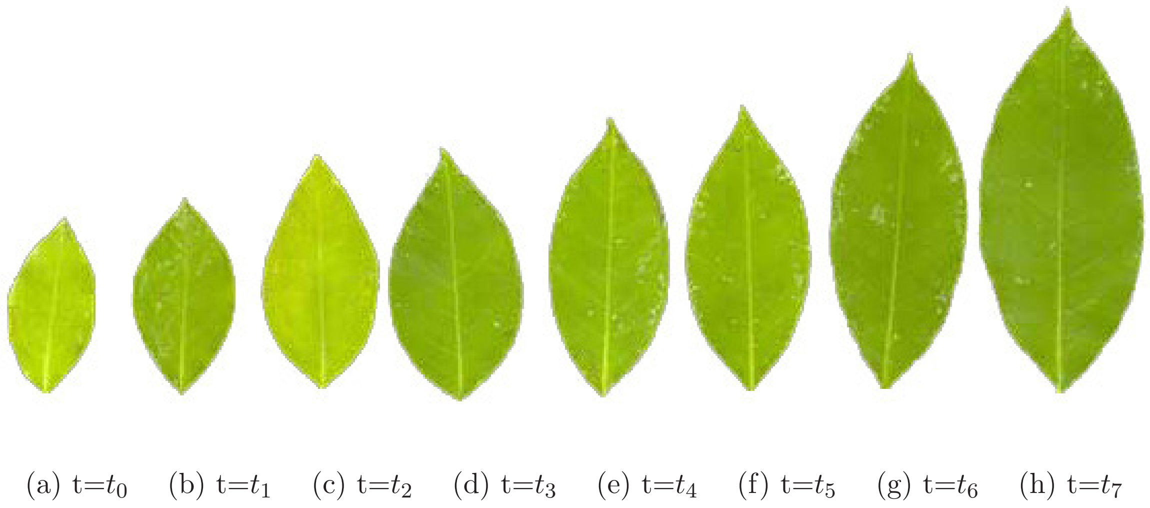

(a) t=t_0 (b) t=t_1 (c) t=t_2 (d) t=t_3 (e) t=t_4 (f) t=t_5 (g) t=t_6 (h) t=t_7

Figure 3.15: Development of growing leaves from $t = t_0$ to $t = t_7$.

At each time step, while the leaf margin is growing, the vein structure is generated simultaneously. The leaf margin expands with its vein structure as a natural leaf expansion. The simple example is shown in Figure 3.17 for second order leaf vein with the product from photosynthesis process. The simulation of the same growth of leaf with high amount of product (red region) is presented in Figure 3.18.

We now have the algorithm to generate leaf veins by fixed and growing leaf margins. In the next section we start to generate the plant structure.

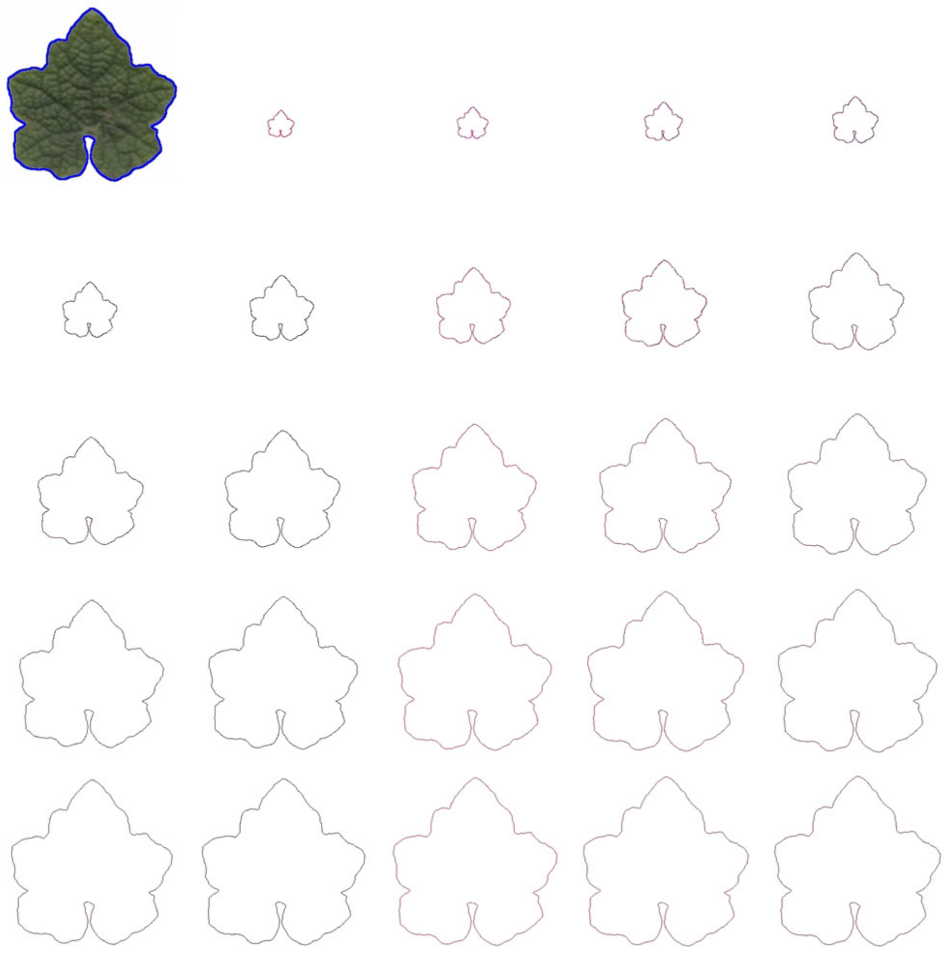


Figure 3.16: Example of growing ivy gourd leaf margin starting from $t = 1$ for every 5 time steps.

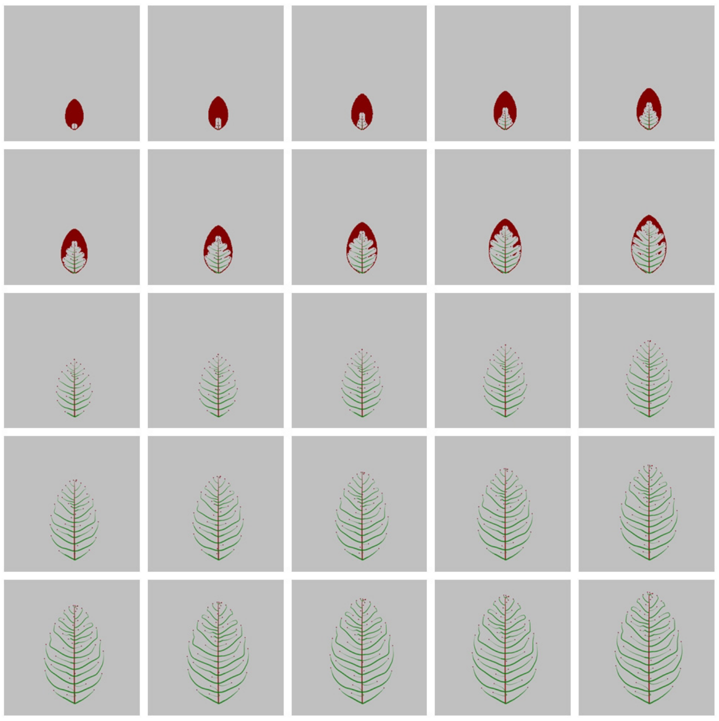


Figure 3.17: Example of growing leaf vein and margin starting from $t = 1$ for every 2 time steps, red region is the product from photosynthesis process.

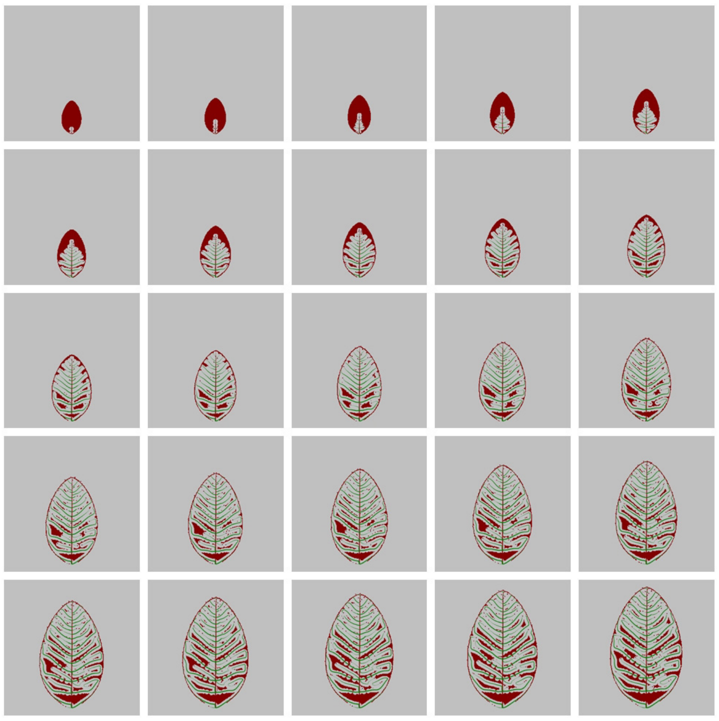


Figure 3.18: Example of growing leaf vein and margin starting from $t = 1$ for every 2 time steps with high amount of product from photosynthesis process.

3.3 Growing Plant Generation Algorithm

This section explains a plant structure generation using PTM. Rodkaew proposed the method to generate tree structures using PTM in Rodkaew *et al* [Rodkaew, 2004]). The results from PTM in [Rodkaew, 2004] are not developed in natural way of plant growth (see the model in Figure 3.19). The result is shown in Figure 3.20. All particles of shoot part move from leaf position to the seed of tree on the soil surface and all particles of root part also move from root tip position to the seed position. The tree structure can be generated only the fixed shoot and root volume.

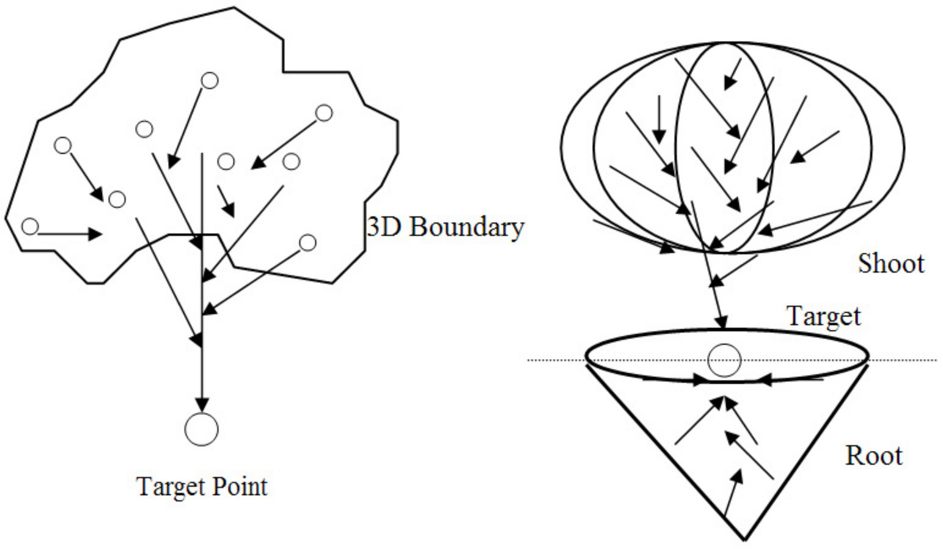


Figure 3.19: Plant Model by PTM in [Rodkaew, 2004].

In this drawback, we improve the new method for growing plant structure from the seed position and grow in the realistic way. We now propose a new algorithm for generating the plant structure in the following Algorithm 3.5 (*PTM-Tree*).

There are seven steps for shoot and root structures as follows.

Algorithm 3.5 ***PTM-Tree***

1. Given a 3D initial volume boundary of shoot and root components,
2. Given a source point $P_0(x, y, z)$ of the seed of plant,
3. Random the set of light particles for the shoot part, set of water particles for the root part,
4. Generate the shoot and root structures to the available set of particles based on the gaining of light for photosynthesis in shoot part, and reaching the moist zone for water uptake in the root part,

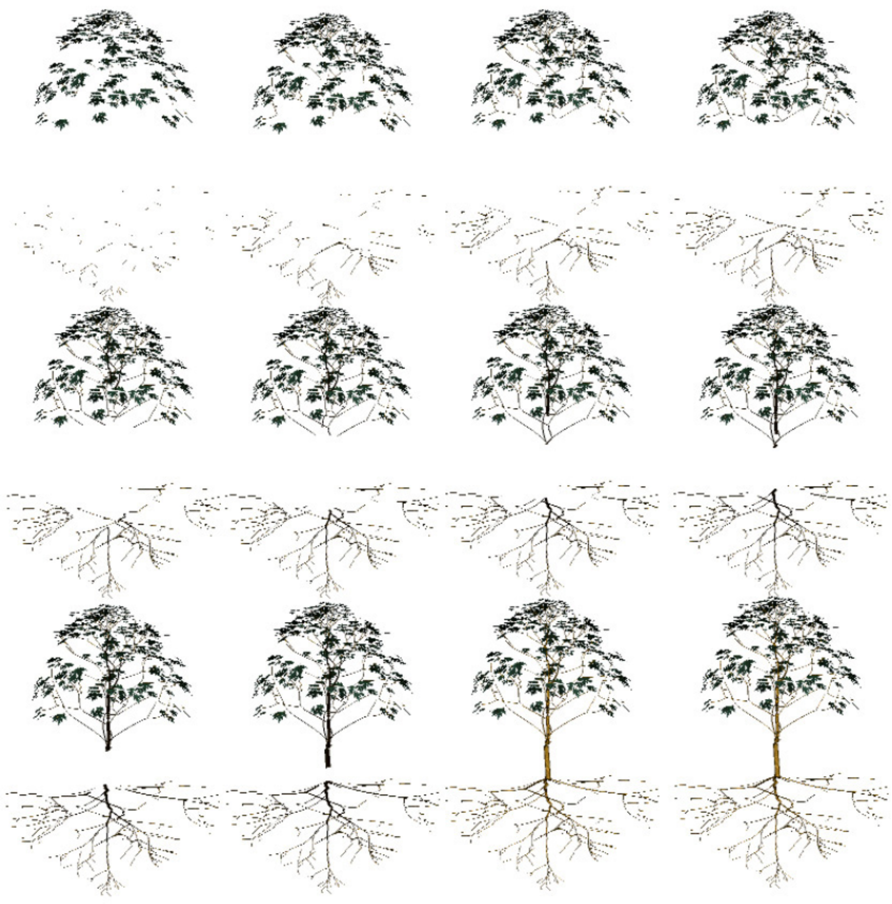


Figure 3.20: Result of tree generation by PTM in [Rodkaew, 2004].

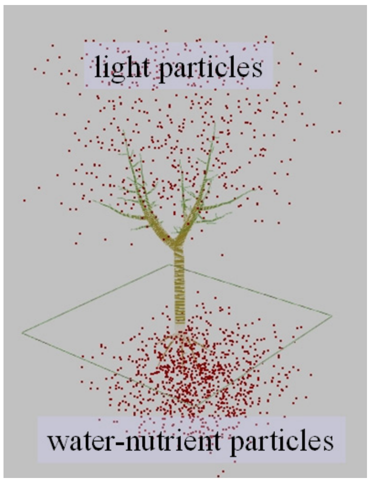


Figure 3.21: Tree structure with light (shoot) and water-nutrient (root) particles in a cubic shape for shoot and root volume.

5. Store the shoot and root networks at the current state,

6. Update the plant components depending on their growth processes in time,
7. Do growing process of shoot and root volume boundary and do the step 3, 4, 5 and 6 until reaching the appropriate time.

To give better detail, we now explain in the next section for each step in the above algorithm.

3.3.1 Shoot and Root Boundary

The shoot and root boundaries are represented by the basic shape such as cone, cylinder, sphere and cube. They can be resized by their parameters both in shoot and root volume. The set of particles P stores the random particle points $P_i(x, y, z)$ in a given boundary domain B of shoot and root volume. Set N represents the set of network points.

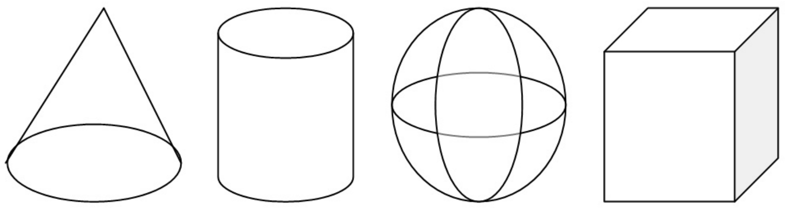


Figure 3.22: Basic boundaries of plant shoot and root domain; cone, cylinder, sphere and cube.

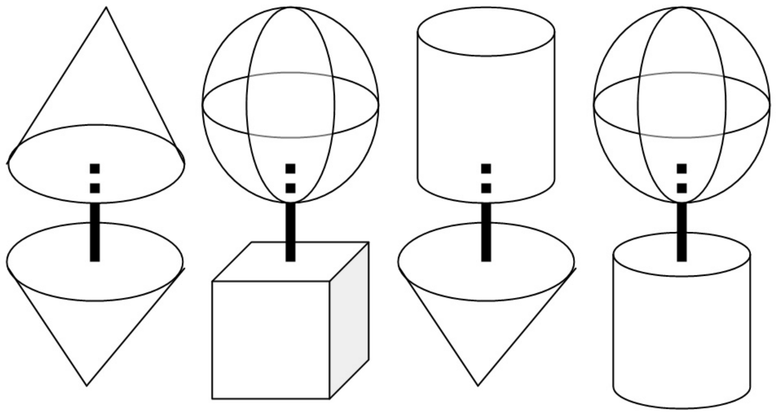


Figure 3.23: Example of the combination of shoot and root domain as a complete structure.

Basic shapes of plant shoot and root boundaries and some examples of combination of shoot and root boundary are illustrated in Figure fig:objectbase and Figure 3.23.

3.3.2 Network Construction and Growing Volume Method

After giving the initial volume of shoot and root parts, the source point P_0 is defined for initial point of plant network. Let a point $n_0(x_0, y_0, z_0)$ be the given initial network point,

d_s and d_r be the growing step size of shoot and root parts, respectively. At each time step t with step size d_t, all random particles are searching for nearest point in the set of network particles N, which its distance is less than R_c. R_c is the radius for considering the distance to particles. Each network point n_i in the set N has its own particles s_k in set S_i, which are respect to it as a nearest point. Each network point n_i will grow as a new network point n_j in the next step if and only if it has more than one particle influence to it as a nearest point. The calculation of the absolute direction of n_j can be calculated from the average direction of all unit vectors v_i with the growing step size d_s or d_r (see Figure 3.24).

The new network point n_j is calculated from the following equations.

$$\begin{aligned} n_j &= n_i + d_s \frac{v_i}{\|v_i\|}, \\ v_i &= \sum_{k=1}^{M_i} \frac{(n_i - S_k)}{\|n_i - S_k\|}, \forall i \in N, \forall k \in S_i. \end{aligned} \tag{3.15}$$

From equation (3.15), M_i is the number of particle which are closet to network point n_i. The new internode of network is increased by drawing edge (n_i, n_j). The number of network point is added every time step if there is at least a point, which is in S (see Figure 3.25).

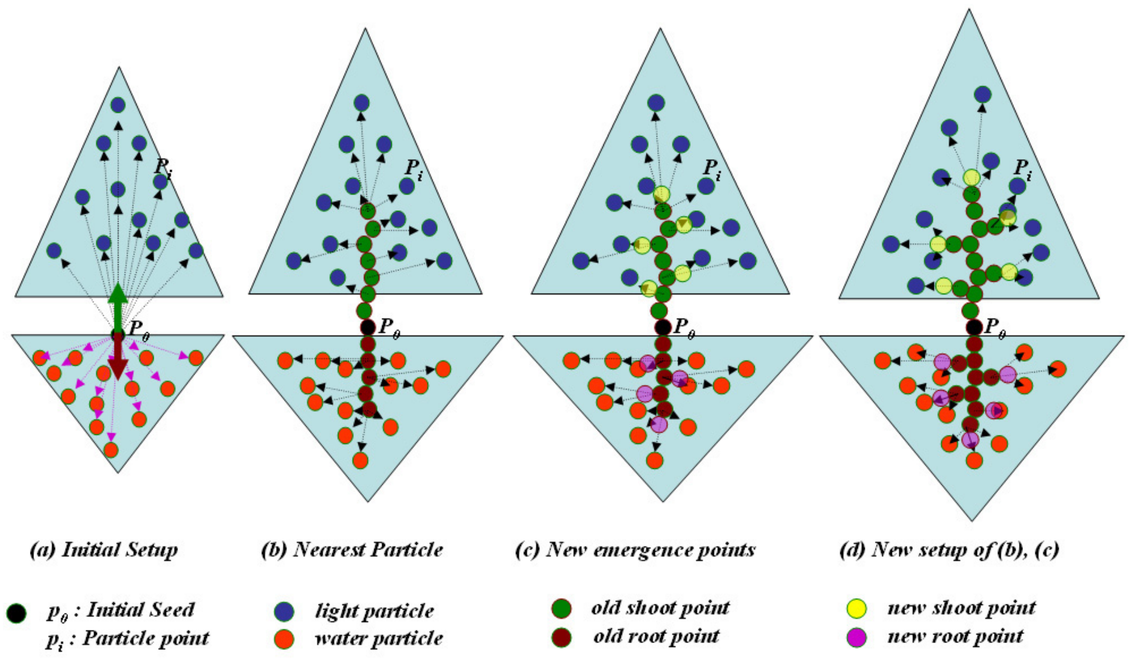


Figure 3.24: Example of network construction from available particles.

We now can generate the plant shoot and root structures. Figure 3.26 illustrates some results from the Algorithm 3.5 for conical shape, cubical shape, spherical shape, and root system. Next section we introduce the arrangement of plant components.

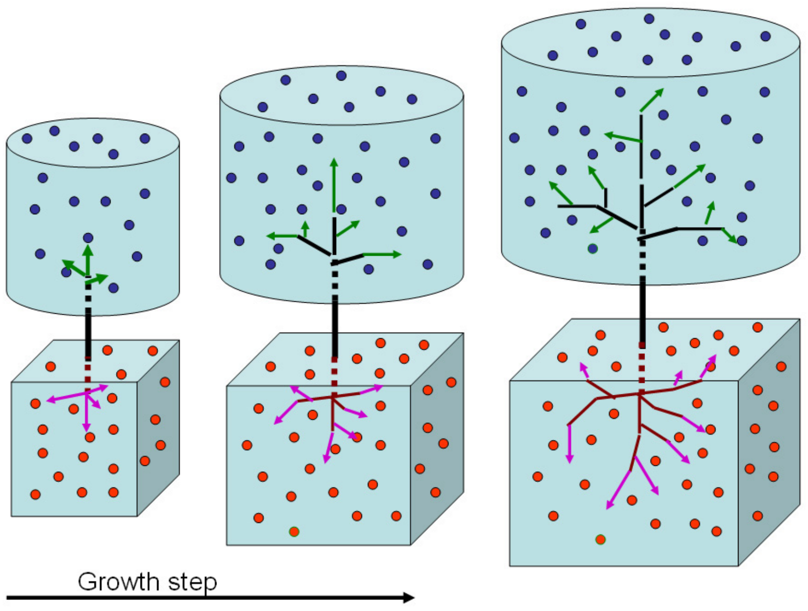


Figure 3.25: Growing process of shoot and root networks.

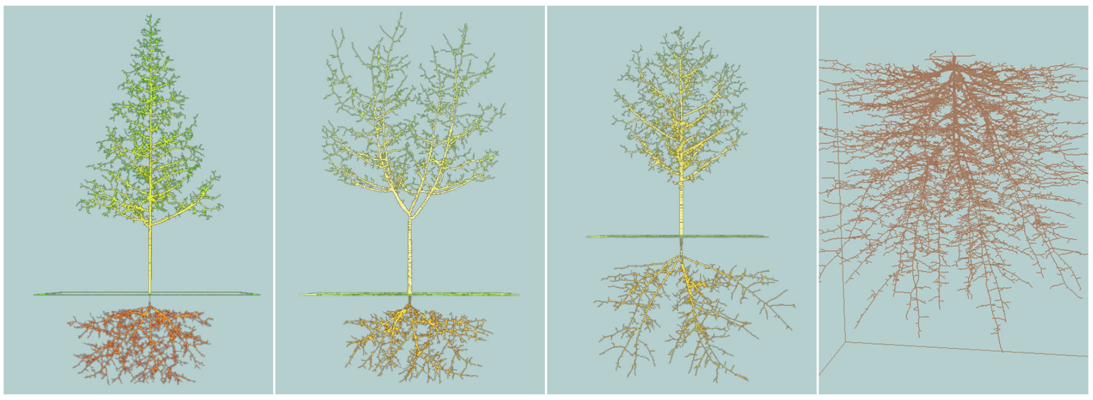


Figure 3.26: Some results from the Algorithm 3.5 for conical shape, cubical shape, spherical shape, and root system.

3.4 Plant Component Arrangements

This section describes the plant component arrangements, e.g. leaf arrangements, flower and fruit arrangements.

3.4.1 Leaf Arrangement

Leaves are formed at the meristem and attached at the node position. At the leaf position, a bud is embedded for a new branch generation under the appropriate condition. The different types of leaf shapes are presented in Figure 3.27. Leaf veins are shown in Figure 3.28 and leaf margins are illustrated in Figure 3.29. The leaf arrangement is called *phyl-*

lotaxis. There are many different kinds of leaf arrangements. Figure 3.30 shows several types of leaf arrangements and leaf compositions of dicotyledon plants.

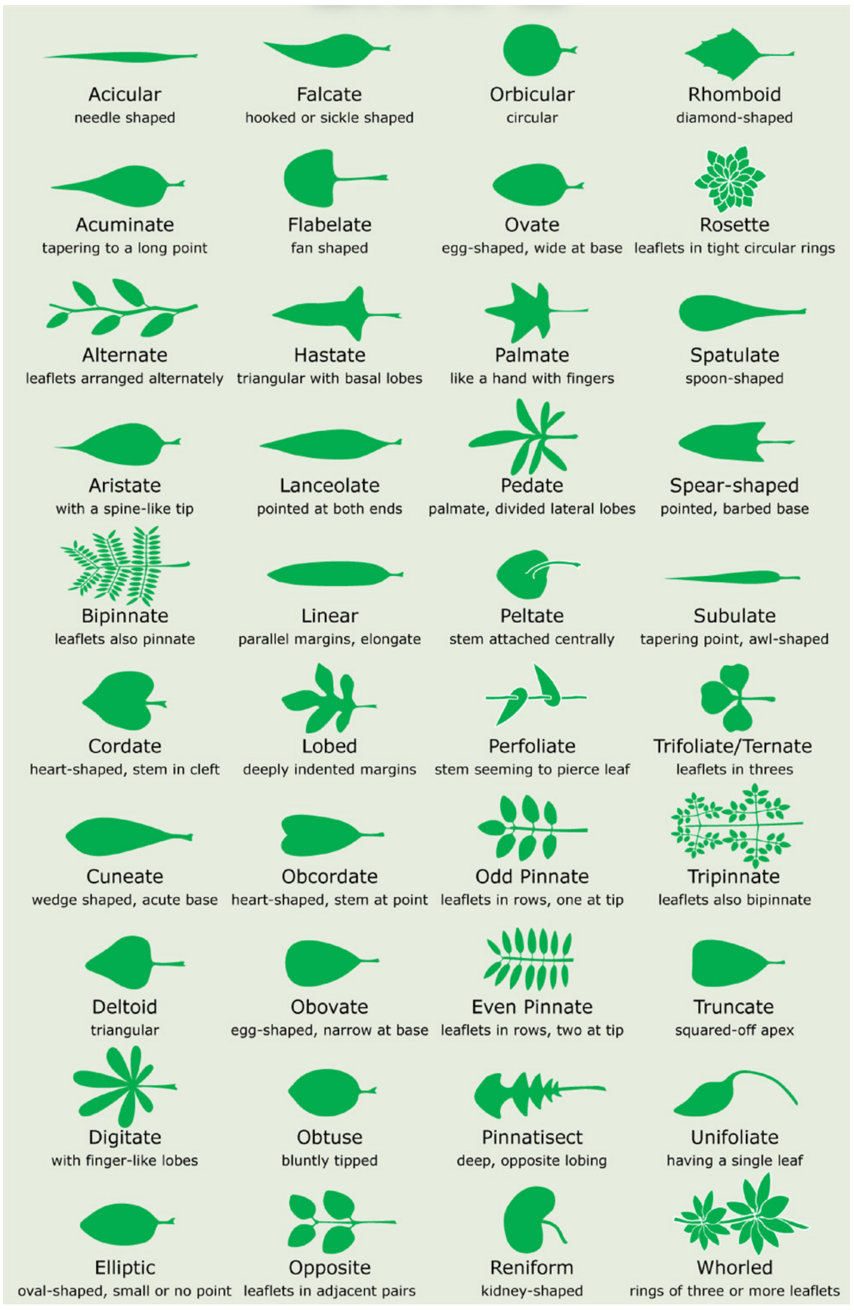


Figure 3.27: Leaf shapes, e.g. acicular, falcate, orbicular, rhomboid, acuminate, flabelate, ovate, rosette, alternate, hastate, palmate, spatulate, aristate, lanceolate, pedate, spear-shaped, bipinnate, linear, peltate, subulate, cordate, lobed, perfoliate, trifoliate/ternate, cuneate, obcordate, odd pinnate, tripinnate, deltoid, obovate, even pinnate, truncate, digitate, obtuse, pinnatisect, unifoliate, ellptic, opposite, reniform, and whorled (image from en.wikipedia.org).

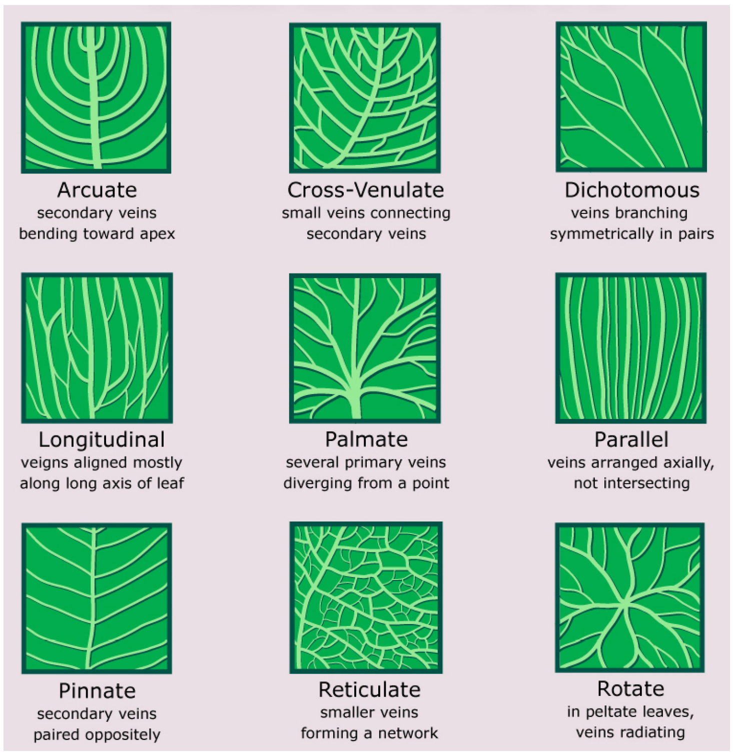

Arcuate
secondary veins bending toward apex
Cross-Venulate
small veins connecting secondary veins
Dichotomous
veins branching symmetrically in pairs
Longitudinal
veigns aligned mostly along long axis of leaf
Palmate
several primary veins diverging from a point
Parallel
veins arranged axially, not intersecting
Pinnate
secondary veins paired oppositely
Reticulate
smaller veins forming a network
Rotate
in peltate leaves, veins radiating

Figure 3.28: Leaf venations, e.g. arcuate, cross-venulate, dichotomous, longitudinal, palmate, parallel, pinnate, reticulate, and rotate (image from en.wikipedia.org).

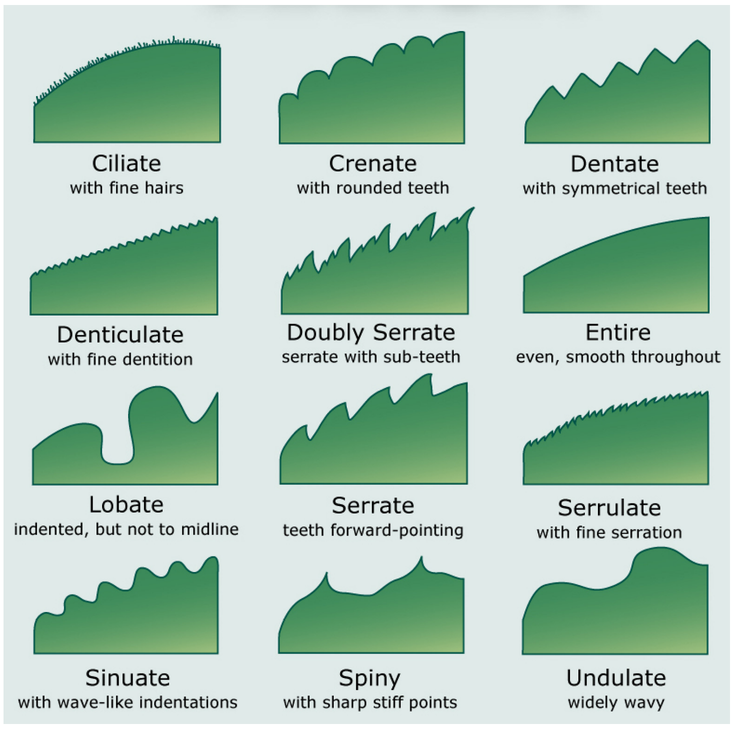


Figure 3.29: Leaf margins, e.g. ciliate, crenate, dentate, denticulate, doubly serrate, entire, lobate, serrate, serrulate, sinuate, spiny, and undulate (image from en.wikipedia.org).

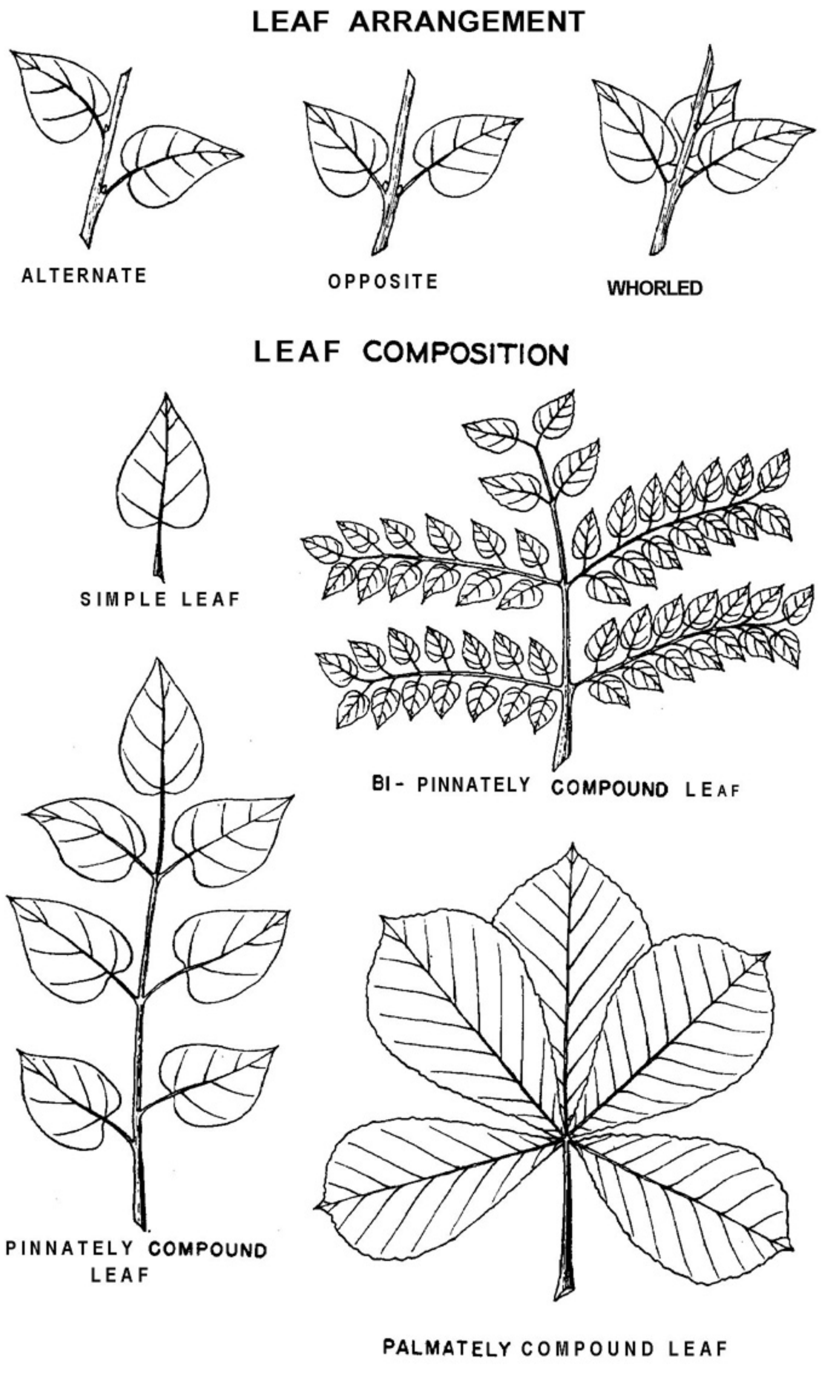
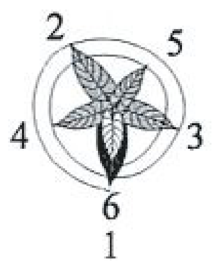


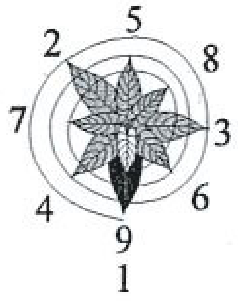
Figure 3.30: Leaf arrangements, e.g. alternate, opposite, whorled; leaf compositions, e.g. simple leaf, bi-pinnately compound leaf, pinnately compound leaf, and palmately compound leaf (image from http://www.forestry.state.al.us).

Figure 3.31 shows how to get from the topmost leaf to the last of the 5 leaves of the plant on the left takes 2 counter-clockwise turns (see Figure 3.31(a)) or 3 clockwise turns (see Figure 3.31(b)). The numbers, 2, 3 and 5 are three consecutive Fibonacci numbers. For the plant on the right, it takes 3 counter-clockwise rotations (see Figure 3.31(c)) or 5 clockwise rotations (see Figure 3.31(d)) to pass 8 leaves. Again, the numbers, 3, 5 and 8 are consecutive Fibonacci numbers. The fraction of number of leaves per number of rounds and its angle are expressed as follows, (a) 5/3 : 5 leaves/2 rounds with (e) 144°, (b) 5/3 with (f) 216°, (c) 8/3 with (g) 135°, and (d) 8/5 with (h) 225°. An estimated 90% of all plants displays this pattern.



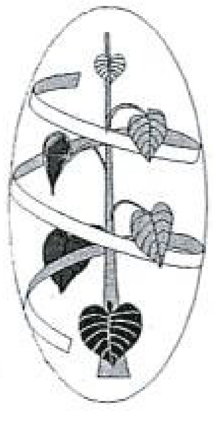
(a) 5/2

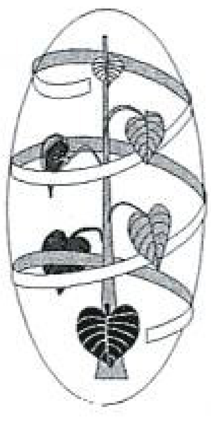
(b) 5/3

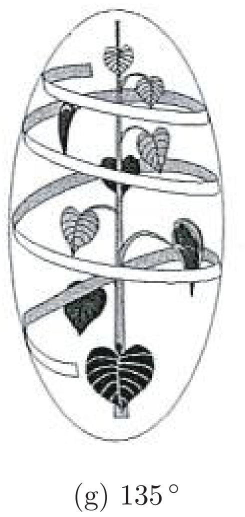


(c) 8/3

(d) 8/5


(e) 144°


(f) 216°


(g) 135°

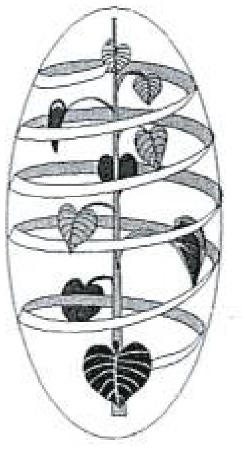

(h) 225°

Figure 3.31: Leaf arrangement with fibonacci number by Michael S. Schneider from *A Beginners Guide To Constructing The Universe*, the fraction of number of leaves and number of rounds, (a) 5/3 : 5 leaves/2 rounds with (e) 144°, (b) 5/3 with (f) 216°, (c) 8/3 with (g) 135°, and (d) 8/5 with (h) 225°.

The angle δ_L in degree is calculated from the following equation (3.16).

$$\delta_L = \frac{360 \cdot N_R}{N_L} \tag{3.16}$$

where δ_L is the angle (degrees) between i^{th} leaf and $(i+1)^{th}$ leaf, N_R is the number of turns from the first leaf to n^{th} leaf in the vertical projection, and N_L is the number of leaves in N_R rounds.

We can also calculate the angle of $(i+1)^{th}$ leaf by the following equation.

$$\delta_l(i+1) = \delta_l(i) + \delta_L \tag{3.17}$$

where $\delta_l(i+1)$ is the accumulative angle from the 1^{st} leaf to $(i+1)^{th}$ leaf, and $\delta_l(1) = 0$.

Leaf arrangement plays an important role for getting enough light for photosynthesis process. Different kinds of plants have their own fraction value and angle of phyllotaxis. The next section explains an arrangement of flowers and fruits.

3.4.2 Flower and Fruit Arrangement

For flowering plants, flower identifies the reproductive stage after vegetative growth stage under the suitable condition. Bud at plant node can be initiated a new vegetative branch, root system, or flower depending on the condition. Flowers are formed from the bud position which is determined to initiate as a flower. The bud position is also located at the leaf position. So, in this experiment we implement the emergence flowers at the bud positions. The growth of leaves, flowers and fruits are explained in the next Section 3.4.3.

3.4.3 Leaf and Flower Growth

In this simulation, leaf vein structure is used as a texture mapping on the 3D leaf surface for the leaf growth in the *PlantVR* software. The leaf and flower growths are based on the sigmoidal growth function (see equation (2.17)).

3.5 Simulation and Results

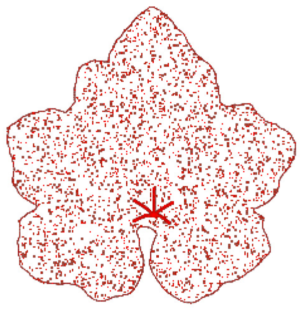
This section illustrates simulated results obtained from the proposed algorithms. An example of some results shows some simulated leaf veins, plant structures of shoot and root part using PTM.

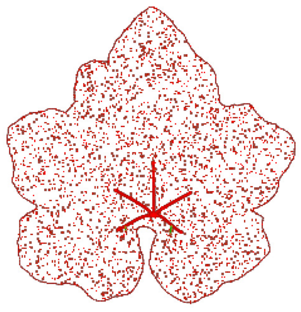
3.5.1 Results of Simulated Leaves

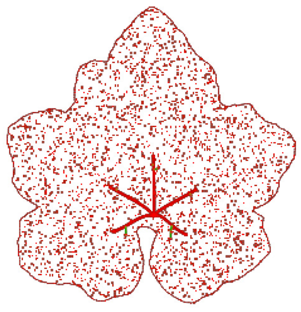
This section presents some simulated leaves obtained in our improvement from [Rodkaew, 2004]. Figure 3.32 shows examples of simulated growing leaf veins starting from $t = 20$ to $t = 130$. The leaf margin is fixed for all iterations. The product particles are randomly allocated in the leaf blade for every time step. The vein structure is initiated from the source point P_0 and growing until the vein density is bounded as the stop condition. The vein structure is labelled for three colors, e.g. red for the first vein order (main vein), green for the second vein order, and blue for the third vein order.

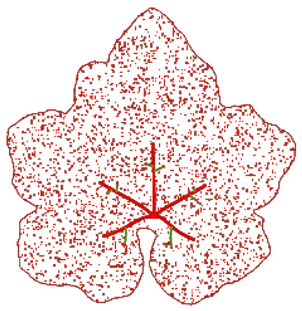
Figure 3.33 illustrates the example of simulated leaf veins by increasing a number of particles, (a) 100 particles, (b) 200 particles, (c) 400 particles, (d) 800 particles, (e) 1600 particles, and (f) 3200 particles. High number of particles gives a better structure, but it costs the simulation times than small number of particles. The red points in the leaf blade show product particles from photosynthesis process.

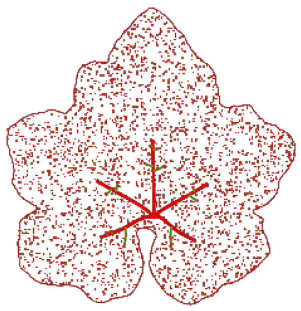
In order to compare some results from real leaf veins and simulated veins, we simulate the vein structures and compare the main important veins. The comparison of synthetic leaf veins and real veins over leaf textures is given in Figure 3.34. The main veins (red branches) in Figure 3.34 (left) are closely to the real veins. The second order to vein structures (green branches) also returns an acceptable results. The vein resolution parameter plays an important role to the leaf vein.

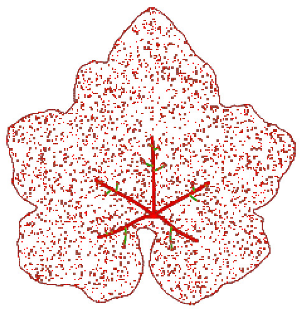

(a) time t=20

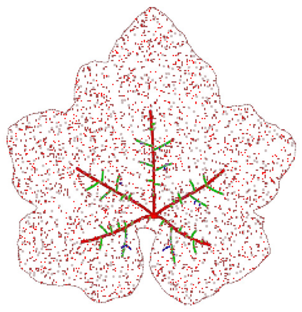

(b) time t=50

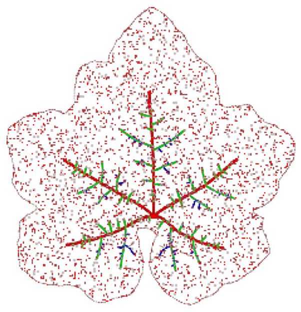

(c) time t=60

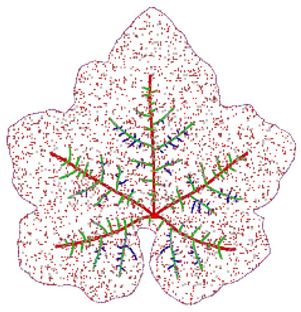

(d) time t=80

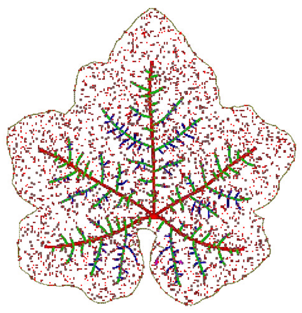

(e) time t=85

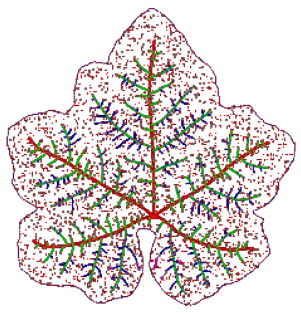

(f) time t=90


(g) time t=95


(h) time t=100


(i) time t=105


(j) time t=110


(k) time t=120

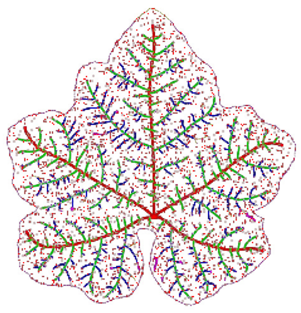
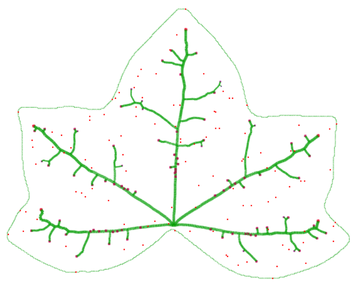
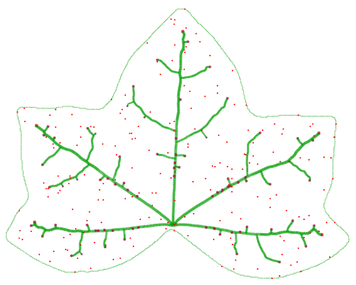
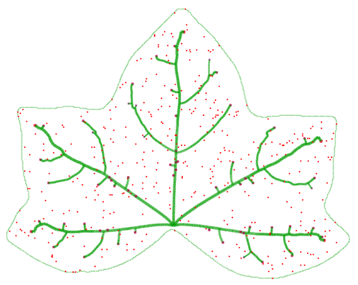
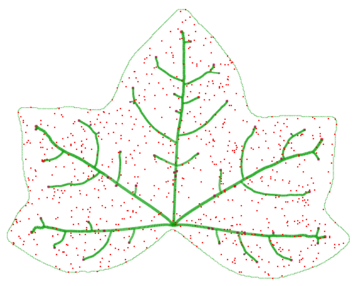

(l) time t=130

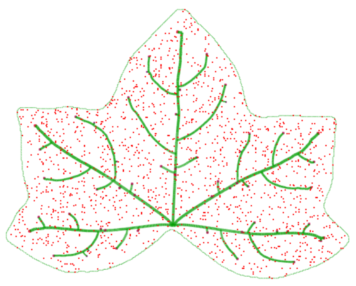
Figure 3.32: Example of simulated growing leaf veins, at time (a) t=20, (b) t=50, (c) t=60, (d) t=80, (e) t=85, (f) t=90, (g) t=95, (h) t=100, (i) t=105, (j) t=110, (k) t=120, and (l) t=130.


(a) 100 particles


(b) 200 particles


(c) 400 particles


(d) 800 particles


(e) 1600 particles

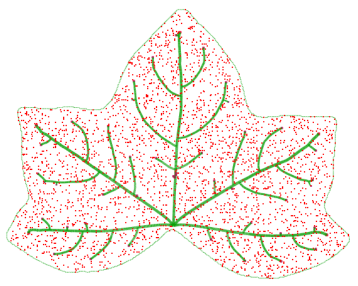

(f) 3200 particles

Figure 3.33: Example of simulated leaf veins with changing number of particles, (a) 100 particles, (b) 200 particles, (c) 400 particles, (d) 800 particles, (e) 1600 particles, and (f) 3200 particles.

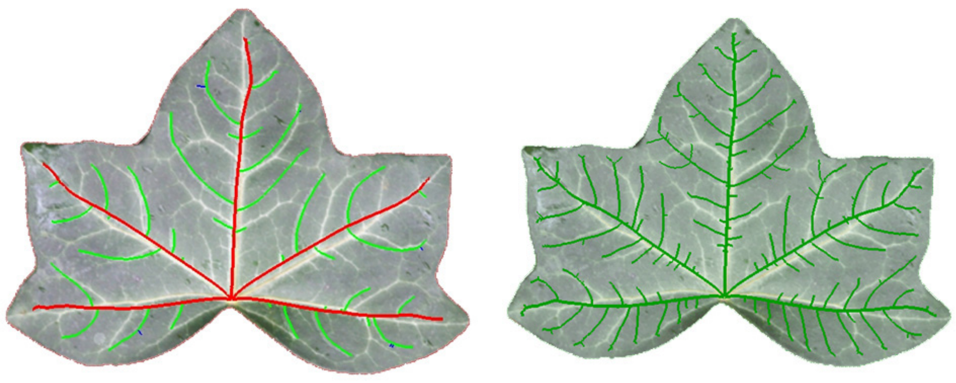


Figure 3.34: Comparison of synthetic leaf veins and real veins in leaf textures.

Figure 3.35 presents the comparison of real leaves (1st row), Rodkaew's leaves (2nd row) using the Algorithm 3.2 (*PTM-II*) in [Rodkaew, 2004] and using the Algorithm 3.4 (*PTM-LeafVein*) (3rd row). The simulated vein structures are acceptable and improved from the original result in [Rodkaew, 2004].

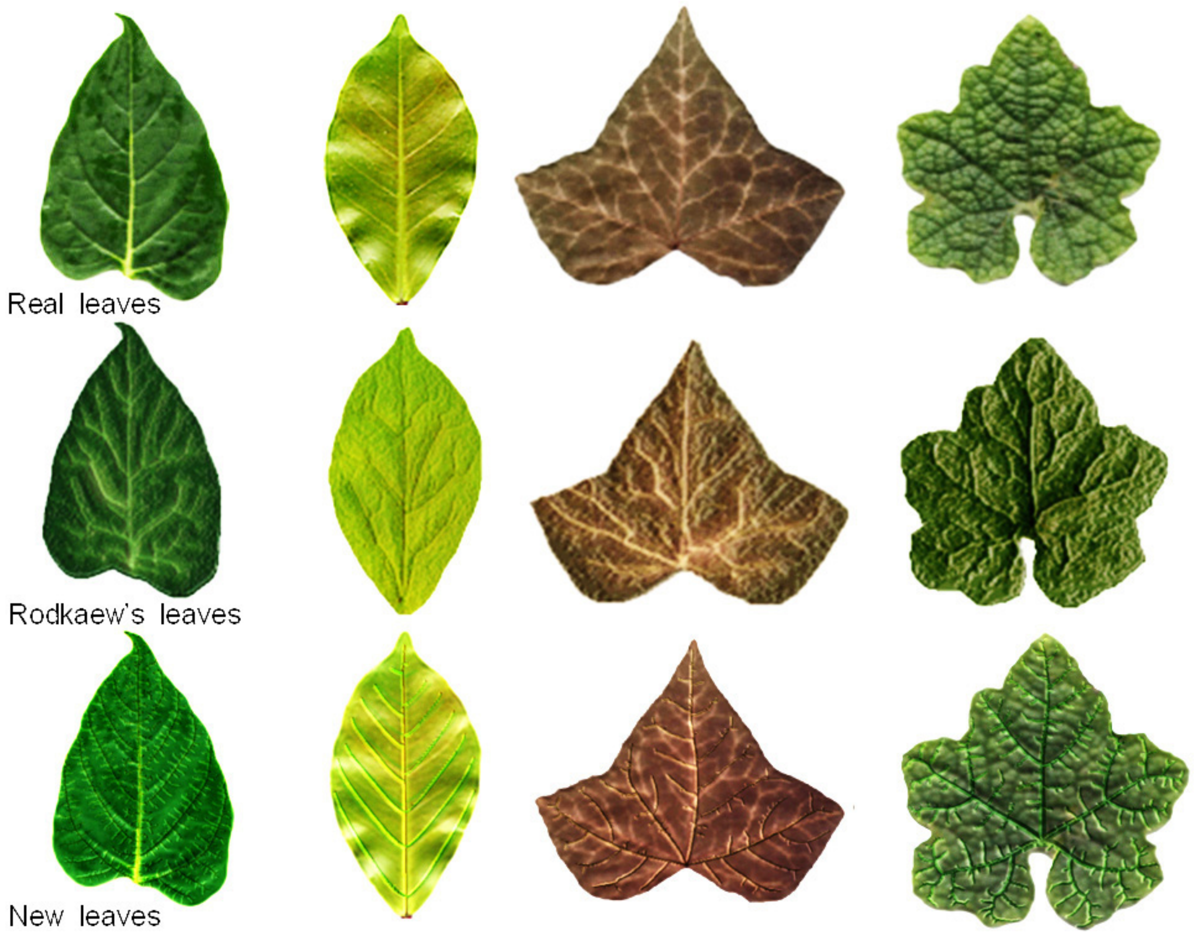


Figure 3.35: Comparison of real leaves (1st row), Rodkaew's leaves (2nd row) and the Algorithm 3.4 (3rd row).

Another example is the comparison of real leaves (1st column), leaf veins (2nd column), and vein with texture (3rd column) using the Algorithm 3.4 (*PTM-LeafVein*) (see Figure 3.36). We also adjust the parameters, e.g. vein resolution value, growth rate of first and second vein orders. Several examples of adjusting these parameters of mango leaf margin are shown in Figure 3.37.

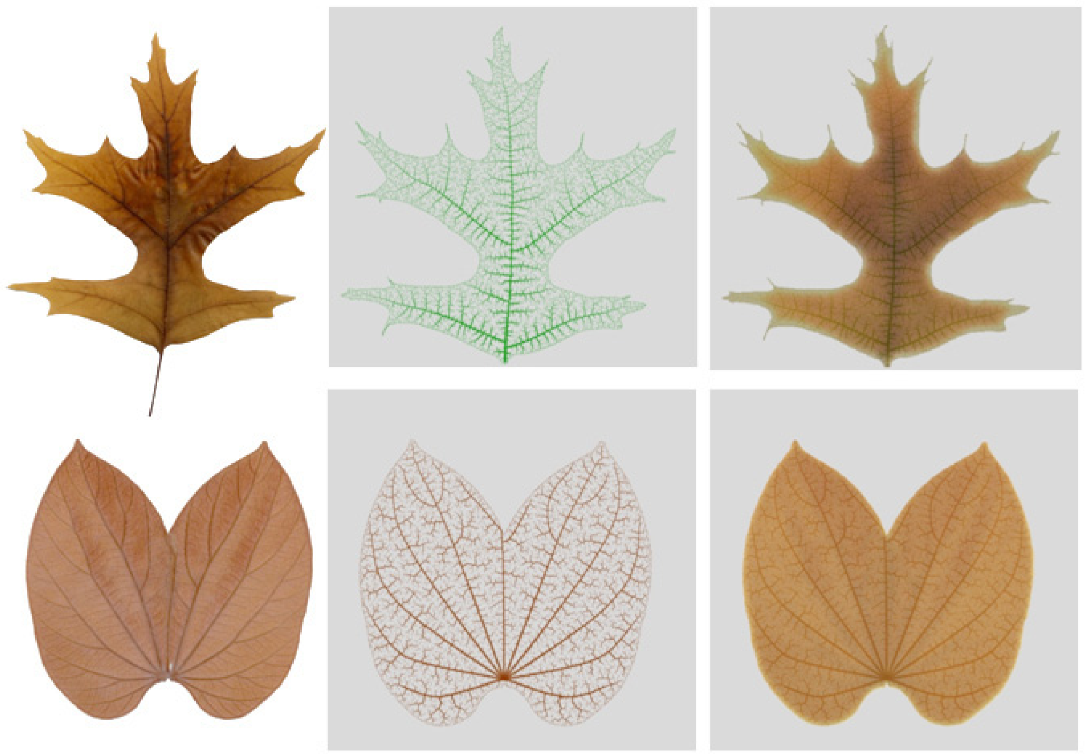


Figure 3.36: Comparison of real leaves (1st column), leaf veins (2nd column), and vein with texture (3rd column).

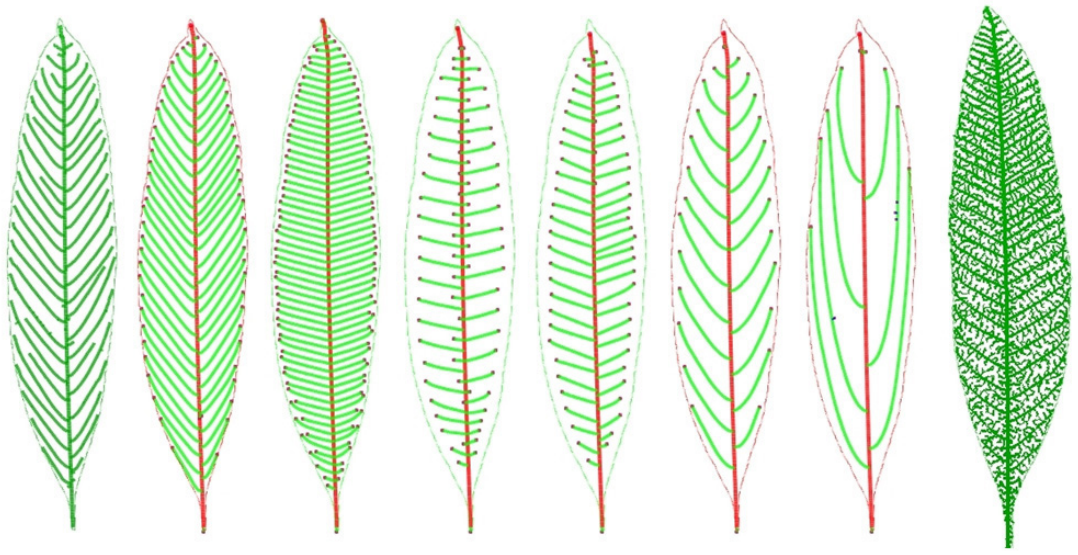
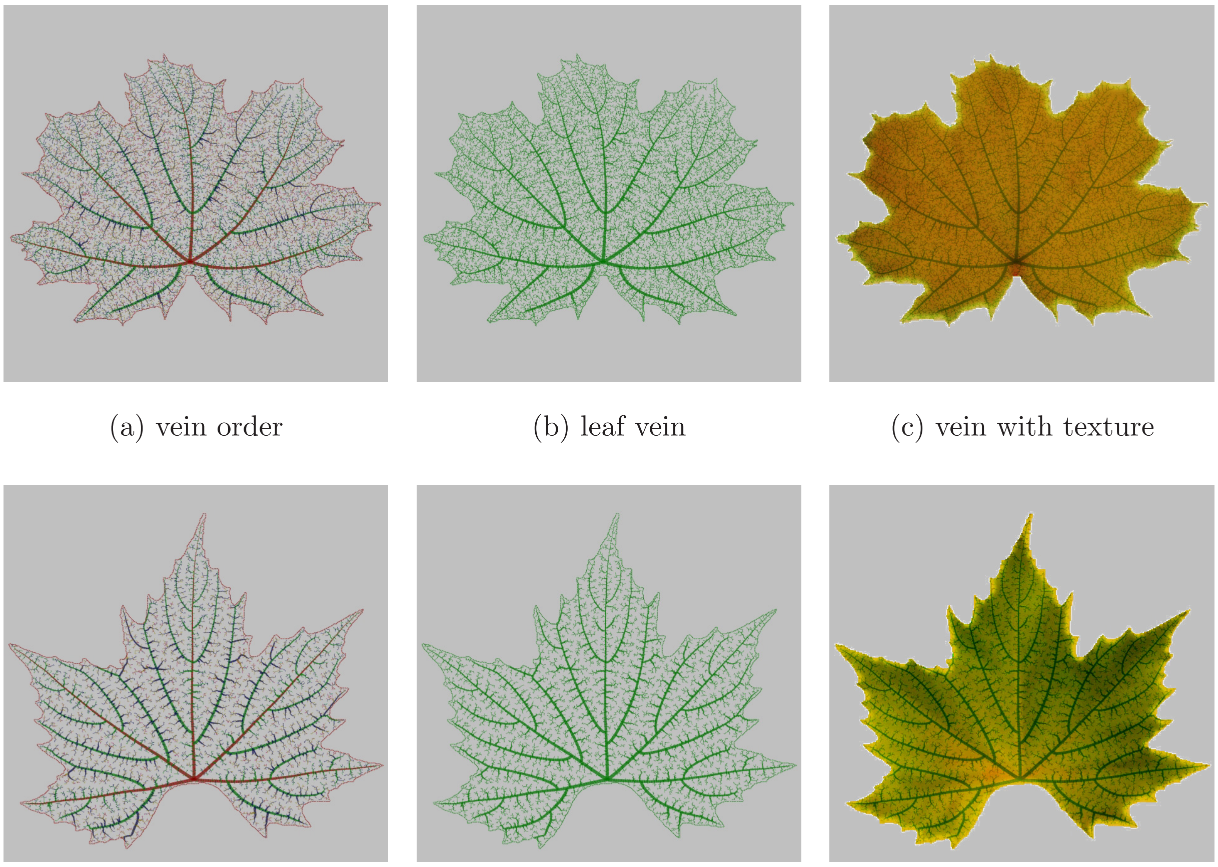


Figure 3.37: Comparison of simulated veins in mango leaf margin.

We now show the example of simulated leaf veins with vein orders in Figure 3.38 (a, d), leaf veins (b, e), and leaf veins with textures (c, f).



(a) vein order (b) leaf vein (c) vein with texture

(d) vein order (e) leaf vein (f) vein with texture

Figure 3.38: Example of simulated leaf veins with showing vein orders (a, d), leaf veins (b, e), and leafs vein with textures (c, f)

The simulation provides some example results of upper leaf surface and lower leaf surface. The examples of this result are given in Figure 3.39, Figure 3.40, Figure 3.41, and Figure 3.42.

Figure 3.43 shows an example of real rubber leaf (left) and simulated leaf (right). The higher detail of rubber vein structure is given in Figure 3.44.

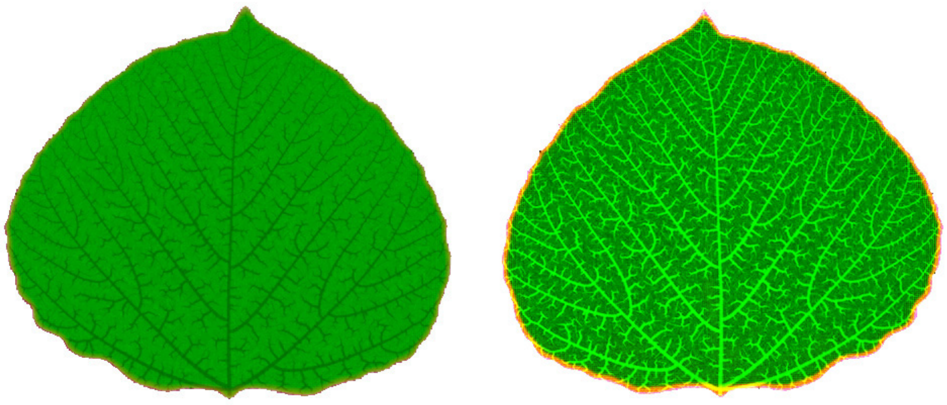


Figure 3.39: Result of upper leaf surface (left) and lower leaf surface (right).

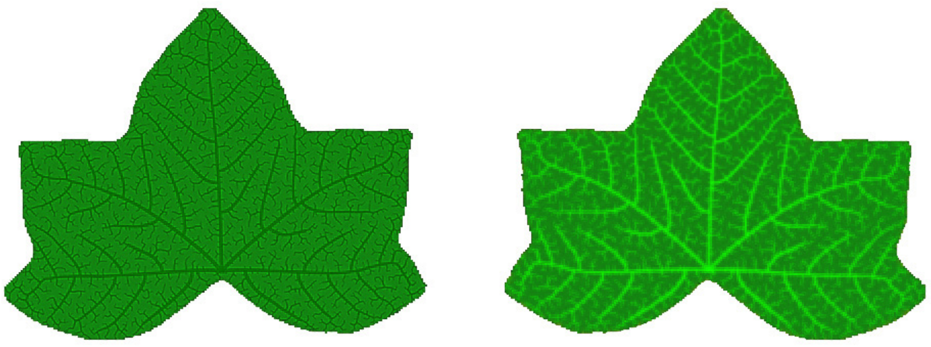


Figure 3.40: Result of upper leaf surface (left) and lower leaf surface (right).

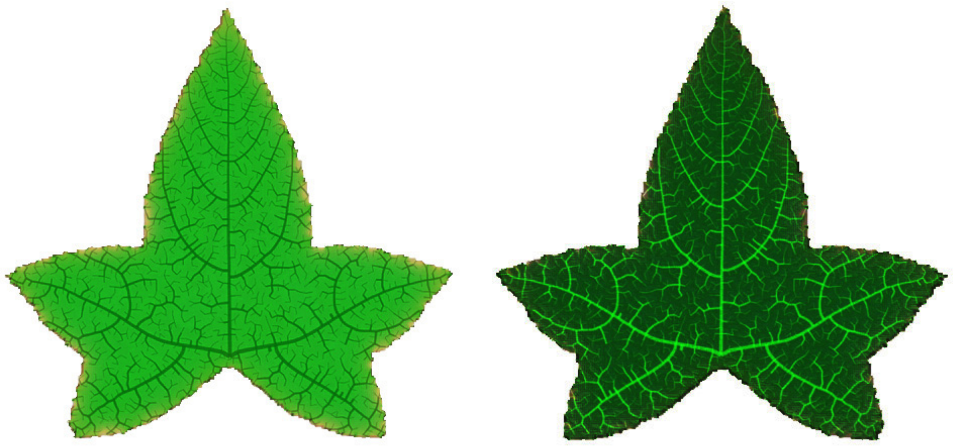


Figure 3.41: Result of upper leaf surface (left) and lower leaf surface (right).

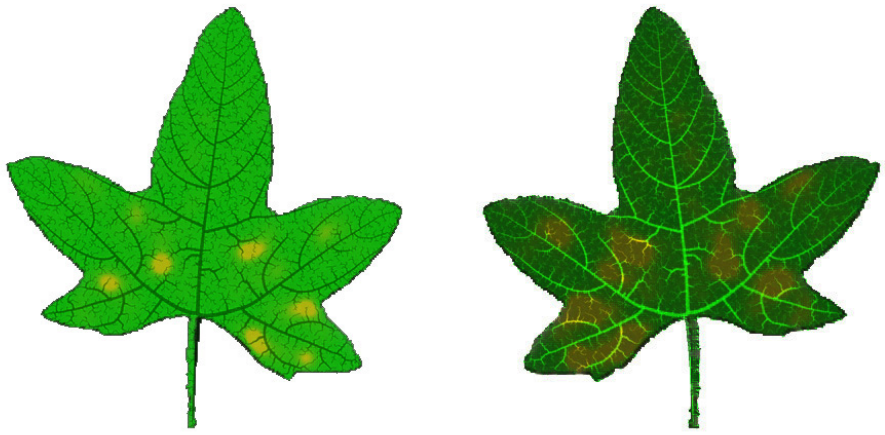


Figure 3.42: Result of upper leaf surface (left) and lower leaf surface (right).

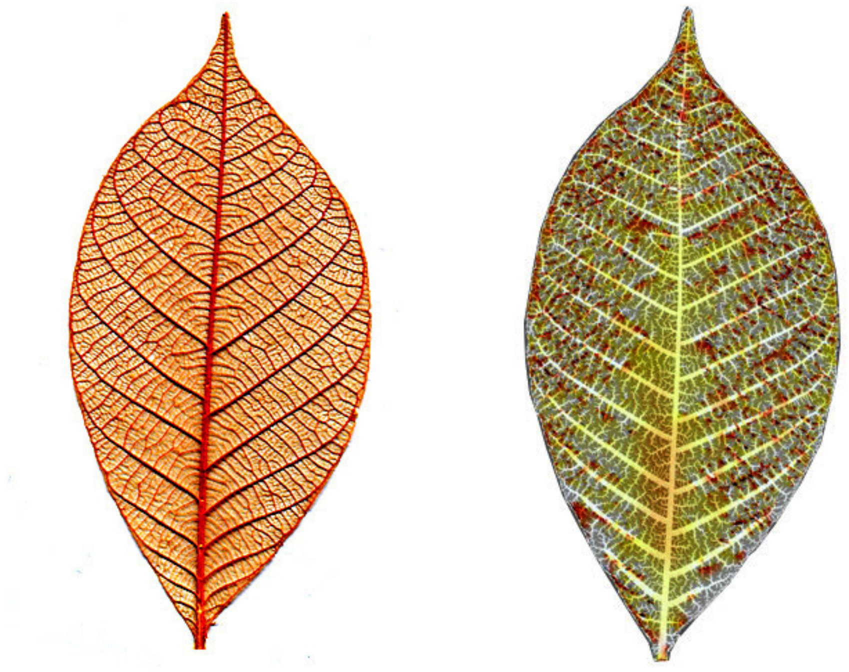


Figure 3.43: An example of real rubber leaf (left) and simulated leaf (right).

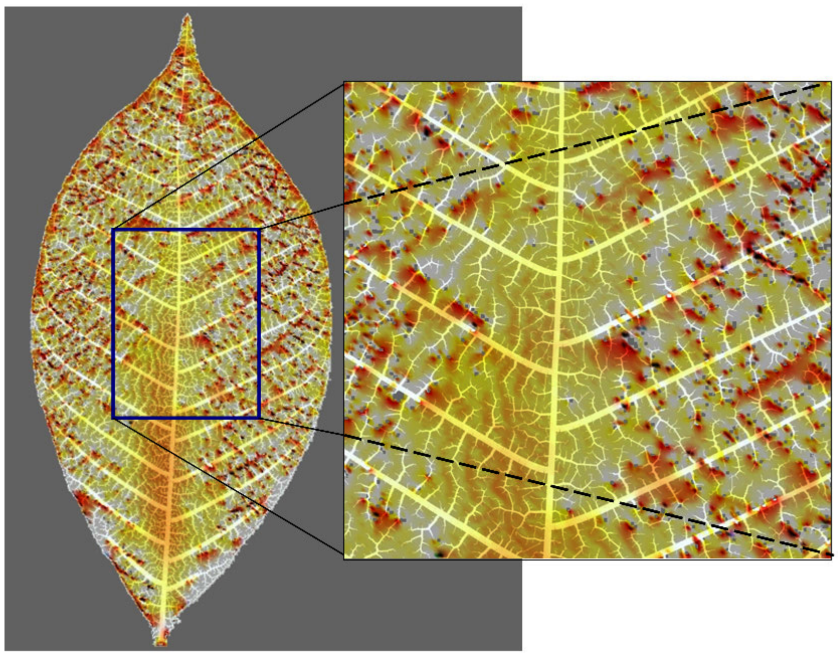


Figure 3.44: Simulated rubber leaf with its zoom-in for vein structure.

We now illustrates the comparison of real rubber leaf vein and its simulated results in Figure 3.45 (a) real rubber leaf, (b) simulated vein, and (c) simulated vein with texture. This simulation allows the closed vein structure. The algorithm provides the combination of two nearest tips when they are close to each other by an appropriate angle.

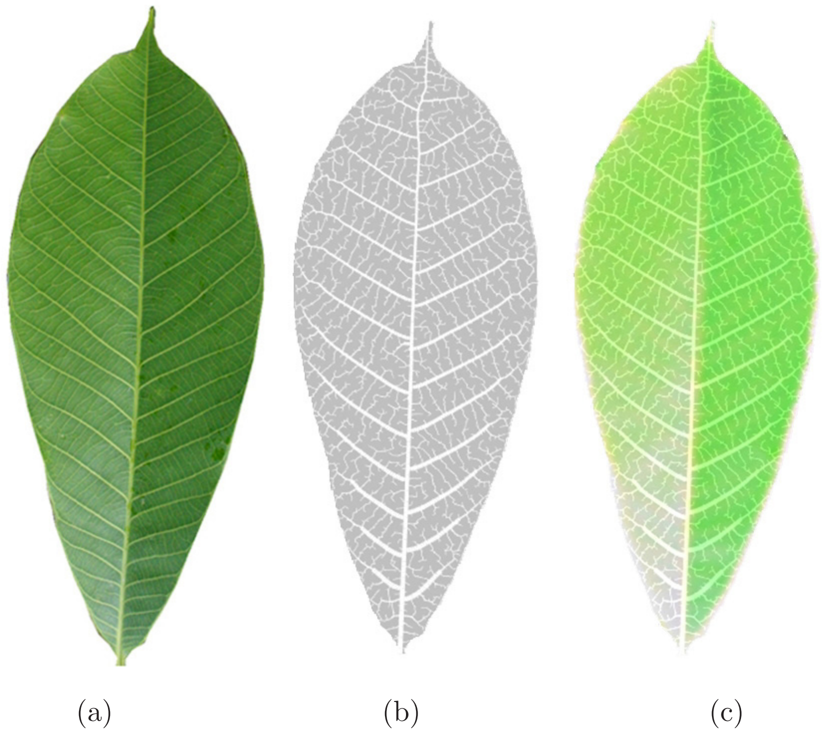


Figure 3.45: Rubber leaf and its simulated leaves, (a) real rubber leaf, (b) simulated vein, and (c) simulated vein with texture.

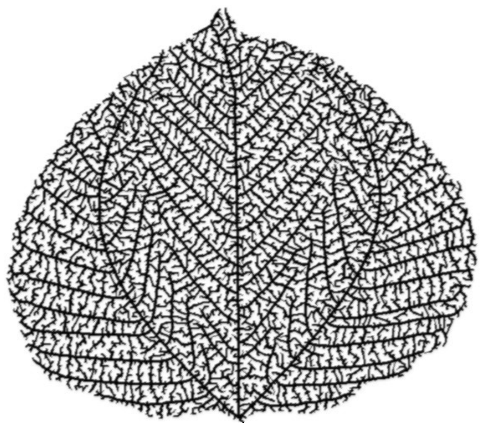


Figure 3.46: Simulated vein in poplar leaf boundary.

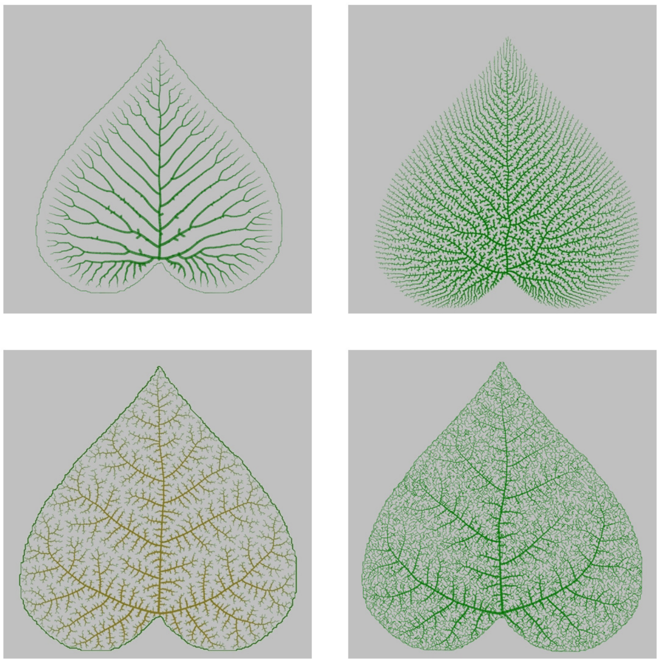


Figure 3.47: Some different simulated veins in heart leaf boundary.

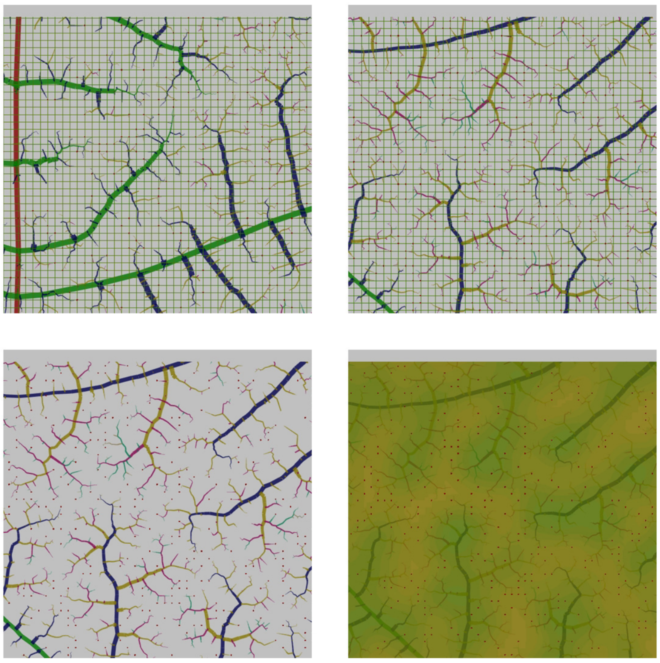


Figure 3.48: Zoom in detail of leaf vein construction with particles and leaf texture.

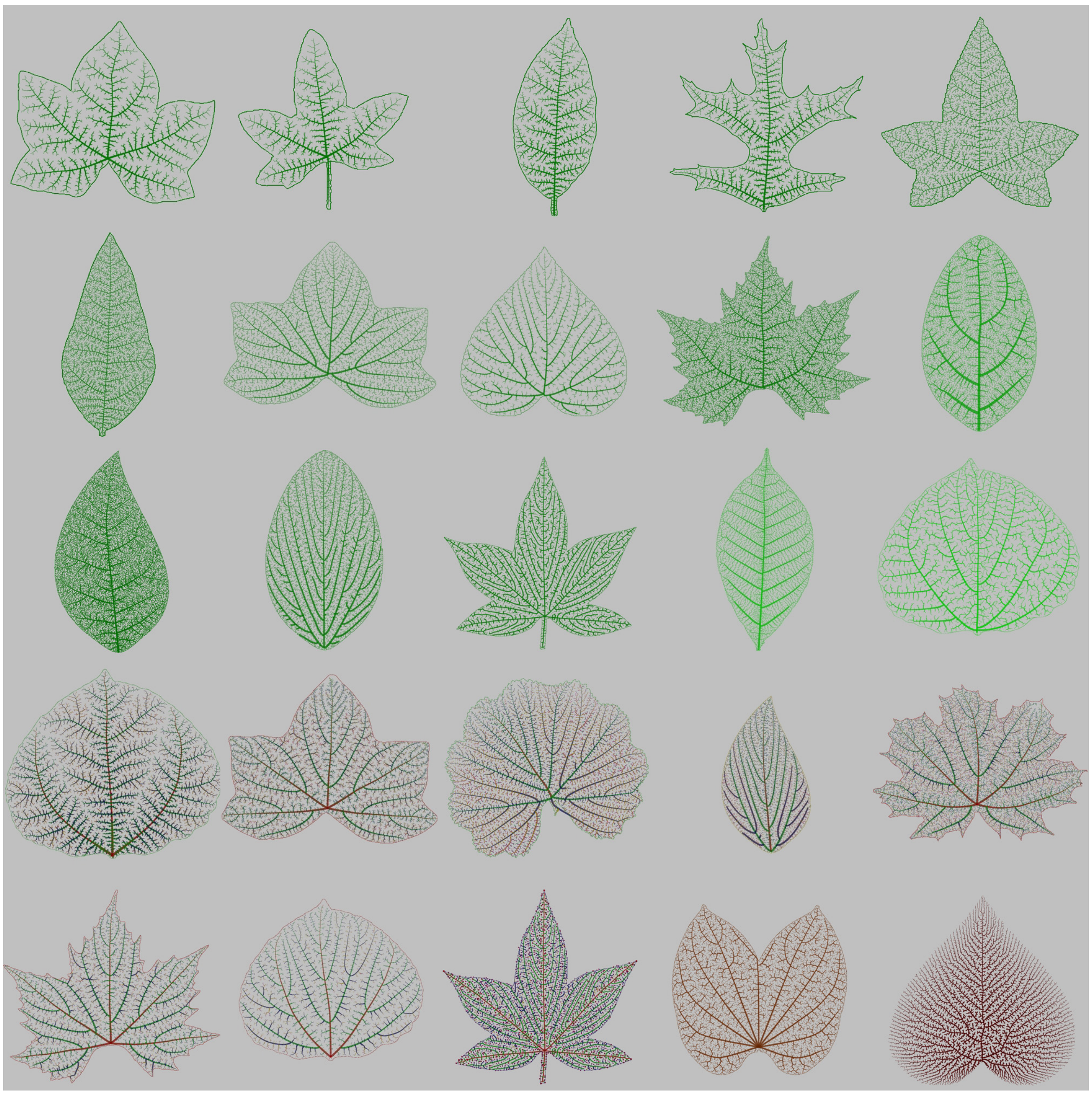


Figure 3.49: Simulated veins in a set of given leaf boundaries.

Since the algorithm 3.4 provides the closed vein property, an example of simulated vein in poplar leaf boundary is presented in Figure 3.46. In the same given leaf boundary, it is possible to generate different veins based on the control parameter of the distribution of particles in leaf blade (see Figure 3.47). We can zoom into detail of leaf network in the leaf blade to see the tip growth, available auxin particles, and change of leaf texture (see Figure 3.48). Finally, we illustrate some examples of leaf veins generated by the Algorithm 3.4 (*PTM-LeafVein*).

Next section, we present some results of simulated plants and trees from the Algorithm 3.5 (*PTM-Tree*).

3.5.2 Results of Simulated Plants/Trees

This section shows some simulated results using PTM for generating plant/tree structures. There are some results of plant structure with different types of initial seeds, e.g. a single plant and two plants.

We do simulate two plants which are growing close to each other. They are growing with sharing the volume space and light competition between both plants. The result of this experiment is shown in Figure 3.50. The interaction of competing light and volume space plays a major role to the number of branches with low branch density.

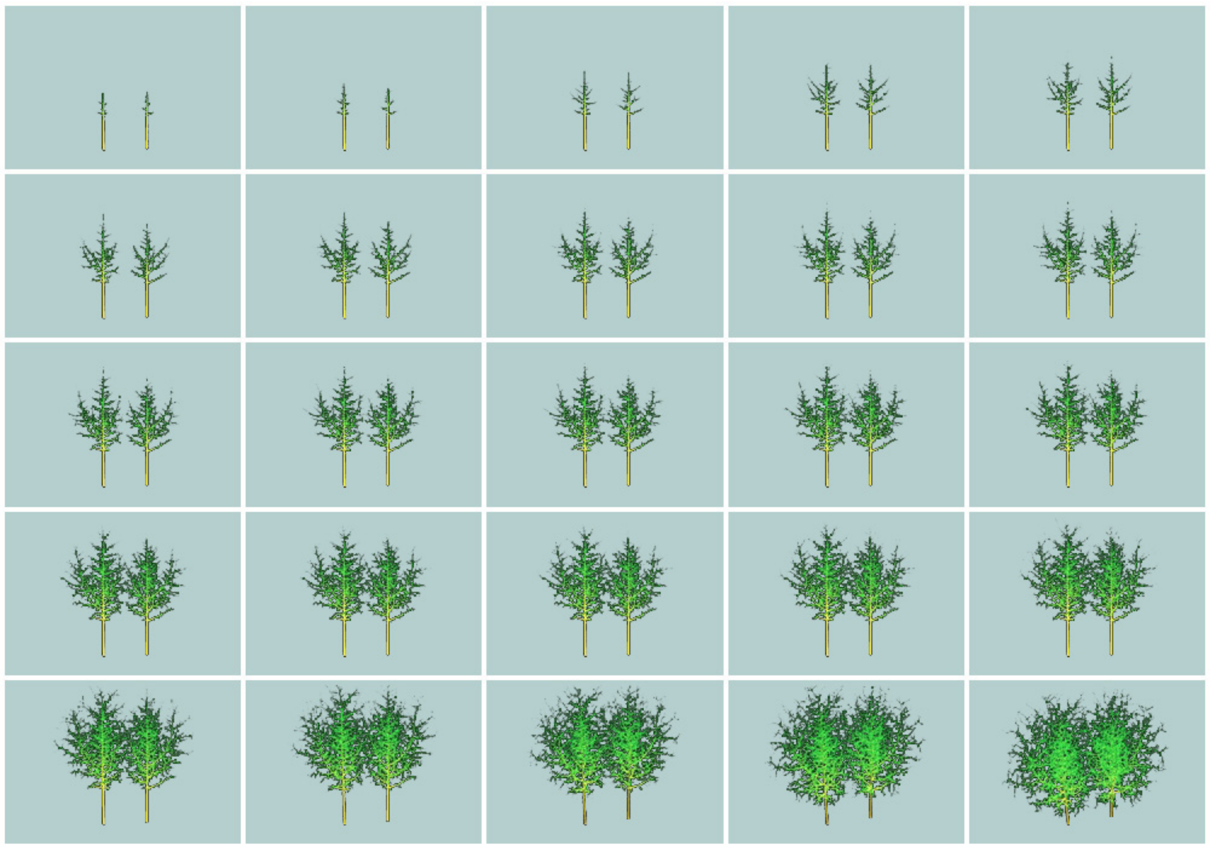


Figure 3.50: Result of two growing trees with single main stem.

We do also simulate the same procedure but we allow one tree can grow with more than one stem. The result is illustrated in Figure 3.51.

In case of defining the volume of shoot and root boundary by shifting them, we would get the different result. The environmental factors; water content and nutrient, are not available or even the plant are grown in the box of concrete that roots cannot grow to that direction. This implementation is visualized in the Figure 3.52. In Figure 3.52, plant was grown in the given environment. Roots grew under the limitation of available nutrient and water in the cubic box while shoot grew in the natural way of spherical shape.

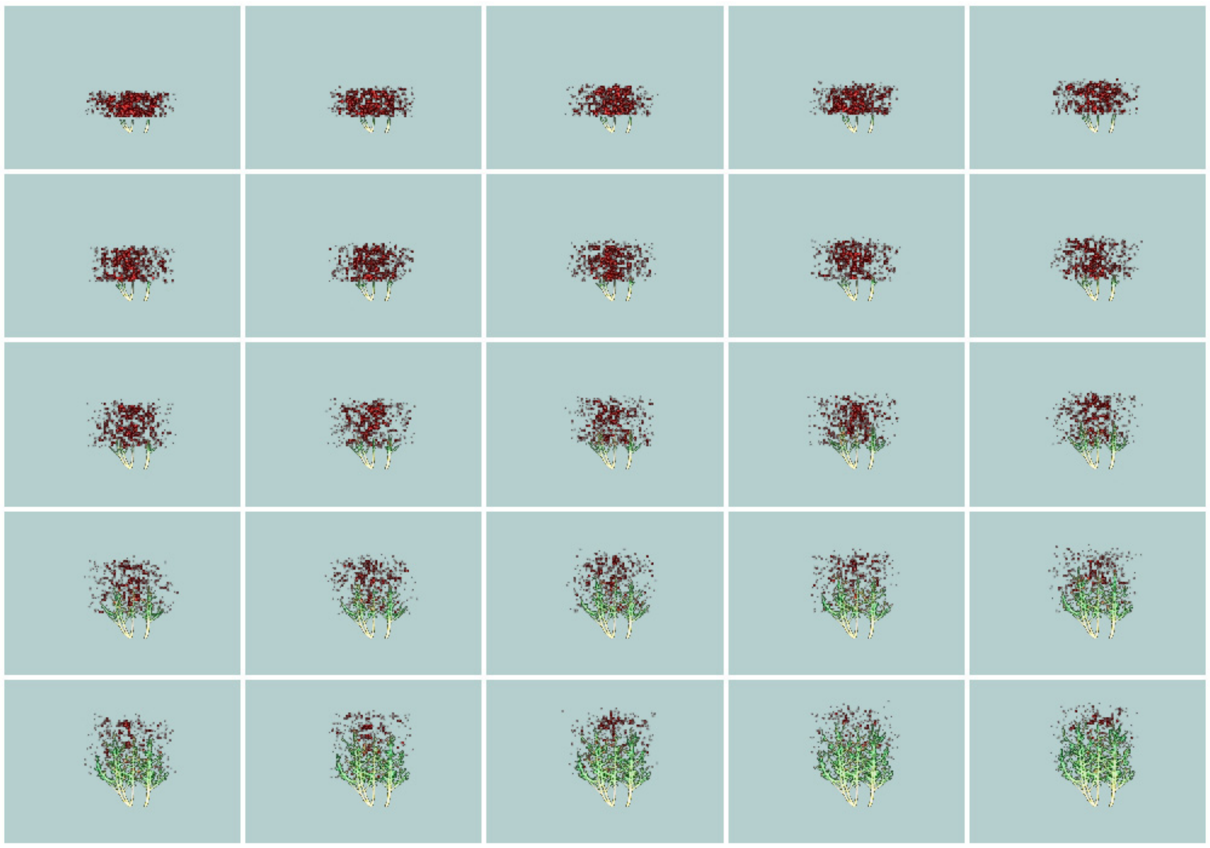


Figure 3.51: Result of two growing trees with multiple main stems and a single main stem.

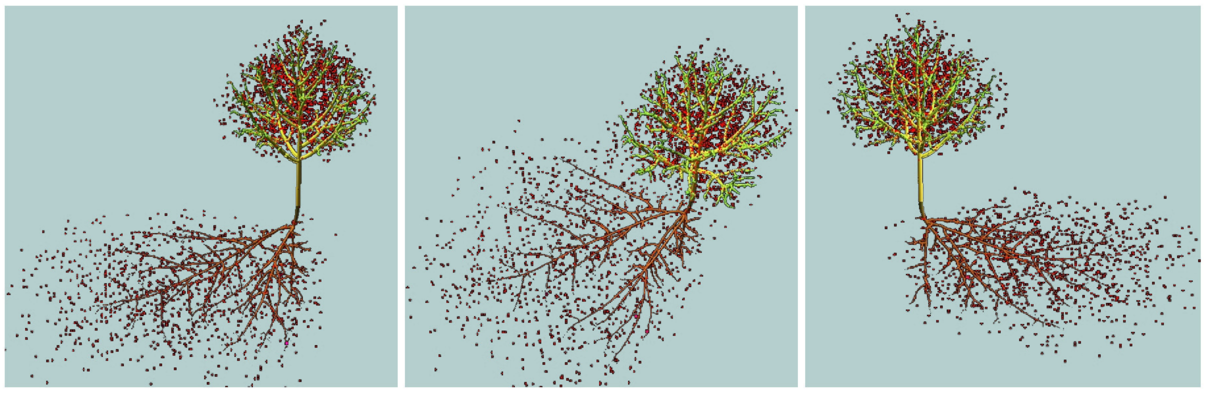


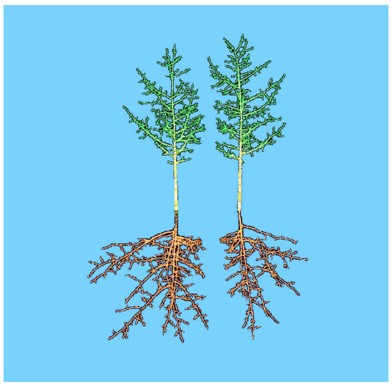
Figure 3.52: Different perspectives of tree structure with environmental limitation of soil volume.

(a) time t=80 days

(b) time t=100 days

(c) time t=110 days

(d) time t=120 days


(e) time t=130 days

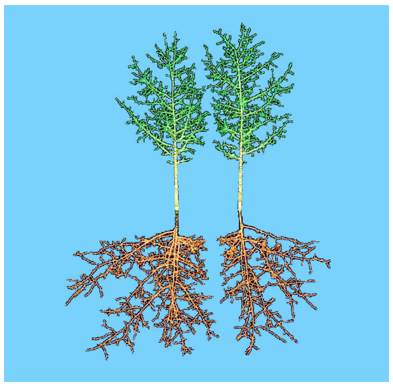
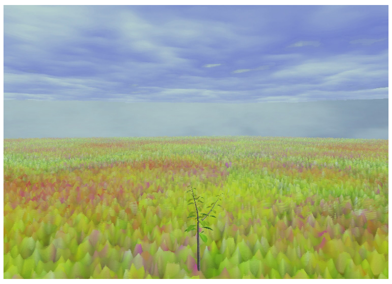
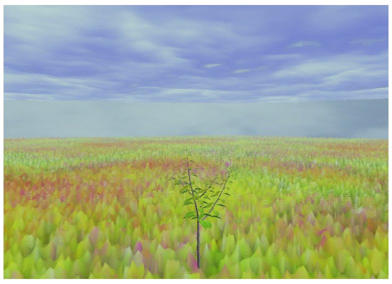
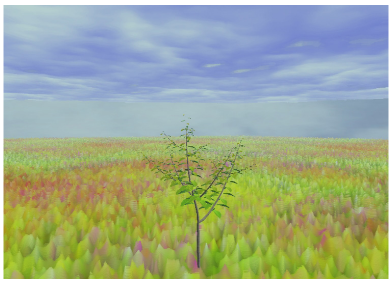

(f) time t=140 days

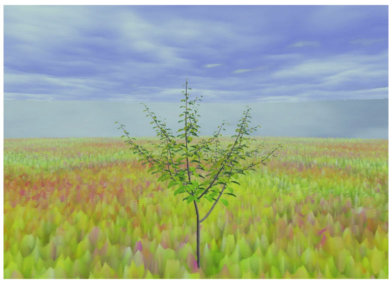
Figure 3.53: Example of simulated growing trees, at time (a) t=80, (b) t=100 days, (c) t=110 days, (d) t=120 days, (e) t=130 days, and (f) t=140 days.

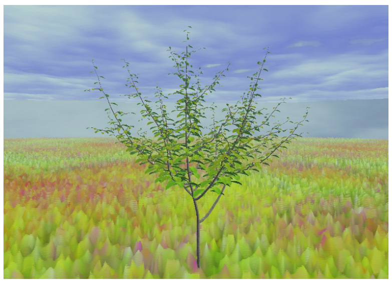
We now simulate the two trees grown close to each other again in both shoot and root parts. Figure 3.53 shows the example of simulated growing trees from $t = 80$ to $t = 140$, (a) time $t = 80$ days, (b) time $t = 100$ days, (c) time $t = 110$ days, (d) time $t = 120$ days, (e) time $t = 130$ days, and (f) time $t = 140$ days. The interactions between two shoot and two root systems are optimized under the available volume space and environmental factors.

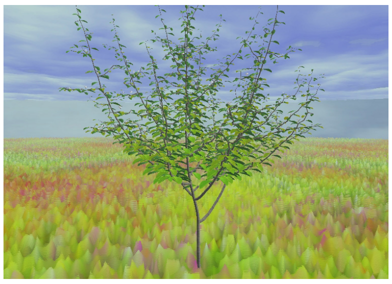

(a) time t=42

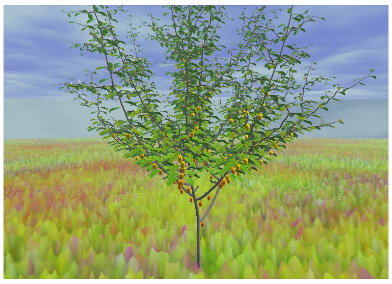

(b) time t=54

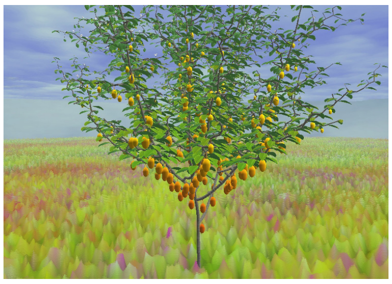

(c) time t=66


(d) time t=78


(e) time t=90


(f) time t=102


(g) time t=114


(h) time t=126

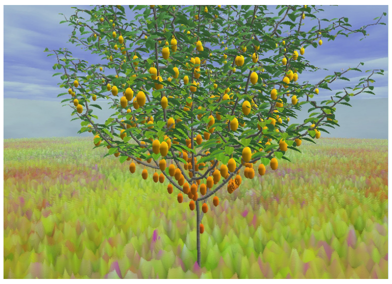

(i) time t=138

Figure 3.54: Example of simulated growing tree, at time (a) t=42, (b) t=54, (c) t=66, (d) t=78, (e) t=90, (f) t=102, (g) t=114, (h) t=126, and (i) t=138.

Figure 3.54 shows the simulation scene of plant growth with its leaves, flower, fruits, and environment; at time (a) t=42, (b) t=54, (c) t=66, (d) t=78, (e) t=90, (f) t=102, (g) t=114, (h) t=126, and (i) t=138.

In order to initialize the shoot and root volume in the easier way, we provide the curve with control points to define the shoot volume (see curve in Figure 3.55). All particles are controlled by the control curve to random in the given volume for branching structure. Figure 3.55 shows the same result with different views.

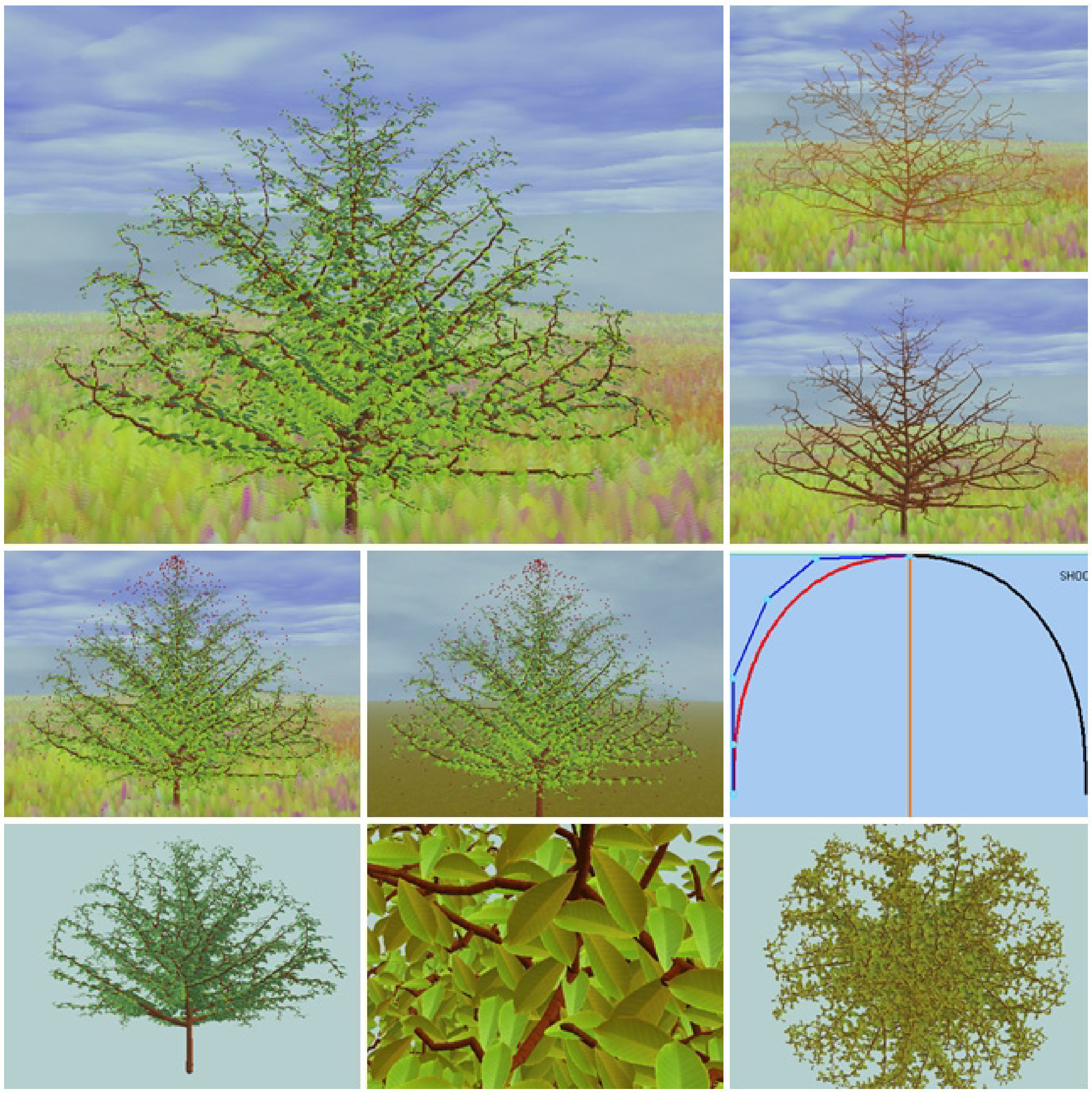


Figure 3.55: Result from the Algorithm 3.5 (PTM-Tree) in *PlantVR* using the control spline curve for shoot volume.

Figure 3.56 presents the simulated plant in both shoot and root parts using the control curve for initiating the given volumes.

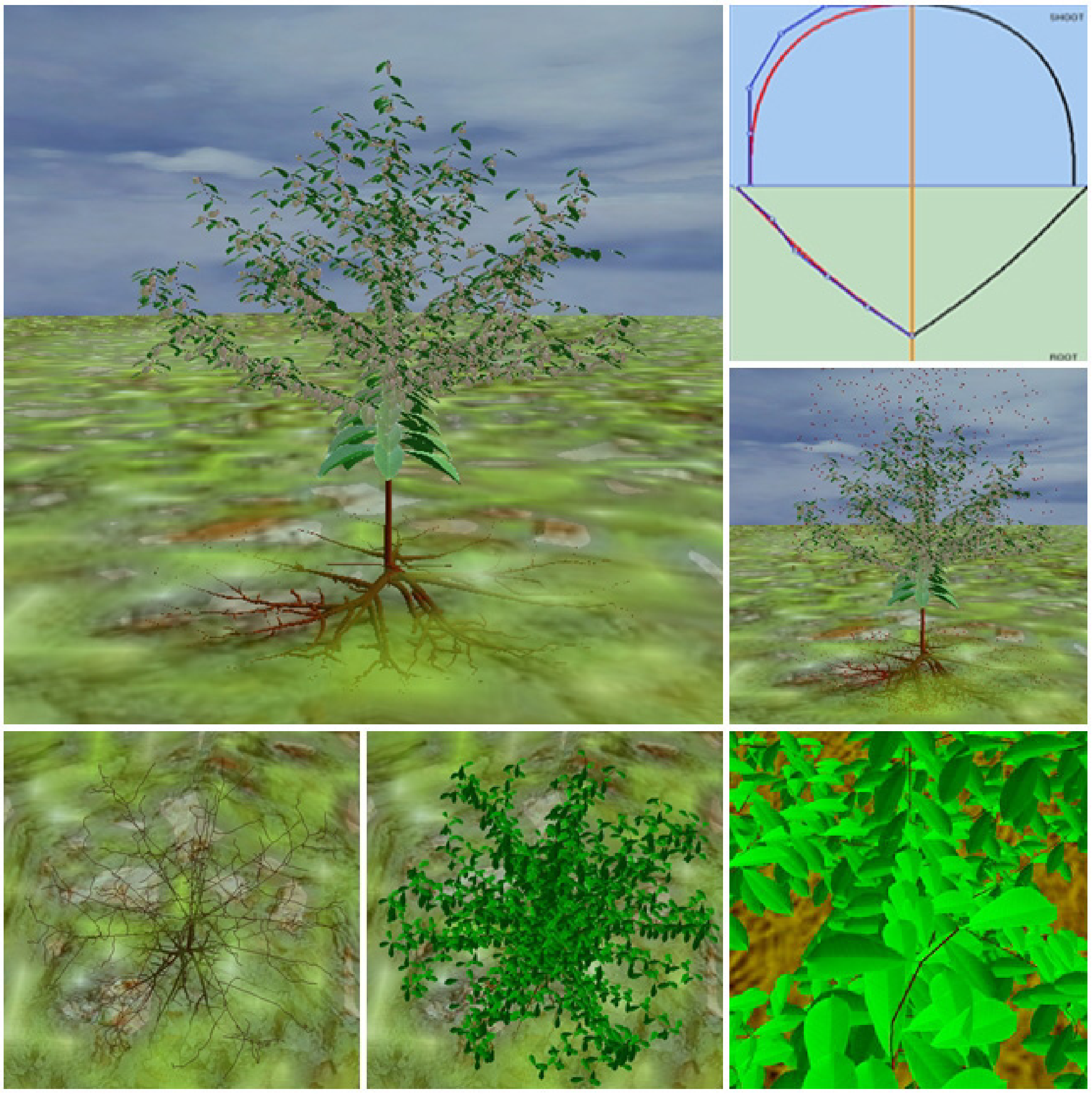


Figure 3.56: Result from the Algorithm 3.5 (PTM-Tree) in *PlantVR* using the control spline curves for shoot and root volume.

3.6 Discussion and Conclusion

This chapter introduced the algorithms for generating leaf vein structures which can also be applied for shoot and root structures from the given shape boundary and the distribution of environmental particles; i.e. auxin distribution or the product from photosynthesis process in leaf blade, light particles for shoot growth, water-nutrient particles for root growth. We improved the Algorithm 3.1 (*PTM-I*) and Algorithm 3.2 (*PTM-II*) and PTM to generate tree structures in a natural way of plant growth (see [Rodkaew, 2004]). The algorithm provides the possibility to grow more than one plant to observe the management of available volume space and environmental resources. The algorithm was implemented in the *PlantVR* software and useful for dicotyledon plants. The parameter of vein resolution is controlled in time for different resolutions at each time step for a dynamic vein adjustment. Leaf vein structure has been mapped on the leaf object as a texture mapping for fast visualization in *PlantVR*. The method will be improved to calculate the water-nutrient transportation from soil to stems and leaves for photosynthesis calculation.

Chapter 4

Root Systems, Water Flow and Nutrient Diffusion

"Look deep into nature,
and then you will understand everything better."
— Albert Einstein (1879 - 1955)

Plant roots are the most important part for absorbing nutrients and water to the main stems and leaves. Roots grow in the path of minor resistance and they extend in the porous spaces of soil. The roots change their directions when they find obstacles that are aggregates of resistant soil. Besides, following the path of minor resistance, roots also grow where the media is better. The roots can avoid the dry and arid soil while they grow in the moist and fertile soil. In this chapter, we are interested in how the roots grow in the soil volume based on nutrient concentration and water content. We assume that root tips grow to the direction of the maximum nutrient concentration, find the best direction and elongate to that direction. During the roots penetrate in soil volume, nutrient concentration is changed by diffusion process. The roots can also uptake nutrient to root stems when the roots reach the local soil volume. At each time step, the nutrient uptake by root can be calculated. The visualization of root growth is presented in this chapter.

4.1 Introduction

The root systems of terrestrial plants perform two primary functions: One is the acquisition of soil-based resources (principally water and dissolved ions or nutrients) and the other is anchorage. Other root system functions such as storage, synthesis of growth regulators, propagation, and dispersal, can be seen as secondary in [Smith et al., 2000]. In Chapter 2 and previous work in [Chuai-Aree et al., 2000], [Chuai-Aree, 2000], [Chuai-Aree et al., 2002], [Chuai-Aree et al., 2003], [Rodkeaw et al., 2004] we use Lindenmayer systems (L-systems) in [Prusinkiewicz and Lindenmayer, 1990] to represent the plant shoot and root structures which are known as a set of production rules of shoot and root parts known in advance. In this chapter we propose the method to represent the root structures using nutrient concentration as an activator of root growth. This method does not need any rule for representing the root structure and root growth. The position of root apices will move to the appropriate nutrient concentration.

The next section describes the water flow in soil volume and mathematical modeling of water uptake by plant roots.

4.2 Water Flow

This section describes the water flow from the flux of water and water root extraction in soil volume. We start from a given soil volume as follows:

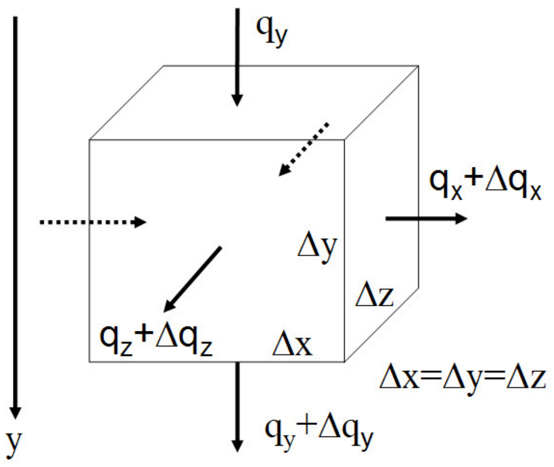


Figure 4.1: Cubic soil with three directions of water flow.

Now suppose that we have box of soil, Δx by Δy by Δz in size (cm), that is partially saturated with volumetric water content, θ $(cm^3\ cm^{-3})$. The volume of the box is $Vt = \Delta x \cdot \Delta y \cdot \Delta z$ (cm^3). The volume of water in the box is $Vw = \theta \cdot Vt = \theta \cdot \Delta x \cdot \Delta y \cdot \Delta z$ (cm^3) at the beginning of Δt. It is $Vw + \Delta Vw = (\theta + \Delta\theta) \cdot Vt$ at the end of Δt. We now consider the flow in the x-direction. We define the flow to be positive in the positive x-direction. The Δ terms are added at the higher end of the respective dimension, $\Delta\theta$ at the end of Δt, and Δq at the end of Δx. The flow is defined as the volumetric flux, q $(cm^3\ cm^{-2} \cdot s^{-1})$ or (cm/s). So that the volume rates of inflow and outflow over time Δt are:

$$\begin{aligned} V_{in} &= q \cdot \Delta y \cdot \Delta z \cdot t, \\ V_{out} &= (q + \Delta q) \cdot \Delta y \cdot \Delta z \cdot t. \end{aligned} \tag{4.1}$$

The volumes (masses) balance can be derived as follows:

$$\begin{aligned}
\Delta V_w &= V_{in} - V_{out}, \\
\Delta V_w &= \Delta\theta \cdot V_t = \Delta\theta \cdot \Delta x \cdot \Delta y \cdot \Delta z, \\
V_{in} - V_{out} &= q \cdot \Delta y \cdot \Delta z \cdot \Delta t - (q + \Delta q) \cdot \Delta y \cdot \Delta z \cdot \Delta t, \\
\Delta\theta \cdot \Delta x \cdot \Delta y \cdot \Delta z &= -\Delta q \cdot \Delta y \cdot \Delta z \cdot \Delta t, \\
\frac{\Delta\theta}{\Delta t} &= -\frac{-\Delta q}{\Delta x}.
\end{aligned} \tag{4.2}$$

so, we now can write the mass balance equation for the flow in x-direction from discrete (Δ) to the continuous derivative (∂) by taking Δt and Δx both to the limit of zero. We finally get the following equation

$$\frac{\partial\theta}{\partial t} = -\frac{\partial q}{\partial x}. \tag{4.3}$$

For the flow q (cm/s), is a term of Darcy's law:

$$q = -K \cdot \frac{\partial H}{\partial x}. \tag{4.4}$$

We can substitute Darcy's law into the mass balance equation (4.3) to get the following equation a so-called one-dimensional Richards' equation:

$$\frac{\partial\theta}{\partial t} = \frac{\partial}{\partial x}\left[K \cdot \frac{\partial H}{\partial x}\right]. \tag{4.5}$$

We can derive the mass balance for flow in all three directions at once. The Darcy's equation can be written in three components of the flow, q_x, q_y and q_z as

$$\begin{aligned}
q_x &= i_x\left(K_{xx}\frac{\partial H}{\partial x} + K_{xy}\frac{\partial H}{\partial y} + K_{xz}\frac{\partial H}{\partial z}\right), \\
q_y &= i_y\left(K_{yx}\frac{\partial H}{\partial x} + K_{yy}\frac{\partial H}{\partial y} + K_{yz}\frac{\partial H}{\partial z}\right), \\
q_z &= i_z\left(K_{zx}\frac{\partial H}{\partial x} + K_{zy}\frac{\partial H}{\partial y} + K_{zz}\frac{\partial H}{\partial z}\right).
\end{aligned} \tag{4.6}$$

The K-component values are defined as follows:

$$\begin{aligned}
K_{xy} &= K_{yx}, \\
K_{xz} &= K_{zx}, \\
K_{yz} &= K_{zy},
\end{aligned} \tag{4.7}$$

where K_{xy} is hydraulic conductivity in xy-direction $(cm\ day^{-1})$, H is total head, and $H(x, y, z)$ (cm). In equation (4.6) we can also express in matrix form as:

$$\begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix} = -\begin{bmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{yx} & K_{yy} & K_{yz} \\ K_{zx} & K_{zy} & K_{zz} \end{bmatrix} . \begin{bmatrix} \dfrac{\partial H}{\partial x} \\ \dfrac{\partial H}{\partial y} \\ \dfrac{\partial H}{\partial z} \end{bmatrix} \tag{4.8}$$

So that, the full-blown Richards' equation can be written in matrix form as:

$$\begin{bmatrix} \dfrac{\partial \theta}{\partial t} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial}{\partial x} & \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial z} \end{bmatrix} . \begin{bmatrix} K_{xx} & K_{xy} & K_{xz} \\ K_{yx} & K_{yy} & K_{yz} \\ K_{zx} & K_{zy} & K_{zz} \end{bmatrix} . \begin{bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \\ \dfrac{\partial}{\partial z} \end{bmatrix} . \; H(x,y,z) \tag{4.9}$$

If the major axes of the soil anisotropy are aligned with the x, y and z axes, the equation (4.9) becomes

$$\begin{bmatrix} \dfrac{\partial \theta}{\partial t} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial}{\partial x} & \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial z} \end{bmatrix} . \begin{bmatrix} K_{xx} & 0 & 0 \\ 0 & K_{yy} & 0 \\ 0 & 0 & K_{zz} \end{bmatrix} . \begin{bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \\ \dfrac{\partial}{\partial z} \end{bmatrix} . \; H(x,y,z) \tag{4.10}$$

If the soil is isotropic, but it is still heterogeneous. The Richards' equation reduces to

$$\begin{bmatrix} \dfrac{\partial \theta}{\partial t} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial}{\partial x} & \dfrac{\partial}{\partial y} & \dfrac{\partial}{\partial z} \end{bmatrix} . \begin{bmatrix} K & 0 & 0 \\ 0 & K & 0 \\ 0 & 0 & K \end{bmatrix} . \begin{bmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \\ \dfrac{\partial}{\partial z} \end{bmatrix} . \; H(x,y,z) \tag{4.11}$$

where K is $K(x, y, z)$. This bring us back to

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial x}\left[K\frac{\partial H}{\partial x}\right] + \frac{\partial}{\partial y}\left[K\frac{\partial H}{\partial y}\right] + \frac{\partial}{\partial z}\left[K\frac{\partial H}{\partial z}\right] \tag{4.12}$$

The root system is a sink that penetrates in the soil profile. The combination of root growth system and water flow in soil solution is presented here as macroscopic way of solving the root water uptake problem. The continuity equation of water flow is combined with a sink term representing water extraction by plant root system.

The three-dimensional root water uptake model can be written as follows

$$\frac{\partial \theta}{\partial t} = -\left(\frac{\partial q}{\partial x} + \frac{\partial q}{\partial y} + \frac{\partial q}{\partial z}\right) - S \tag{4.13}$$

where θ is the soil water content ($cm^3\ cm^{-3}$), t is time ($days$), y is the vertical coordinate (cm) take positive upward, q is the Darcian soil water flux density ($cm\ day^{-1}$) taken positive upward, and S is the actual root water uptake rate ($cm^3\ cm^{-3}\ day^{-1}$).

Combination of equation (4.13) and equation (4.6) is so-called as Richards' equation:

$$\frac{\partial \theta}{\partial t} = -C(h)\frac{\partial h}{\partial t} = \frac{\partial \left[K(h)(\frac{\partial h}{\partial y} + 1)\right]}{\partial y} - S(y) \tag{4.14}$$

where C is the differential water capacity ($d\theta/dh$) (cm^{-1}), and h is soil water pressure head (cm). It is the slope of the soil water characteristic. Van Genuchten [van Genuchten, 1980] has provided analytical expression for the strongly non-linearly behaving soil hydraulic characteristics $\theta(h)$ and $K(h)$.

Next section presents nutrient diffusion process in soil profile and mathematical modeling of nutrient uptake by plant roots.

4.3 Nutrient Diffusion

Nutrient diffusion can be described by the Fick's second law which predicts how diffusion causes the nutrient concentration field to change in time:

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2} \tag{4.15}$$

where c is the nutrient concentration (mol/cm^3), t is time (s), D is the diffusion coefficient (cm^2/s), and x is the position (cm).

The Fick's second law can be derived from Fick's first law and mass balance as follows:

$$\frac{\partial c}{\partial t} = -\frac{\partial}{\partial x}J = \frac{\partial}{\partial x}\left(D\frac{\partial}{\partial x}c\right). \tag{4.16}$$

Assuming the diffusion coefficient D to be a constant, the diffusion equation can be rewritten in the following form:

$$\frac{\partial c}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial}{\partial x}c\right) = D\frac{\partial^2 c}{\partial x^2}. \tag{4.17}$$

The diffusion equation can be expand to 3D form with a constant diffusion coefficient D as follows:

$$\frac{\partial c}{\partial t} = D\left(\frac{\partial^2 c}{\partial x^2}+\frac{\partial^2 c}{\partial y^2}+\frac{\partial^2 c}{\partial z^2}\right). \tag{4.18}$$

To combine the diffusion equation with the convection term caused by the pore water velocity and the sink term for nutrient uptake by plant roots. The diffusion equation is given in equation (4.19). We use the equation (4.19) as a model.

$$\frac{\partial c}{\partial t} = D\left(\frac{\partial^2 c}{\partial x^2}+\frac{\partial^2 c}{\partial y^2}+\frac{\partial^2 c}{\partial z^2}\right) - v(\frac{\partial c}{\partial x}+\frac{\partial c}{\partial y}+\frac{\partial c}{\partial z}) - S, \tag{4.19}$$

where $c(x, y, z, t)$ is the nutrient concentration (mol/cm^3), D is the diffusion coefficient (cm^2/s), v is the pore water velocity (cm/s), and $S(x, y, z, t)$ is the sink term for nutrient uptake by plant roots Somma et al. [1998].

We now propose a method to generate root structures based on nutrient concentration. We assume that the nutrient concentration is stored in each small soil voxel in soil volume. Nutrient concentration will be changed in time based on the diffusion process. If the soil voxel does not contain the part of roots, then sink term is zero for that voxel. The initial nutrient concentration is randomly for every voxel at $t = 0$ and possible to set as function of nutrient distribution in soil volume. Nutrient concentration could not diffuse through the boundary. Assume that we grow the plant in a pot. We implement this simulation using small voxels in soil volume to study how the diffusion process behaves. So finally, we get the numerical calculation of the nutrient concentration at time t by the following equation 4.20.

$$\begin{aligned} c^{t}_{i,j,k} &= c^{t-1}_{i,j,k} + R_1 * (c^{t-1}_{i-1,j,k} + c^{t+1}_{i-1,j,k} + c^{t-1}_{i,j-1,k} + c^{t-1}_{i,j+1,k} + c^{t-1}_{i,j,k-1} + c^{t-1}_{i,j,k+1} \\ &\quad -6 * c^{t-1}_{i,j,k}) - R_2 * ((c^{t-1}_{i,j-1,k} - c^{t-1}_{i,j+1,k}) + (c^{t-1}_{i-1,j,k} - c^{t-1}_{i+1,j,k}) \\ &\quad +(c^{t-1}_{i,j,k-1} - c^{t-1}_{i,j,k+1}) - (dt) * (S^{t-1}_{i,j,k}). \end{aligned} \tag{4.20}$$

where $c(x_i, y_j, z_k, t)$ is the nutrient concentration at the voxel (x_i, y_j, z_k) at time t, $R_1 = \frac{(dt)D}{(dx)^2}$, $R_2 = \dfrac{(dt)\,v}{(2*(dx))}$, $D \le \frac{(dx)^2}{2(dt)}$, dx is the size of cubic voxel, and dt is the time step size.

The nutrient uptake by plant roots is given in equation 4.21.

$$S(x, y, z, t) = \alpha(x, y, z, t)\beta'(x, y, z, t)T_{pot} \tag{4.21}$$

where $\alpha(x, y, z, t)$ is the water-extraction function [-], β' is the normalized nodal value of β [-], and T_{pot} is potential transpiration rate. The nodal values of $\alpha(x, y, z, t)$ is calculated using the expression proposed by van Genuchten [van Genuchten, 1987], considering the combined effects of matric and osmotic potential on water uptake rate:

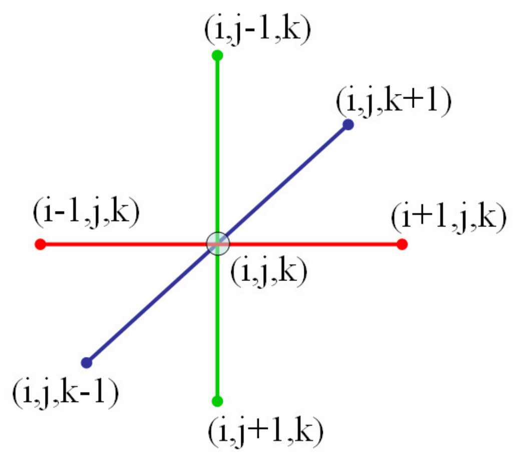


Figure 4.2: Soil voxel diagram of position (i, j, k) and its neighbors.

$$\alpha(x, y, z, t) \quad = \quad \frac{1}{[1 + (h/h_{50})^{p1}] * [1 + (\pi/\pi_{50})^{p2}]}, \tag{4.22}$$

where h_{50} and π_{50} are the soil-water pressure head and the osmotic head at which the uptake rate is reduced by 50%, respectively, and $p1$ and $p2$ are fitting parameters (see [Somma et al., 1998]).

The normalized nodal value of β is required to calculate $\beta'(x, y, z, t)$ in the following equation:

$$\beta'(x, y, z, t) \quad = \quad \frac{\beta(x, y, z, t)}{\int_D \beta(x, y, z)dD}, \tag{4.23}$$

where D is integration over the complete soil domain. The $S(x, y, z, t)$ is calculated at each time step as in F. Somma [Somma et al., 1998].

At each time step t, the nutrient concentration is calculated by using the previous nutrient concentration at the same voxel and its neighbors shown in Figure 4.2. The nutrient concentration at the boundary of the soil volume is equal to its neighbor inside the volume. The growth direction follows to the position of the maximum concentration of its neighbors. In the case of there are many branches at the same position, one branch grows to the position of maximum concentration and the other branches move to the lower concentration. While the root is moving to the appropriate direction, it has taken up the nutrient to the root stem automatically.

Next section, we present how to generate root structures with nutrient concentration in soil volume.

4.4 Root Structures and Root Systems

This section presents root structures and root systems in soil volume. Root tip can turn to an appropriate direction based on soil environment.

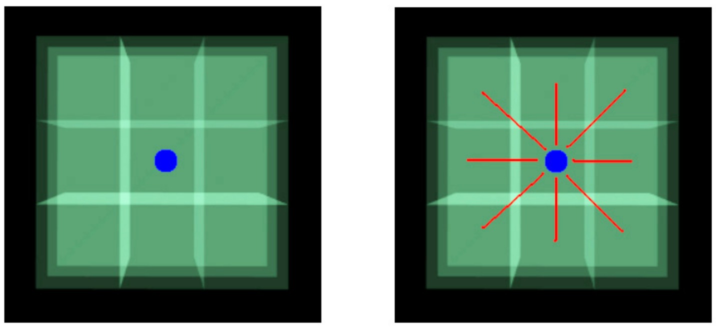


Figure 4.3: A seed position from the top view of soil volume and the possible direction for growing roots.

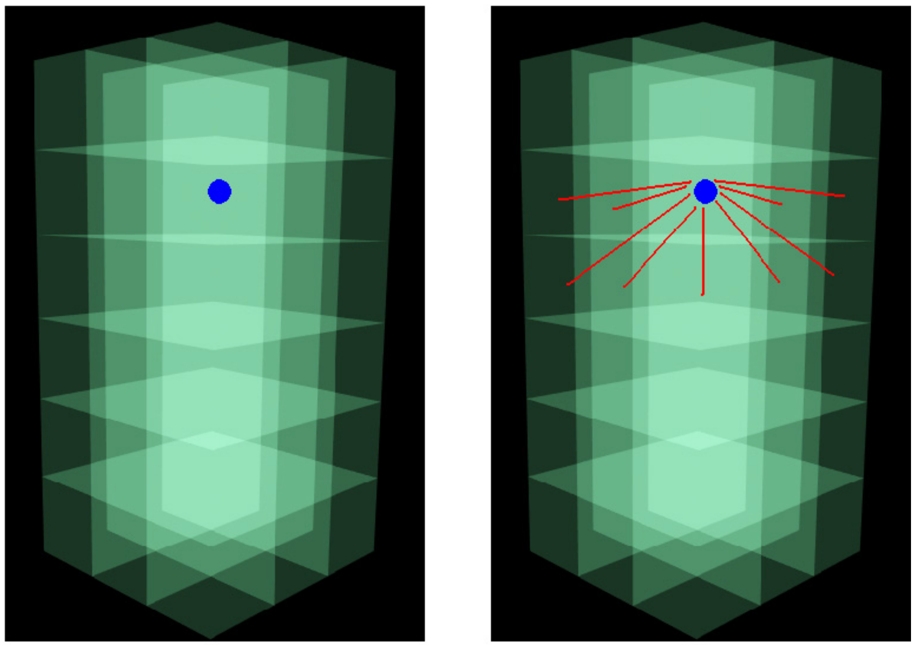


Figure 4.4: A seed position and possible direction for moving root to downward direction.

Figure 4.3 and Figure 4.4 show the seed position and the possible directions of the root movement. There are 26 neighbors which have to be considered excluding the previous root position. The root structures from the simulation are given in Figure 4.5.

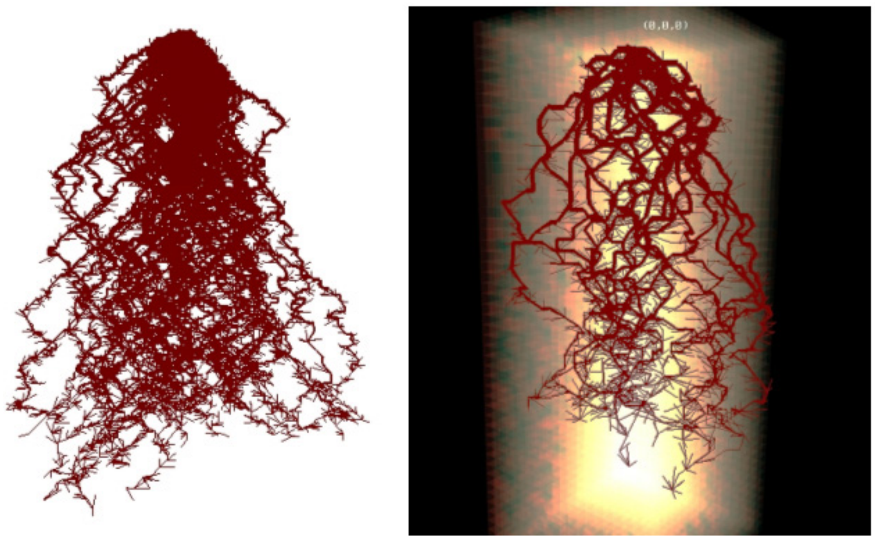
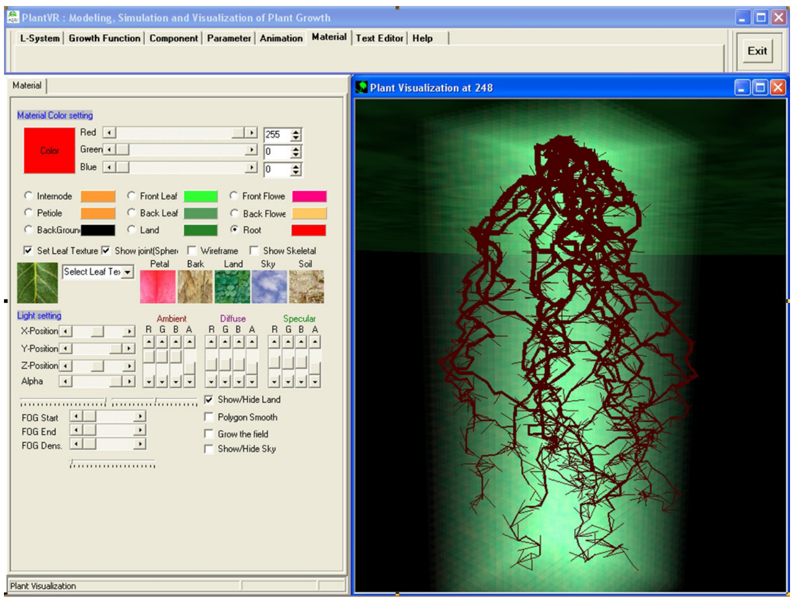


Figure 4.5: Root structures from the simulation [Chuai-Aree et al., 2004a].

The numbers of roots and root depth are required for the simulation. Hairy roots are randomly generated concerning nutrient concentration. The root diameter depends on the root age of each segment. This algorithm is available for the monocotyledon and dicotyledon plants. The root growth process and diffusion processes are changing simultaneously. The root structure visualized in *PlantVR* software is shown in Figure 4.6d and the development of root growth is illustrated in Figure 4.7.



Red 255
Green 0
Blue 0

[✓ Root]

[✓] Set Leaf Texture [✓] Show joint(Sphere) [ ] Wireframe [ ] Show Skeletal

[✓] Show/Hide Land
[ ] Polygon Smooth
[ ] Grow the field
[ ] Show/Hide Sky

Figure 4.6: Simulated root structure in soil volume visualized in *PlantVR* software [Chuai-Aree et al., 2004a].

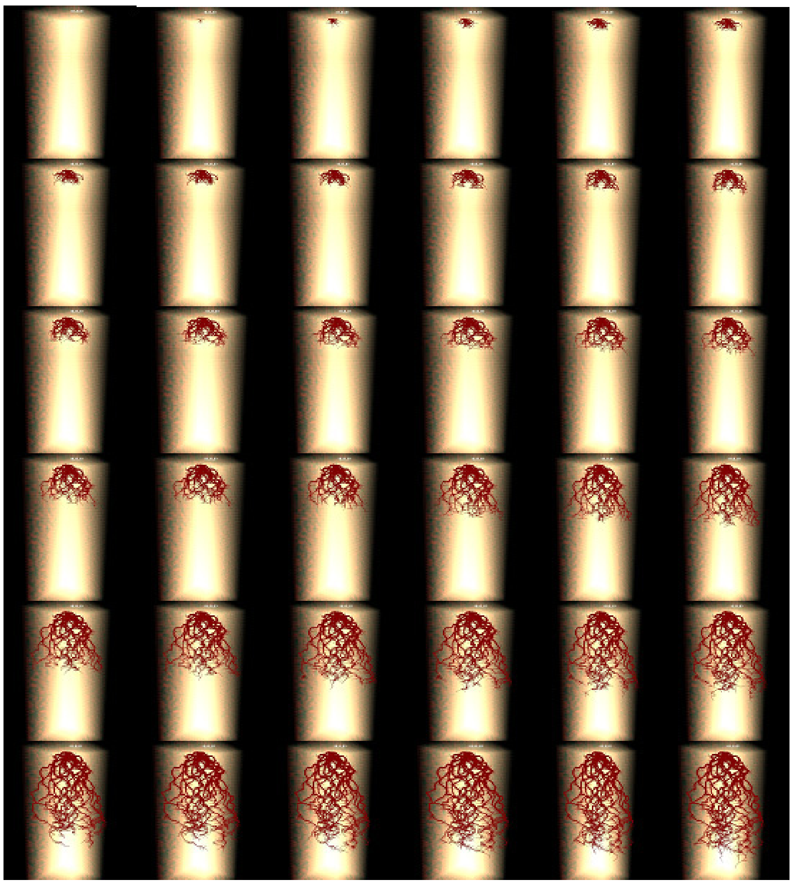


Figure 4.7: Development of root growth in soil volume [Chuai-Aree et al., 2004a].

Next section, we present how to generate root systems by applying PTM from Chapter 3.

4.5 Root Systems using PTM

This section describes the way to combine the root systems with the particle transportation method (see Chapter 3). In Chapter 3 we use random nutrient-water particles for growing root systems. In order to cooperate with root systems in this section, we apply the water content in soil volume to stimulate the root growth. The random water particles are transferred to the water content in each soil voxel. The structured grid for soil volume is applied for stimulating the root growth.

Root Growth Process Algorithm

From Figure 4.8 let the circle be the root tip and eight neighbors are particles which stimulate the direction of tip for the growth process. The root growth process is given in the following algorithm.

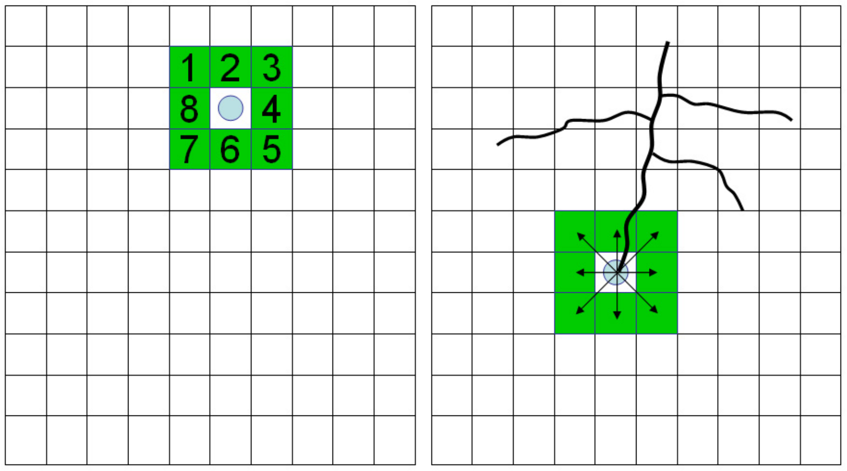


Figure 4.8: 2D grid domain for root growth.

Algorithm 4.1 ***RGP***

1. Initialize a given seed position P_0,
2. Calculate the preferred direction of the root tip P_0,
   - Calculate absolute direction for water content θ,

$$\vec{E} = \vec{d_i} = \sum_{j=1}^{n} \theta_j \vec{v}_{i,j} \tag{4.24}$$

     where n is the number of neighbor particles, θ_j is water content at the position j, and $v_{i,j}$ is the unit vector from the root tip to the position j. It is $\vec{e_i} = |\vec{d_i}| = \frac{\vec{d_i}}{\|\vec{d_i}\|}$,
   - Calculate absolute direction for all factors,

$$\vec{N} = \vec{O} + \vec{E} + \vec{G} + \vec{M} \tag{4.25}$$

     where $\vec{N}$ is the new direction of next root growth, $\vec{O}$ is an old unit vector of the previous direction of root tip, $\vec{E}$ is the unit gradient vector of water content, $\vec{G}$ is the unit vector by gravitropism effect, and $\vec{M}$ is the unit vector by soil strength (see Figure 4.9),
   - Elongate the root tip P_i

$$\vec{p_i^*} = e_r \vec{e_i} \tag{4.26}$$

     where $e_r = e\nabla\theta = \frac{eT_\theta}{(1+\nabla\theta)}$, e_r is elongation rate of the root tip corresponding to gradient of water content $\nabla\theta$ or nutrient concentration ∇c, e is elongation rate constant, and T_θ is the total uptake,

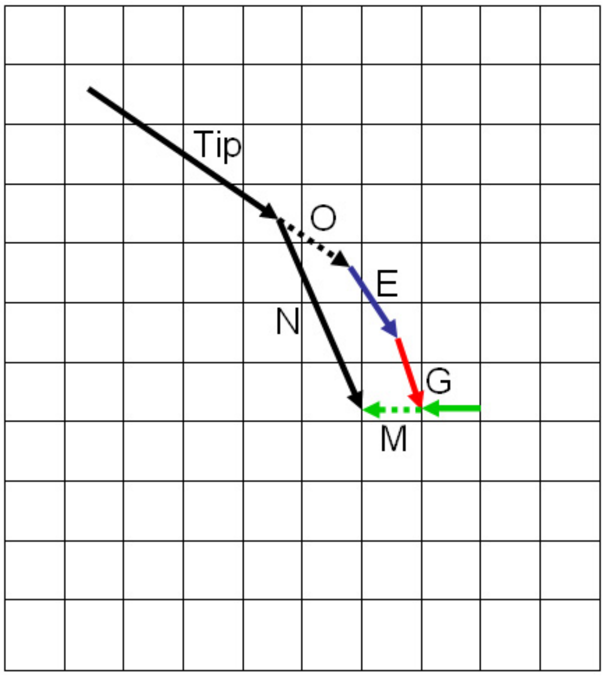


Figure 4.9: 2D grid and root growth direction.

3. Solve nutrient diffusion in equation (4.20) or water flow in equation (4.13),
4. Do the step 2 until the calculation time is reached.

Next section describes shortly about the *MuPhi* library for water flow calculation in soil volume.

4.6 Root Water Uptake Using *MuPhi* Library

This section describes water uptake by plant roots cooperating with the *MuPhi* (or $\mu\varphi$) library for water flow in soil volume. *MuPhi* library was developed by O. Ippisch at the *Institut für Parallele und Verteilte Systeme* (IPVS), University of Stuttgart (see [Ippisch, 2001]). It is used to calculated the water flow in soil volume. Water flow in *MuPhi* solves the Richards' equation in soil volume using Finite Volume Method (FVM). At each time step *MuPhi* is called by *PlantVR* software to ask for the water flow in soil domain. *PlantVR* gets the solution of water content and uses water content for the growth of plant and uptake some water from the soil domain and sends the rest of water after taking up to the *MuPhi* library for the same procedure. A better detail of sending and receiving data between *MuPhi* and *PlantVR* will be explained again in next chapter.

4.7 Results

This section shows some results from the simulation. Figure 4.10 and Figure 4.11 show root growth for every three time steps starting from $t = 63$ using PTM from front and top views, respectively.

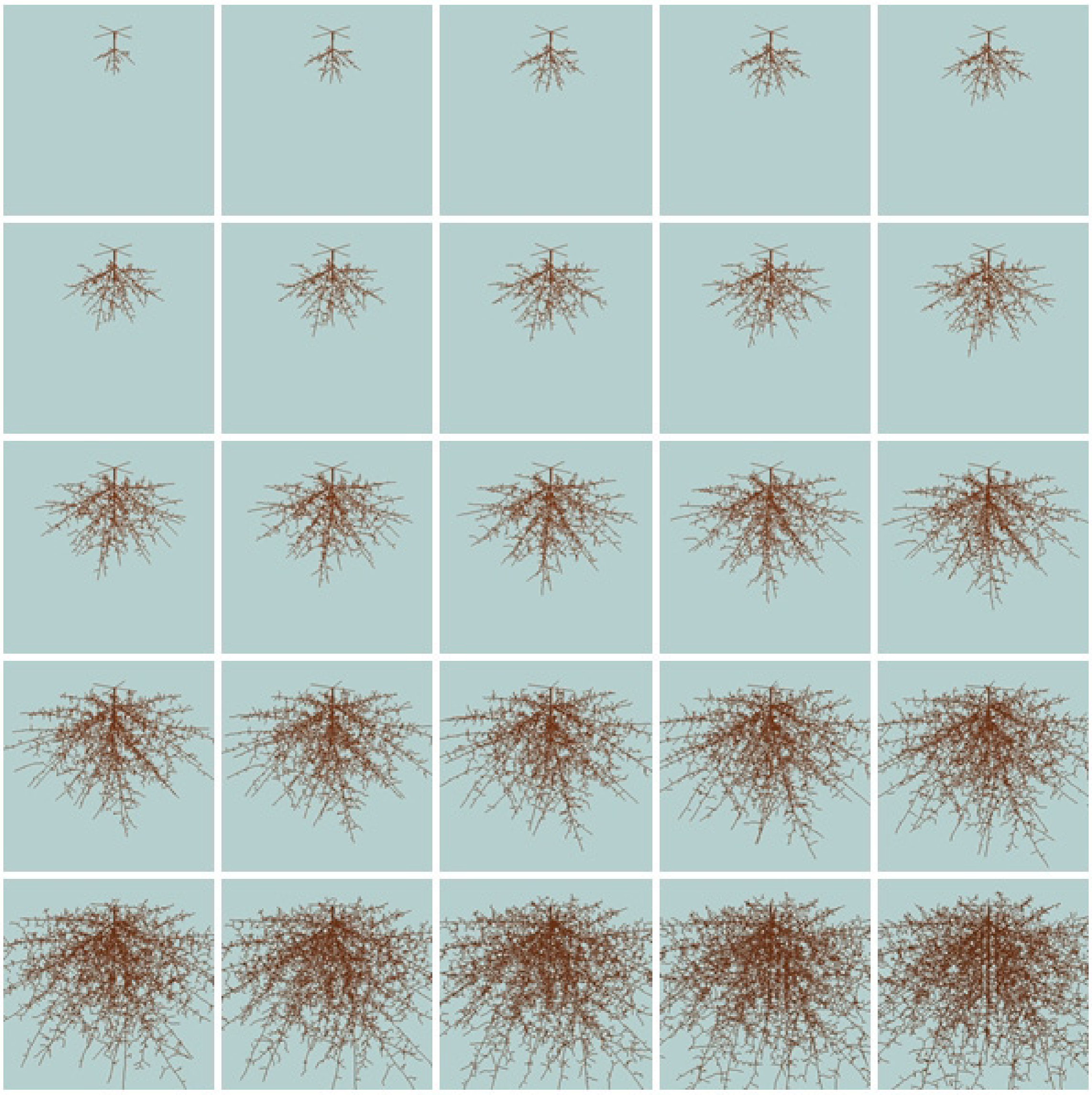


Figure 4.10: Root growth using PTM for every 3 time steps starting from $t = 63$.

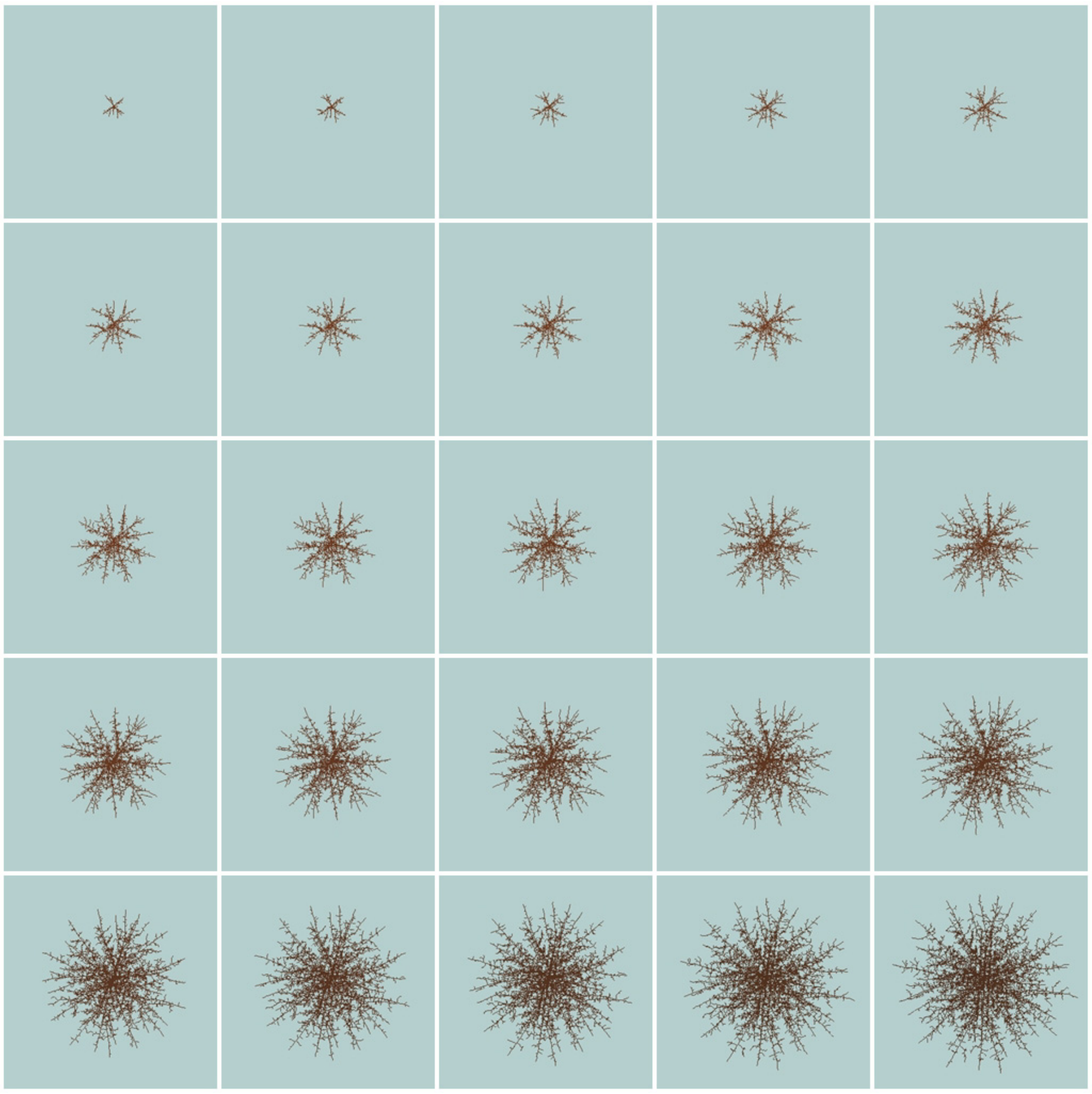


Figure 4.11: Root growth using PTM for every 3 time steps starting from $t = 63$ from top view.

Figure 4.12 shows root growth with its growth curve of root length starting from $t = 3$ for every eight time steps $dt = 8$. Figure 4.14 illustrates the growing root with nutrient uptake by considering the root age. Only young root zone can only uptake nutrient from soil solution.

3D root growth and water uptake using *MuPhi* library is visualized in Figure 4.13. Figure 4.15 illustrates 3D plant growth and water uptake for every three time steps starting from $t = 7$.

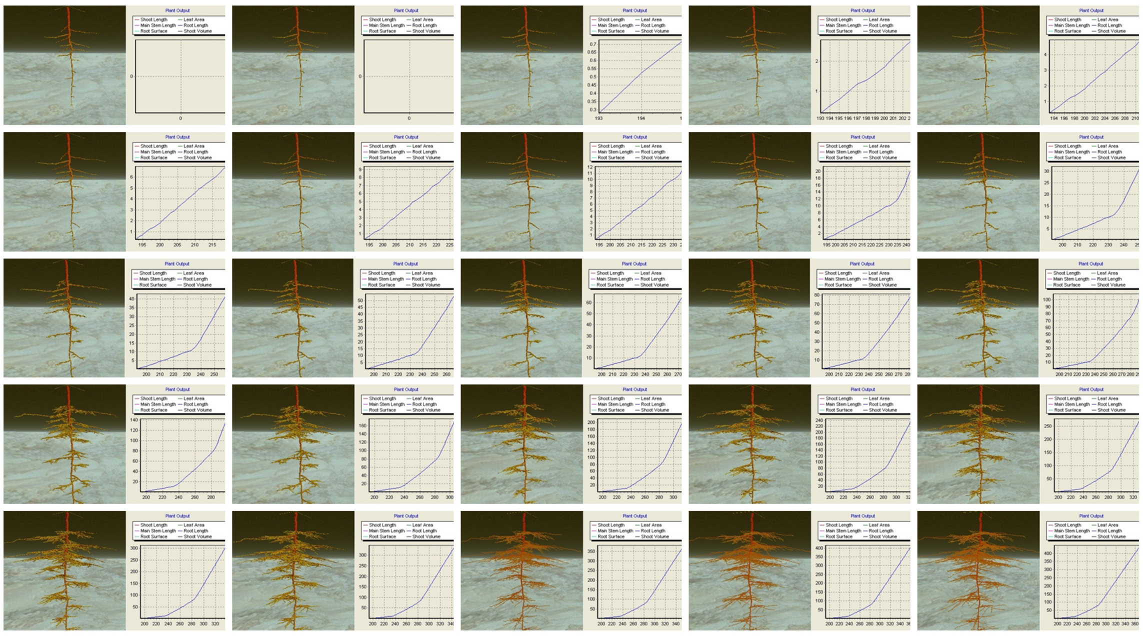


Figure 4.12: Root growth with its growth curve of root length starting from $t = 3$ for every eight time steps $dt = 8$.

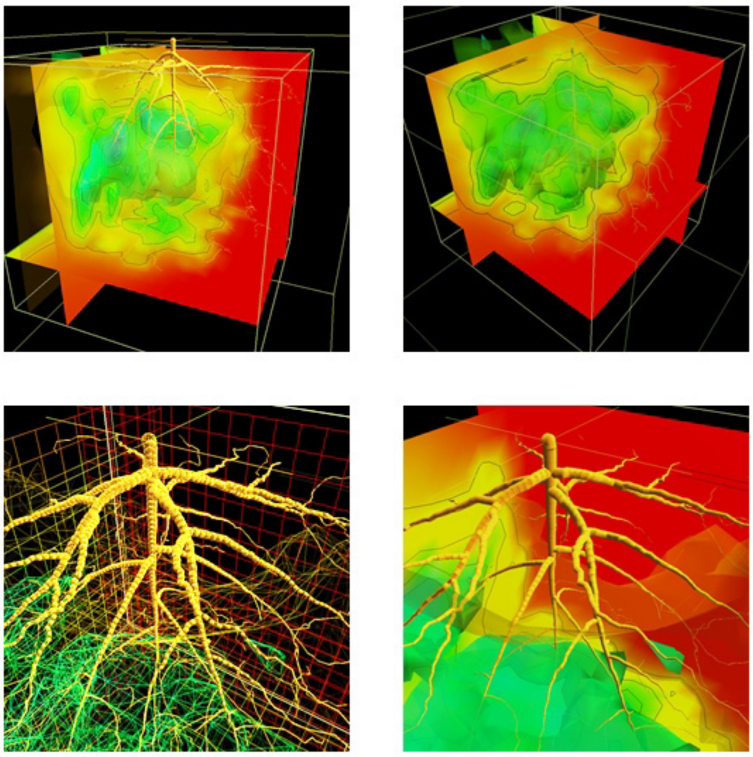


Figure 4.13: 3D root growth and water uptake using *MuPhi* library visualized in *PlantVR*.

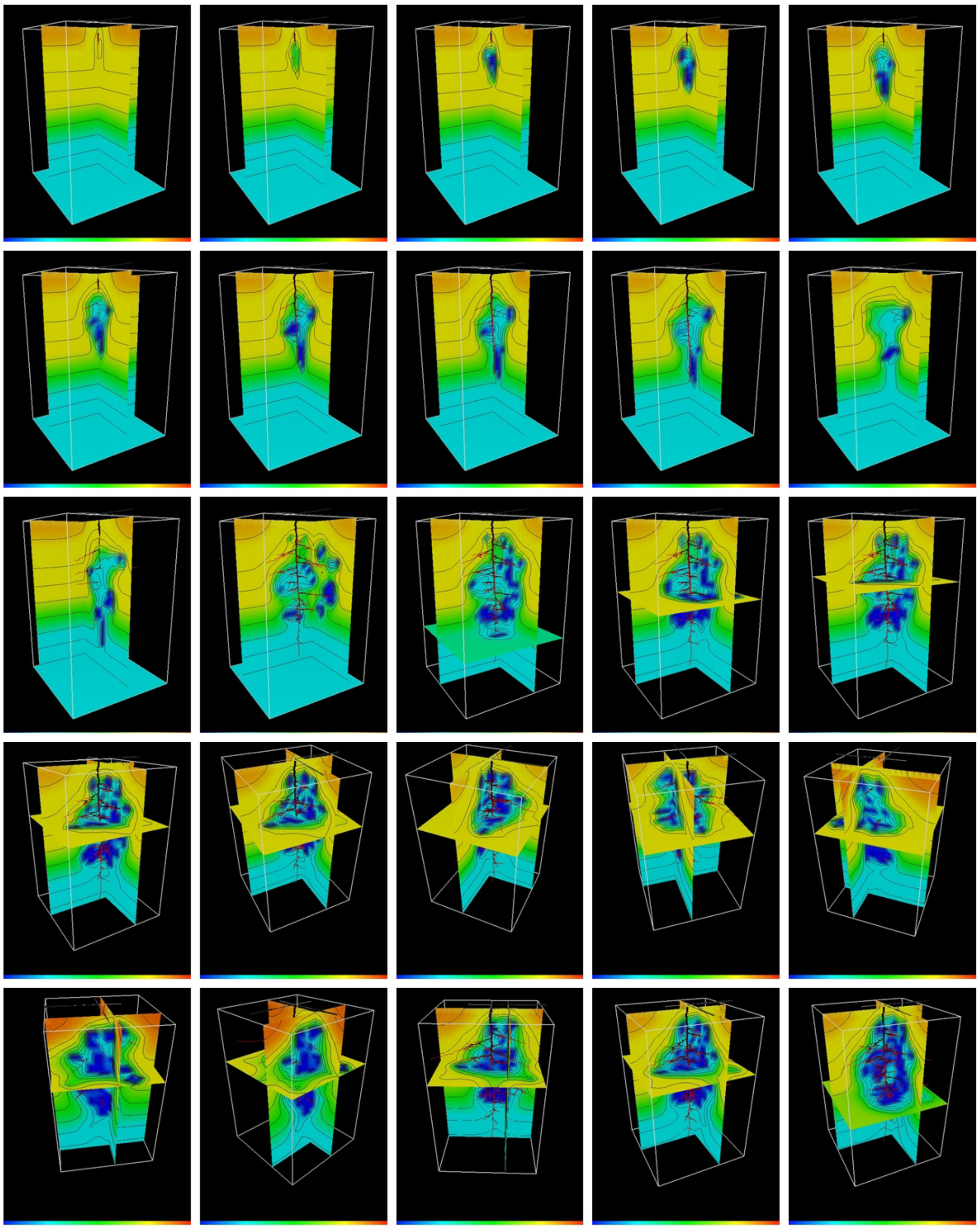


Figure 4.14: Root growth with nutrient uptake by considering the root age.

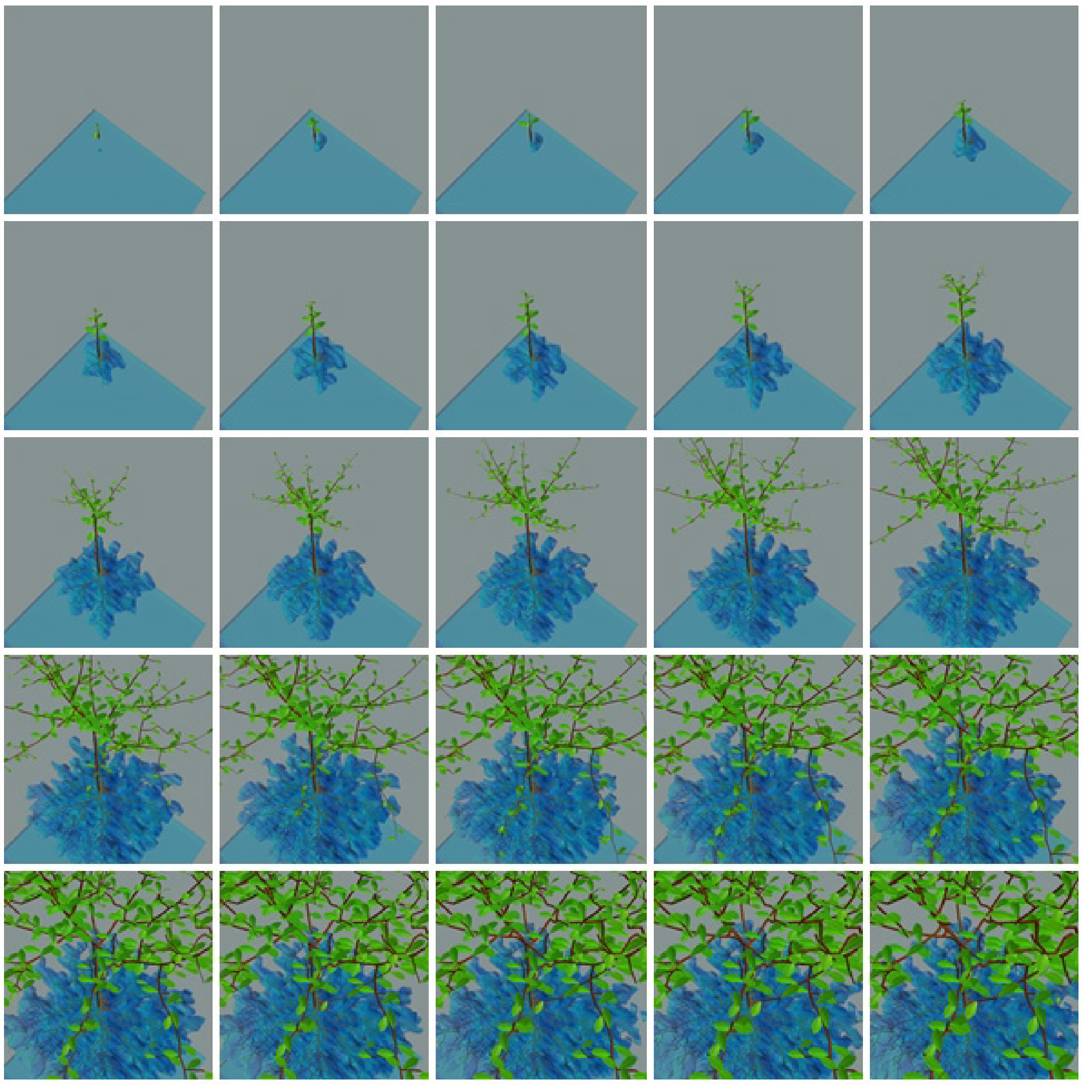


Figure 4.15: 3D plant growth and water uptake for every 3 time steps starting from $t = 7$.

4.8 Discussion and Conclusion

This chapter introduced the root systems, water flow and nutrient diffusion in the soil volume based on the diffusion process of nutrient concentration. Water flow was described by Richards' equation. Plant root growth and water uptake by plant roots was modelled by water flow and sink term from root density in soil volume. Nutrient diffusion was used for the calculation of nutrient uptake by plant roots. The amount of nutrient at each time step in the root stem was coupled with the shoot parts for plant growth. The root growth process algorithm was explained to generate the root system in soil volume cooperating with PTM. Water flow and nutrient diffusion are not coupled together as a solute transport in soil volume in *MuPhi* library yet. It will be improved for the further work. The *MuPhi* library is used in *PlantVR* software.

Chapter 5

Software Tools and Results

"Theory without practice is fruitless, but practice without theory is rootless."
—The land of wisdom, India

This chapter introduces five softwares programmed within the scope of this dissertation, e.g. *PlantVR*, *LeafDS*, *LeafVein*, *BranchRecon*, and *PlantVR-MuPhi*. *PlantVR* is a software for generating plant structures using L-systems based on bracketed L-system, stochastic L-system, parametric L-system and particle transportation method (PTM) based on the environmental particles in both shoot and root parts (see Chapter 2 and Chapter 3). It simulates the growth of plant in time both in shoot and root parts. Leaf3D software was programmed to create the 3D leaf surface used in *PlantVR* as a predefined surface. *LeafVein* software simulates the leaf veins using PTM by defining leaf boundary and the source point P_0 of leaf (see Chapter 3). *BranchRecon* software was developed to solve the inverse problem of L-systems by giving a set of input images contained branching structures and returning the L-system description which can be used in *PlantVR*, Finally, *PlantVR-MuPhi* is a communication library between the *PlantVR* and *MuPhi* library written by O. Ippisch [Ippisch, 2001] for calculating water flow in soil volume and sending the numerical solution to *PlantVR* for plant growth and water uptake by plant roots.

This chapter is organized into six sections for each software and conclusion: *PlantVR*, *LeafDS*, *LeafVein*, *BranchRecon*, *PlantVR-MuPhi*, and discussion and conclusion. Next section describes more detail of *PlantVR* software.

5.1 PlantVR Software

This section introduces *PlantVR* procedure, simulation and visualization of *PlantVR*. *PlantVR* is a software tool to simulate and visualize a model of plant growth such as the growth of leaves, shoots and roots. This software can generate the plant shoot and root structures based on L-systems and PTM depending on their environmental factors. There are two methods for generating plants in *PlantVR*. The first method uses L-systems based on *PlantVR*'s syntax. In order to create the plant structures in an easy way, the PTM is provided in the second method which is designed to generate leaf veins, plant shoots, and roots. The growth depends on environmental factors. For example, leaf growth is based on auxin hormone and glucose distribution in the leaf blade, while shoot growth is based on sunlight, available volume space, and resources from root part. Root growth is based on the available water, nutrient, volume space in the soil, and resources from shoot part. The software allows users to expand from a single plant to an entire crop. The user can export the results to an animation file and display the growth at each time step.

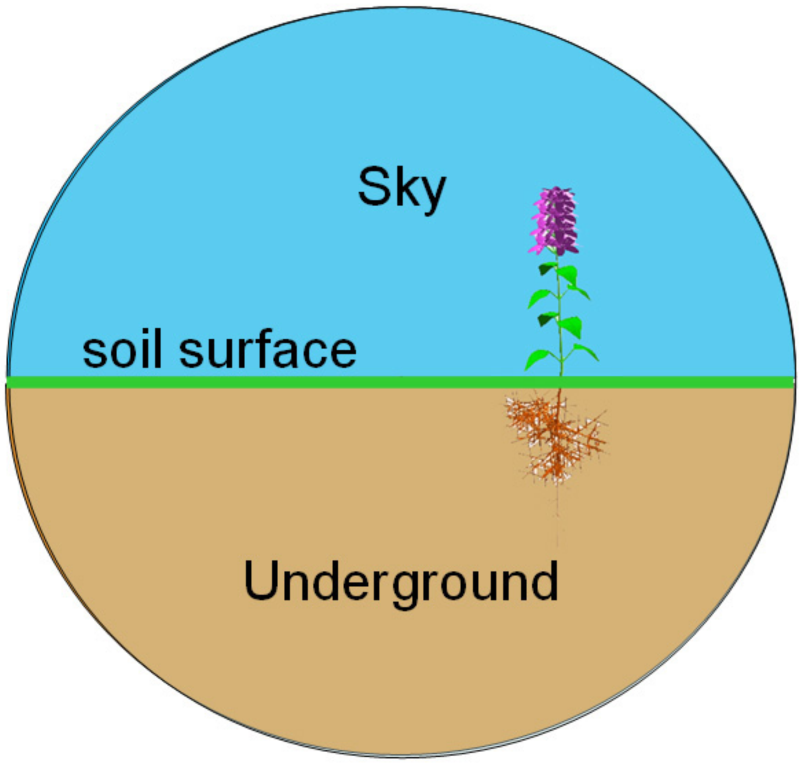


Figure 5.1: Diagram of *PlantVR* visualizer.

To motivate this section and overview the *PlantVR* software, Figure 5.1 illustrates the diagram of *PlantVR* visualizer. The spherical world represents the world inside for above ground and underground.

This section is organized into five parts: procedure of *PlantVR* prototype, L-systems interpretation algorithm, collected data, growth function and curve fitting, leaf and flower in predefined object, and visualization procedure.

5.1.1 Procedure of PlantVR Prototype

The biological data from actual plant observations are very important to simulate the plant development [Chuai-Aree, 2000]. A flow diagram of a prototype of *PlantVR*, simulation and visualization of plant growth is shown in Figure 5.2. This section presents a prototype for creating computer models that capture the development of plants using L-systems and mathematical model incorporating biological data. The L-systems are used for qualitative model in order to represent the plant topology and development. This method has six consecutive steps, (1) defining a qualitative model constructed from observations of plant growth in their life cycle, (2) measuring key characteristics collected from actual plants by L-systems description, (3) converting raw data to growth functions based on sigmoidal function approximations using curve fitting and parameter estimation, (4) defining a quantitative model composed from the qualitative model and growth functions, (5) visualizing the quantitative model in three dimensional space, and (6) evaluating the models. The design of L-system in this dissertation is based on bracketed L-system, stochastic L-system and parametric L-system.

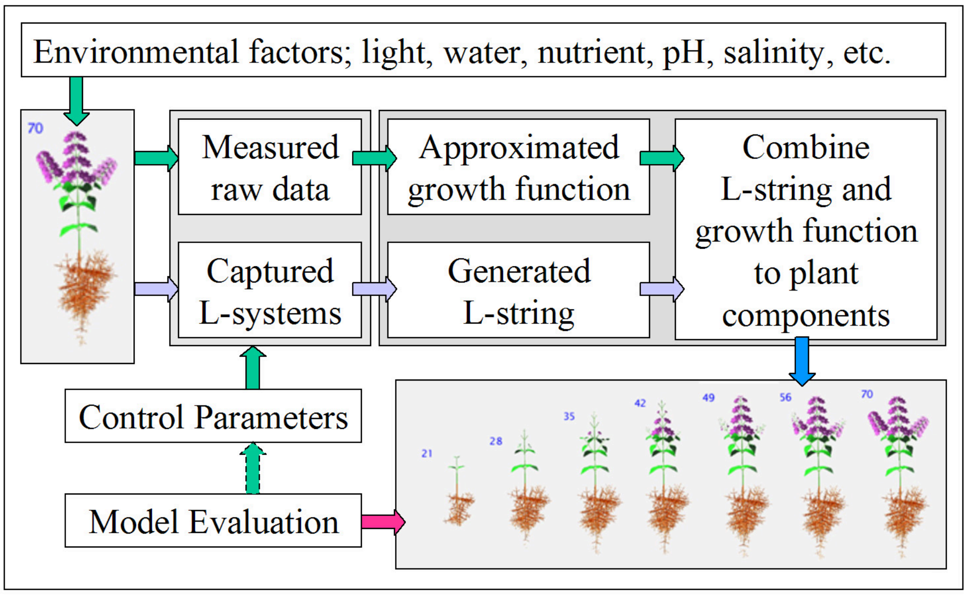

Environmental factors; light, water, nutrient, pH, salinity, etc.
70
Measured raw data
Approximated growth function
Combine L-string and growth function to plant components
Captured L-systems
Generated L-string
Control Parameters
Model Evaluation
21
28
35
42
49
56
70

Figure 5.2: Diagram of plant simulation and visualization in *PlantVR*.

5.1.2 L-systems Interpretation Algorithm

This section introduces interpretation algorithm of L-systems which is related to all symbols in Chapter 2 (Table 2.5) and their interpretations in L-systems prototype for both plant shoot and root parts.

Figure 5.3 shows a flow diagram of L-systems interpretation. The process starts from reading the input L-systems code. A given L-systems code is compiled and checked for syntax error. If there are no syntax errors, the L-string will be generated and interpreted to plant components and their properties. The programming code of rewriting algorithm is similar to Pascal programming langauge given in Appendix A.

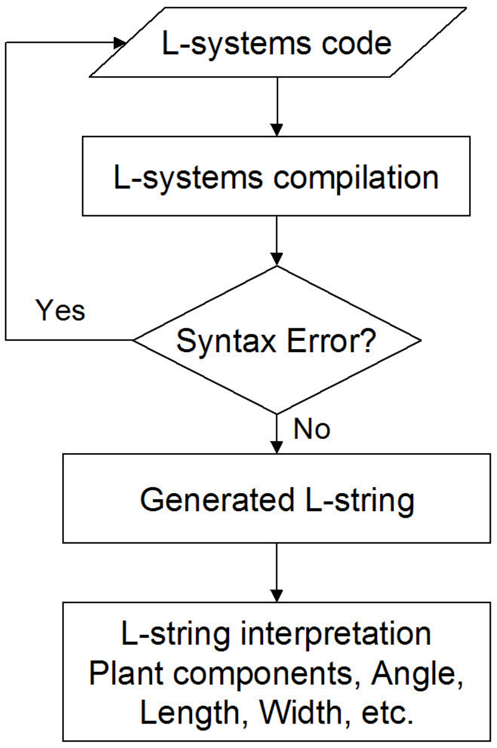


Figure 5.3: L-systems interpretation flow diagram.

5.1.3 Collected Data, Growth Function and Curve Fitting

The *PlantVR* software provides an option to import the measured data from actual plants for approximating the growth function by parameter estimation based on sigmoidal function. The soybean (*Glycine max*) data were firstly collected in 1999 as a case study (see [Chuai-Aree, 2000]). It is suitable for new data observation in the same procedure.

For the measured data, they were collected by manual method using rulers for length and width, vernier caliper for diameter, protractor for angle. Figure 5.4 shows the observation of soybeans which were grown from the seeds in their life time approximately 120 days. Figure 5.5 shows the soybean structure and its label for measurement. The internodes are measured from the seed position (soil surface) to the first node called *N1*. Figure 5.6 illustrates the diagram of internode measurement. The diagrams of petiole and leaf measurements are shown in Figure 5.7 and Figure 5.8, respectively. There are some abbreviations labelled in the figures, i.e. *N4LL* is the *left* leaf *length* of the 4^{th} internode, *N4LW* is the *left* leaf *width* of the 4^{th} internode, *N4MW* is the *middle* leaf *width* of the 4^{th} internode.

All collected data are imported to the system and approximated all parameter values based on N-pulses sigmoidal function using parameter estimation (see Section 2.5.1).

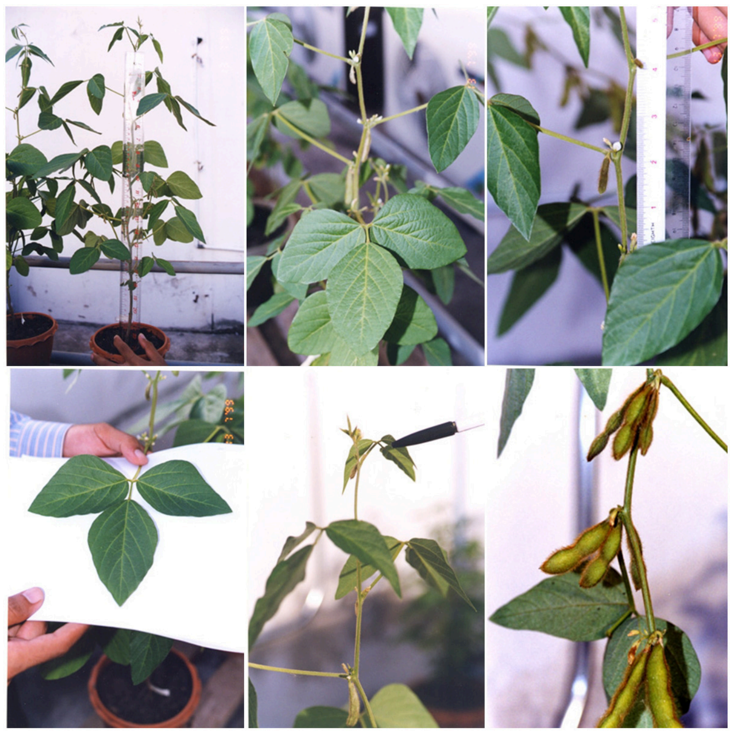


Figure 5.4: Data collection from soybeans (*Glycine max*).

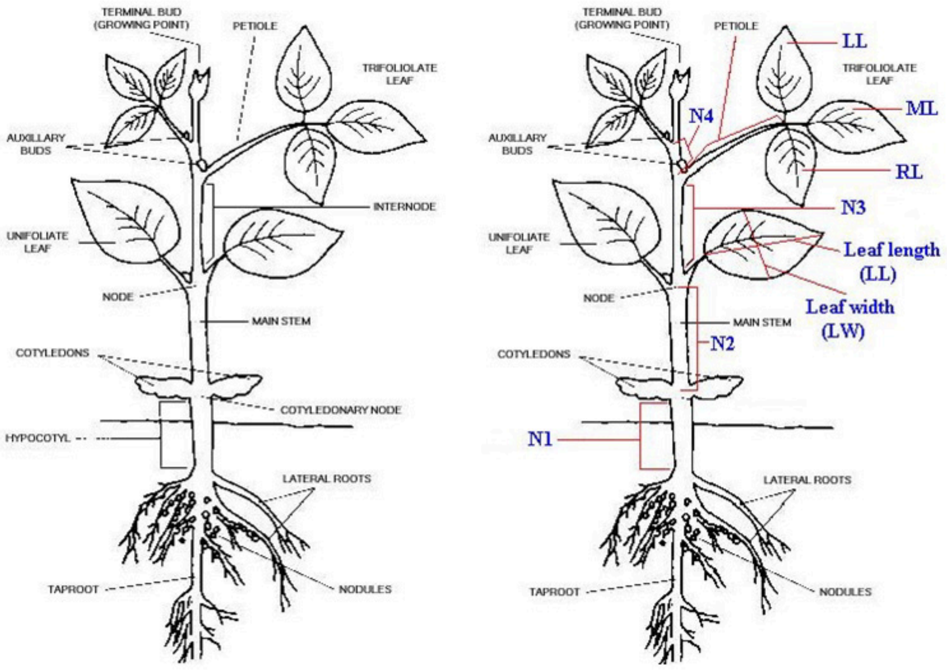


Figure 5.5: The structure of soybean (*Glycine max*).

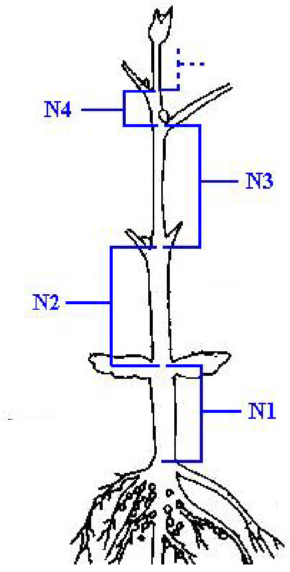


Figure 5.6: The diagram of internode measurement, N_i is the i^{th} internode.

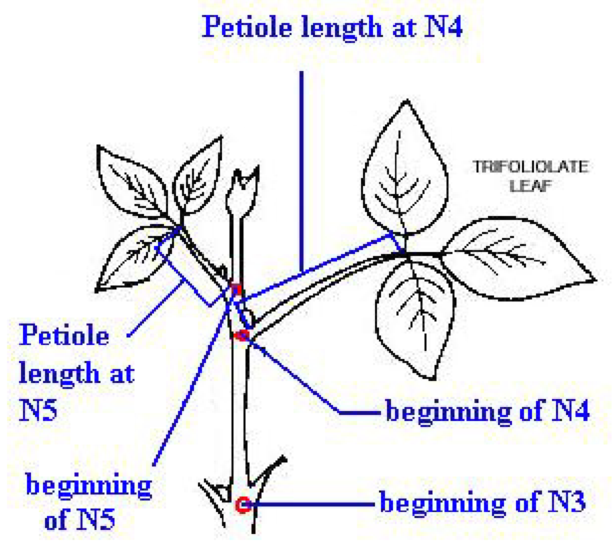


Figure 5.7: The diagram of petiole measurement, petiole length data.

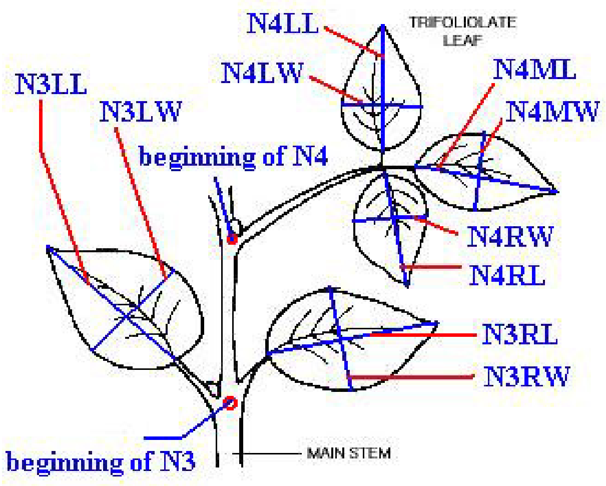


Figure 5.8: The diagram of leaf measurement, the length and width data of left, middle and right leaf.

The description of leaf measurement is shown in Figure 5.8. At the beginning of third internode, there are two leaves. The fourth internode and upper internode, there are trifoliolate - three leaves. The symbol description of each leaf is given below:

$NiLL$	:	the left leaf length of i^{th} internode,
$NiLW$	:	the left leaf width of i^{th} internode,
$NiRL$	:	the right leaf length of i^{th} internode,
$NiRW$	:	the right leaf width of i^{th} internode,
$NiML$	:	the middle leaf length of i^{th} internode, and
$NiMW$	:	the middle leaf width of i^{th} internode.

5.1.4 Qualitative Model

The modeling process begins with the specification of the qualitative model. It captures the aspects of a plant which can be obtained through the observations and are deemed essential to its form and development. These include the topology and the sequence of activities of various plant modules. The main components of the plant are distinguished and their developmental stages are identified. The connections between these components are also defined. In this dissertation, the qualitative parameters are obtained from a soybean. The qualitative parameters of the soybean model consists of three main parts: internodes, petioles, and leaves.

The simulation begins with first a pair of leaves. This is captured by the L-system axiom, or the initial string of modules. The axiom in Figure 5.9 represents an internode I, a pair of leave L with its short internode i, and an apex A. The apex A is initially contained within the petioles. After some iterations, the apex A is substituted by an internode I, a right petiole $[-P]$, an internode I, a left petiole $[+B]$ and an apex A shown by the following production rule.

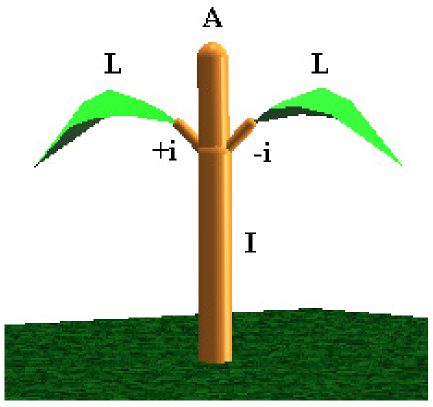


Figure 5.9: Axiom = $I[+iL][-iL]A$.

$$A \;=\; I[-P]I[+B]A. \tag{5.1}$$

Each petiole consists of some internodes, some short petioles, a left leaf, a right leaf, and a middle leaf. The production rules of the left and right petioles are defined as follows:

$$\begin{aligned} P &= IIIII[\backslash pL][/pL][-pL], \\ B &= IIIII[\backslash pL][/pL][+pL] \end{aligned} \tag{5.2}$$

The “Endrule” is defined as follows:

$$\begin{aligned} B &= IL, \\ P &= IL, \\ A &= IL \end{aligned} \tag{5.3}$$

The above *Endrules* are called at the last iteration. The left petiole B, the right petiole P as well as the apex A are substituted by an internode and a leaf. By using the previous defined production rules with some specific parameters, the L-system description of a soybean is given in the following format.

```
Soybean {
    Shoot {
        Iterations=6
        Angle=45
        Diameter=1.5
        Axiom=I[+iL][-iL]A
        A = I[-P]I[+B]A
        P = IIII[\pL][/pL][-pL]
        B = IIII[\pL][/pL][+pL]
        Endrule
        B = IL
        P = IL
        A = IL
    }
}
```

Actually, L-system code consists of a module of shoot and root parts (see Table 2.6). In this case, we show only the shoot part of soybean for a better explanation. In the above L-systems code, there are six iterations; the initial branch angle is 45 degrees, and the diameter of first internode is 1.5 centimeters. After the sixth iteration, the *Endrule* productions are called to terminate the substitution process.

The last L-string at the sixth iteration is

```
I[+iL][-iL]I[-IIII[\pL][/pL][-pL]]I[+IIII[\pL][/pL][+pL]]I[-IIII[\pL][/pL]
[-pL]]I[+IIII[\pL][/pL][+pL]]I[-IIII[\pL][/pL][-pL]]I[+IIII[\pL][/pL][+pL]]
I[-IIII[\pL][/pL][-pL]]I[+IIII[\pL][/pL][+pL]]I[-IIII[\pL][/pL][-pL]]I
[+IIII[\pL][/pL][+pL]]I[-IIII[\pL][/pL][-pL]]I[+IIII[\pL][/pL][+pL]]A.
```

The string P and B are not appeared in the last string of the above symbol string, but A remains in the last letter. A is replaced with *endproduction* $A = IL$. Observe that the *endproduction* $B = IL$ and $P = IL$ is not necessary to define in the soybean module. After the above L-string is replaced with the *endproduction*, the final string is

```
I[+iL][-iL]I[-IIII[\pL][/pL][-pL]]I[+IIII[\pL][/pL][+pL]]I[-IIII[\pL]
[/pL][-pL]]I[+IIII[\pL][/pL][+pL]]I[-IIII[\pL][/pL][-pL]]I[+IIII[\pL]
[/pL][+pL]]I[-IIII[\pL][/pL][-pL]]I[+IIII[\pL][/pL][+pL]]I[-IIII[\pL]
[/pL][-pL]]I[+IIII[\pL][/pL][+pL]]I[-IIII[\pL][/pL][-pL]]I[+IIII[\pL][/pL][+pL]]IL.
```

After the sixth iteration, the L-string is substituted by *Endrule* to the symbol defined in all end production rules. The L-string is interpreted by the turtle's interpretations of the L-systems definition. This string is transformed to the associated plant structures with the growth function at each time step.

L-system String and Plant Interpretation

The last string of generated L-system is interpreted as the plant structure which consists of internodes, petioles, leaves, apices, and flowers that compose to the main stem. The branches is defined from the bracket symbols “[” and “]”. The example of the L-system prototype is given in an example *Plant2*. The simplest L-system is given for understanding the L-system code and L-system interpretation in *Plant2* prototype.

```
Plant2 {
    Shoot {
        Iterations=1
        Angle=45
        Diameter=2
        Axiom=IA
        A=[+pL][-pL]I[-I1][+I1]I[/I2][\I2]IF
        1=[/iL][\iL]P
        2=[-iL][+iL]P
        Endrule
        P=iiF
    }
}
```

The *Plant2* prototype has one iteration, the angle of 45 degrees, and two units of diameter. An axiom consists of an internode I and an apex A. At the first iteration, the apex A is substituted by left short petiole and leaf $[+pL]$, right short petiole and leaf $[-pL]$, internode I, left petiole and special symbol $[-I1]$, right petiole and special symbol $[+I1]$, internode I, back petiole and special symbol $[/I2]$, front petiole and special symbol $[\backslash I2]$, internode I, and flower F. The second production, a symbol 1 is replaced by back short internode and its leaf $[/iL]$, front short internode and its leaf $[\backslash iL]$, and the symbol P. The third production, a symbol 2 is replaced by short right petiole and its leaf $[-iL]$, short left petiole and its leaf $[+iL]$, and the symbol P. The *endproduction* P is substituted by two short internode ii and flower F for every symbol P in the last L-system string. After the first iteration, the last string is given below.

```
I[+pL][-pL]I[-I[/iL][\iL]iiF][+I[/iL][\iL]iiF]I[/I[-iL][+iL]iiF][\I[-iL][+iL]iiF]IF
```

The internode order, leaf order, and flower order of the L-string are described in Figure 5.10. The internode order is counted from the first to the last symbol of L-string for symbol I, i, P, and p. The leaf L and flower F order are attached to the previous internode or petiole. All internode and petiole have their own attributes for leaf and flower. All attributes are described in Section 5.1.6.

The visualization of the L-string is presented in Figure 5.11. It shows only the internodes and petioles. The left nearest component of the internode or the petiole is the parent of component i. Figure 5.12 shows the number of internode order.

In Figure 5.12, the number of each internode is counted following the L-string as in Figure 5.10. It is used to determine the parent of each component. The number of symbol

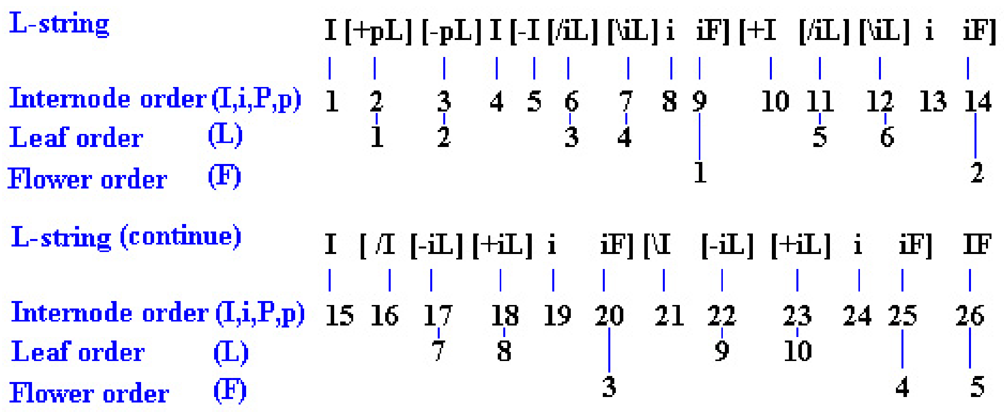

L-string
I [+pL] [-pL] I [-I [/iL] [\iL] i iF] [+I [/iL] [\iL] i iF]
Internode order (I,i,P,p) 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Leaf order (L) 1 2 3 4 5 6
Flower order (F) 1 2
L-string (continue)
I [/I [-iL] [+iL] i iF] [\I [-iL] [+iL] i iF] IF
Internode order (I,i,P,p) 15 16 17 18 19 20 21 22 23 24 25 26
Leaf order (L) 7 8 9 10
Flower order (F) 3 4 5

Figure 5.10: The component order of L-string

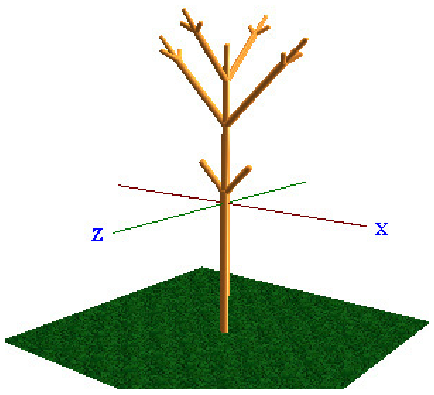


Figure 5.11: All internodes and petioles

[in the L-string is used to generate the main stem of plant. The level of each component is used to calculate the initial time of each component. The number of bracket [is calculated from the following equation (5.4).

$$N_{[} \quad = \quad N([) - N(]) \tag{5.4}$$

where

$N_{[}$	:	Number of bracket [,
$N([)$	:	Number of open bracket [, and
$N(])$	:	Number of close bracket].

The parent is used to set the attribute of each component in the visualization procedure, for example in Figure 5.12, the 1st internode is the parent of the 2nd, 3rd and 4th internode. The parent value of component can be calculated from the number of bracket [and the number of main internode in the current state. Each stack of every state with the order number of internode or petiole of the internode I, short internode i, petiole P, or short petiole p are shown in Figure 5.13.

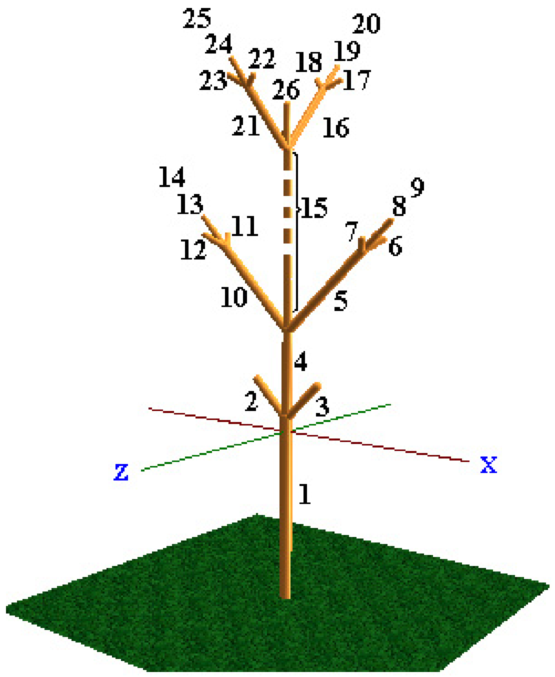


Figure 5.12: All internodes and their number labels.

At the last order of Figure 5.13, the main stem of the plant will be arranged in the stack. For example, the main stem of this example is the first, the fourth, the 15^{th}, and the 26^{th} internode, respectively. The level of each component is considered from the order of the parent calculated from the following equation (5.5):

$$L_i = L_{P_i} + 1 \tag{5.5}$$

where the level of the parent the first component is zero $L_{P_0} = 0$. Figure 5.14(a) shows the level of the plant. It is used to set the initial time of its component. It will be shown in Section 5.1.5. The level of component, the main stem order, the parent of component, and the number of bracket [are shown in Table 5.1, where

L_i	:	Level of component i,
M_i	:	Main stem order of component i,
P_i	:	Parent of component i, and
$N([)$	:	Number of open bracket [.

From this prototype, the leaf L and the flower F are added to the system and visualized in Figure 5.14(b).

In the case of soybean, the L-string is more complicated, so we reduced the process to two iterations. The last L-string is given below.

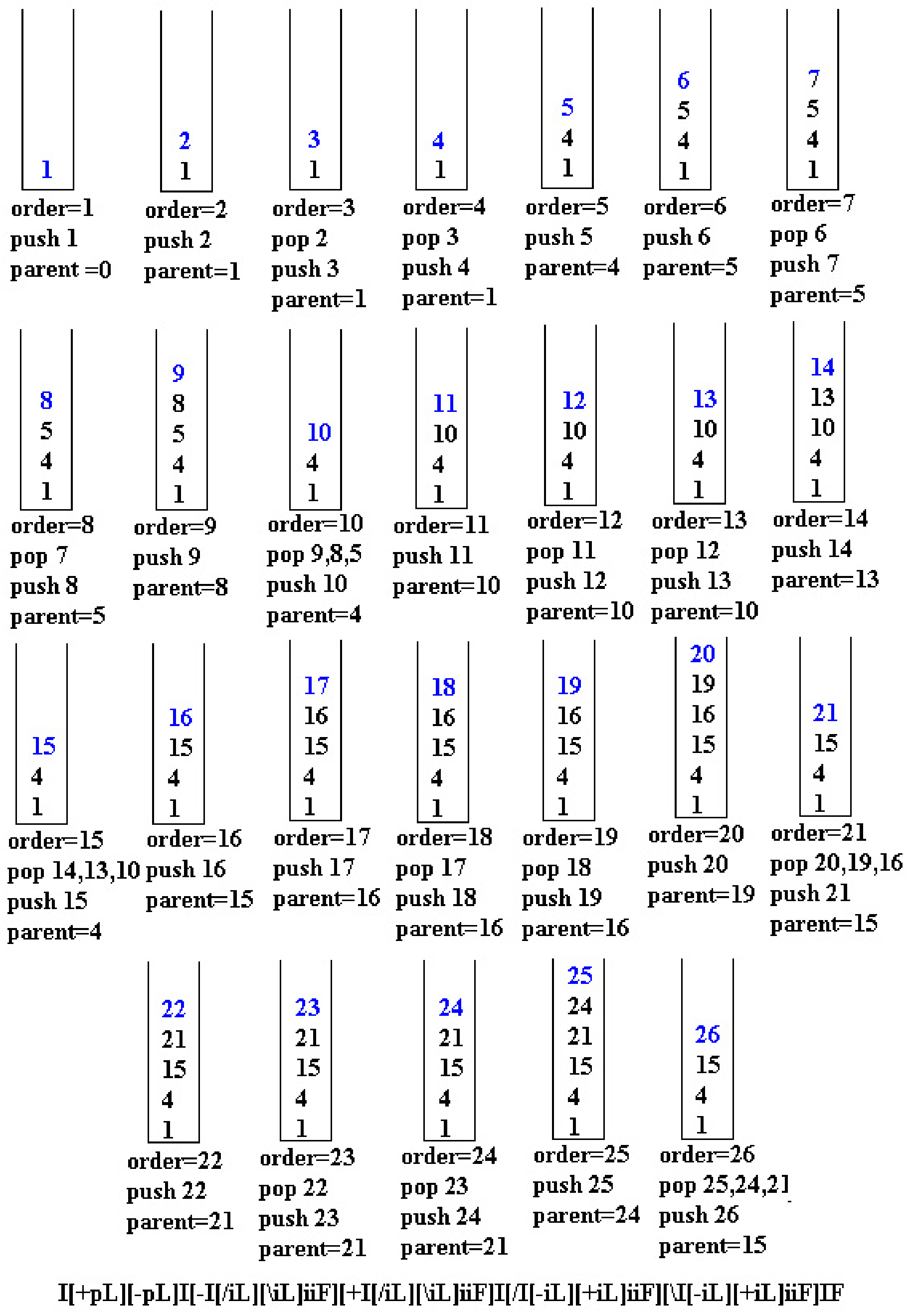
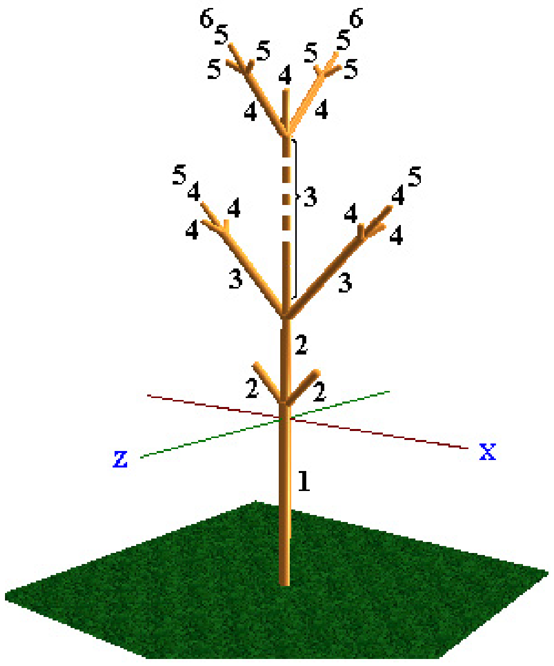


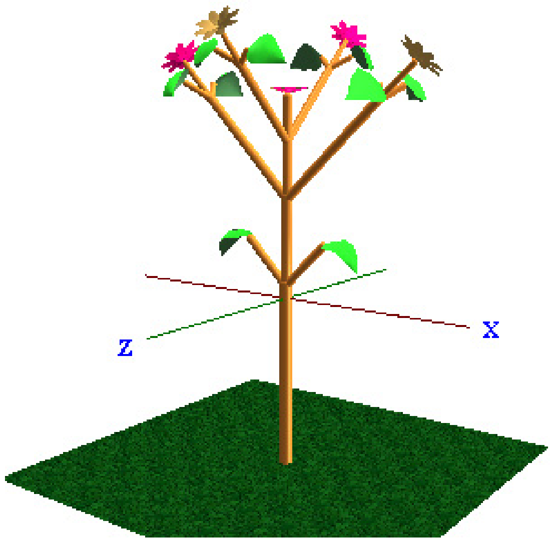
Figure 5.13: Push and pop stack for interpreting the parent of each internode and petiole.

Table 5.1: The value of leaf number, flower number, number of [, parent of each component, main stem order, and level of each component.

Component number i	Symbol	Leaf number	Flower Number	$N([)$	Pi	Mi	Li
1	I	-	-	0	0	1	1
2	p	1	-	1	1	-	2
3	p	2	-	1	1	-	2
4	I	-	-	0	1	2	2
5	I	-	-	1	4	-	3
6	i	3	-	2	5	-	4
7	i	4	-	2	5	-	4
8	i	-	-	1	5	-	4
9	i	-	1	1	8	-	5
10	I	-	-	1	4	-	3
11	i	5	-	2	10	-	4
12	i	6	-	2	10	-	4
13	i	-	-	1	10	-	4
14	i	-	2	1	13	-	5
15	I	-	-	0	4	3	3
16	I	-	-	1	15	-	4
17	i	7	-	2	16	-	5
18	i	8	-	2	16	-	5
19	i	-	-	1	16	-	5
20	i	-	3	1	19	-	6
21	I	-	-	1	15	-	4
22	i	9	-	2	21	-	5
23	i	10	-	2	21	-	5
24	i	-	-	1	21	-	5
25	i	-	4	1	24	-	6
26	I	-	5	0	15	4	4



(a) The level of each component

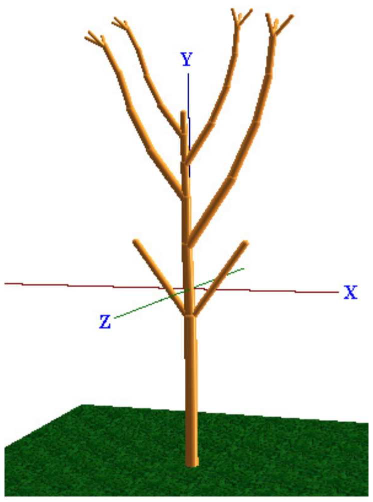


(b) adding leaves and flowers

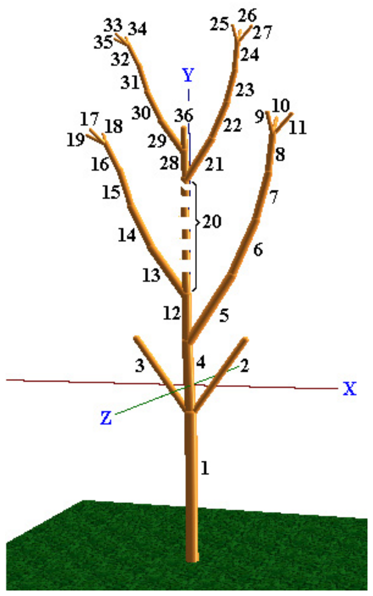
Figure 5.14: The level of each component (a) and the visualized image after adding leaves and flowers to the plant structure. (b).

I[-iL][+iL]I[-IIII[\pL][/pL][-pL]]I[+IIII[\pL][/pL][+pL]]I[-IIII[\pL]
[/pL][-pL]]I[+IIII[\pL][/pL][+pL]]IL

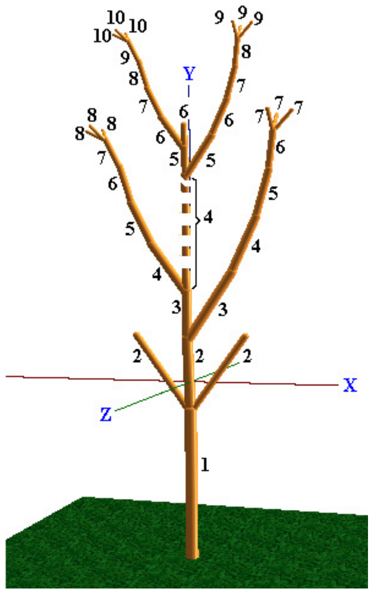
The internodes and petioles of the soybean model are shown in Figure 5.15(a), and the number of internode and petiole are shown in Figure 5.15(b). The level of each component and its parent is shown in Figure 5.15(c) by following the concept of the *Plant2* prototype. After adding the leaf L to the plant, the visualization is shown in Figure 5.15(d), and the 6th iteration of *soybean* prototype of its structure is expressed in Figure 5.16.



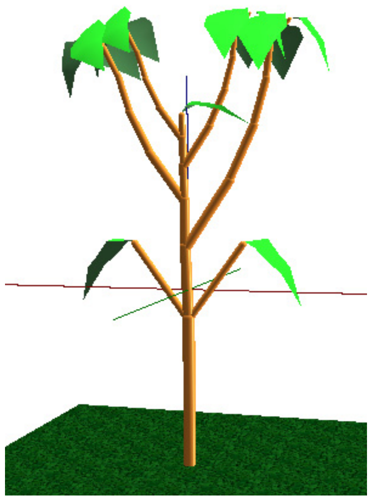
(a) soybean at 2^{nd} iteration



(b) soybean and component labels



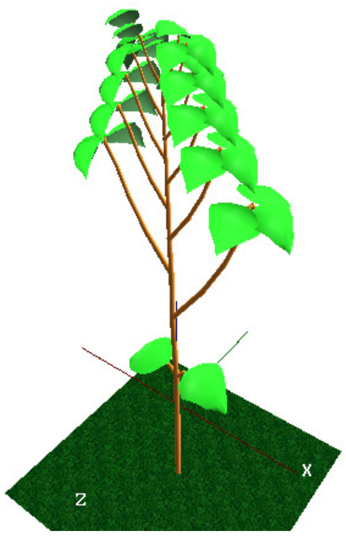
(c) soybean and level of all components



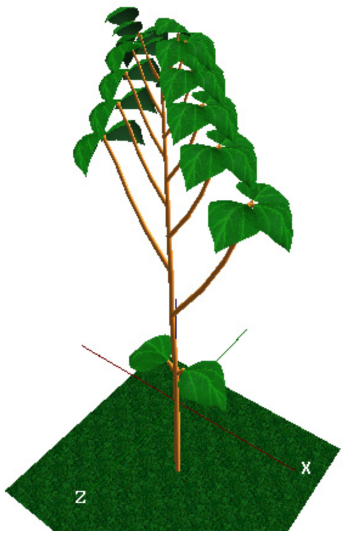
(d) soybean with leaves

Figure 5.15: Soybean structure (a) at the 2^{nd} iteration and its component labels (b).

The implemented codes of rewriting algorithm and L-system interpretation algorithm are given in Appendix A.1.



(a) soybean at 6th iteration



(b) soybean and leaf texture mapping

Figure 5.16: Soybean's structure (a) at 6th iteration and its leaves with texture mapping (b).

5.1.5 Quantitative Model

The quantitative model combines the L-systems string with the approximated growth functions of each component. The plant model can simulate its growth with continuous development in the virtual reality form. In Section 5.1.4, the qualitative model or the L-system string is defined and generated to the plant structure. In Section 5.1.3, the approximated growth function is constructed from the raw data corresponding to the time in their life cycle. The slope m, the minimum value L, the maximum value U, and the time T are set to its component. Every component of the plant such as the internode I, the petiole P, the leaf L, the flower F, and the apex A are controlled by a self-growth function with either same or different slope m, the time T, the maximum U, and the minimum L. For example, the L-system string of Soybean prototype in Section 5.1.4 is shown below:

I[-iL][+iL]I[-IIII[\pL][/pL][-pL]]I[+IIII[\pL][/pL][+pL]]I[-IIII[\pL]
[/pL][-pL]]I[+IIII[\pL][/pL][+pL]]I[-IIII[\pL][/pL][-pL]]I[+IIII[\pL]
[/pL][+pL]]I[-IIII[\pL][/pL][-pL]]I[+IIII[\pL][/pL][+pL]]I[-IIII[\pL]
[/pL][-pL]]I[+IIII[\pL][/pL][+pL]]I[-IIII[\pL][/pL][-pL]]I[+IIII[\pL][/pL][+pL]]IL

The value of m, L, U, T of every component is shown in the Table 5.2.

The approximated growth function of each component is shown in Figure 5.17. The internode I, the short internode i, and the apex A are shown in Figure 5.17(a), the petiole

Table 5.2: The value of L, U, m, T.

Symbols	L value	U value	m value	T value
I and i	0.1275	4.039	0.65	7
P and p	0	6.8737	0.38	8
L width	0	2.7909	0.49	11
L length	0	5.2613	0.54	11
A	0.1275	4.039	0.65	7

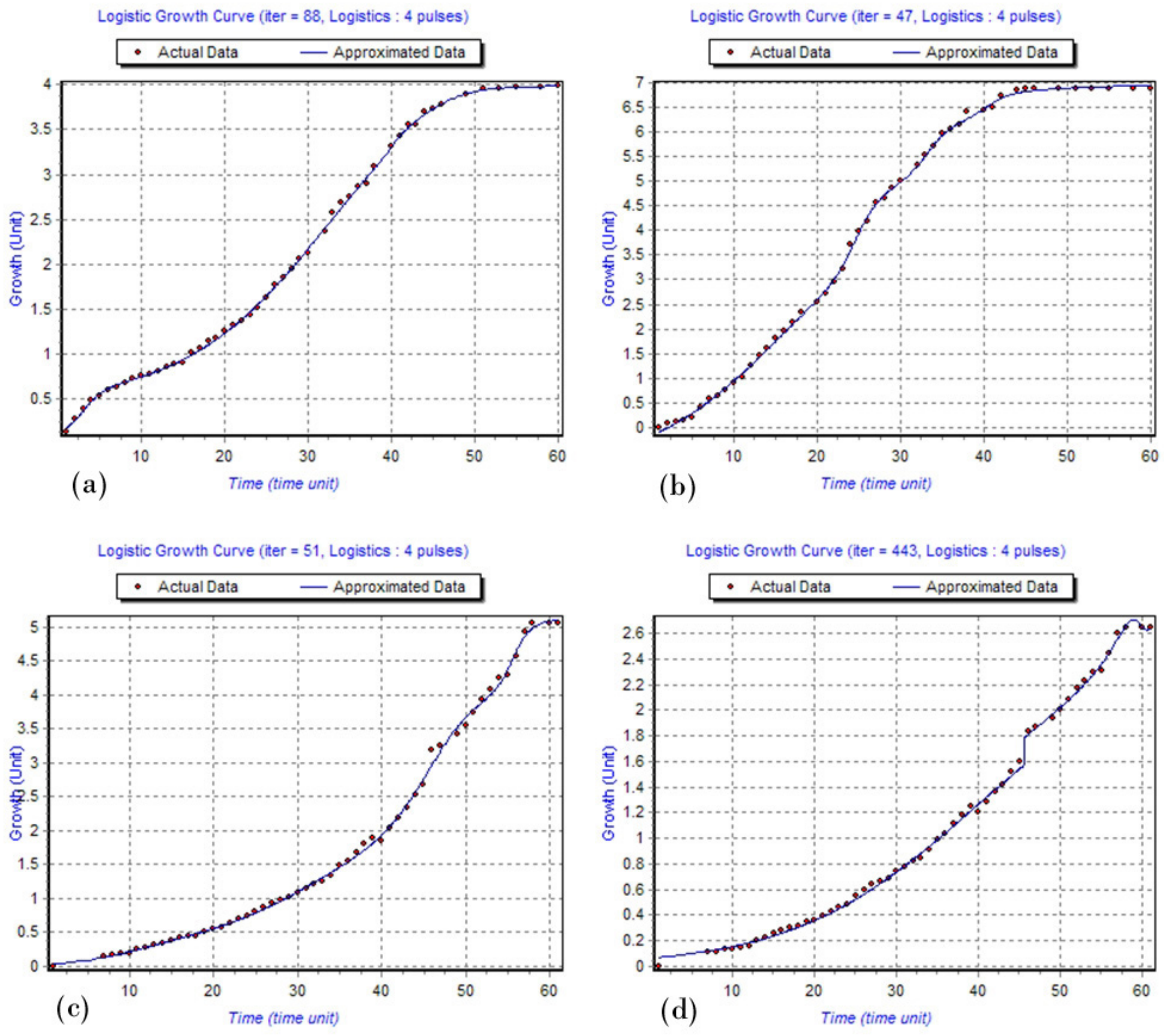


Figure 5.17: Approximated growth functions of internode and apex (a), petiole (b), leaf length (c) and leaf width (d). The growth unit and time unit are centimeters and days, respectively.

P and the short petiole p are presented in Figure 5.17(b), Figure 5.17(c) shows the leaf length L, the leaf width L is shown in Figure 5.17(d).

Figure 5.18 shows the approximated function of each component I, i, P, p, L for their initial time functions.

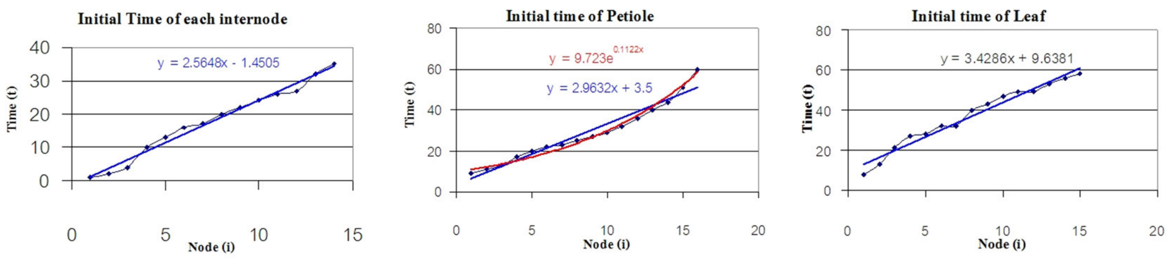
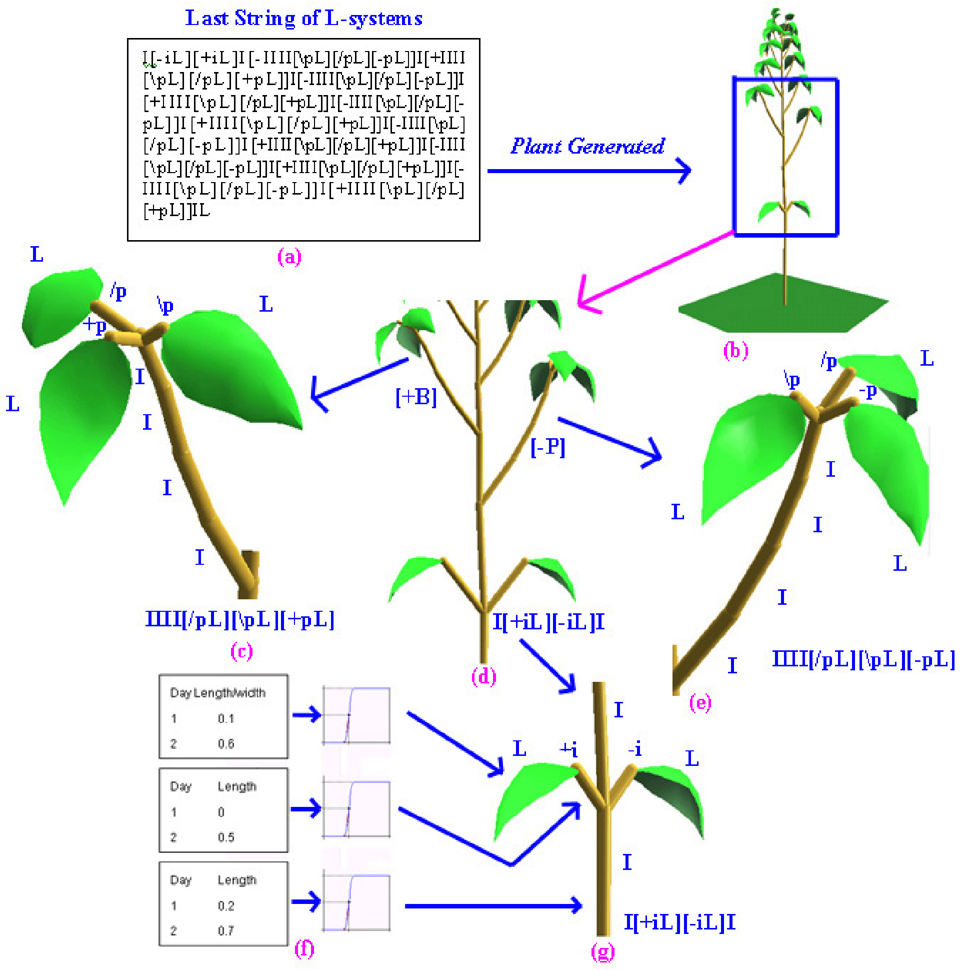


Figure 5.18: Examples of approximation of initial time of internode, petiole and leaf of soybean using linear approximation.



```
I[-iL][+iL]I[-IIII[\pL][/pL][-pL]]I[+IIII
[\pL][/pL][+pL]]I[-IIII[\pL][/pL][-pL]]I
[+IIII[\pL][/pL][+pL]]I[-IIII[\pL][/pL][-
pL]]I[+IIII[\pL][/pL][+pL]]I[-IIII[\pL]
[/pL][-pL]]I[+IIII[\pL][/pL][+pL]]I[-IIII
[\pL][/pL][-pL]]I[+IIII[\pL][/pL][+pL]]I[-
IIII[\pL][/pL][-pL]]I[+IIII[\pL][/pL]
[+pL]]IL
```

Day	Length/width
1	0.1
2	0.6

Day	Length
1	0
2	0.5

Day	Length
1	0.2
2	0.7

Figure 5.19: Structure of simulation generated from the production rules of soybean.

Figure 5.19 illustrates a graphical image of a soybean drawn from the L-string defined in the production rules in Section 5.1.4. Figure 5.19(a) shows the L-string obtained after the rewriting method. The plant structure in Figure 5.19(b) is constructed from the L-system string shown in Figure 5.19(a) and its graphical image of the axiom of the plant is illustrated in Figure 5.19(g). Figure 5.19(d) gives the details of a part of soybean. Figure

5.19(c) is the left petiole component. The right petiole is shown in Figure 5.19(e). In Figure 5.19(f), the raw data were converted to the growth function corresponding to each symbol.

5.1.6 Visualization Procedure

This section describes the visualization method of each component such as the internode I, the short internode i, the petiole P, the short petiole p, the leaf L, the apex A, and the flower F. Each component uses the primitive geometric shape; for example, cylinder, line, triangular polygon, sphere, rectangular polygon, and texture bitmap.

Internode

A cylinder represents the plant internode, while a sphere represents the internode joint. Figure 5.20 shows the internode of plant. The short internode i is similar to internode I but its length is shorter than internode I.

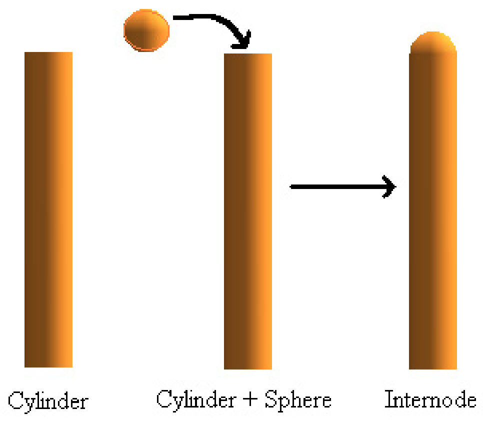


Figure 5.20: Internode of plant structure.

The structure of internode in the prototype is defined by the following type:

```
Procedure Internode(wx, wy : real)
begin
    Cylinder(wx, wx, wy, slices, stacks)
    Sphere(wx, longitudes, latitudes)
end;
```

From above procedure, internode receive 2 arguments, wx and wy where wx represents the base radius of the internode, and wy represents the height of the internode. A cylinder has five arguments, namely, base radius, top radius, height, edges, and slices. Spheres have three arguments, namely, radius, latitudes, and longitudes. The cylinder and the sphere are given in Figure 5.21.

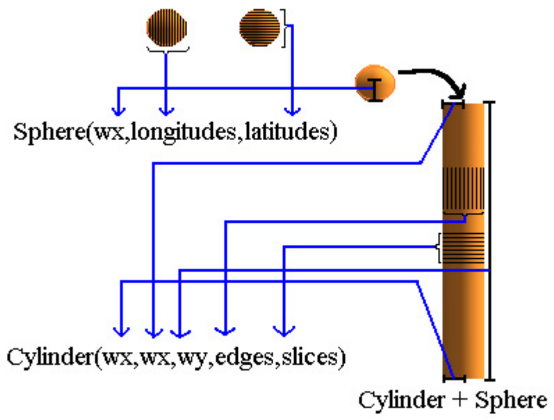

I
Sphere(wx,longitudes,latitudes)
Cylinder(wx,wx,wy,edges,slices)
Cylinder + Sphere

Figure 5.21: The arguments used in cylinder and sphere function.

Petiole

A petiole and a main stem have the same topology as internode with either same or different size. The petiole component is represented by symbols I, i, P, or p. Figure 5.22 shows the petiole and the main stem of the plant structure. The short petiole structure is similar to the petiole with either same or different size. The main stem was given in Section 5.1.4. Its direction is upward to Y-axis.

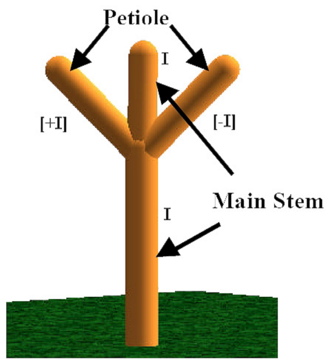


Figure 5.22: The petiole and the main stem of plant topology $I[-I][+I]I$.

Leaf

Leaves are very important for plant visualization to make plant look more realistic. The plant is defined by more than three points in the three-dimensional space. It consists of the source point which is attached to the internode or the petiole tip. In this dissertation, the leaf library is designed in the following format.

```
Name=Leaf_Name
Source=S
```

```
Point=P1x P1y P1z
Point=P2x P2y P2z
Point=P3x P3y P3z
     ...
Point=Pnx Pny Pnz
Triangle=T11 T12 T13
Triangle=T21 T22 T23
Triangle=T31 T32 T33
     ...
Triangle=Tm1 Tm2 Tm3
```

The meaning of each keyword is given as follows:

Name=Leaf_Name
Leaf_Name is the name of leaf.

Source=S
A source point is a point in a set of the leaf points. It is attached to the petiole or the internode with its leaf.

Point=P_{nx} P_{ny} P_{nz}
A keyword is a coordinate (x, y, z) of leaf point. The value P_{nx}, P_{ny}, and P_{nz} are the x, y, z at the n^{th} point where $x, y, z \in \mathbb{R}$.

Triangle=T_{m1} T_{m2} T_{m3}
A triangle consists of three points. The values of T_{m1}, T_{m2} and T_{m3} are the first point, the second point, and the third point of the m^{th} triangle, respectively. They are members in the set of the leaf points. Figure 5.23 illustrates the leaf surface structure.

For example, the leaf surface prototype (see [Chuai-Aree, 2000, Chuai-Aree et al., 2000]) is given below.

```
Name=Leaf,Soybean
Source=2
Point=0 0 0
Point=0 0 0
Point=0 0 0
Point=-33 -50 8
Point=0 -50 11
Point=33 -50 8
Point=-44 -100 11
Point=0 -100 17
Point=47 -100 11
Point=-33 -150 5
Point=0 -150 12
```

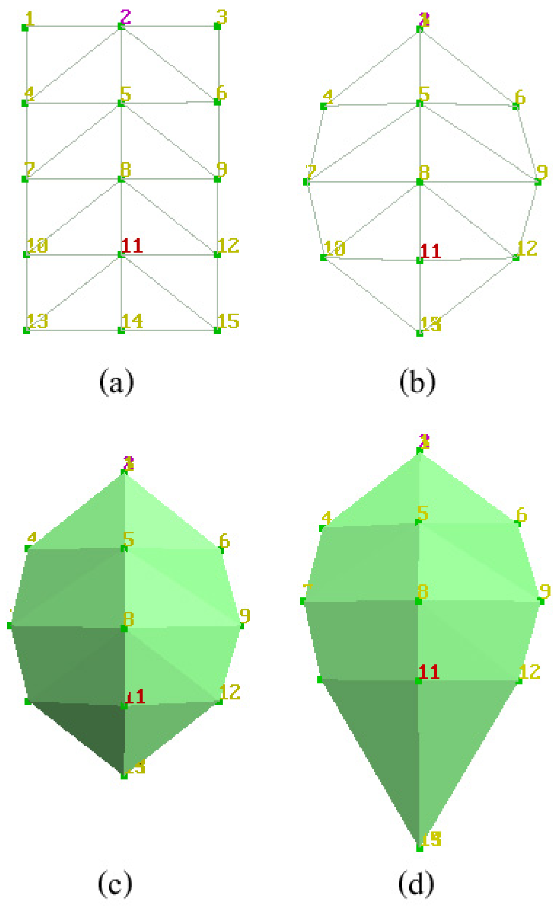


Figure 5.23: The leaf surface used in *PlantVR*.

```
Point=33 -150 8
Point=0 -200 0
Point=0 -200 0
Point=0 -200 0
Triangle=1 4 2
Triangle=4 5 2
Triangle=2 5 6
Triangle=2 6 3
Triangle=4 7 5
Triangle=7 8 5
Triangle=5 8 9
Triangle=5 9 6
Triangle=7 10 8
Triangle=10 11 8
Triangle=8 11 12
Triangle=8 12 9
```

```
Triangle=10 13 11
Triangle=13 14 11
Triangle=11 14 15
Triangle=11 15 12
```

This example illustrates a soybean leaf prototype. The second point is the source point of the leaf. There are 15 points (size 5×3), and 16 triangular polygons. Figure 5.23 shows the soybean leaf topology. At the first, there are 15 points as shown in Figure 5.23(a) in XY-coordinate. The first point and the third point are set similar to the second point as same as the 13^{th} point and the 15^{th} point are set to the 14^{th} point. The seventh point is moved leftward while the ninth is moved rightward. The fifth point, the eighth point, and the 11^{th} point are moved upward. The result is shown in Figure 5.23(b). All triangular polygon is set to every triangle in Figure 5.23(c). In Figure 5.23(d), the 13^{th} point, 14^{th} point, and 15^{th} point are moved downward. It makes the leaf longer than the leaf Figure 5.23(c).

The leaf grid in this thesis is designed for $J \times K$ points, where J and K are 3, 5, 7, 9, or 11. The leaf size supports odd number, because this thesis assumes that the leaf is symmetric with respect to the column. A size of leaf depends on its complicated structure. In Figure the string 5.9 shows the simple plant with its leaves. The source point of leaf is attached to the internode tip. Therefore the second point is attached to the internode. The leaves in this thesis can be resized and rotated following the user adjustment. The simple L-system of Figure 5.9 is $I[-iL][+iL]A$. The previous symbol of the leaf symbol L must be only attached to the internode I, the short internode I, the petiole P, or the short petiole p. Therefore, the leaf cannot be separated from the symbol I, the short internode i, the petiole P, or the short petiole p.

Flower

A flower of plant in this dissertation is defined only rounded flower. The number of petals will be changed by users. The petal structure is similar to the leaf structure. It has a source point, set of points, set of triangular polygons. For example, Figure 5.24 shows a simple plant with its leaves and flower with the L-system as follows:

$I[-iL][+iL]IiF$

The plant consists of eight components, there are two internodes I and one short internode i for main stem. There are two petioles, left petiole $[+iL]$, and right petiole $[-iL]$ with their leaves L. The flower F must follow the internode I, the short internode i, the petiole P, or the short petiole p. The symbol F is imported from the flower library as well as the leaf library.

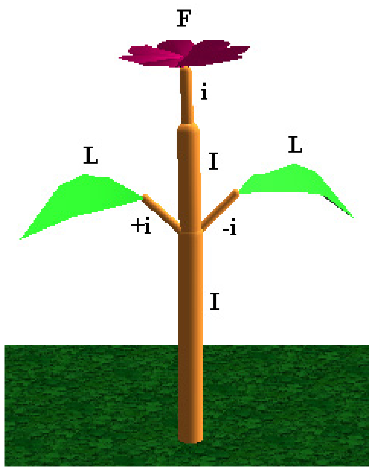


Figure 5.24: A simple plant with its leaves and flower : $I[-iL][+iL]IiF$.

Root

Root components in *PlantVR* are implemented by the symbols I, i, P, p, and A. The structure of each component are used as shoot part without leaves and flowers.

Environments

PlantVR provides environmental elements such as ground, sky, fog, grass and soil for making the visualization of the scene more realistic.

Texture mapping

Texture is a set to images, e.g. leaf, trunk, ground, sky, grass, soil. The leaf texture is designed for 2D texture mapping to 3D space of leaf polygon. The texture size must be $2^n \times 2^m$ pixels, where n, m are integer, 3, 4, 5,..., k. It is cropped from the actual leaf images and set to the appropriate size. The textures can be changed the color by user adjustment. The user can create a new texture or select from the given texture library. The texture coordinate is mapped to a range of [0,1]. For example, if the texture size is 64x64 pixels, the texture coordinate will map to 1×1. Figure 5.25 shows the mapping coordinate. The texture coordinate (0,0) is mapped to p0 (0,0), (63,0) is mapped to p1(1,0), (0,63) is mapped to p2(0,1), (63,63) is mapped to p3(1,1), (31,31) is mapped to p4(0.5,0.5) as well as other points.

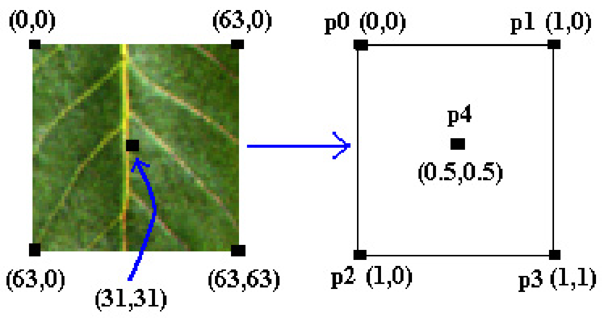


Figure 5.25: The texture coordinate.

Figure 5.26 shows a texture mapping from a cropped texture to a leaf polygon. A leaf coordinate in three-dimensional space (Figure 5.26(a)) is mapped to range of [0,1] coordinate in XY-plane (Figure 5.26(b)) corresponding to texture coordinate in Figure 5.26(d), and the texture coordinate is mapped to the actual leaf coordinate as Figure 5.26(c).

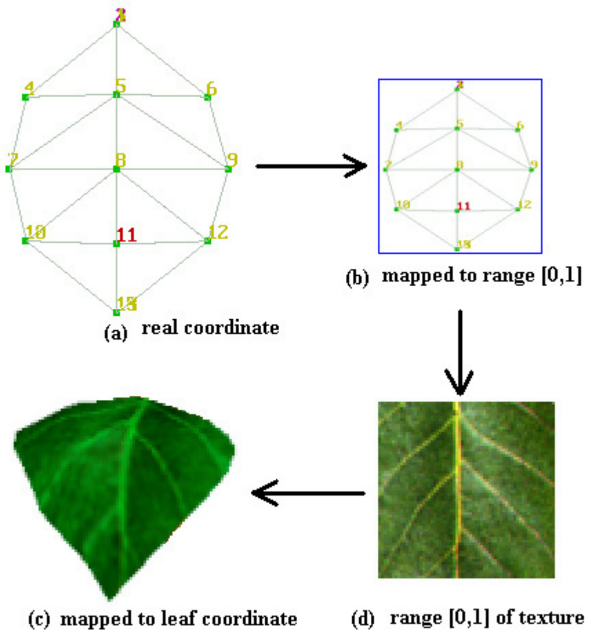
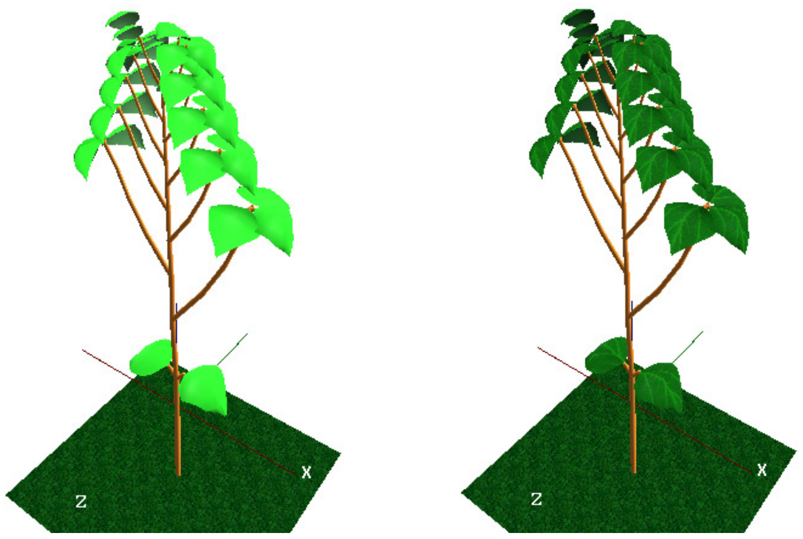

(a) real coordinate
(b) mapped to range [0,1]
(c) mapped to leaf coordinate
(d) range [0,1] of texture

Figure 5.26: The texture mapping procedure.

The flow of leaf texture mapping is shown in Figure 5.26. The texture is acquired by scanners or cameras from actual leaves, and cropped to an appropriate size, then mapped to the 3D leaf polygon. The 3D leaf can be changed the material color by users.

In the case of soybean, the texture of the soybean leaf is cropped from the actual soybean in this thesis experiment. It is mapped to the leaf polygon of soybean plant. Figure 5.27(a) visualizes the Soybean prototype in Section 5.1.4 using six iterations as well as in Figure 5.27(b) is added the soybean texture to its leaves.



(a) soybean at 6th iteration (b) soybean with leaf texture mapping

Figure 5.27: Soybean's structure at 6th iteration without leaf texture mapping (a), and with texture mapping (b).

Changing of Plant Components in Time

Plant components have been changed their properties, e.g. color, angle, size, in time. *PlantVR* provides the interpolation color of plant components. The mapping functions of red R, green G, and blue B convert the object color (R, G, B) to three channels R, G, B, and interpolate each color channel depending on time. Finally, all channels of R, G, B are combined to the object color (R, G, B) for visualizing at the current time. Angle of each component depends on the property in L-system code. Leaf angle can be interpolated in the angle corresponding to XYZ-axes. Size of component changes with growth function at each time step. Figure 5.28 illustrates the plant growth and changing color of plant components in time.

Programming language

PlantVR is a software for simulation and visualization of plant growth. It works on windows operating systems based on *OpenGL* library. The implemented code is based on Pascal object oriented language namely Delphi.

User interface

Delphi provides a fast user interface implementation based on graphical user interface (GUI). The objects in Delphi allow users to interact with 2D and 3D rendering view port linking to *OpenGL*.

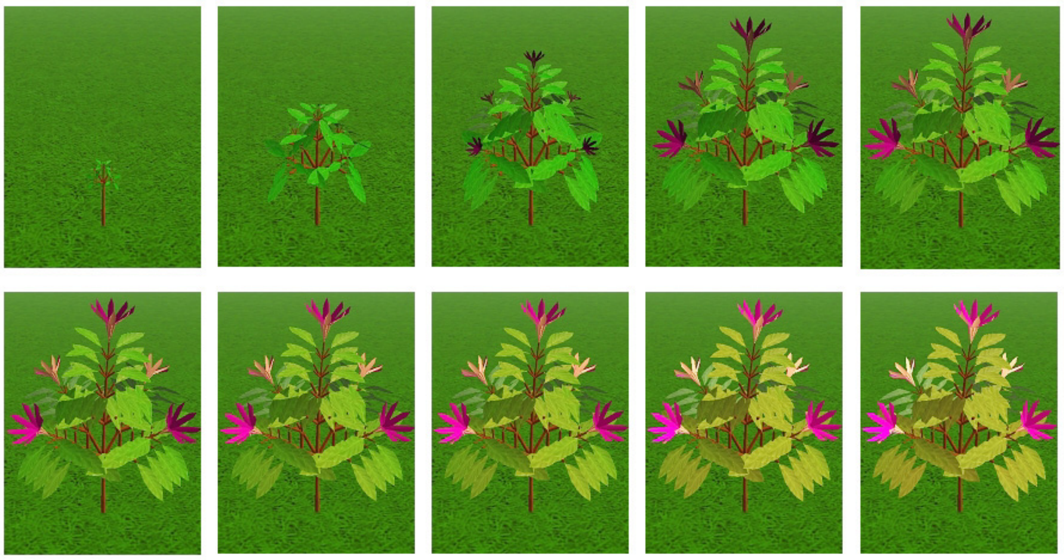


Figure 5.28: Plant growth and changing color of plant in time space.

OpenGL implementation

OpenGL is free library for graphic implementation in both 2D and 3D based on simple geometry such as point, line, triangle, polygon. *OpenGL* provides also advanced features for environmental effects, i.e. blending function, transparency, fog, etc.

Visualization results

Figure 5.29 and Figure 5.30 show some results from *PlantVR* software. The gallery of some additional results is shown in Appendix C.

5.1.7 Model Evaluation

The *PlantVR* software presented in this dissertation has the capability to adjust the parameters of the plant model interactively. It allows users to verify the production rules and to modify the appearance of the graphical image of the generated plant in real time mode. In addition, if there are any flaws presented in the plant model due to the production rules, the users can edit the rules and recompile the L-system description.

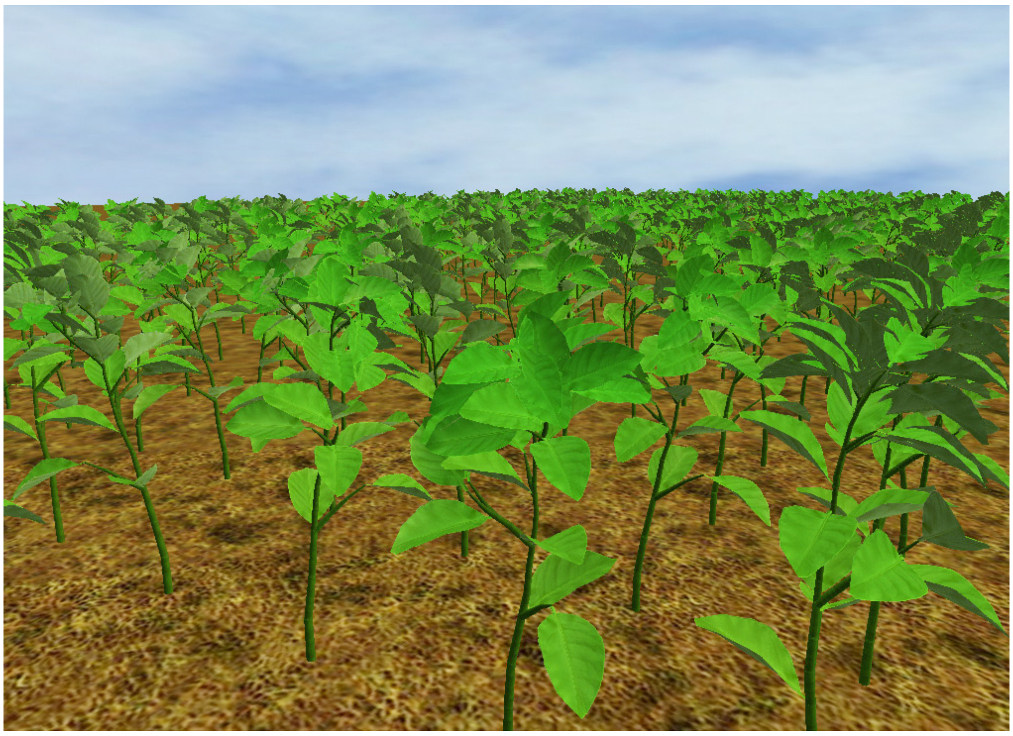


Figure 5.29: Plant field.

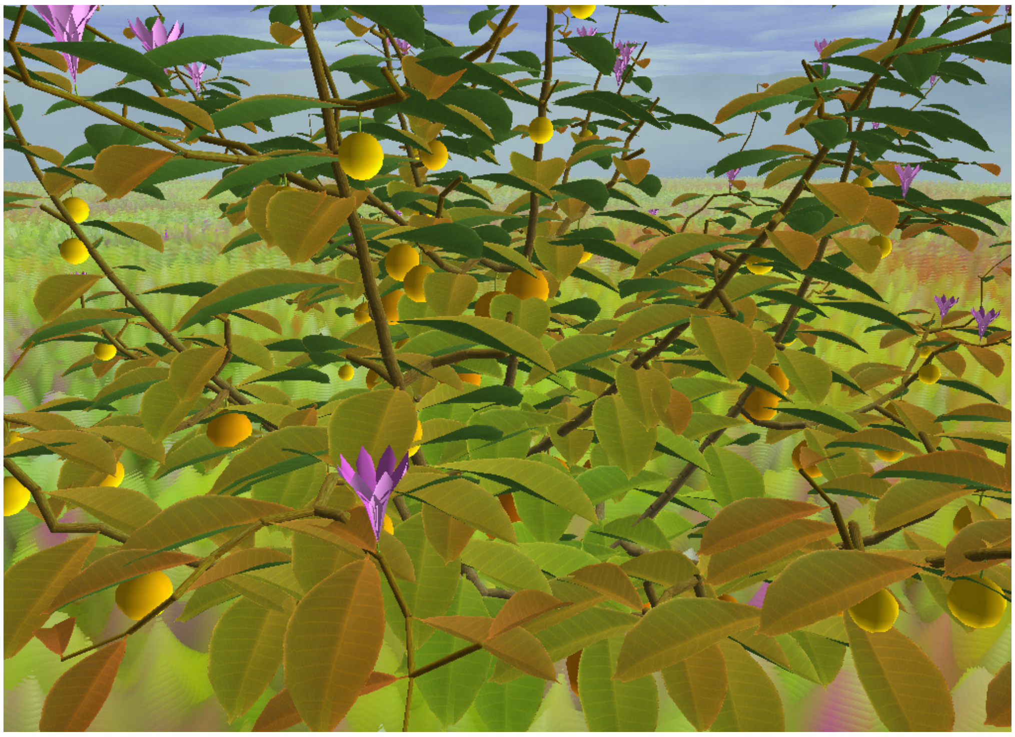
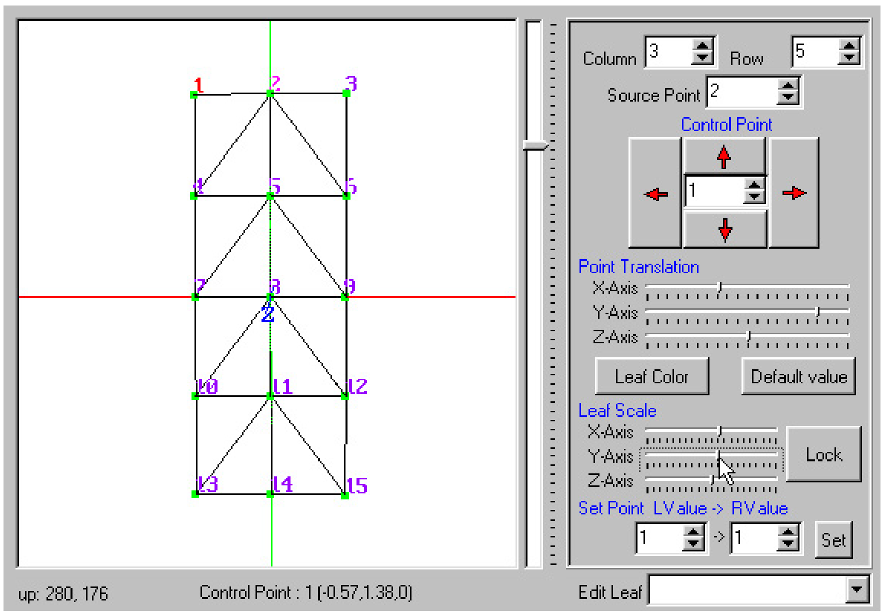


Figure 5.30: A simulated plant with flowers and fruits in *PlantVR*.

5.2 LeafDS Software

The section describes the *LeafDS* (Leaf Design) software for creating the 3D leaf surface (file extension .sur) used in *PlantVR* as a predefined leaf surface of leaf and petal. The leaf surface is based on 2D mesh (leaf plane). It allows the user to move and modify each vertex to the appropriate position. The *set point* operation provides to set a vertex to the target position for some unnecessary vertices.


Column 3 Row 5

Source Point 2

Control Point 1

Set Point LValue -> RValue 1 -> 1

up: 280, 176

Control Point : 1 (-0.57,1.38,0)

Edit Leaf

Figure 5.31: *LeafDS* software : Initial leaf grid.

Figure 5.31 shows the setting up of 2D mesh for 3 columns and 5 rows. The point number 2 is defined as a source point which is attached to the petiole or internode. We now show an example for creating a simple leaf surface. Firstly, the point number 1 and 3 are set to the point number 2, and the point number 13 and 15 are set to the point number 14. Then we now get the result in Figure 5.32.

All points on the leaf surface can be moved to any direction for creating the 3D leaf surface. Figure 5.33 shows leaf surface after editing by user in wire frame. The leaf surface with polygon fill is illustrated in Figure 5.34. The *LeafDS* allows the user to load and modify the existing leaf library in .sur file format and save it back to the library.

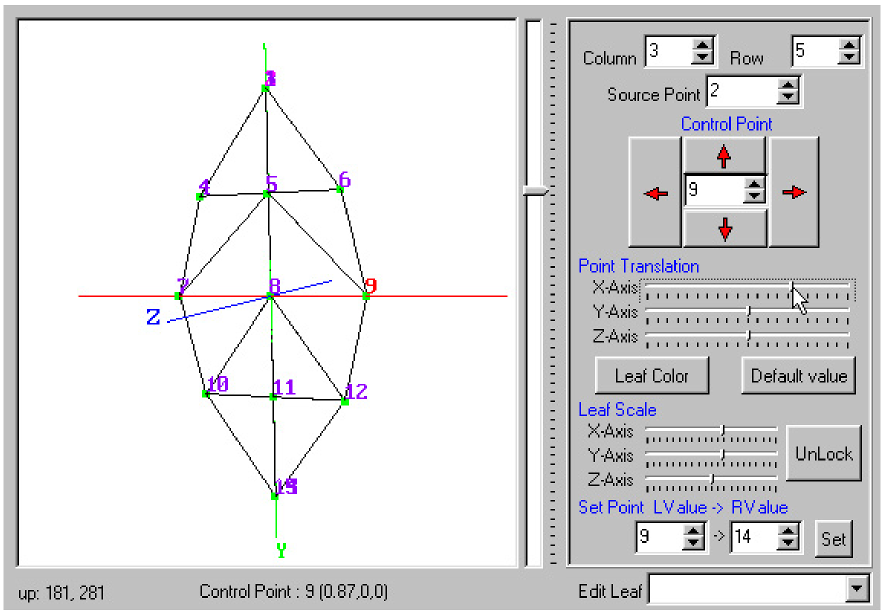


Figure 5.32: *LeafDS* software : Leaf control points.

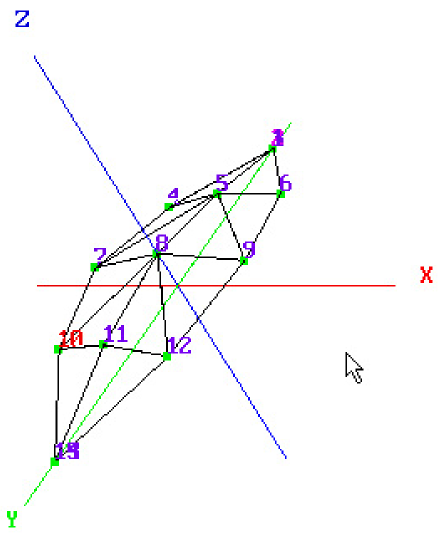


Figure 5.33: *LeafDS* software : Leaf surface in wire frame.

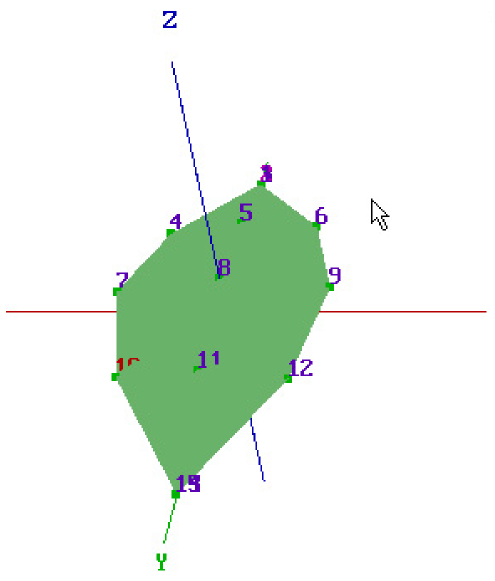


Figure 5.34: *LeafDS* software : Leaf surface with polygon fill.

The leaf surface library in *LeafDS* is imported to *PlantVR* for optional leaf and petal library to attach the leaf shape to the branch structure. The growth of leaf and petal is controlled in sigmoidal growth functions from data collection.

5.3 LeafVein Software

LeafVein software was designed to generate the leaf vein structure based on PTM (see Chapter 3). The input of *LeafVein* requires leaf margin from scanned leaf or other resources. Figure 5.35 illustrates the input leaf margin, leaf edge detection and leaf margin representation.

The results from leaf vein generation are used as leaf textures and mapped on 3D leaf surface (result from *LeafDS* software). *LeafVein* was mainly implemented for simulating leaf vein structures, but trees and roots were also included in the software for showing in rough detail. Figure 5.36 shows simulated leaf vein of ivy leaf.

The vein resolution parameter is controlled by Bezier curve as shown in Figure 5.37.

Figure 5.38 shows an example of simulated leaf vein from LeafVein software and used in PlantVR for leaf texture mapping.

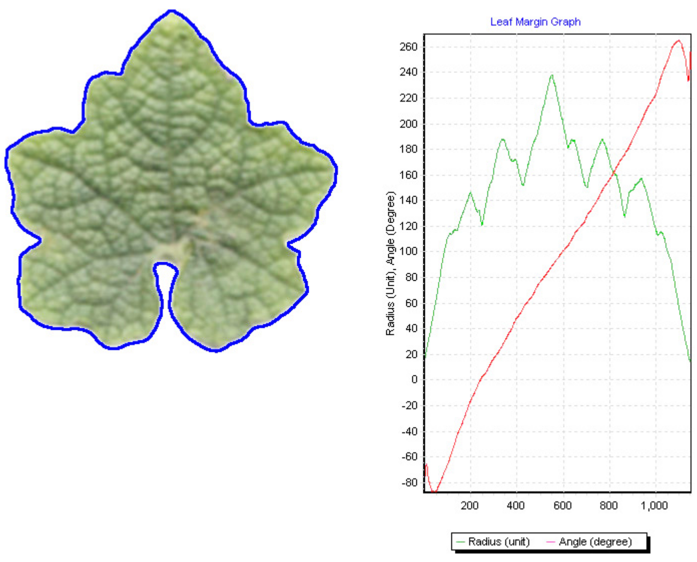
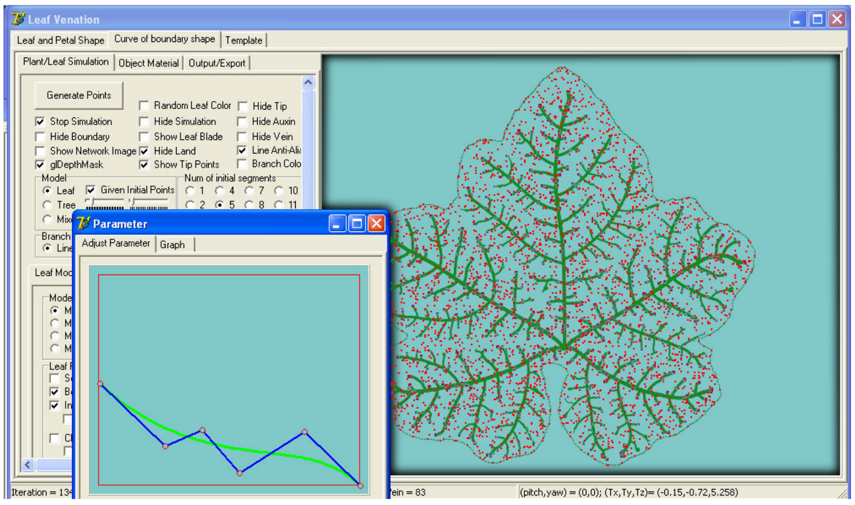


Figure 5.35: Leaf edge detection and leaf margin representation.



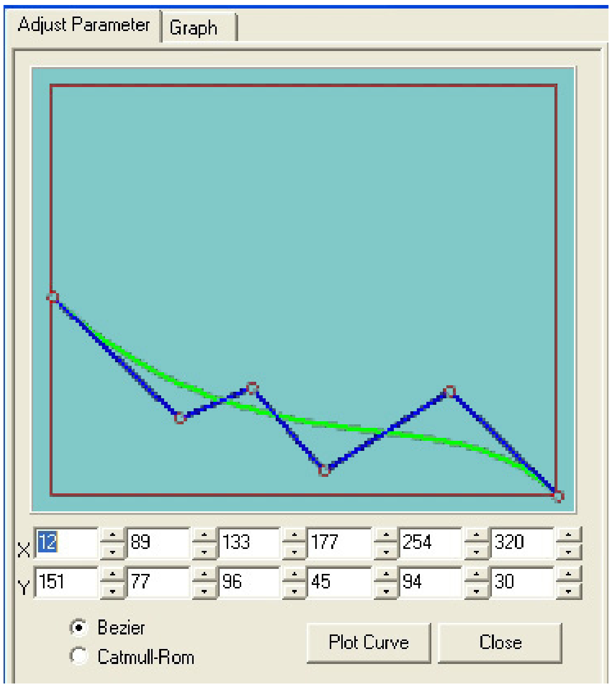
[✓] Stop Simulation
[ ] Hide Boundary
[ ] Show Network Image
[✓] glDepthMask
[ ] Random Leaf Color
[ ] Hide Simulation
[ ] Show Leaf Blade
[✓] Hide Land
[✓] Show Tip Points
[ ] Hide Tip
[ ] Hide Auxin
[ ] Hide Vein
[✓] Line Anti-Alia
[ ] Branch Colo

Model: [✓ Leaf] Tree Mix
[✓] Given Initial Points

Num of initial segments: 1 4 7 10 2 [✓ 5] 8 11

Iteration = 134 ein = 83 (pitch,yaw) = (0,0); (Tx,Ty,Tz)= (-0.15,-0.72,5.258)

Figure 5.36: *LeafVein* software.



X 12 89 133 177 254 320
Y 151 77 96 45 94 30

[✓ Bezier] Catmull-Rom

Figure 5.37: Leaf vein resolution.

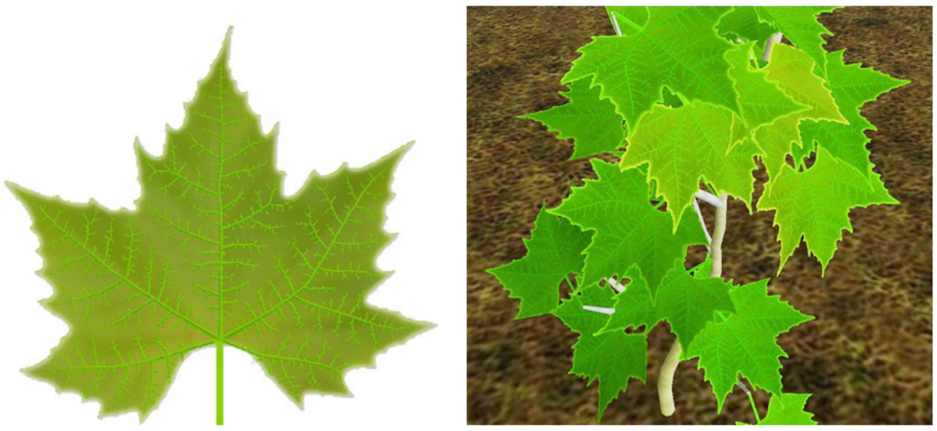


Figure 5.38: Simulated leaf vein (left) by *LeafVein* software and simulated plant with texture mapping (right) in *PlantVR*.

5.4 BranchRecon Software

The *BranchRecon* software is implemented to solve the inverse problem of L-system (see section 2.7). The flowchart of the software was shown in Figure 5.39. The procedure starts from given 2D/3D input data, proceeds a preprocessing for noise reduction, executes region or volume growing method, runs thinning and skeletonization processes, generates network and resolution reduction, produces L-systems rules or L-string code, and print L-system string for further uses. Figure 5.40 shows the software after entering an input branching structure image.

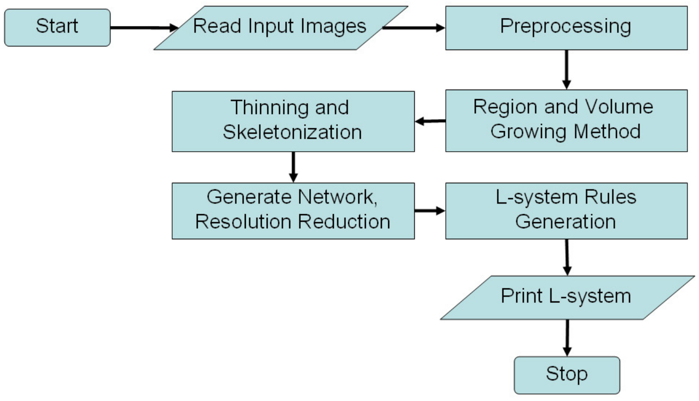
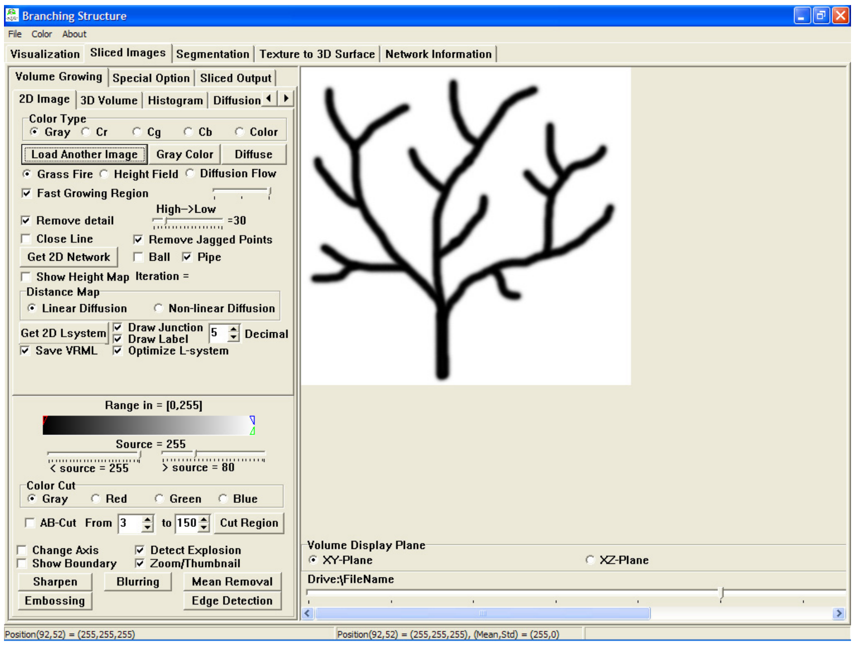

Start
Read Input Images
Preprocessing
Region and Volume Growing Method
Thinning and Skeletonization
Generate Network, Resolution Reduction
L-system Rules Generation
Print L-system
Stop

Figure 5.39: Flow diagram of reconstruction process of branching structure described in algorithm 2.2 [Chuai-Aree et al., 2007].

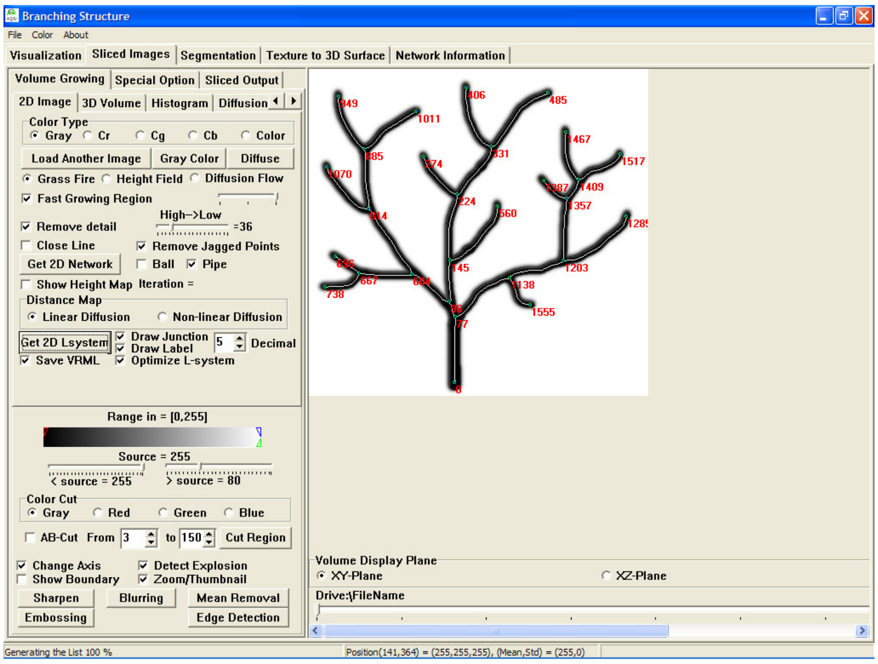


Color Type: [✓ Gray] Cr Cg Cb Color
[✓ Grass Fire] Height Field Diffusion Flow
[✓ Fast Growing Region]
High–>Low
[✓ Remove detail] =30
Close Line [✓ Remove Jagged Points]
Ball [✓ Pipe]
Show Height Map Iteration =
Distance Map: [✓ Linear Diffusion] Non-linear Diffusion
[✓ Draw Junction] [✓ Draw Label] 5
[✓ Save VRML] [✓ Optimize L-system]
Range in = [0,255]
Source = 255
< source = 255 > source = 80
Color Cut: [✓ Gray] Red Green Blue
AB-Cut From 3 to 150
Change Axis [✓ Detect Explosion]
Show Boundary [✓ Zoom/Thumbnail]
Volume Display Plane: [✓ XY-Plane] XZ-Plane
Drive:\FileName

```
Position(92,52) = (255,255,255)          Position(92,52) = (255,255,255), (Mean,Std) = (255,0)
```

Figure 5.40: *BranchRecon* software with input branch structure.

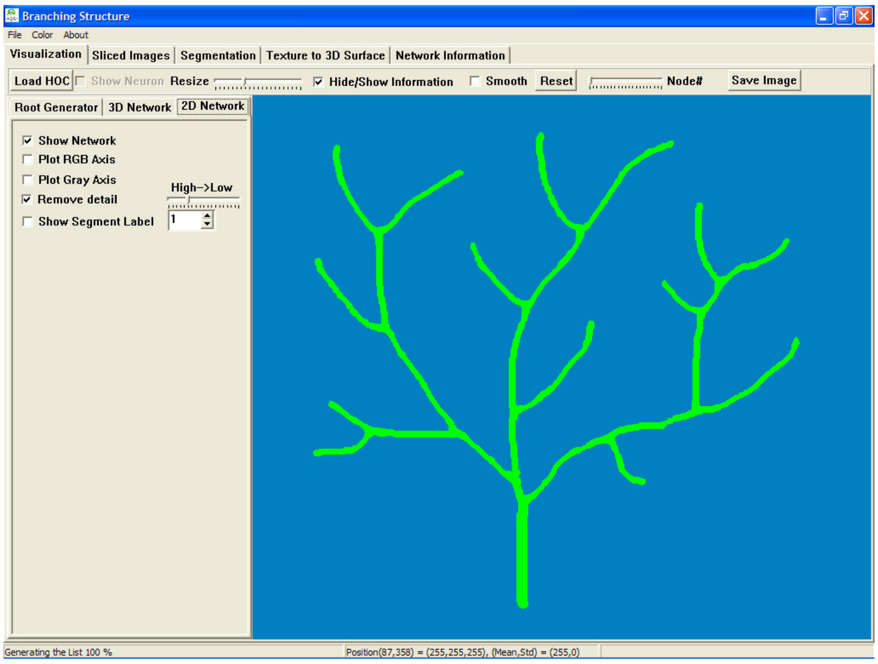
After applying the region growing algorithm and the skeletonizing algorithm, the skeleton of branching structure is constructed and labelled all appropriate nodes (see Figure 5.41). The 3D structure of reconstructed network is visualized in Figure 5.42. provides



Generating the List 100 %

Position(141,364) = (255,255,255), (Mean,Std) = (255,0)

Figure 5.41: Input branch structure and its skeleton with node labels.



Generating the List 100 %

Position(87,358) = (255,255,255), (Mean,Std) = (255,0)

Figure 5.42: Skeleton of reconstructed structure.

Figure 5.43 and Figure 5.44 illustrate 3D view of reconstructed rat neuron structures. 3D structures can be converted from .HOC file to .NLS (Network L-system) file format.

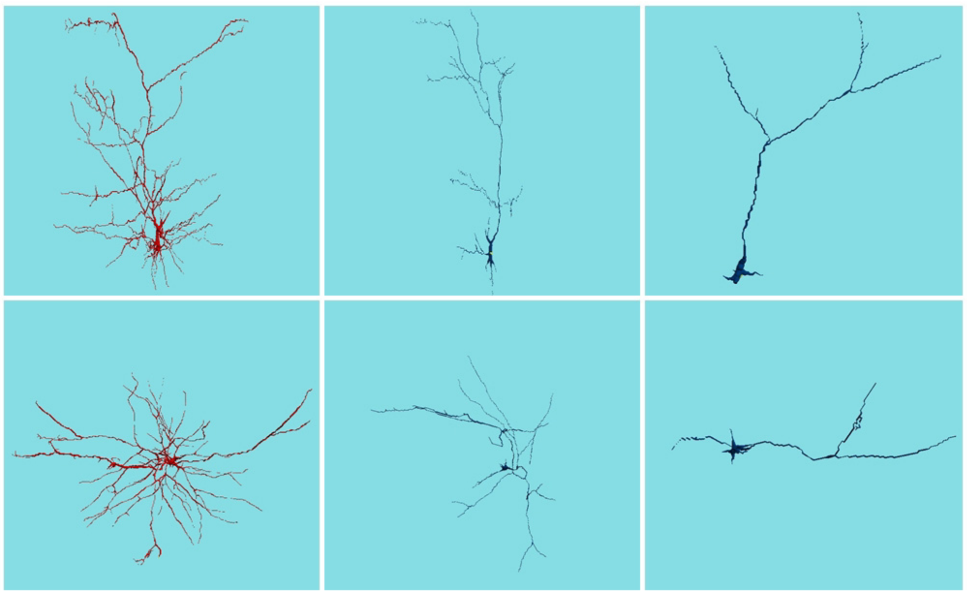


Figure 5.43: Some rat neuron structures reconstructed by P. J. Broser [Broser, 2006] and visualized in *BranchRecon* and converted to .NLS (Network L-systems) file format in form of L-systems description, 1st row (side view) and 2nd row (top view).

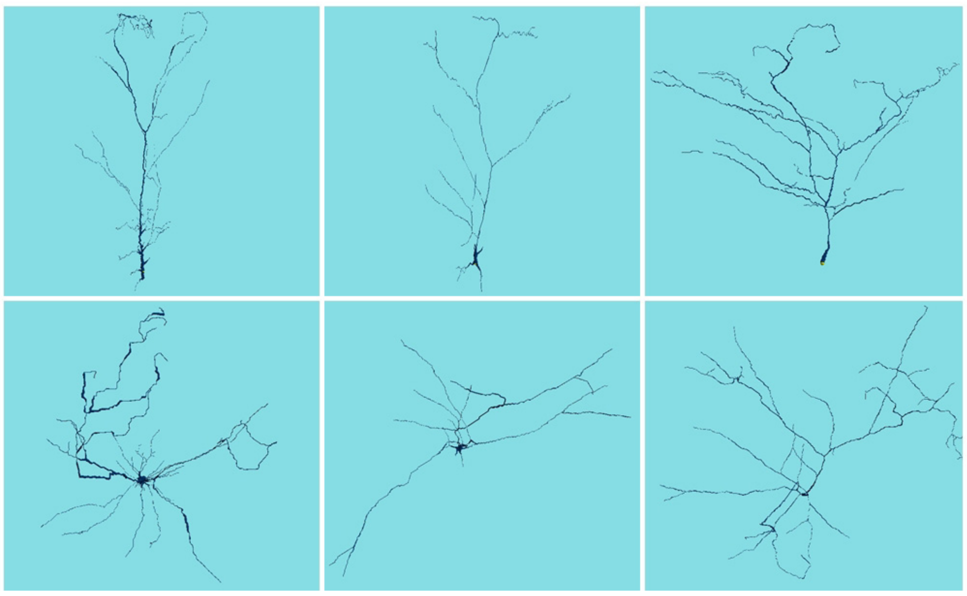


Figure 5.44: Some rat neuron structures reconstructed by P. J. Broser [Broser, 2006] and visualized in *BranchRecon* and converted to .NLS (Network L-systems) file format in form of L-systems description, 1st row (side view) and 2nd row (top view).

The side view and top view of reconstructed canola root structures are shown in Figure 5.45 and Figure 5.46. 3D structures can be converted from .MV3D file to .NLS (Network L-system) file format.

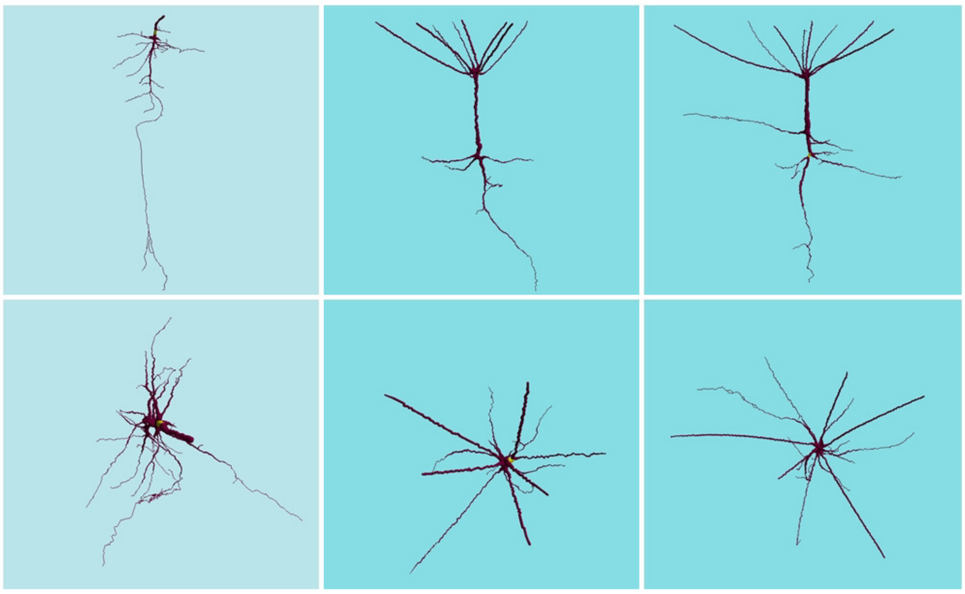


Figure 5.45: Some canola root structures reconstructed by P. Kolesik [Kolesik, 2004] and visualized in *BranchRecon* and converted to .NLS (Network L-systems) file format in form of L-systems description, 1st row (side view) and 2nd row (top view).

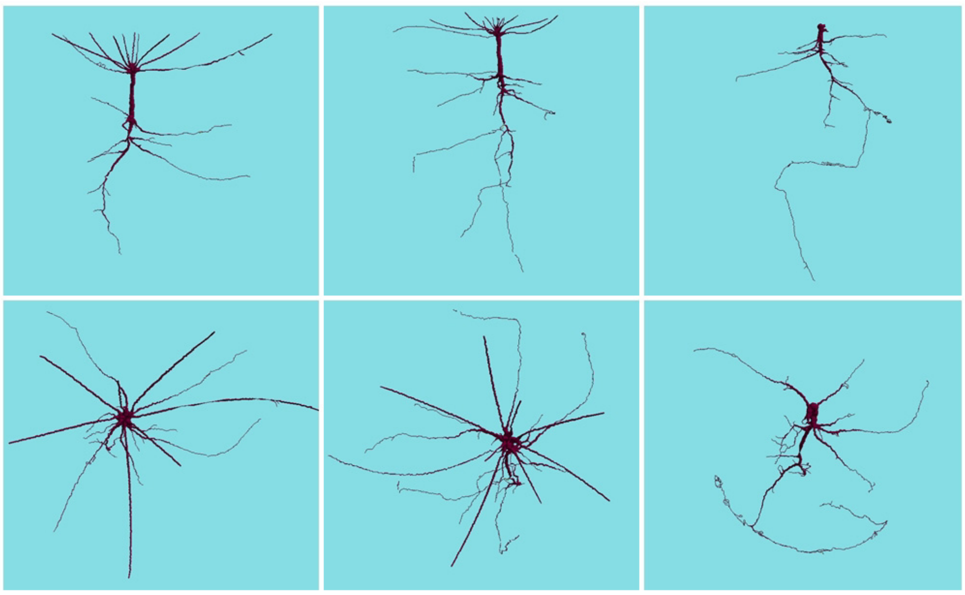
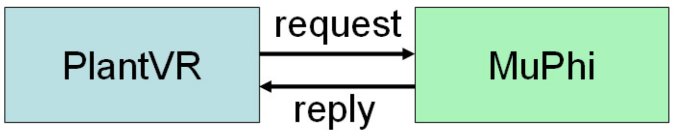


Figure 5.46: Some canola root structures reconstructed by P. Kolesik [Kolesik, 2004] and visualized in *BranchRecon* and converted to .NLS (Network L-systems) file format in form of L-systems description, 1st row (side view) and 2nd row (top view).

5.5 PlantVR - MuPhi Communication

This section describes the interaction between PlantVR software and *MuPhi*(or $\mu\varphi$) library. *MuPhi* library was developed by O. Ippisch [Ippisch, 2001] at the *Institut für Parallele und Verteilte Systeme* (IPVS), University of Stuttgart. It is used to calculated the water flow in soil volume. In this section, the explanation of the communication between both softwares is shown in Figure 5.47.



1.Prepare input file input.dat		
2.Call MuPhiInitialize	$\longleftrightarrow$	Initialize input file input.dat
3.Define the gridsize of domain	$\longleftrightarrow$	Initialize the gridsize
4.Call MuPhiSetSources(sources)	$\longleftrightarrow$	Set the source/sink term
5.Call MuPhiDoTimeStep(dt)	$\longleftrightarrow$	Calculate water flow in the domain
6.Call MuPhiGetThetaArray(myTheta)	$\longleftrightarrow$	Return array of *Theta* values
7.Call MuPhiGetPotentialArray(myPotential)	$\longleftrightarrow$	Return array of *Potential* values
8.Use *myTheta* and *myPotential* for plant		
9.Calculate plant growth and water uptake		
10.Do 4. to 9. until satisfies the condition		
11.Call MuPhiFinished()	$\longleftrightarrow$	Finalize the MuPhi

Figure 5.47: Flow diagram of PlantVR-MuPhi interaction.

There are 11 consecutive steps to proceed the communication for all calculations between PlantVR-MuPhi. All steps are describes in detail as follows.

1. Prepare input file input.dat

2. Call MuPhiInitialize

3. Define the grid size of domain

4. Call MuPhiSetSources(sources)

5. Call MuPhiDoTimeStep(dt)

6. Call MuPhiGetThetaArray(myTheta)

7. Call MuPhiGetPotentialArray(myPotential)

8. Use *myTheta* and *myPotential* for plant

9. Calculate plant growth and water uptake

10. Do step 4 to step 9 until it satisfies the condition of the target time step.
11. Call MuPhiFinished()

5.6 Discussion and Conclusion

This chapter introduced our software packages for modeling, simulation and visualization of plant growth. We have introduced five softwares programmed within the scope of this thesis, e.g. *PlantVR*, *LeafDS*, *LeafVein*, *BranchRecon*, and *PlantVR-MuPhi*. *PlantVR* software is a software for generating the plant structures using L-systems based on bracketed L-system, stochastic L-system, parametric L-system and particle transportation method (PTM) based on the environmental particles in both shoot and root parts (see Chapter 2 and Chapter 3). It simulates the growth of plant in time both in shoot and root. Leaf3D software is programmed to create the 3D leaf surface used in *PlantVR* as a predefined surface. *LeafVein* software simulates the leaf veins using PTM by defining leaf boundary and the source point P_0 of leaf (see Chapter 3). *BranchRecon* software is developed to solve the inverse problem of L-system by giving the input image contained branching structure and returning the L-system description which can be used in *PlantVR*, Finally, *PlantVR-MuPhi* is a communication library between the *PlantVR* and *MuPhi* library for calculating the water flow in soil volume and send the numerical solutions to *PlantVR*.

Chapter 6

Conclusion and Outlook

"Structure without function is only half, but structure with function is one."

6.1 Conclusion

In this thesis, we proposed two methods for generating plant structures using L-systems and PTM. We also presented the algorithm for reconstructing the network structure from data acquisition such as camera, scanner, CT-scanner, the so-called method *inverse problem of L-systems*.

Plant structures can be modelled using deterministic and stochastic L-systems. A set of simple production rules after decomposition, the plant structure was generated to the complex plant or tree structures. The growth functions based on sigmoidal function were applied to all plant components and controlled by starting time. The whole plant structure is growing depending on the global time. Each plant component has their own local time for starting the growth. We used Levenberg-Marquardt algorithm to estimate the parameters of the N-pulses sigmoidal growth function. To calibrate the model, we acquired real data from the experiments.

To generate trees quickly, we selected the PTM as algorithm which is a proper tool to produce complicated structures starting from given the shape of shoot and root parts. It also generates leaf veins for fixed and growing leaf margins. Using the interface we prescribed spline curves to control the branching process of the network density. The given leaf margin and tree boundary shape play a big role for tree network construction. Varying the vein velocities of the main stem and the lateral branches, different network structures and shapes of trees could be reproduced.

Solving the inverse problem of L-systems, we developed a tool to reconstruct the automatic L-string code. It can be used both in 2D image and 3D volume data. The presented reconstruction algorithms can be adapted to a wide scope of applications in medicine, e.g. by describing the vessel branching in human body. The input data are required to proceed the pre-processing for noise reduction using anisotropic diffusion filtering. We applied the region and volume growing methods to reduce the object to its skeleton. The network structure is reconstructed and represented in its L-string code. Furthermore, we offer a tool to reduce the resolution of the network.

The nutrient diffusion and water flow in soil play a big role for root growth. In this thesis, we simply calculated nutrient diffusion and water flow in soil interacting with the root system. We used *MuPhi* library written by O. Ippisch for computing the water flow and root growth simultaneously. The library is not completed for solute transport with water flow, but the solute transport will be developed in the future.

Finally, five softwares are developed for investigatation in this thesis, e.g. *PlantVR*, *LeafVein*, *BranchRecon*, *LeafDS* and *PlantVR-MuPhi*. They are working on windows operating systems.

6.2 Outlook

As in any research project there is a number of aspects which were not treated and that shall be the object of future research. Here, we are going to discuss some of them briefly.

We see the following possibilities and the need for improvements in future research:

1. improving the mathematical modeling of processes inside the plant based on physiological information,
2. developing new features and tool for the L-systems in order to treat other classes of structures,
3. introducing a hierarchical structure allowing to pass between different scales,
4. transfer of the concepts and software tools to other applications, e.g. vessels systems, systems of neurons in medicine,
5. improving the coupling of the different compartments in the system, including investigation on PTM,
6. including the micro-ecology interacting with the root system,
7. combining the photosynthesis process in the plant growth,
8. interacting of different plant species in root and shoot parts, and
9. combination of solute transport with water flow in PlantVR-MuPhi interface.

Not all of these topics are of the same importance and posing the same difficulties. Improving the modeling of the physiological processes in the plant has the highest priority. In fact, it is a long term project for the community which needs intensive cooperation between the biologists and specialists in modeling and simulation. We see a good chance to transfer our approaches to structures which can be represented by networks. We already started cooperation with scientists in medicine investigating network of blood vessels and neurons. It is already a challenge to characterize and to quantify their geometries.

Computing processes on such complex networks is even more challenging. Those topics which belong to the more technological parts seem to be easier solvable and we hope to achieve the demanded improvement in the near future.

Appendices

Appendix A

Algorithms

This appendix describes the rewriting algorithm and data structure of plant structures presented in Section 2.2.

A.1 Rewriting Algorithm

We define a production consisting of a predecessor *Pred* and a successor *Succ* as the following record declaration.

```
Type  production  =  record
      Pred        :  Character
      Succ        :  String
End
```

The variables are defined as follows:

i, j, and k	:	a positive integer,
iter	:	a number of the L-system iterations,
rulenum	:	a number of production rules,
allstr	:	a last L-system string,
rule	:	an array of the production rule,
prev	:	a previous string of current character,
last	:	a next string of current character,
Endrule_Check	:	a flag of the endrule keyword, it is true if there are endrule keyword,
Length()	:	a length function return the number of argument string,
EndRuleNum	:	a number of endproduction rules,
Endrule	:	an array of the production rule.

The rewriting algorithm of the L-system is given below.

Rewriting Algorithm

```
BEGIN
    FOR i := 1 TO iter DO
    BEGIN
        FOR j := 1 TO rulenum DO
        BEGIN
            k := 1
            WHILE (k ≤ length(allstr)) DO
            BEGIN
                IF allstr[k]=rule[j].pred THEN
                BEGIN prev := Copy(allstr,1,k-1)
                    last := Copy(allstr,k+1,length(allstr)-k)
                    allstr := prev + rule[j].succ + last
                    k := k+length(rule[j].succ)
                END
                ELSE
                    k:=k+1
            END
        END
    END

    IF EndRule_Check THEN
    BEGIN
        FOR j := 1 TO EndRuleNum DO
        BEGIN
            k := 1
            WHILE (k ≤ length(allstr)) DO
            BEGIN
                IF allstr[k]=EndRule[j].pred THEN
                BEGIN
                    prev := Copy(allstr,1,k-1)
                    last := Copy(allstr,k+1,length(allstr)-k)
                    allstr := prev + EndRule[j].succ + last
                    k := k+length(EndRule[j].succ)
                END
                ELSE
                    k:=k+1
            END
        END
    END
END.
```

A.2 L-system interpretation algorithm

After executing the rewriting algorithm, the L-system string is interpreted by the L-system interpretation algorithm presented Section 2.3.1. The structure and variables are defined as follows:

The structure of plant is given the following type:

```
Type  TTree                          =  record
      StringType                     :  Char;
      Length                         :  Real;
      BigT                           :  Real;
      T                              :  Real;
      t_start,t_stop                 :  Real;
      Angle_H, Angle_U, Angle_L      :  GLFloat;
      Angle                          :  GLFloat;
      angle_azimut                   :  GLFloat;
      allchild                       :  Integer;
      child                          :  array[1..maxchild] of integer;
      myparent                       :  Integer;
      mylevel                        :  Integer;
      Leaf                           :  Boolean;
      Flower                         :  Boolean
End
```

The description of *TTree* attribute is given as follows:

Attribution name	**Meaning**
StringType	a symbol of the L-system string,
Length	a length of internode, or petiole,
BigT	a maximum life time of internode or petiole,
t	a time variable between the internode or petiole life cycle,
t_start,t_stop	a start time and stop time of the internode or petiole,
Angle_H, Angle_L, Angle_U	a angle respect to X, Y, Z axis, respectively,
allchild	a number of all child of internode or petiole,
child	an array of child internode or petiole,
myparent	a parent of the internode or the petiole,
mylevel	a level of the internode or the petiole,
Leaf	a flag for internode leaf, it is TRUE if there is a leaf,
Flower	a flag for internode flower, it is TRUE if there is a flower,

The variables are defined as follows:

Variable name	**Meaning**
CountNode	a number of symbols I, A, P, i, and p,
Showmessage()	a procedure that display the argument message,
Exit	an exiting procedure,
NOT	a boolean operator,
Maxatree	a maximum length of the L-system string,
Atree	an array of plant TTree type,
Maxatree	a maximum number of symbols I, A, P, i, and p,
Current_Str[k]	a k^{th} symbol of the L-system string,
Main_Stem[k]	an array of the main stem order at kth level,
MainStemOfBracket[k]	a number of bracket [at the kth level,
Bracket	a number of Bracket [,
Ang_U	an angle respect to Z axis,
Ang_L	an angle respect to Y axis,
Ang_H	an angle respect to X axis,
TopStack	a length of stack,
Stack_Angle_U[TopStack]	a Topstack angle respect to Z axis,
Stack_Angle_L[TopStack]	a Topstack angle respect to Y axis,
Stack_Angle_H[TopStack]	a Topstack angle respect to X axis,
RU_Change	a flag of turtle angle on Z axis,
RL_Change	a flag of turtle angle on Y axis,
RH_Change	a flag of turtle angle on X axis,
AllNode	a number of all component I, A, P, i, and p,
Current_Node	a current symbol of the L-system string,
TopMainStem	a current length of the main stem, and
MainStem[TopMainStem]	a main stem order at TopMainStem level.

The L-system interpretation algorithm presented Section 2.3.1 is given below.

L-system interpretation algorithm

```
BEGIN
  CountNode := 0;
  FOR k := 1 TO length(allstr) DO
  BEGIN
    IF allstr[k] IN [‘I’,‘A’,‘P’,‘i’,‘p’] THEN
      CountNode := CountNode + 1
    IF NOT (allstr[k] in [‘I’,‘i’,‘A’,‘P’,‘p’,‘L’,‘F’,‘[’,‘]’,‘-’,‘+’,‘/’,‘\’,‘^’,‘&’,‘|’]) THEN
    BEGIN
      Showmessage(allstr[k]+‘is not in {I,i,A,P,p,L,F,[,],/,\,+,-,ˆ,&,| }’)
      EXIT
    END
  END
  IF CountNode > Maxatree THEN
  BEGIN
    Showmessage('The String is too long.')
    EXIT
  END

  FOR k := 1 TO CountNode DO
  BEGIN
    atree[k].Angle_U := 0
    atree[k].Angle_L := 0
    atree[k].Angle_H := 0
    atree[k].Leaf := False
    atree[k].Flower := False
  END

  IF CountNode > MaxaTree THEN
  BEGIN
    Showmessage('The number of node is too much.')
    EXIT
  END
  ELSE
  FOR k:= 1 TO length(allstr) DO
  BEGIN
    Current_Str[k] := 0
    Main_Stem[k] := 0
    MainStemOfBracket[k] := 0
  END

//———— Plant generator start ————
  k := 1
  WHILE k ≤ length(allstr) DO
  BEGIN
    CASE allstr[k] OF
      '[' : BEGIN
            Bracket := Bracket + 1
            MainStemOfBracket[Bracket] := 0

            Ang_U := 0
            Ang_L := 0
            Ang_H := 0

            Stack_Angle_U[TopStack] := Ang_U
            Stack_Angle_L[TopStack] := Ang_L
            Stack_Angle_H[TopStack] := Ang_H
          END
‘]’ : BEGIN
      Bracket := Bracket - 1
      Current_Str[TopStack] := 0 // Clear the top of Stack

      Ang_U := Stack_Angle_U[TopStack]
      Ang_L := Stack_Angle_L[TopStack]
      Ang_H := Stack_Angle_H[TopStack]

      TopStack := TopStack - MainStemOfBracket[Bracket+1]
      MainStemOfBracket[Bracket+1] := 0

      RU_Change := True
      RL_Change := True
      RH_Change := True
    END

‘A’,‘P’,‘I’,‘i’,‘p’ : BEGIN
      AllNode := AllNode + 1
      TopStack := TopStack + 1
      Current_Node := CountNode + 1
      Current_Str[TopStack] := Current_Node

      MainStemOfBracket[Bracket] := MainStemOfBracket [Bracket] + 1

      IF (RU_Change) THEN
      BEGIN
        IF (allstr[k+1] IN [‘I’,‘A’,‘P’,‘i’,‘p’]) THEN
        BEGIN
          RU_Change :=True
          atree[Current_Node].Angle_U := Ang_U
        END
        ELSE
        RU_Change :=False
      END

      IF (RL_Change) THEN
      BEGIN
        IF (allstr[k+1] IN [‘I’,‘A’,‘P’,‘i’,‘p’]) THEN
        BEGIN
          RL_Change :=True
          atree[Current_Node].Angle_L := Ang_L
        END
        ELSE
          RL_Change :=False
      END
```

```
IF (RH_Change) THEN
BEGIN
  IF (allstr[k+1] IN [‘I’,‘A’,‘P’,‘i’,‘p’]) THEN
  BEGIN
    RH_Change :=True
    atree[Current_Node].Angle_H := Ang_H
  END
  ELSE
    RH_Change :=False
END
IF Bracket = 0 THEN // Check for MainStem
BEGIN
  inc(TopMainStem) // = 0 is empty node.
  Main_Stem[TopMainStem] := Current_Node
  IF (allstr[k-1] IN [‘[’,‘I’,‘A’,‘P’,‘]’,‘i’,‘p’]) THEN
    atree[Current_Node].angle := 0
  Ang_U := 0.01*Random(5)
  Ang_L := 0.01*Random(5)
  Ang_H := 0.01*Random(5)
  IF allStr[k] = ‘I’ THEN
    atree[Current_Node].StringType := ‘I’
  ELSE IF allStr[k] = ‘i’ THEN
    atree[Current_Node].StringType := ‘i’
  ELSE
    atree[Current_Node].StringType := allstr[k]
END
ELSE IF allStr[k] IN [‘I’,‘i’] THEN
BEGIN
  IF allStr[k] = ‘I’ THEN
    atree[Current_Node].StringType := ‘P’
  ELSE
    atree[Current_Node].StringType := ‘p’;
  // Reset the angle of Petiole after first Node

  IF allStr[k-1] IN [‘I’,‘i’] THEN
  BEGIN
    IF RU_Change AND (atree[Current_Node1].Angle_U > 0) THEN
      Ang_U := 0.05
    ELSE
      Ang_U := -0.05
    IF RL_Change AND (atree[Current_Node1].Angle_L > 0) THEN
      Ang_L := 0.05
    ELSE
      Ang_L := -0.05
```

```
            IF RH_Change AND (atree[Current_Node1].Angle_H > 0) THEN
                Ang_H := 0.05
            ELSE
                Ang_H := -0.05
        END
    END

    ELSE
        atree[Current_Node].StringType := allstr[k]

    // Set Leaf attach to this node
    IF allstr[k+1] = ‘L’ THEN
    BEGIN
        atree[Current_Node].Leaf := True
        k :=k+1
    END

    // Set Flower attach to this node
    ELSE
    IF allstr[k+1] = ‘F’ THEN
    BEGIN
        atree[Current_Node].Flower := True
        k := k+1
    END

    atree[Current_Node].myparent := Current_Str[TopStack-1]

    atree[Current_Node].Angle_U := Ang_U
    atree[Current_Node].Angle_L := Ang_L
    atree[Current_Node].Angle_H := Ang_H

    atree[Current_Node].mylevel := Bracket
  END

‘−’ : BEGIN
        Ang_U := Ang_U - sigma
        RU_Change := True
    END

‘+’ : BEGIN
        Ang_U := Ang_U + sigma
        RU_Change := True
    END

‘|’ : BEGIN
        Ang_U := Ang_U + 180.0
        RU_Change := True
    END
```

```
        '&' : BEGIN
                 Ang_L := Ang_L + sigma
                 RL_Change := True
              END
        '^' : BEGIN
                 Ang_L := Ang_L - sigma
                 RL_Change := True
              END
        '\' : BEGIN
                 Ang_H := Ang_H + sigma
                 RH_Change := True
              END
        '/' : BEGIN
                 Ang_H := Ang_H - sigma
                 RH_Change := True
              END
        END
        k := k + 1
      END
  END.
```

A.3 Stochastic Structure Implementation and Natural Selection

The stochastic L-system presented in Section 2.3.3 is applied to L-systems prototype based on natural selection.

The attributions of production rule in the stochastic L-system are defined as follows:

```
Type  production  =  record
      ruleno      :  Integer;
      pred        :  String;
      succ        :  array[1..MaxSuccessor] of String;
      prob        :  array[1..MaxSuccessor] of Single;
End
```

The detail of each attribution is given in the following table.

Attribution name	**Meaning**
ruleno	all successor rules,
pred	predecessor of production rule in stochastic L-system,
succ	array of all successors in the same production rule,
prob	array of probability value of each successor,

The variables are defined as follows:

Variable name	**Meaning**
MaxBall	a maximum number of ball for natural selection process,
StochRule	stochastic rule,
Ball	an array of all labelled balls in the system,
BallTmp	a maximum number of ball in each case concerning probability value,
BallNo	running number of ball number,
BallNew	a random new ball index,
BallNewValue	the index of BallNew,
BallOld	a random old ball index,
BallOldValue	the index of BallOld,

We implemented the natural selection algorithm for stochastic production rules selection in L-systems. The following source code of natural selection algorithm is given below.

Natural Selection Algorithm

```
const MaxBall = 100;

function NaturalSelection(StochRule : Production) : Integer;
var
    Ball : array[1..MaxBall] of Integer; // To store the kind of ball
    i,j,k : Integer;
    BallTmp, BallNo : Integer;
    BallNew,BallNewValue,BallOld,BallOldValue : Integer;

begin
    BallNo := 0;
    for i:= 1 to StochRule.ruleno do
    begin
      BallTmp := Round(StochRule.prob[i]*MaxBall);

      for j:= 1 to BallTmp do
      begin
        inc(BallNo);
        Ball[BallNo] := i;
      end;
    end;
    // Shaking the Ball Process.
    for i:=1 to 5000 do
    begin
      BallOld := Random(MaxBall)+1; // Random old Address
      BallOldValue := Ball[BallOld]; // store the old value

      BallNew := Random(MaxBall)+1; // Random New address
      BallNewValue := Ball[BallNew]; // Store New vlaue

      Ball[BallOld] := BallNewValue; // move New value to Old address
      Ball[BallNew] := BallOldValue; // move Old value to New address

    end;

    // Return the selected ball to the caller..
    Result := Ball[Random(MaxBall)+1];
end;
```

Appendix B

Parameter Estimation

This appendix describes the Hessian matrix of multi-logistic parameter estimation shown in Section 2.5.

Hessian Matrix

From equation (2.35), let N be a number of N-pulses. In order to make the example easy, we show for 3-pulses ($N = 3$).

$$\begin{aligned} G(t) &= \frac{U_1}{1+e^{m_1(T_1-t)}} + \frac{U_2}{1+e^{m_2(T_2-t)}} + \frac{U_3}{1+e^{m_N(T_N-t)}} \\ &= \sum_{k=1}^{3}(\frac{U_k}{1+e^{m_k(T_k-t)}}) \end{aligned} \tag{B.1}$$

We now have 3-pulses with 9 parameters: $U_1, m_1, T_1, U_2, m_2, T_2, U_3, m_3, T_3$. Let a_i be all parameters where $i, j = 1, 2, 3, ..., 9$. The equation (2.40) can be rewritten as follows:

$$\begin{aligned} \nabla^2 E(a_t) &= \begin{bmatrix} \frac{\partial^2 E(a_t)}{\partial a_{1t}\partial a_{1t}} & \frac{\partial^2 E(a_t)}{\partial a_{1t}\partial a_{2t}} & \frac{\partial^2 E(a_t)}{\partial a_{1t}\partial a_{3t}} & \cdots & \frac{\partial^2 E(a_t)}{\partial a_{1t}\partial a_{9t}} \\ \frac{\partial^2 E(a_t)}{\partial a_{2t}\partial a_{1t}} & \frac{\partial^2 E(a_t)}{\partial a_{2t}\partial a_{2t}} & \frac{\partial^2 E(a_t)}{\partial a_{2t}\partial a_{3t}} & \cdots & \frac{\partial^2 E(a_t)}{\partial a_{2t}\partial a_{9t}} \\ \frac{\partial^2 E(a_t)}{\partial a_{3t}\partial a_{1t}} & \frac{\partial^2 E(a_t)}{\partial a_{3t}\partial a_{2t}} & \frac{\partial^2 E(a_t)}{\partial a_{3t}\partial a_{3t}} & \cdots & \frac{\partial^2 E(a_t)}{\partial a_{3t}\partial a_{9t}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E(a_t)}{\partial a_{9t}\partial a_{1t}} & \frac{\partial E(a_t)}{\partial a_{9t}\partial a_{2t}} & \frac{\partial E(a_t)}{\partial a_{9t}\partial a_{3t}} & \cdots & \frac{\partial^2 E(a_t)}{\partial a_{9t}\partial a_{9t}} \end{bmatrix} \\ &= [h_{i,j}] \\ &= \left[\frac{\partial^2 E(a_t)}{\partial a_{it}\partial a_{jt}}\right] \quad \forall i, j; \qquad \textit{where } i, j = 1, 2, 3, ..., 9\,(m = 3(3) = 9) \end{aligned} \tag{B.2}$$

The Hessian matrix is symmetry. The $h_{i,j}$ elements are listed in the following equations.

$$h_{1,1} = h_{1,4} = h_{1,5} = h_{1,6} = h_{1,7} = h_{1,8} = h_{1,9} = 0 \tag{B.3}$$

$$h_{1,2} = \frac{\partial^2 E(a_t)}{\partial a_{1t} \partial a_{2t}} = h_{2,1} = -\frac{(dd)(x-a_3)e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}\right)^2 (a_2)^2} \tag{B.4}$$

$$h_{1,3} = \frac{\partial^2 E(a_t)}{\partial a_{1t} \partial a_{3t}} = h_{3,1} = -\frac{(dd)e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}\right)^2 (a_2)} \tag{B.5}$$

$$h_{2,2} = \frac{\partial^2 E(a_t)}{\partial a_{2t} \partial a_{2t}} = -\frac{2a_1(dd)^2(x-a_3)^2 \left(e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}\right)^2}{\left(1+e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}\right)^3 (a_2)^4}$$

$$+\frac{2a_1(dd)(x-a_3)e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}\right)^2 (a_2)^3}$$

$$-\frac{a_1(dd)^2(x-a_3)^2e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}\right)^2 (a_2)^4} \tag{B.6}$$

$$h_{2,3} = \frac{\partial^2 E(a_t)}{\partial a_{2t} \partial a_{3t}} = h_{3,2} = -\frac{2a_1(dd)^2(x-a_3)\left(e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}\right)^2}{\left(1+e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}\right)^3 (a_2)^3}$$

$$+\frac{a_1(dd)e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}\right)^2 (a_2)^2}$$

$$-\frac{a_1(dd)^2(x-a_3)e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}\right)^2 (a_2)^3} \tag{B.7}$$

$$h_{2,4} = h_{2,5} = h_{2,6} = h_{2,7} = h_{2,8} = h_{2,9} = 0 \tag{B.8}$$

$$h_{3,3} = \frac{\partial^2 E(a_t)}{\partial a_{3t} \partial a_{3t}} = -\frac{2a_1(dd)^2\left(e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}\right)^2}{\left(1+e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}\right)^3 (a_2)^2}$$

$$-\frac{a_1(dd)^2e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_3)}{a_2}\right)}\right)^2 (a_2)^2} \tag{B.9}$$

$$h_{3,4} = h_{3,5} = h_{3,6} = h_{3,7} = h_{3,8} = h_{3,9} = 0 \tag{B.10}$$

$$h_{4,1} = h_{4,2} = h_{4,3} = h_{4,4} = h_{4,7} = h_{4,8} = h_{4,9} = 0 \tag{B.11}$$

$$h_{4,5} = \frac{\partial^2 E(a_t)}{\partial a_{4t} \partial a_{5t}} = h_{5,4} = -\frac{(dd)(x-a_6)e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}\right)^2 (a_5)^2} \tag{B.12}$$

$$h_{4,6} = \frac{\partial^2 E(a_t)}{\partial a_{4t} \partial a_{6t}} = h_{6,4} = -\frac{(dd)e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}\right)^2 (a_5)} \tag{B.13}$$

$$h_{5,5} = \frac{\partial^2 E(a_t)}{\partial a_{5t} \partial a_{5t}} = -\frac{2a_4(dd)^2(x-a_6)^2\left(e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}\right)^2}{\left(1+e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}\right)^3 (a_5)^4}$$

$$+\frac{2a_4(dd)(x-a_6)e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}\right)^2 (a_5)^3}$$

$$-\frac{a_4(dd)^2(x-a_6)^2e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}\right)^2 (a_5)^4} \tag{B.14}$$

$$h_{5,6} = \frac{\partial^2 E(a_t)}{\partial a_{5t} \partial a_{6t}} = h_{6,5} = -\frac{2a_4(dd)^2(x-a_6)\left(e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}\right)^2}{\left(1+e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}\right)^3 (a_5)^3}$$

$$+\frac{a_4(dd)e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}\right)^2 (a_5)^2}$$

$$-\frac{a_4(dd)^2(x-a_6)e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}\right)^2 (a_5)^3} \tag{B.15}$$

$$h_{5,7} = h_{5,8} = h_{5,9} = 0 \tag{B.16}$$

$$h_{6,6} = \frac{\partial^2 E(a_t)}{\partial a_{6t} \partial a_{6t}} = -\frac{2a_4(dd)^2\left(e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}\right)^2}{\left(1+e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}\right)^3 (a_5)^2}$$

$$-\frac{a_4(dd)^2 e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_6)}{a_5}\right)}\right)^2 (a_5)^2} \tag{B.17}$$

$$h_{6,7} = h_{6,8} = h_{6,9} = 0 \tag{B.18}$$

$$h_{7,1} = h_{7,2} = h_{7,3} = h_{7,4} = h_{7,5} = h_{7,6} = h_{7,7} = 0 \tag{B.19}$$

$$h_{7,8} = \frac{\partial^2 E(a_t)}{\partial a_{7t} \partial a_{8t}} = -\frac{(dd)(x-a_9)e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}\right)^2 (a_8)^2} \tag{B.20}$$

$$h_{7,9} = \frac{\partial^2 E(a_t)}{\partial a_{7t} \partial a_{9t}} = -\frac{(dd)e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}\right)^2 (a_8)} \tag{B.21}$$

$$h_{8,1} = h_{8,2} = h_{8,3} = h_{8,4} = h_{8,5} = h_{8,6} = 0 \tag{B.22}$$

$$h_{8,7} = \frac{\partial^2 E(a_t)}{\partial a_{8t} \partial a_{7t}} = h_{7,8} = -\frac{(dd)(x-a_9)e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}\right)^2 (a_8)^2} \tag{B.23}$$

$$h_{8,8} = \frac{\partial^2 E(a_t)}{\partial a_{8t} \partial a_{8t}} = -\frac{2a_7(dd)^2(x-a_9)^2\left(e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}\right)^2}{\left(1+e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}\right)^3 (a_8)^4}$$

$$+\frac{2a_7(dd)(x-a_9)e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}\right)^2 (a_8)^3}$$

$$-\frac{a_7(dd)^2(x-a_9)^2 e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}\right)^2 (a_8)^4} \tag{B.24}$$

$$h_{8,9} = \frac{\partial^2 E(a_t)}{\partial a_{8t}\partial a_{9t}} = h_{9,8} = -\frac{2a_7(dd)^2(x-a_9)\left(e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}\right)^2}{\left(1+e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}\right)^3 (a_8)^3}$$

$$+\frac{a_7(dd)e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}\right)^2 (a_8)^2}$$

$$-\frac{a_7(dd)^2(x-a_9)e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}}{\left(1+e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}\right)^2 (a_8)^3} \tag{B.25}$$

$$h_{9,1} = h_{9,2} = h_{9,3} = h_{9,4} = h_{9,5} = h_{9,6} = 0 \tag{B.26}$$

$$h_{9,7} = h_{7,9} \tag{B.27}$$

$$h_{9,8} = h_{8,9} \tag{B.28}$$

$$h_{9,9} = \frac{\partial^2 E(a_t)}{\partial a_{9t} \partial a_{9t}} = -\frac{2a_7(dd)^2 \left(e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)} \right)^2}{\left(1 + e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)} \right)^3 (a_8)^2}$$

$$-\frac{a_7(dd)^2 e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)}}{\left(1 + e^{\left(-\frac{(dd)(x-a_9)}{a_8}\right)} \right)^2 (a_8)^2} \qquad \text{(B.29)}$$

Appendix C

Gallery Results

This appendix shows some simulated results in the dissertation obtained from *PlantVR* software. The description of each simulated result is given at its caption.

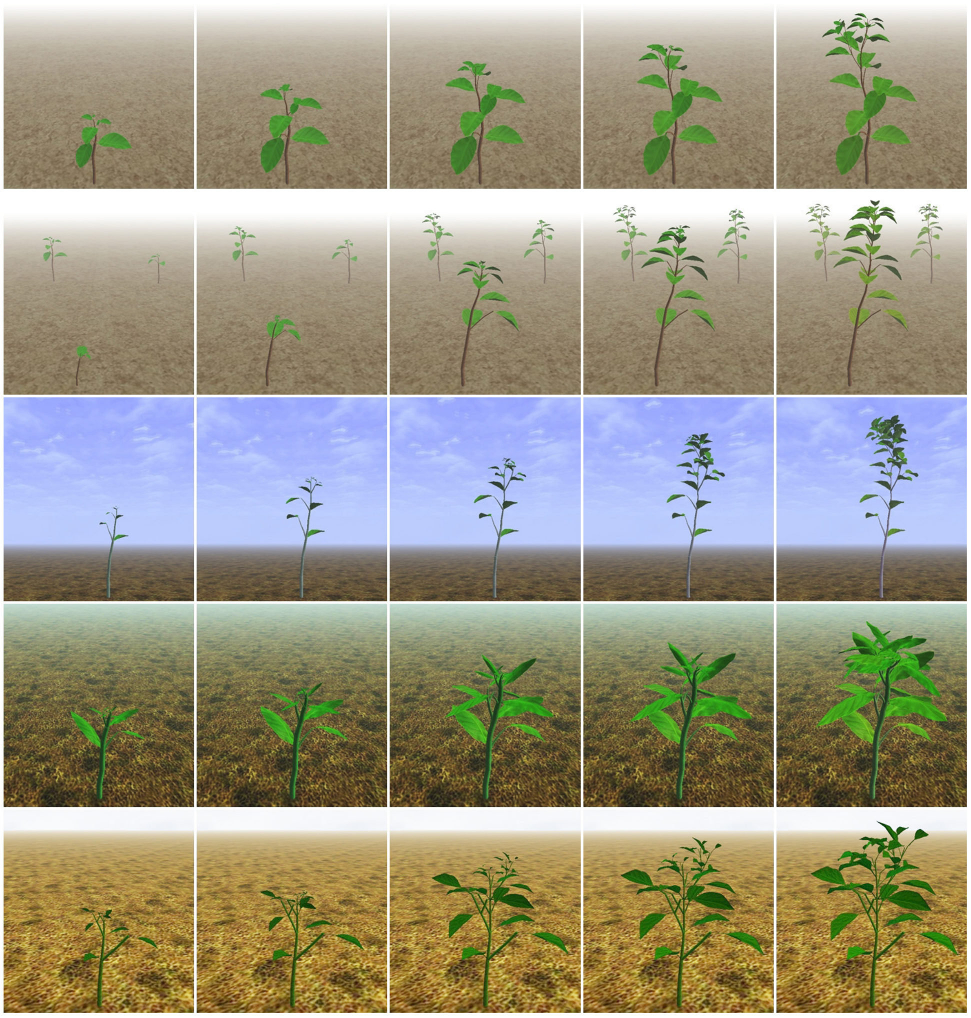


Figure C.1: Some examples of plant growth using *PlantVR*.

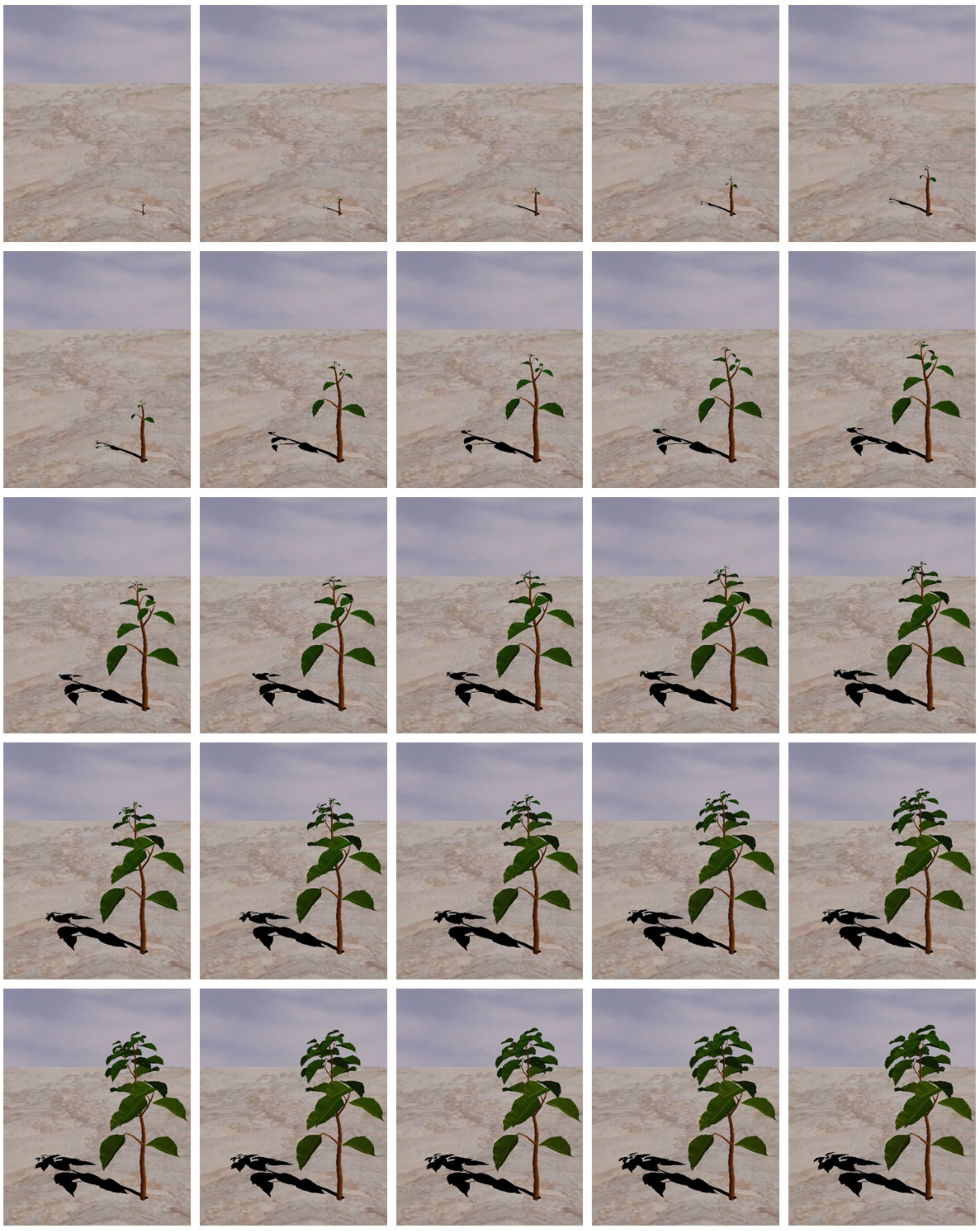


Figure C.2: Plant development with its shadow starting at $t = 20$ days for every two time steps $dt = 2$ days.

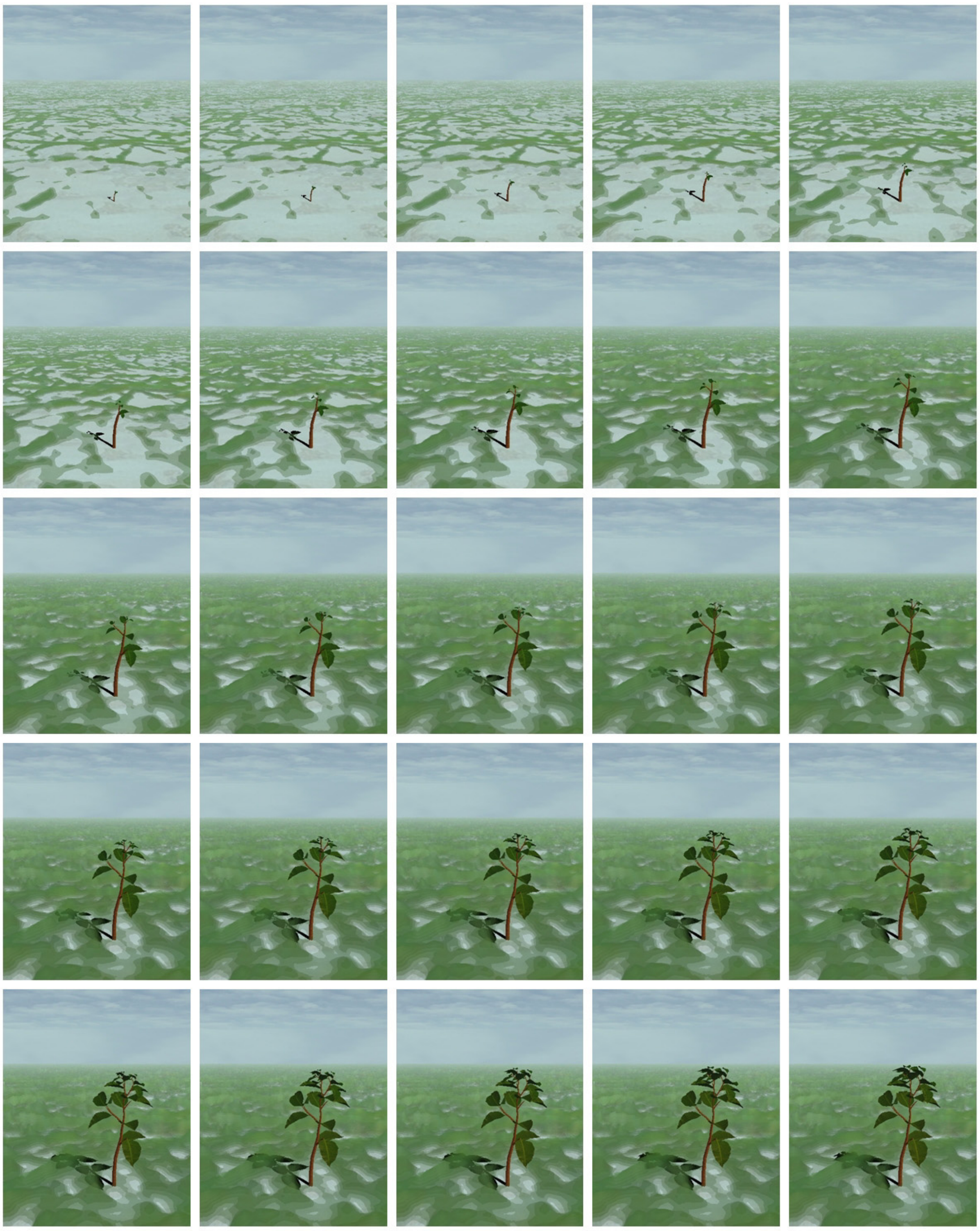


Figure C.3: Plant development and its shadow with growing grass starting at $t = 20$ days for every two time steps $dt = 2$ days.

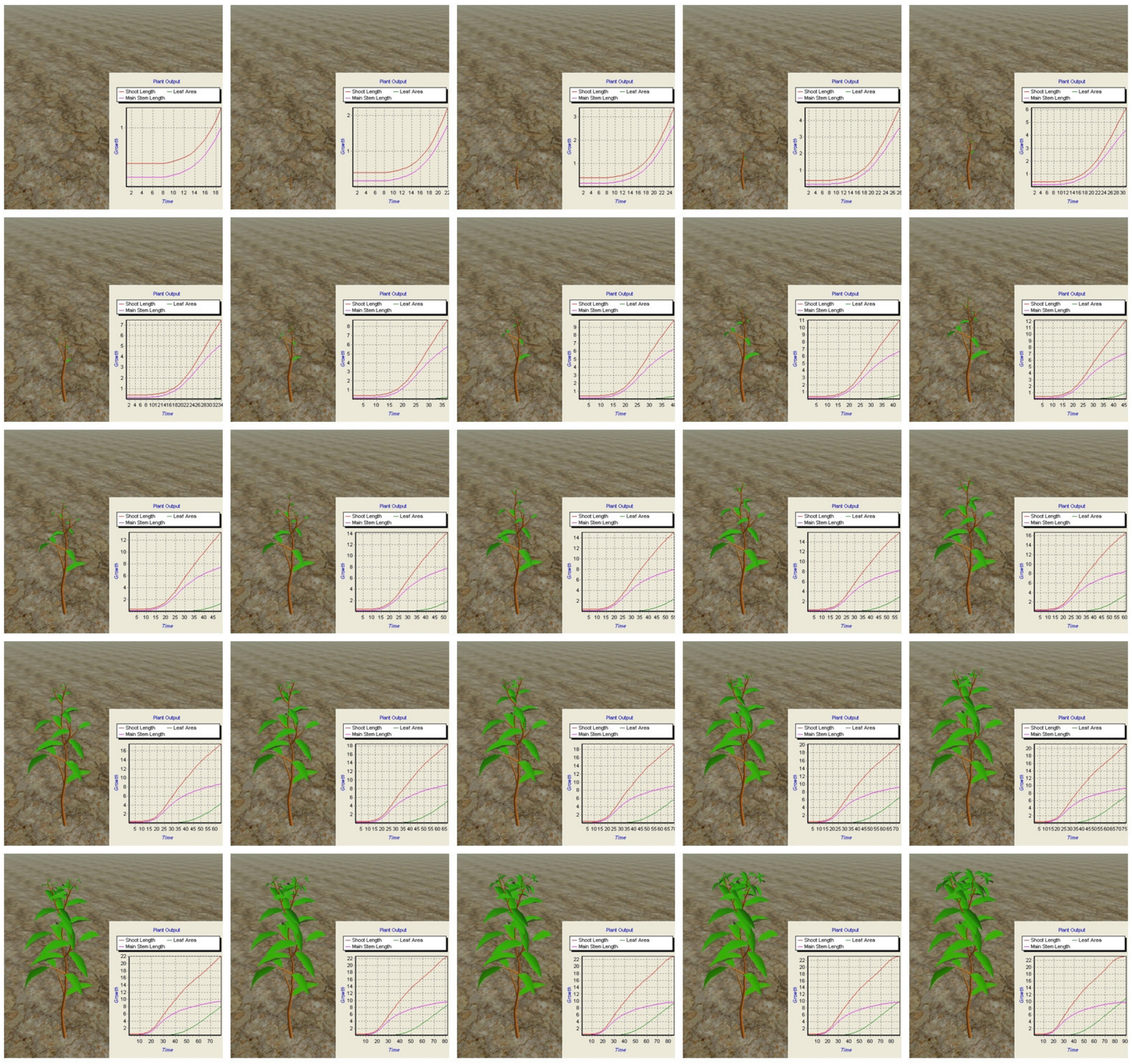


Figure C.4: Plant development with their growth curves starting at $t = 20$ days for every three time steps $dt = 3$ days.

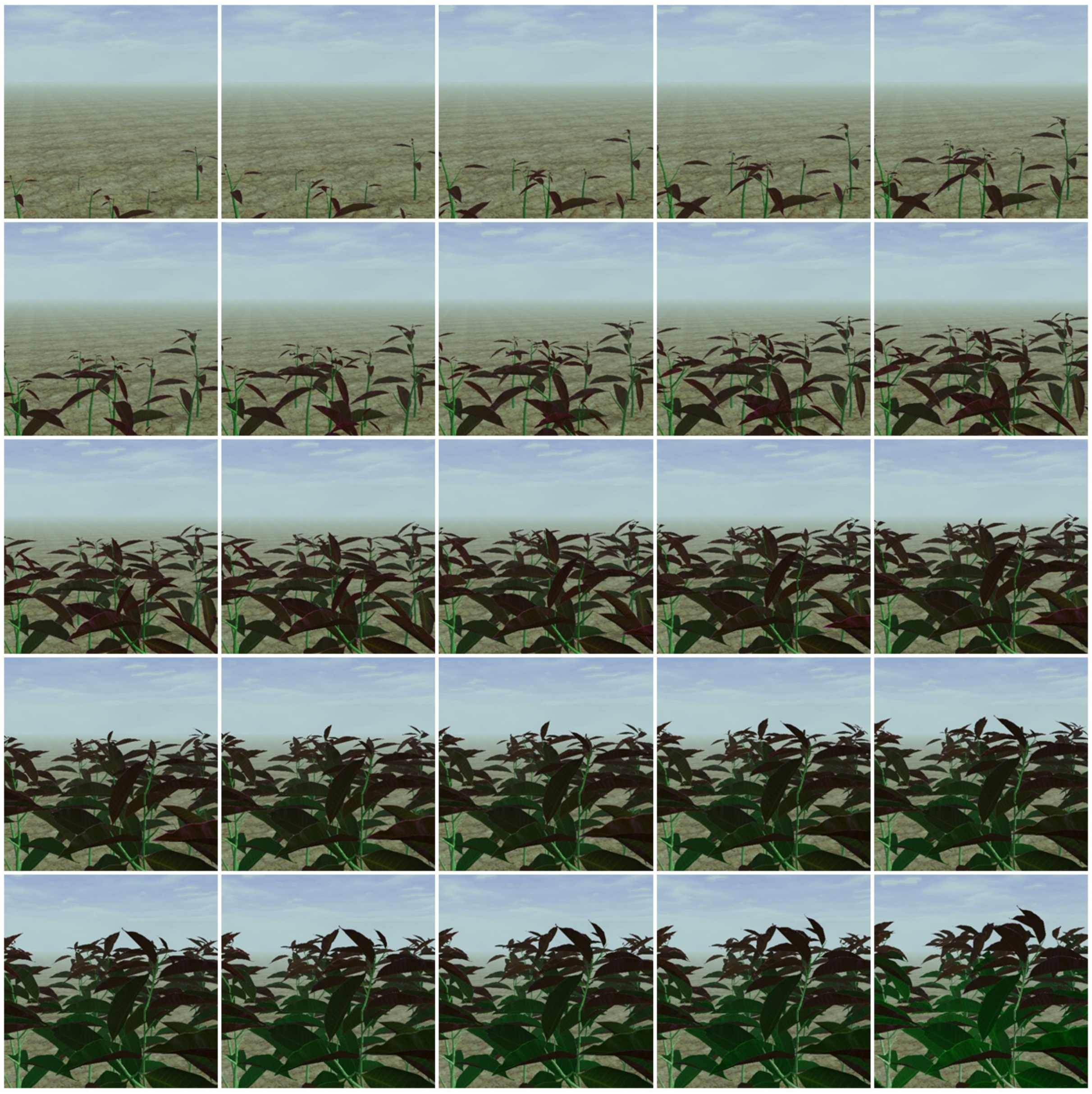


Figure C.5: Simulated mango tree with changing leaves color.

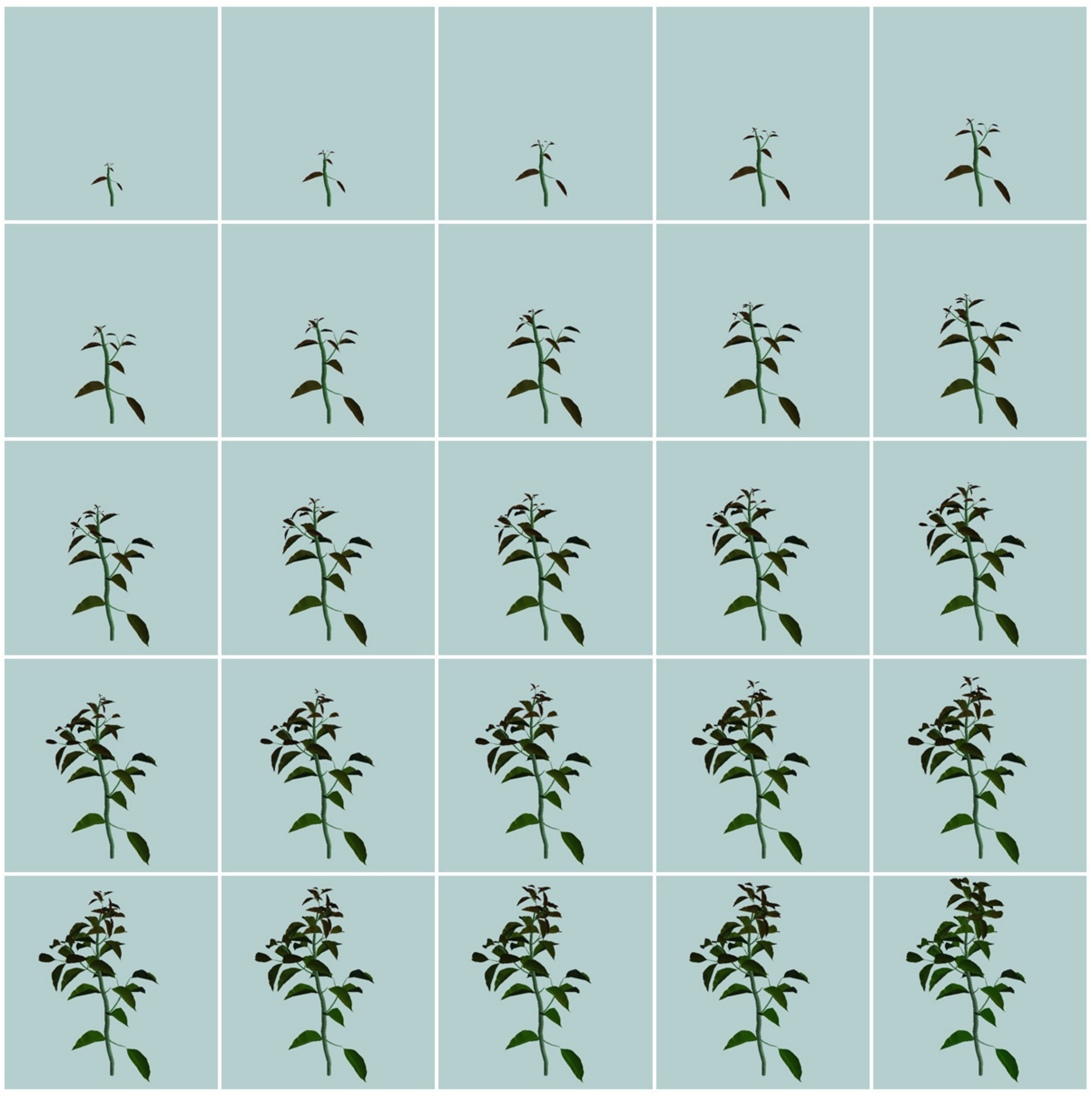


Figure C.6: Simulated growing mango tree starting at $t = 28.25$ for every 10 frames.

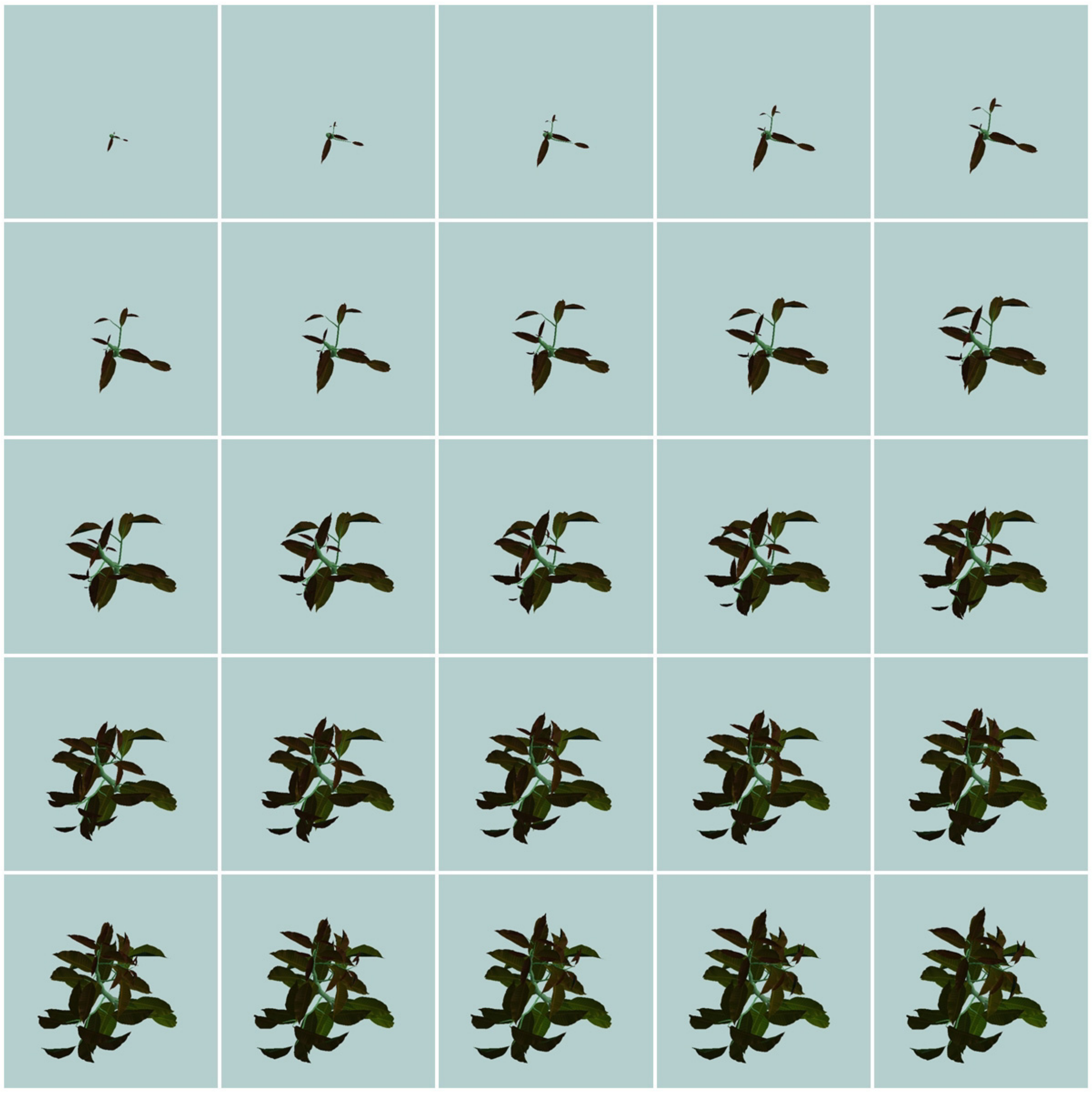


Figure C.7: Simulated growing mango tree starting at $t = 28.25$ for every 10 frames (from top view). It is similar to Figure C.6 mapped frame by frame.

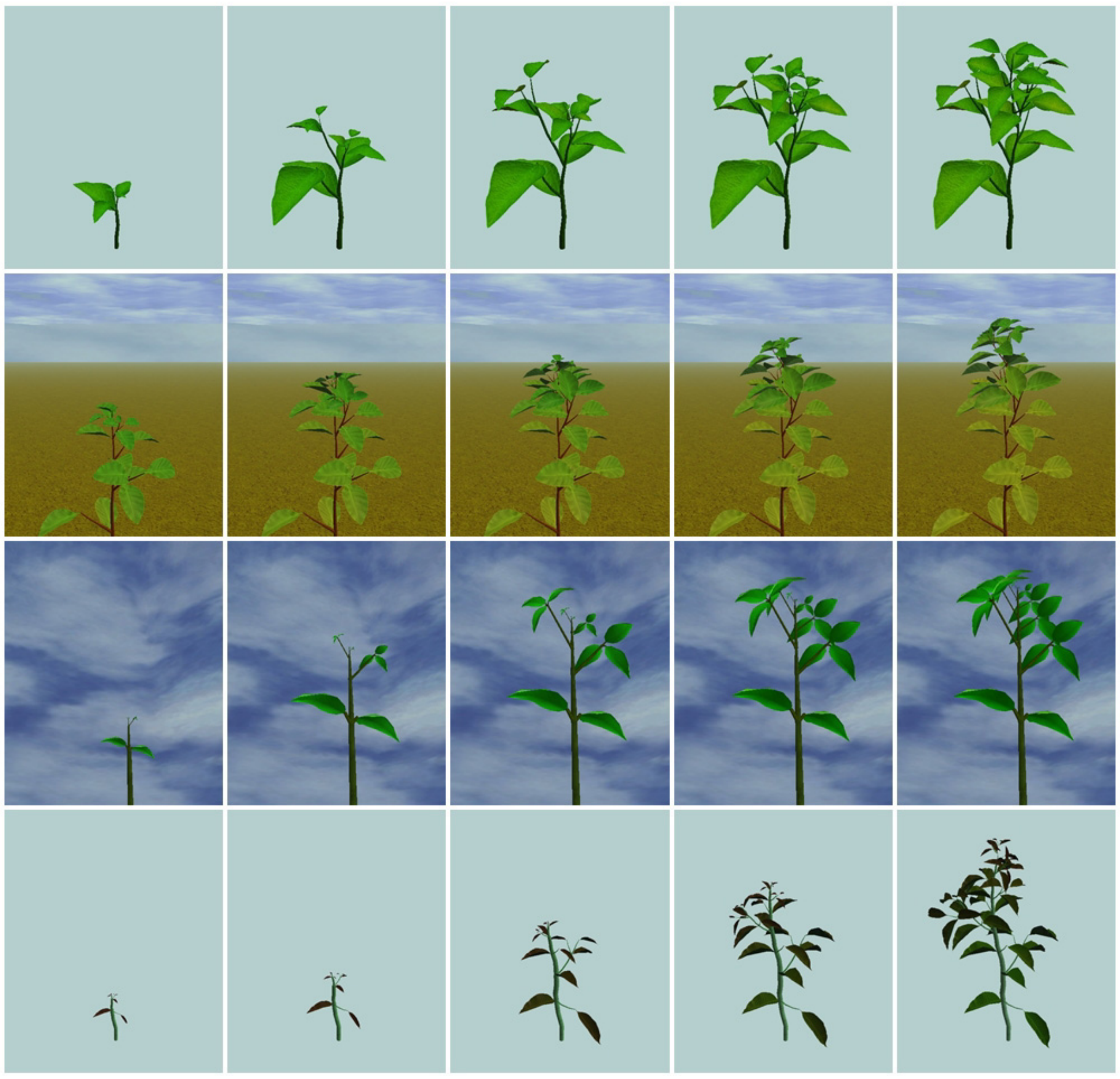


Figure C.8: Four different simulated growing plants based on sigmoidal growth functions.

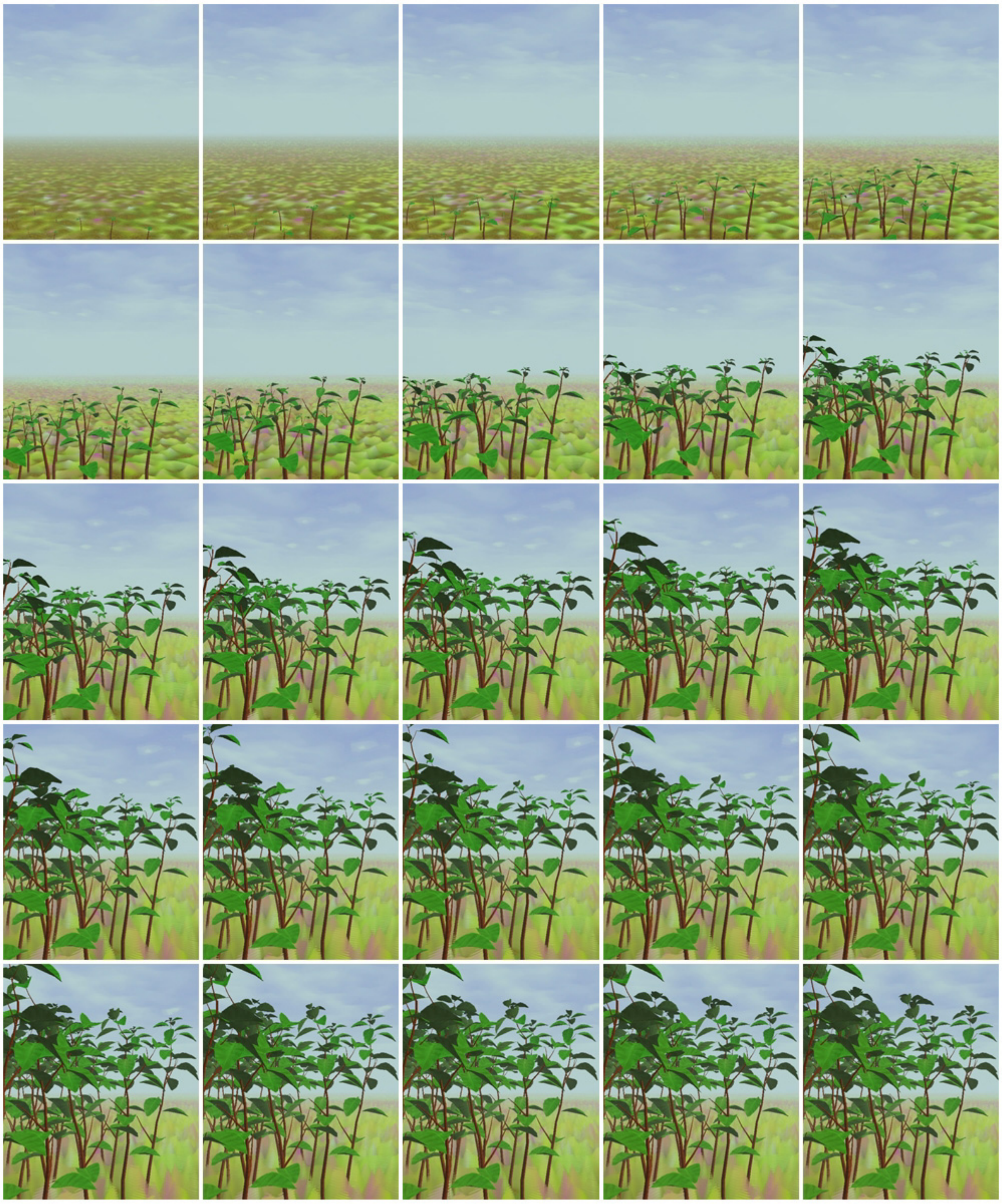


Figure C.9: Simulated growing entire crop in the field based on sigmoidal growth functions.

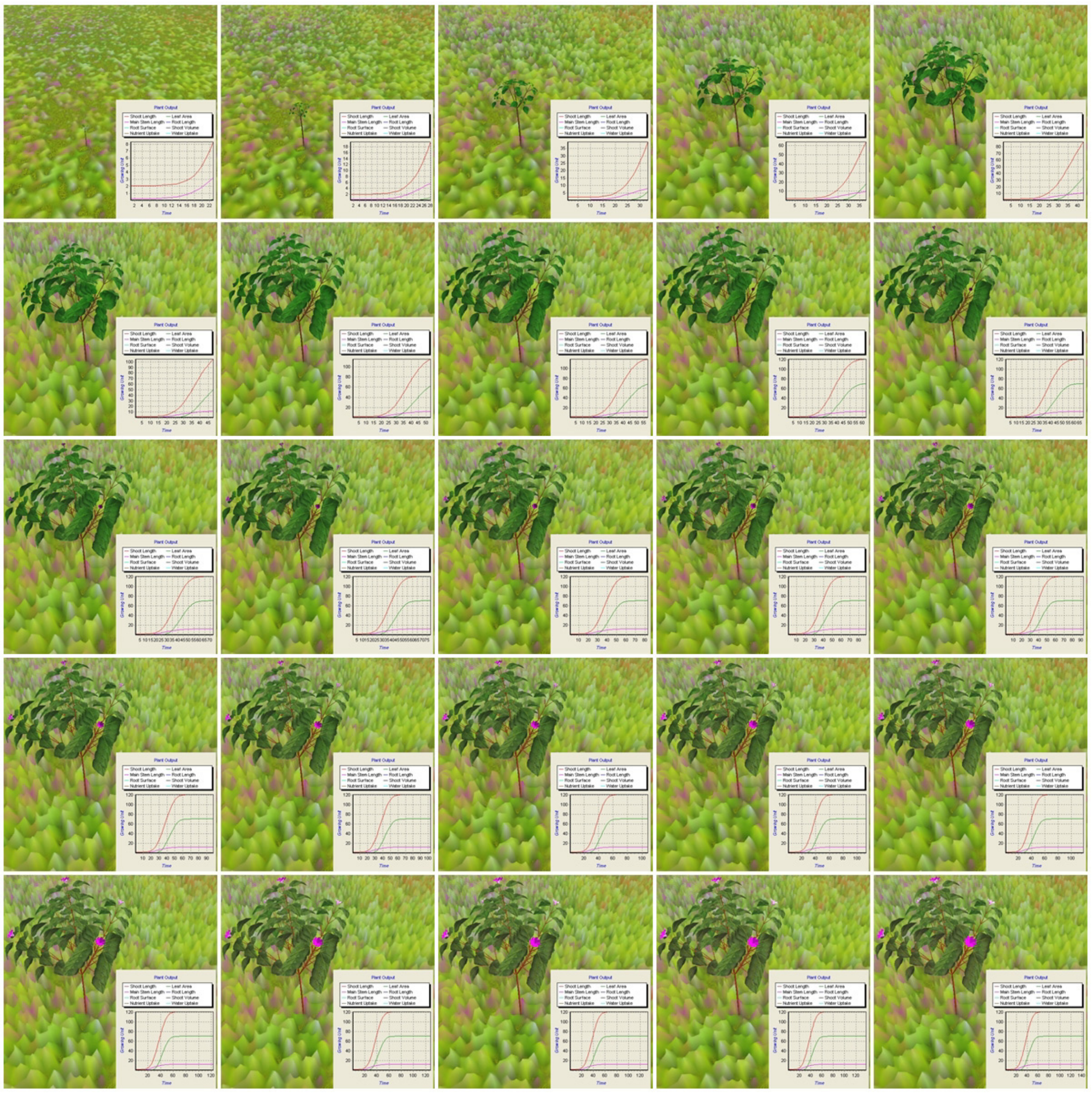


Figure C.10: Simulated growing plant with growth curve of plant components, shoot length, main step length, and leaf area. Users can observe plant growth behavior.

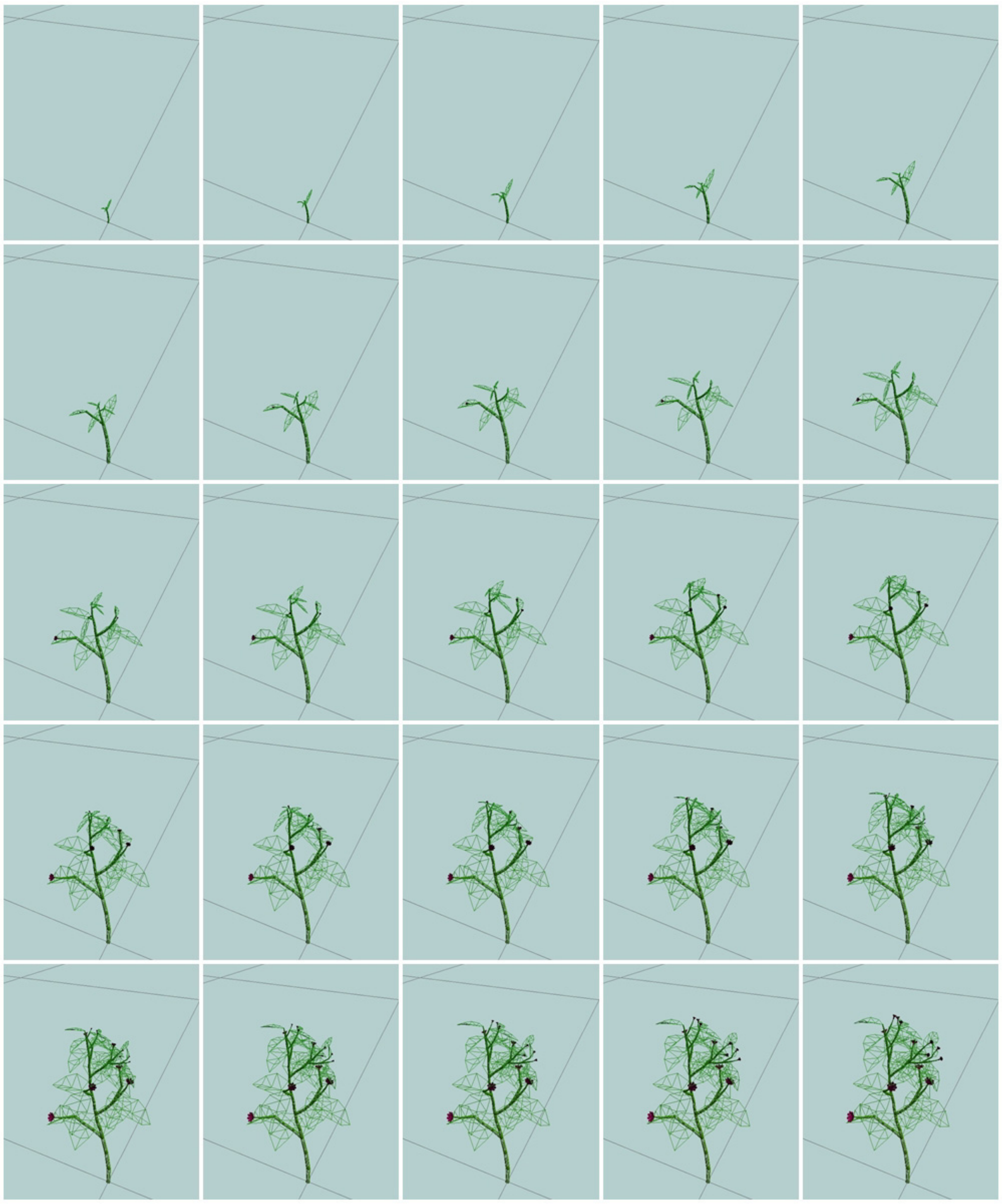


Figure C.11: Simulated growing plant in wire frame.

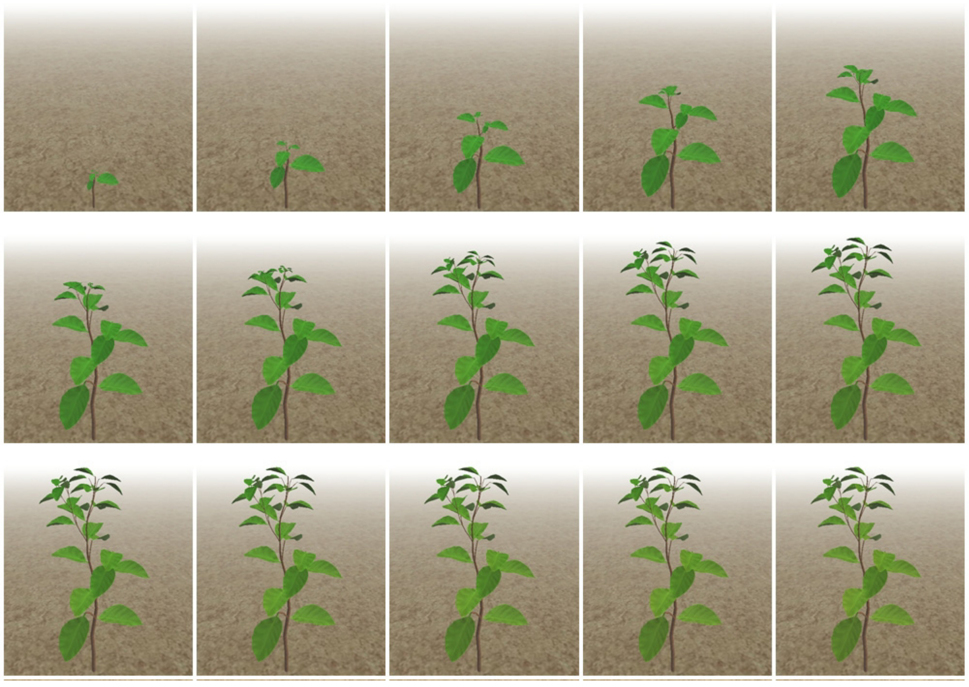


Figure C.12: Simulated growing plant with changing leaf color depending on leaf age.

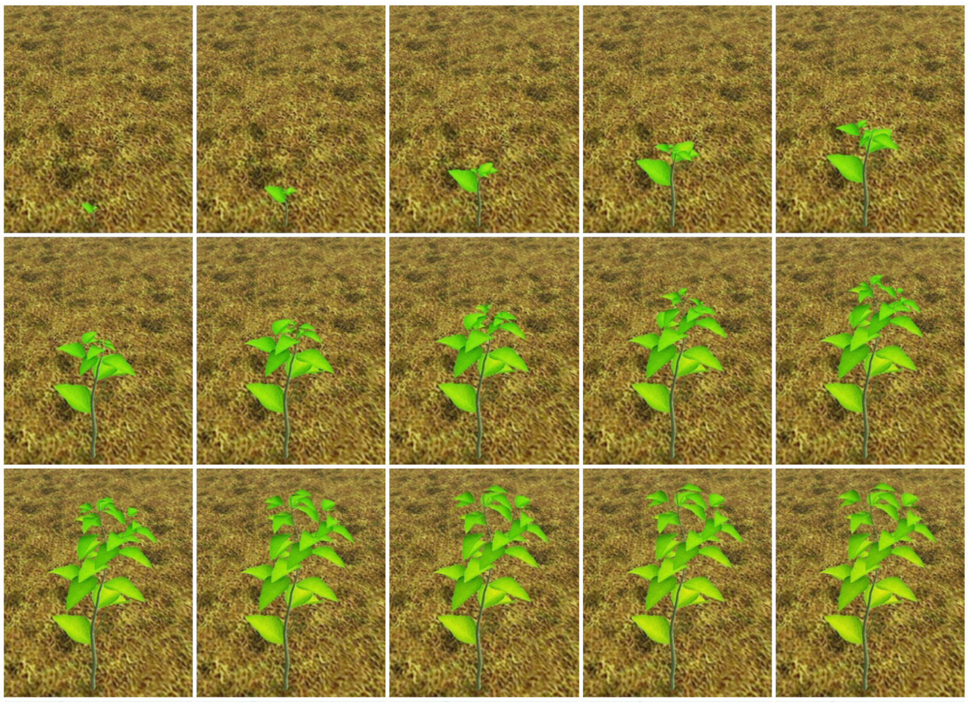


Figure C.13: Simulated growing soybean with changing leaf material corresponding leaf age.

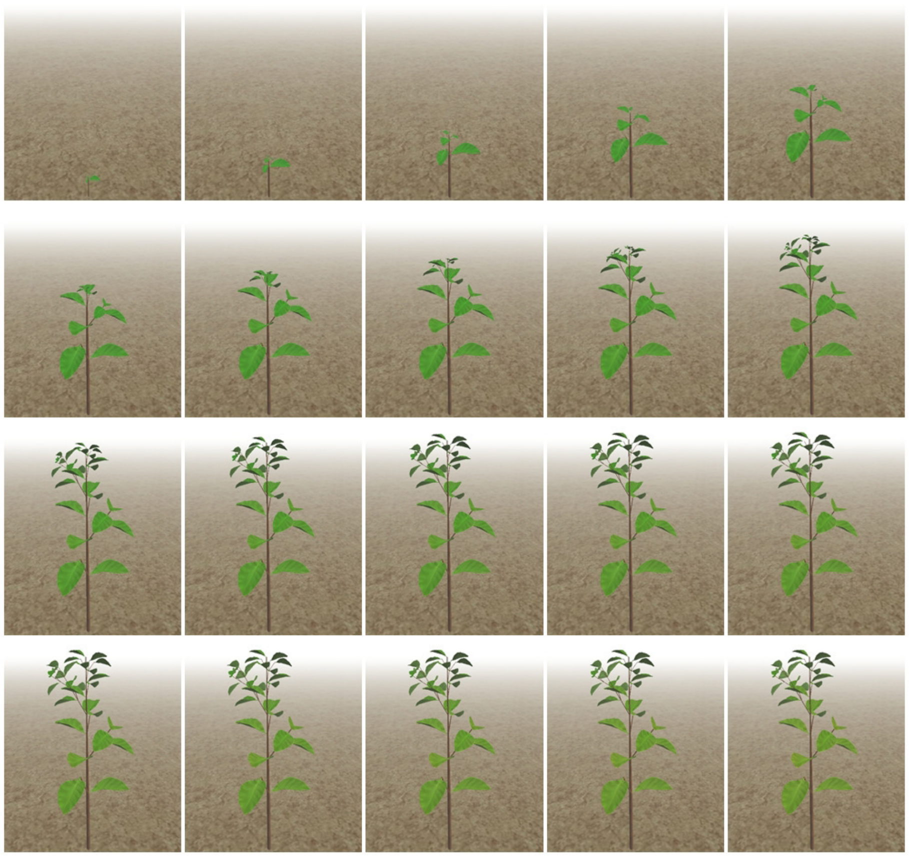


Figure C.14: Simulated growing plant with changing leaf material corresponding leaf age, straight main stem, *PlantVR* provides an option to adjust the main stem look more realistic.

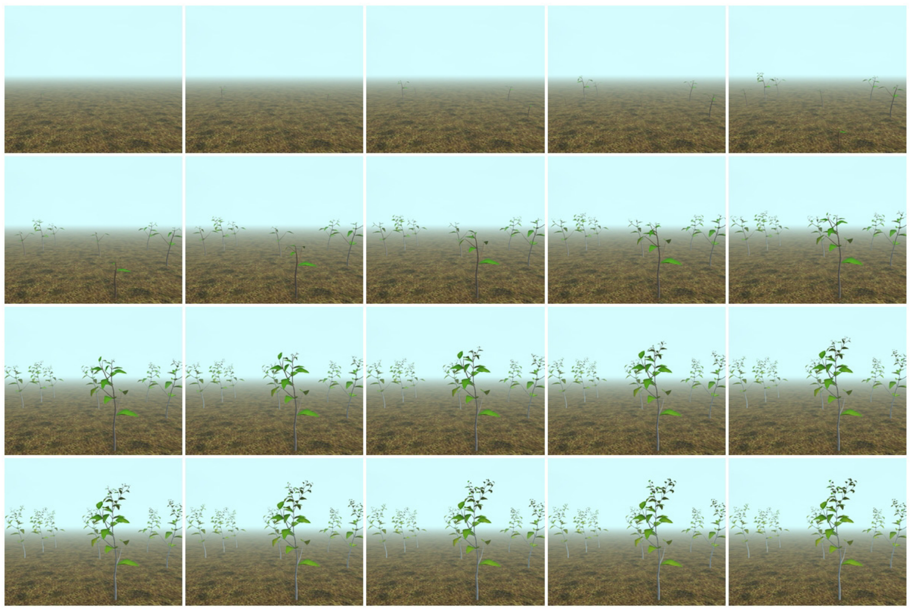


Figure C.15: Simulated stochastic L-system applied to soybean prototype and growing as entire crop.

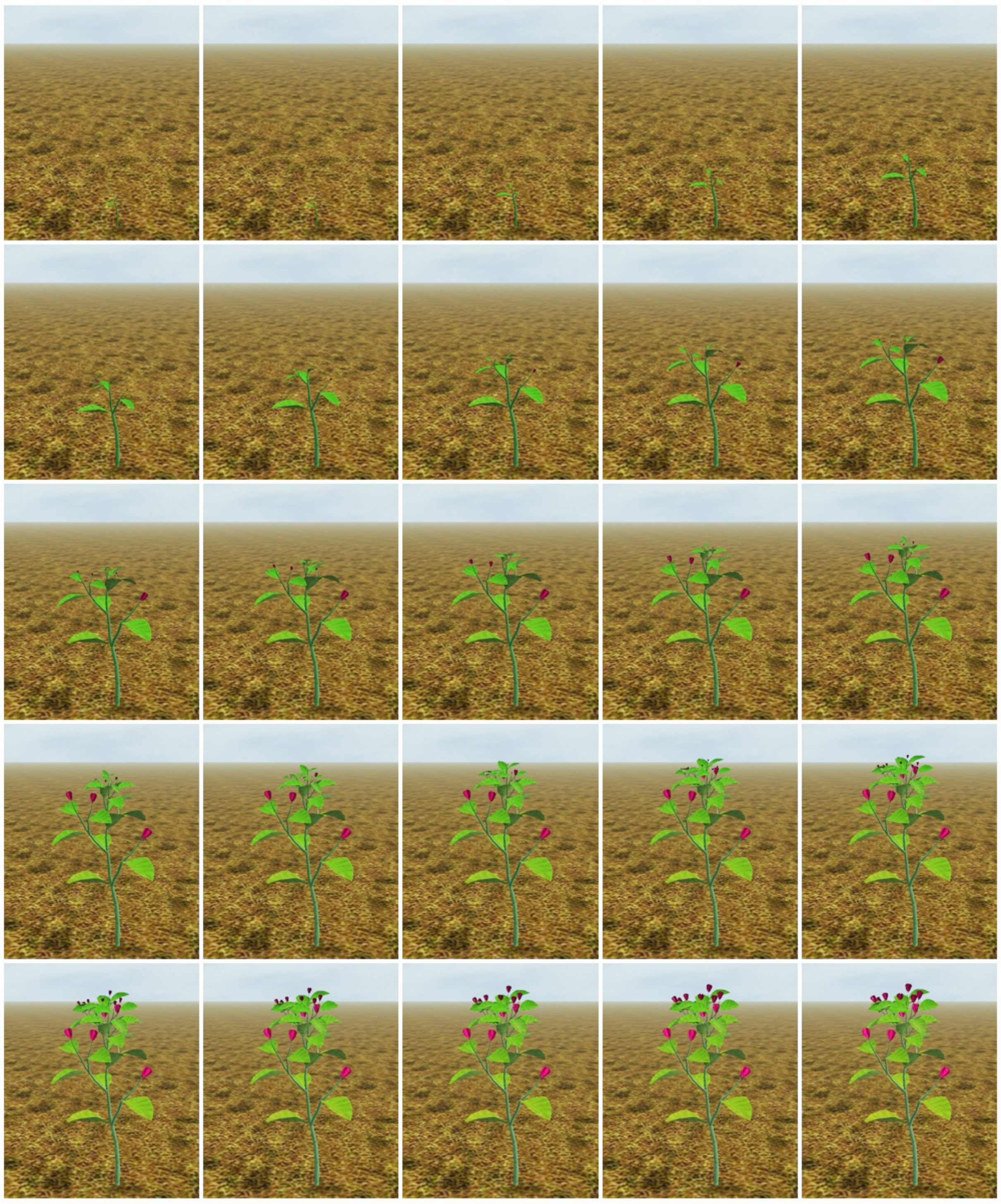


Figure C.16: Simulated growing plant with red flowers and changing leaf color in time, the main stem with realistic mode.

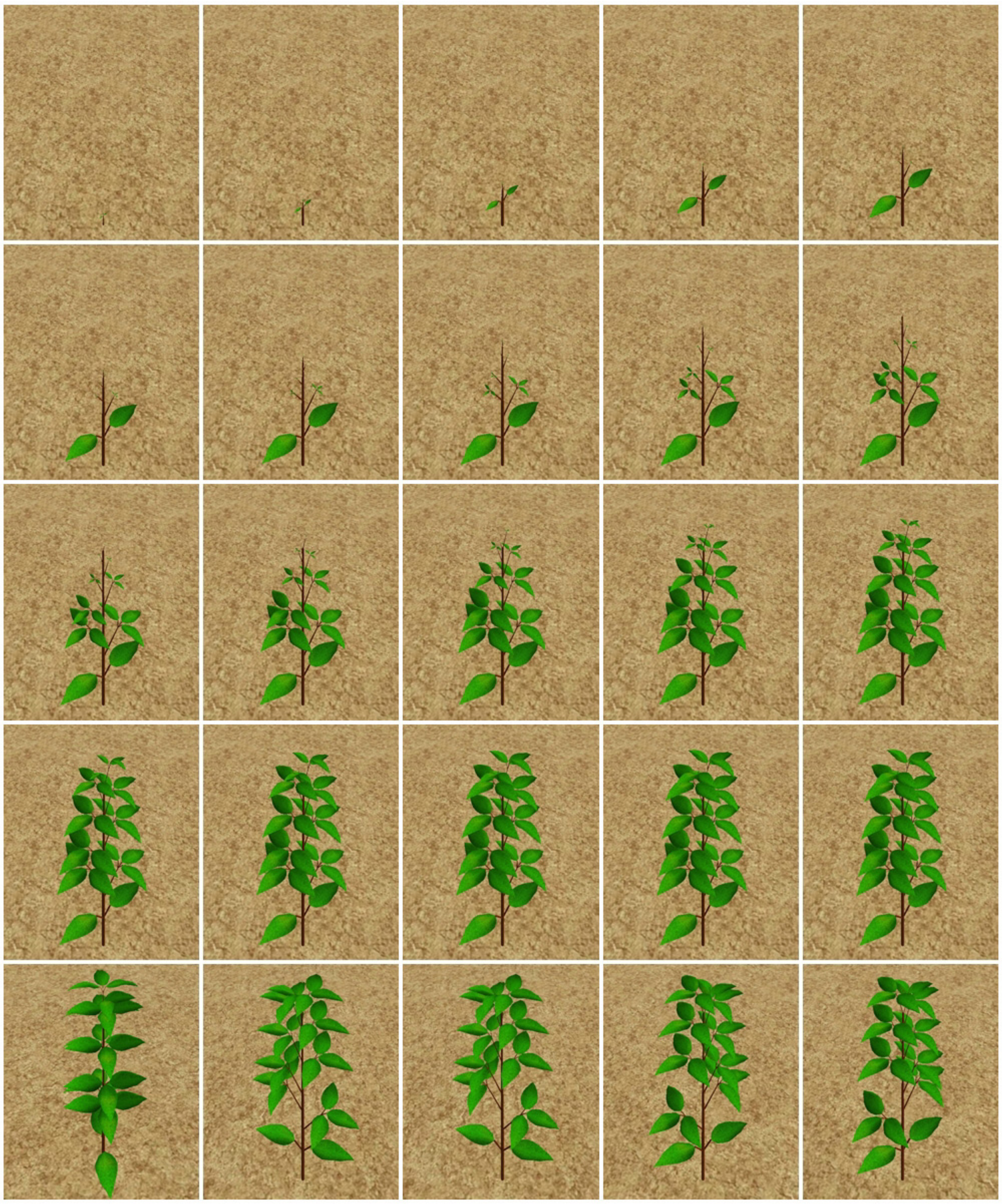


Figure C.17: Simulated growing soybean using stochastic L-system.

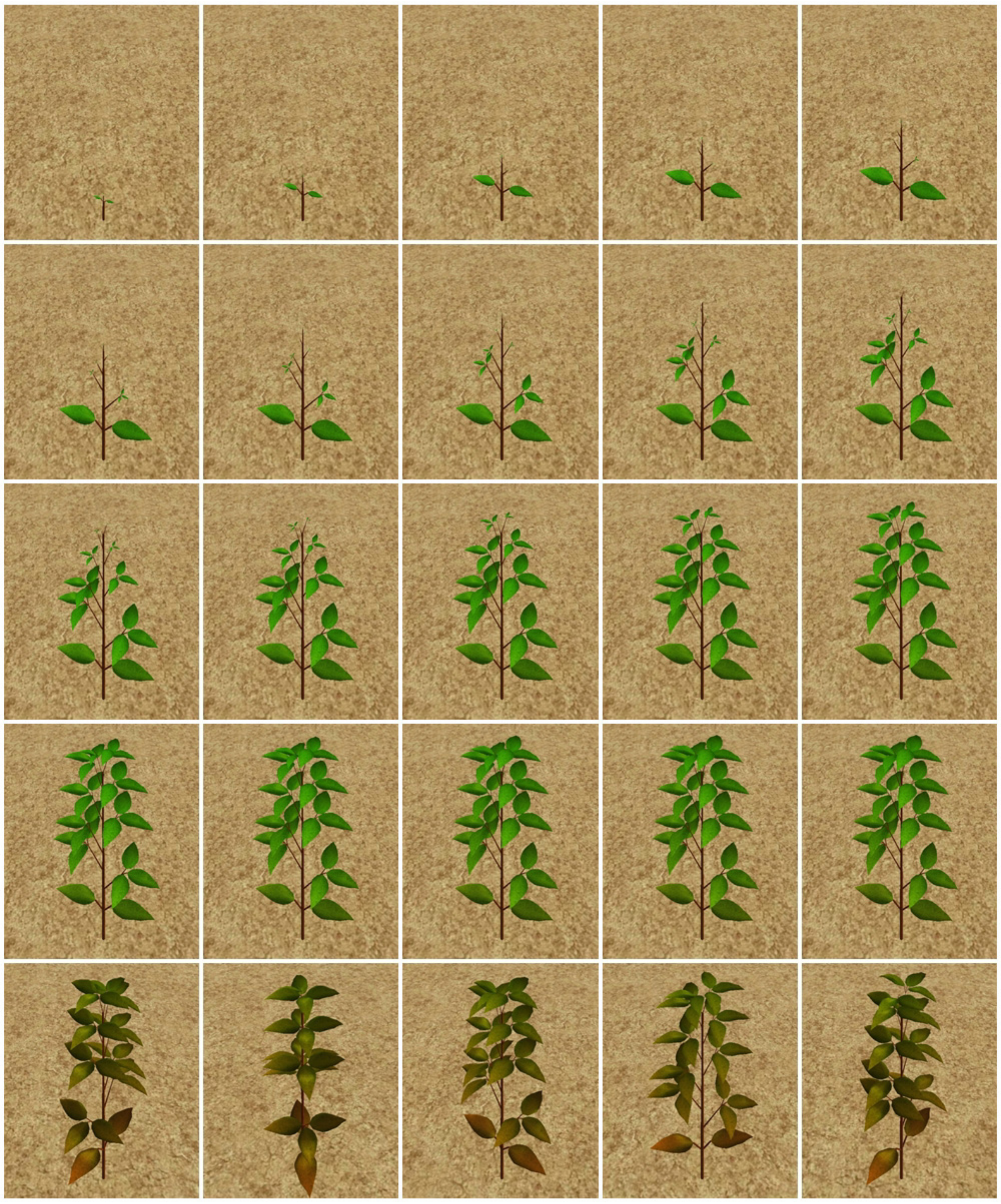


Figure C.18: Simulated growing soybean using stochastic L-system with changing leaf color depending on leaf age.

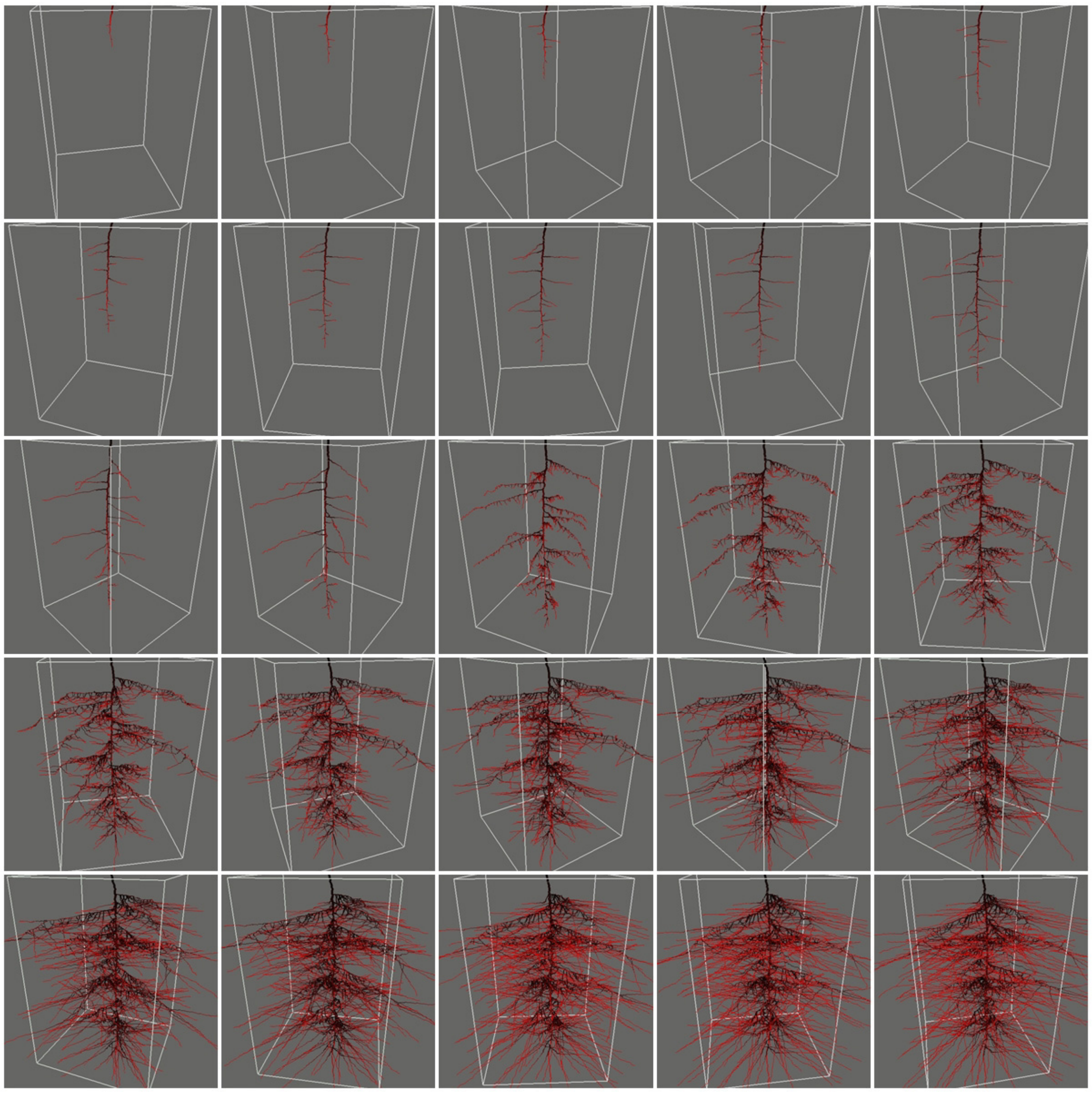


Figure C.19: Simulated growing root system using root-growth-process algorithm, root color presents the root age. Red root is younger than black root.

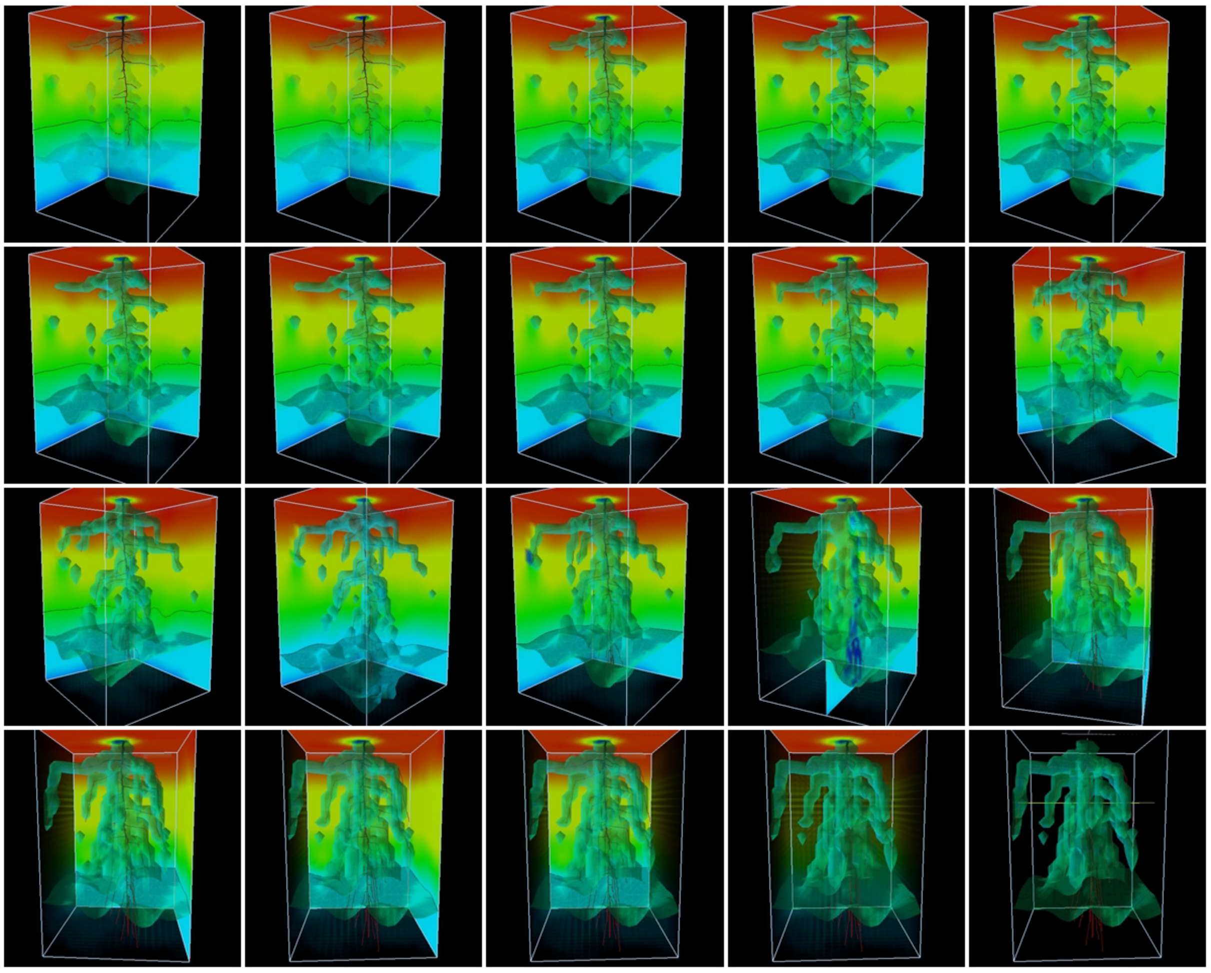


Figure C.20: Simulated growing root and nutrient uptake by plant root, the color planes present the nutrient concentration at each position in soil volume. Red is high concentration and blue is low concentration. The color was coded from red to blue for high to low concentration, respectively.

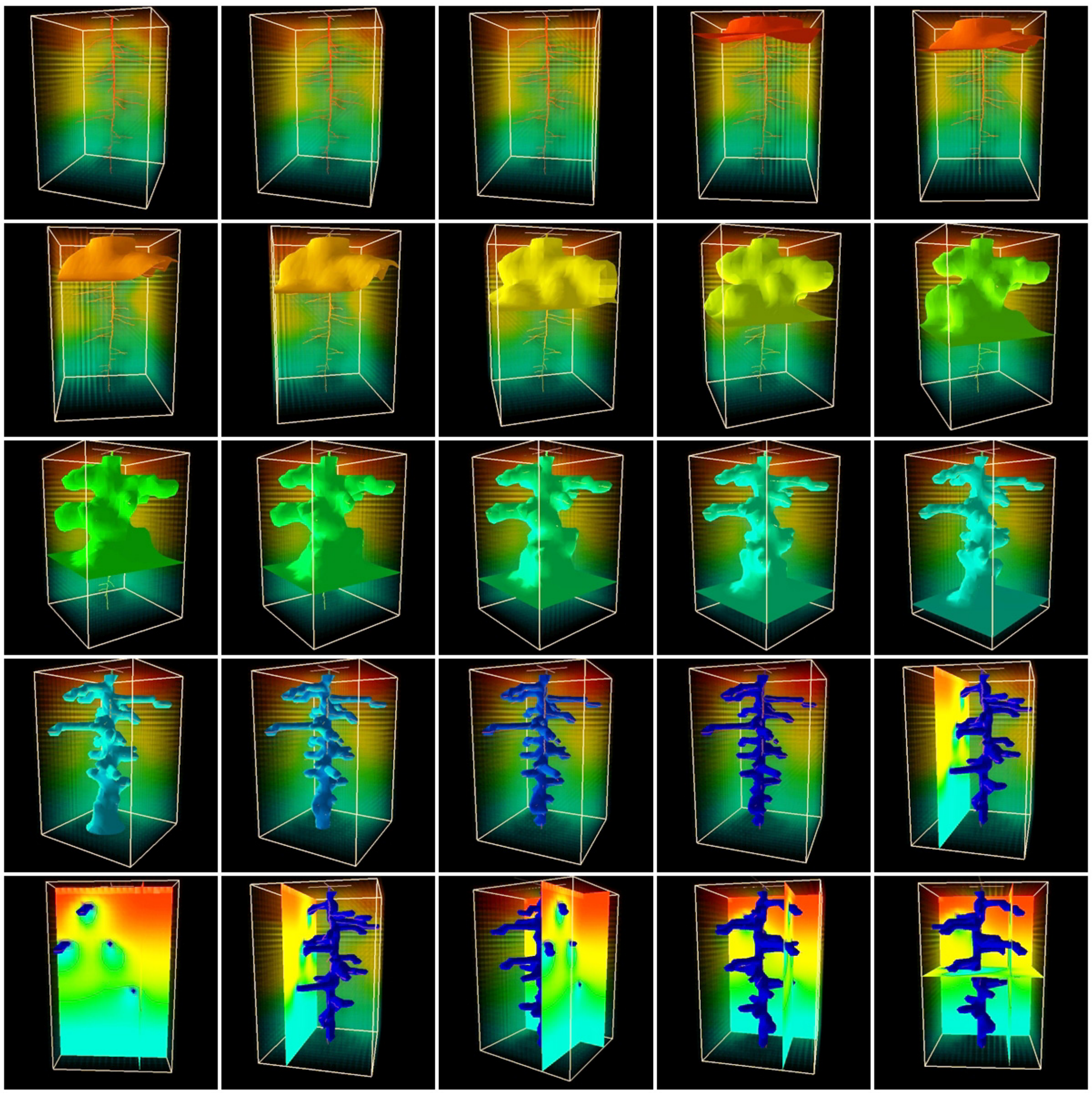


Figure C.21: Visualization of root growth and nutrient diffusion, the iso-surface presents the nutrient concentration around root zone. The color was coded from red to blue for high to low concentration, respectively.

List of Figures, Tables, Abbreviations and Notations

List of Figures

List of Tables

Abbreviations and Notations

Abbreviations and Notations

This appendix describes all abbreviations and notational conventions used in the text. It is divided into two sections: Abbreviations and Notations.

List of Abbreviations

Abbreviations	Meaning
2D	2-dimensional space
3D	3-dimensional space
BrachRecon	Branching Reconstruction software
CBN	Algorithm for Constructing the Branching Network
CT	Computed Tomography
Delphi	Delphi Borland Cooperation
DFV	Depth Field Value
DOL	Deterministic and Context-Free L-systems
FVM	Finite Volume Method
GUI	Graphical User Interface
HFM2D	Height Field Map Algorithm for 2D Images
HFV3D	Height Field Volume Algorithm for 3D Volume Data
HOC	Neuron Network File Format
IAA	Indole-3-Acetic Acid (Auxin Hormone)
LCon	L-string Construction Algorithm
L-systems	Lindenmayer Systems
LeafDS	Leaf Design Surface Software
LeafVein	Leaf Vein Generator Software
LED	Leaf Edge Detection algorithm
LevMar	Levenberg-Marquardt's Algorithm
LM	Levenberg-Marquardt Method
MRI	Magnetic Resonance Imaging
MuPhi	$\mu\phi$ Library for Calculating Water Flow
NLS	Network L-system File Format
OL	Context-Free L-systems
OpenGL	Open Graphics Library
PlantVR	Plant Virtual Reality software
PlantVR-MuPhi	PlantVR Software Linked to MuPhi(or $\mu\varphi$) Library

Abbreviations	Meaning
PM	The Perona-Malik model
PTM	Particle Transportation Method
PTM-LeafVein	Particle Transportation Method for Leaf Vein Generation
PTM-Tree	Particle Transportation Method for Tree/Plant Network Generation
ReconProc	Branching Structure Reconstruction Procedure
ResRed	Resolution Reduction Algorithm
RGM2D	Region Growing Method for 2D Images
RGP	Root Growth Process Algorithm
SKel2D	Skeletonizing Algorithm for 2D Images
SKel3D	Skeletonizing Algorithm for 3D Volume Data
SSE	Sum Square Error
VGM3D	Volume Growing Method for 3D Volume Data

List of Notations

Symbols	Meaning	Defined in
a	parameters of nonlinear regression equation	2.5.3
B_i	initial time of component i	2.5
$BB_{i,j}$	blue intensity of pixel i, j of image I	2.7.5
c	nutrient concentration (mol/cm^3)	4.3
C	differential water capacity ($d\theta/dh$) (cm^{-1})	4.2
$C(\Sigma)$	a *logical expression* with parameters from Σ	2.2.8
$\delta_l(i+1)$	accumulative angle from the 1^{st} leaf to $(i+1)^{th}$ leaf	3.4.1
δ_L	angle (degrees) between the i^{th} leaf and the $(i+1)^{th}$ leaf	3.4.1
δ_r	an initial angle between the first root internode with vertical axis	2.4
δ_s	an initial angle between the first shoot internode with vertical axis	2.4
d_l	growing step size of leaf vein segment	3.2.3
D	diffusion coefficient (cm^2/s)	4.3
$\vec{D}_i$	direction of particle p_i	3.1
D_r	an initial diameter of the first root internode in plant module design	2.4
D_s	an initial diameter of the first shoot internode in plant module design	2.4
$DFV_{i,j,k}$	depth field value at the voxel i, j, k	2.7.5
$\vec{E}$	unit gradient vector of water content	4.5
$E(a)$	sum square error of nonlinear regression equation	2.5.3
E_{i_new}	energy of new particle after combining the particle i and its nearest particle	3.1
$E(\Sigma)$	an *arithmetic expression* with parameters from Σ	2.2.8
g	vector of first order partial derivative of $E(a_t)$	2.5.3
G	growth function of nonlinear regression equation or dependent variable	2.5.3
$\vec{G}$	unit vector by gravitropism effect	4.5
G_i	growth value of measured data at i^{th} data point	2.5.3
$G_l(t)$	growth value of leaf at time t	3.2.3
$G(t)$	growth value of sigmoidal function	2.5
$GG_{i,j}$	green intensity of pixel i, j of image I	2.7.5
h	soil water pressure head (cm)	4.2
H	Hessian matrix of second order partial derivative of $E(a_t)$	2.5.3
$H(x,y,z)$	total head (cm)	4.2
J	Jacobian matrix derived from H, $H \cong 2J \cdot J^T$	2.5.3
χ	a successor of production rules for rewriting systems	2.2.1
$K_{i\_nearest}$	the nearest particle of the particle i	3.1

Symbols	Meaning	Defined in
K_{i_new}	the new particle after combining the particle i and its nearest particle	3.1
K_{xy}	hydraulic conductivity in xy-direction ($cm\ day^{-1}$)	4.2
λ	a modest value of step size in Levenberg-Marquardt's Algorithm	2.5.3
λ_u	conduction coefficient of gray intensity u	2.7.4
L	minimum value of sigmoidal growth function	2.5
m	slope of sigmoidal growth function	2.5
m_r	n^{th} endproduction rule of root module in plant module design	2.4
m_s	n^{th} endproduction rule of shoot module in plant module design	2.4
$\vec{M}$	unit vector by soil strength	4.5
M^{New}	2D or 3D array of gray intensity at the *new* iteration	2.7.5
$M^{New}_{i,j,k}$	gray intensity of the voxel i, j, k at the *new* iteration	2.7.5
M^{Old}	2D or 3D array of gray intensity at the *old* iteration	2.7.5
$M^{Old}_{i,j,k}$	gray intensity of the voxel i, j, k at the *old* iteration	2.7.5
n	number of iteration for rewriting systems	2.2.1
n_r	n^{th} production rule of root module in plant module design	2.4
n_s	n^{th} production rule of shoot module in plant module design	2.4
n_i	a network point of leaf vein	3.2.3
N_L	number of leaves in N_R rounds	3.4.1
N_R	number of turns from the first leaf to the n^{th} leaf in the vertical projection	3.4.1
$\vec{N}$	new direction of next root growth	4.5
$N_{[}$	number of bracket [	5.1.4
$N([)$	number of open bracket [	5.1.4
$N(])$	number of close bracket]	5.1.4
$NiLL$	left leaf length of i^{th} internode	5.1.3
$NiLW$	left leaf width of i^{th} internode	5.1.3
$NiML$	middle leaf length of i^{th} internode	5.1.3
$NiMW$	middle leaf width of i^{th} internode	5.1.3
$NiRL$	right leaf length of i^{th} internode	5.1.3
$NiRW$	right leaf width of i^{th} internode	5.1.3
N_r	iteration number of root module in plant module design	2.4
N_s	iteration number of shoot module in plant module design	2.4
ω	an axiom for rewriting systems	2.2.1
ω_r	a given axiom of root module in plant module design	2.4
ω_s	a given axiom of shoot module in plant module design	2.4
$\vec{O}$	old unit vector of the previous direction of root tip	4.5
π	a set of production probabilities mapped one to one into P	2.2.6
$\hat{p}_i$	a unit vector direction from a particle p_i to a presumed target point	3.1
P	a finite set of productions for rewriting systems	2.2.1

Symbols	Meaning	Defined in
$P_0(x, y)$	a source point in the input leaf image	3.2.1
$P_0(x, y, z)$	a source point of the seed of plant	3.3
P_{i_new}	position of new particle after combining the particle i and its nearest particle	3.1
$\hat{q}_i$	a unit vector direction from a particle p_i to its nearest neighbor	3.1
q_x	volumetric flux in x-direction (cm/s)	4.2
$r(t)$	distance from a source point $P_0(x, y)$ to a current point $P_t(x, y)$	3.2.1
R_{i_new}	radius of new particle after combining the particle i and its nearest particle	3.1
$RR_{i,j}$	red intensity of pixel i, j of image I	2.7.5
s	absolute gradient of gray intensity u, $s = \|\nabla u\|$	2.7.4
S	a set of symbols containing elements that remain fixed constants	2.2.1
$S(x, y, z, t)$	nutrient uptake rate by plant roots at time t	4.3
Σ	a set of formal parameters	2.2.8
θ	volumetric water content $(cm^3\ cm^{-3})$	4.2
$\theta(t)$	angle between a source point $P_0(x, y)$ to a unit vector $\overrightarrow{(1, 0)}$ and a source point $P_0(x, y)$ to a current point $P_t(x, y)$	3.2.1
t	time independent variable (s)	4.3
t	time variable of growth function	2.5
t_i	time variable or independent variable at i^{th} data point	2.5.3
$t_{U10\%}$	the growth value at 10% of U	2.5
$t_{U90\%}$	the growth value at 90% of U	2.5
T	time at growth function value $G(t) = \frac{(U+L)}{2}$	2.5
U	maximum value of sigmoidal growth function	2.5
v	pore water velocity (cm/s)	4.3
v_i	absolute direction of network point n_i	3.2.3
V	set of alphabet or variables for rewriting systems	2.2.1
V^*	set of all words over V for rewriting systems	2.2.1
w_p	weighting parameter from a particle p_i to a target point	3.1
w_q	weighting parameter from a particle p_i to a nearest point	3.1
x	position (cm)	4.3
Y_i	length of internode i	2.5
$YY_{i,j}$	gray intensity of pixel i, j of image I	2.7.5

Bibliography

R. Adams and L. Bischof. Seeded region growing. *IEEE Transaction on PAMI*, 16(7):641–647, 1994.

F. A. Aharonian et al. H.E.S.S. (High Energy Stereoscopic System) letter of intent. Technical report, Max-Planck-Institut für Kernphysik, 1997.

J. Arvo and D. Kirk. Modeling plants with environment-sensitive automata. pages 27–33, 1988. URL `http://citeseer.ist.psu.edu/arvo88modeling.html`.

A. Ash, B. Ellis, L.J. Hickey, K. Johnson, P. Wilf, and S. Wing. Manual of leaf architecture- morphological description and categorization of dicotyledonous and net-veined monocotyledonous angiosperms by leaf architecture working group. Technical Report 65p., 1999a.

A. Ash, B. Ellis, L. J. Hickey, K. Johnson, P. Wilf, and S. Wing. Manual of leaf architecture- morphological description and categorization of dicotyledonous and net-veined monocotyledonous angiosperms by leaf architecture working group. Technical Report 65p, 1999b.

D. Azar and G. Toussaint. Hilditch's algorithm for skeletonization, 1997. URL `http://jeff.cs.mcgill.ca/ godfried/teaching/projects97/azar/skeleton.html`.

G. V. G. Baranoski, J. G. Rokne, and P. Shirley. Simulation of light interaction with plants, 2000. URL `http://citeseer.ist.psu.edu/baranoski00simulation.html`.

M. F. Barnsley. *Fractals everywhere*. Boston: Academic press, Boston, 1988.

A. D. Bell and A. Bryan. Plant structure leaves stems and roots, 1993a. URL `http://www.rbgkew.org.uk/ksheets/pdfs/b3plant.pdf`.

A. D. Bell and A. Bryan. *Plant Form: an Illustrated Guide to Flowering Plant Morphology*. Oxford University Press, Oxford, 1993b.

B. Beneš. Fast estimation of growth direction in virtual plants simulation. volume 1, pages 1–10. University of West Bohemia Press, 1997. URL `http://citeseer.ist.psu.edu/499315.html`.

T. R.J. Bossomaier and D. G. Green. *Complex Systems*. Cambridge University Press, Cambridge, 2000.

P. J. Broser. Max-planck-institut fuer medizinische forschung, germany, 2006. URL `http://www.mpimf-heidelberg.mpg.de/public_html/pbroser`.

J. Cermak. Mendel university of agriculture and forestry, brno, faculty of forestry and wood technology, institute of forest ecology, czech republic, Retrived 2007. URL `http://academic.evergreen.edu/projects/ican/research/researchers/cv_jan%20cermak.pdf`.

S. G. Chen, R. Ceulemans, and I. Impens. A fractal-based populus canopy structure model for the calculation of light interception. *Forest Ecology and Management*, 69(01):97–110, 1994.

S. Chuai-Aree, S. Siripant, and C. Lursinsap. Animating plant growth in l-system by parametric functional symbols. In *International Conference on Intelligent Technology 2000*, pages 135–143, 2000.

S. Chuai-Aree, W. Jäger, H. G. Bock, and S. Siripant. Smooth animation for plant growth using time embedded component and growth function. *East-West Journal of Mathematics*, Special Volume:285–295, 2002.

S. Chuai-Aree, W. Jäger, H. G. Bock, S. Siripant, and C. Lursinsap. Plantvr : An algorithm for generating plant shoot and root growth using applied lindenmayer systems. In B. Hu and M. Jaeger, editors, *2003' Intern. Symposium on Plant Growth Modeling, Simulation, Visualization and Their Applications*, pages 55–68, 2003.

S. Chuai-Aree, W. Jäger, H. G. Bock, and S. Siripant. Modeling, simulation and visualization of plant root growth and diffusion processes in soil volume. In C. Godin, J. Hanan, W. Kurth, A. Lacointe, A. Takenaka, P. Prusinkiewicz, T. DeJong, C. Beveridge, and B. Andrieu, editors, *Proc. of 4TH International Workshop on Functional-Structural Plant Models*, pages 289–293, 2004a.

S. Chuai-Aree, W. Jäger, H. G. Bock, and S. Siripant. Plantvr: Software for simulation and visualization of a model for plant growth. In C. Godin, J. Hanan, W. Kurth, A. Lacointe, A. Takenaka, P. Prusinkiewicz, T. DeJong, C. Beveridge, and B. Andrieu, editors, *4th International Workshop on Functional-Structural Plant Models*, page 418, 2004b.

S. Chuai-Aree, W. Jäger, H. G. Bock, and S. Siripant. Simulation and visualization of plant growth using lindenmayer systems. In H. G. Bock, E. Kostina, H. X. Phu, and R. Rannacher, editors, *Modeling, Simulation and Optimization of Complex Processes. Proc. Int. Conf. High Performance Scientific Computing*, pages 115–126. Springer, 2005a.

S. Chuai-Aree, W. Jäger, H. G. Bock, and S. Siripant. Inverse problem of lindenmayer systems on branching structures. In *International Conference in Mathematics and Applications (ICMA-MU 2005)*, volume Special Volume, pages 223–238, 2005b.

S. Chuai-Aree, W. Jäger, H. G. Bock, and S. Siripant. Inverse problem of lindenmayer systems on branching structures. In H. G. Bock, E. Kostina, H. X. Phu, and R. Rannacher, editors, *Modeling, Simulation and Optimization of Complex Processes. Proc. Int. Conf. High Performance Scientific Computing*. Springer, 2007.

S. Chuai-Aree. an algorithm for simulation and visualization of plant growth. Master's thesis, Chulalongkorn University, 2000.

M. F. Cohen and J. R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Boston, MA, 1993. URL `http://citeseer.ist.psu.edu/cohen93radiosity.html`.

N. G. Dengler. Regulation of vascular development. *Journal of Plant Growth Regulation*, 6:967981, 2001.

O. Deussen, P. Hanrahan, B. Linterman, R. Mech, M. Pharr, and P. Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. *Computer Graphics*, 32(Annual Conference Series):275–286, 1998. URL `http://citeseer.ist.psu.edu/deussen98realistic.html`.

C. H. Diaz-Ambrona, A. M. Tarquis, and M. T. Minguez. Faba bean canopy modelling with a parametric open l-system: a comparison with the monsi and saeki model. *Field Crops research*, 58:1–13, 1998.

P. Dimitrov, C. Phillips, and K. Siddiqi. Robust and efficient skeletal graphs. In *Conference on Computer Vision and Pattern Recognition*, 2000.

F. D. Fiore, W. V. Haevre, and F. V. Reeth. Rendering artistic and believable trees for cartoon animation. In *Proceedings of Computer Graphics International (CGI 2003)*, pages 144–151, July 2003. URL `http://citeseer.ist.psu.edu/difiore03rendering.html`.

P. Frankl, K. Ota, and N. Tokushige. Exponents of uniform l-systems. *Journal of Combinatorial Theory Series A*, 75(1):23–43, 1996. URL `http://citeseer.ist.psu.edu/frankl95exponents.html`.

W. V. Haevre and P. Bekaert. A simple but effective algorithm to model the competition of virtual plants for light and space. 11(3):464–471, february 2003. URL `http://citeseer.ist.psu.edu/vanhaevre03simple.html`.

M. S. Hammel and P. Prusinkiewicz. Visualization of developmental processes by extrusion in space-time. In *Proceedings of Graphics Interface 96*, pages 246–258, 1996.

M. S. Hammel, P. Prusinkiewicz, and B. Wyvil. Modelling compound leaves using implicit countours. In *Proceedings of CG International' 92*, pages 119–212, June 1992.

J. S. Hanan and P. M. Room. Practical aspects of virtual plant research. *Plant to ecosystems: Advances in Computational Life Sciences*, 1:28–44, 1997.

J. Hanan. Virtual plants—integrating architectural and physiological plant models. *Proceedings of ModSim 95*, 1:44–50, 1995.

V. F. Hess. Observation of penetrating radiation of seven ballon flights. *Physikalische Zeitschrift*, 13: 1084, 1912.

O. Ippisch. *Coupled Transport in Natural Porous Media*. PhD thesis, Ruprecht-Karls Universität Heidelberg, 2001.

C. Jacob. Genetic l-system programming: Breeding and evolving artificial flowers with mathematica. In *Proceeding of First International Mathematica Symposium, IMS 95*, pages 215–222. Faculty of Electrical Engineering and Computer Science BUT, 1995a. URL `http://citeseer.ist.psu.edu/55092.html`.

C. Jacob. Modeling growth with l-systems and mathematica, 1995b. URL `http://citeseer.ist.psu.edu/jacob95modeling.html`.

J. Kang and N. Dengler. Vein pattern development in adult leaves of arabidopsis. *Journal of Plant Science*, 165:231–242, 2004.

P. Kolesik. Soil and land systems, university of adelaide waite campus sa5064, australia, 2004. URL `http://www.regional.org.au/au/asssi/supersoil2004/s14/oral/1611_McNeilla.htm`.

M. Kolka and I. Provaznik. Connecting of turtles in a programming language based on l-systems. In *Proceedings of 7th Conference Student FEI 2001*, pages 244–248. Faculty of Electrical Engineering and Computer Science BUT, May 2001. ISBN 80-214-1860-5. URL `http://citeseer.ist.psu.edu/kolka01connecting.html`.

W. Kurth and B. Sloboda. Growth grammars simulating trees — an extension of l-systems incorporating local variables and sensitivity, 1997. URL `http://citeseer.ist.psu.edu/kurth97growth.html`.

W. Kurth and B. Sloboda. Tree and stand architecture and growth described by formal grammars, 1999. URL `http://citeseer.ist.psu.edu/kurth99tree.html`.

W. Kurth. Morphological mmodels of plant growth: Possibilities and ecological relevance. *Ecological Modelling*, 75:299–308, 1994a.

W. Kurth. Growth grammar interpreter grogra 2.4 - a software tool for the 3-dimensional interpretation of stochastic, sensitive growth grammars in the context of plant modelling, 1994b. URL `http://citeseer.ist.psu.edu/kurth94growth.html`.

A. Lindenmayer. Mathematical models for cellular interaction in development, parts i and ii. *Journal of Theoretical Biology*, 18:280–315, 1968.

L. Linsen, B. J. Karis, E. G. McPherson, and Bernd Hammann. Tree growth visualization. *The Journal of WSCG*, 13, 2005.

B. Lintermann and O. Deussen. Interactive modeling of plants. *IEEE Computer Graphics and Applications*, 19(1), 1999a. URL `http://graphics.uni-konstanz.de/publikationen/`.

B. Lintermann and O. Deussen. A modelling method and user interface for creating plants. *The Eurographics Association*, 17:73–82, 1998.

B. Lintermann and O. Deussen. Interactive modeling of plants. *IEEE Computer Graphics and Applications*, 19:56–65, 1999b.

M. S. Longair. *Galaxy Formation*, chapter 7. Springer, Germany, 1998.

D. R. Lungley. The growth of root systems a numerical computer simulation model. *Plant Soil*, 38: 145–159, 1973.

B. B. Mandelbrot. *The fractal geometry of nature.* USA: W.N. Freeman, New York, 1983.

J. P. McCormack. Interactive evolution of L-system grammars for computer graphics modelling. In D. G. Green and T. Bossomaier, editors, *Complex Systems: from biology to computation*, pages 118–130. ISO Press, Amsterdam, 1993. URL `http://citeseer.ist.psu.edu/507995.html`.

R. Měch and P. Prusinkiewicz. Visual models of plants interacting with their environment. In *SIGGRAPH*, pages 397–410, 1996a. URL `http://citeseer.ist.psu.edu/mech96visual.html`.

R. Měch and P. Prusinkiewicz. Visual models of plants interacting with their environment. In *Proceeings in Computer Graphics (SIGGRAPH'96)*, pages 397–410, August 1996b.

H. Meinhardt. *Models of Biological Pattern Formation.* Academic Press, London, 1982.

M. Mori. Status of the CANGAROO-III project. In F. A. Aharonian and H. J. Völk, editors, *High Energy Gamma-Ray Astronomy*, pages 578–581, 2000.

H. Noser. Lworld: An animation system based on rewriting, Jul 2002. URL `http://citeseer.ist.psu.edu/article/noser02lworld.html`.

R. A. Ong et al. The VERITAS project. arXiv:astro-ph/0302610v1, March 2003.

P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in Fortran, The Art of Scientific Computing, Second Edition.* Cambridge University Press, America, 1992.

P. Prusinkiewicz and M. S. Hammel. Escape-time visualization method for language-restricted iterated function systems. pages 213–223, 1992a. URL `http://citeseer.ist.psu.edu/479459.html`.

P. Prusinkiewicz and M. S. Hammel. Escape-time visualization method for language-restricted iterated function systems. In *Proceeding of Graphics Interface '92*, pages 213–223, May 1992b.

P. Prusinkiewicz and J. Hanan. Lindenmayer systems, fractals, and plants. *Lecture Notes in Biomathematics, New-York: Springer Verlag*, 75, 1989.

P. Prusinkiewicz and L. Kari. Subapical Bracketed L-Systems. In J. E. Cuny, H. Ehrig, G. Engels, and G. Rozenberg, editors, *Proceeding of 5th International Workshop on Graph Grammars and their Application to Computer Science*, volume 1073, pages 550–564. Springer-Verlag, 1996a. URL `http://citeseer.ist.psu.edu/article/prusinkiewicz96subapical.html`.

P. Prusinkiewicz and L. Kari. Subapical bracketed l-systems. *Lecture Notes in Computer Science*, 1073: 550–564, 1996b.

P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer, New York, 1990.

P. Prusinkiewicz, M. S. Hammel, and E. Mjolsness. Animation of plant development. *Computer Graphics*, 27(Annual Conference Series):351–360, 1993a. URL `http://citeseer.ist.psu.edu/article/prusinkiewicz93animation.html`.

P. Prusinkiewicz, M. S. Hammel, and E. Mjolsness. Animation of plant development. In *Proceedings of Computer graphics SIGGRAPH'93*, pages 351–360, August 1993b.

P. Prusinkiewicz, M. James, and R. Měch. Synthetic topiary. *Computer Graphics*, 28(Annual Conference Series):351–358, 1994a. URL `http://citeseer.ist.psu.edu/480394.html`.

P. Prusinkiewicz, W. Remphrey, C. Davidson, and M. Hammel. Modeling the architecture of expanding fraxinus pennsylvanica shoots using l-systems. *Canadian Journal of Botany*, 72:701–714, 1994b.

P. Prusinkiewicz, M. Hammel, R. Mech, and J. Hanan. The artificial life of plants. In D. Terzopoulos, editor, *SIGGRAPH' 95 course notes*. SIGGRAPH, 1995. URL `http://citeseer.ist.psu.edu/219337.html`.

P. Prusinkiewicz, M. S. Hammel, J. Hanan, and R. Měch. *Handbook of formal language: Visual model of plant development*. Springer-Verlag, Germany, 1996a.

P. Prusinkiewicz, M. Hammel, J. Hanan, and R. Mech. *Visual models of plant development*. Springer–Verlag, Berlin, 1996b. URL `http://citeseer.ist.psu.edu/article/prusinkiewicz96visual.html`.

P. Prusinkiewicz, L. Muendermann, R. Karwowski, and B. Lane. The use of positional information in the modeling of plants. In *Proceedings of SIGGRAPH 2001 (Los Angeles, California)*, pages 289–300, August 2001.

P. Prusinkiewicz. Modeling and Vizualisation of Biological Structures. In *Proceeding of Graphics Interface '93*, pages 128–137, May 1993. URL `http://citeseer.ist.psu.edu/prusinkiewicz93modeling.html`.

W. T. Reeves. Particle systems–a technique for modeling a class of fuzzy objects. *ACM Transactions On Graphics*, 2(2):359–375, 1983.

Y. Rodkaew, S. Siripant, C. Lursinsap, and P. Chongstitvatana. An algorithm for generating vein images for realistic modeling of a leaf. In *In Proceedings of Computational Mathematics and Modeling (CMM2002)*, 2002.

Y. Rodkaew, S. Siripant, C. Lursinsap, and P. Chongstitvatana. Particle systems for plant modeling. In B.-G. Hu and M. Jaeger, editors, *In Proceedings of Plant Growth Modeling and Applications (PMA03)*, pages 210–217, 2003.

Y. Rodkaew. *An alogorithm for generating plant models*. PhD thesis, Chulalongkorn University, 2004.

Y. Rodkeaw, S. Chuai-Aree, C. Lursinsap snd P. Sophatsathit, S. Siripant, and P. Chongstitvatana. Animating plant growth in l-system by parametric functional symbols. *International Journal of Intelligent Systems*, 19(1-2):9–23, 2004.

P. M. Room and J. S. Hanan. Virtual cotton: a new tool for research,management and training. In *Proceedings of the World Cotton Research Conference*, pages 14–17, February 1994.

P. M. Room and J. S. Hanan. Virtual plants: New perspectives for ecologists, pathologists, and agricultural scientists. *Trends in Plant Science*, 1:33–38, 1996.

P. M. Room, L. Maillette, and J. Hanan. Module and metamer dynamics and virtual plants. *Advances in Ecological Research*, 25:105–157, 1994.

A. Runions, M. Fuhrer, B. Lane, P. Federl, A. Roland-Lagan, and P. Prusinkiewicz. Modeling and visualization of leaf vanation patterns. *ACM Transactions on Graphics*, 24(3):702–711, 2005.

J. C. Russ. *The Image Processing Handbook*. CRC Press, 1995.

A. Samal, B. Peterson, and D. J. Holliday. Recognizing plants using stochastic l-systems. *IEEE International Conference*, 1:183–187, 1994.

J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, London, 1999.

I. Shlyakhter, M. Rozenoer, J. Dorsey, and S. Teller. Reconstructing 3d tree models from instrumented photographs. *IEEE Computer Graphics and Applications*, 21(3):53–61, 2001. URL `http://citeseer.ist.psu.edu/article/shlyakhter01reconstructing.html`.

L. G. Sieburth. Auxin is required for leaf vein pattern in arabidopsis. *Plant Physiology*, 121:1179–1190, 1999.

A. L. Smith, A. G. Bengough, C. Engels, M. van Noordwijk, S. Pellerin, and van de S.C. Geijn. *Root Methods: A Handbook*. Springer-Verlag, Berlin Heidelberg New York, 2000.

A. R. Smith. Plants, fractals, and formal languages. *ACM SIGGRAPH*, 18:1–10, 1984.

M. Soch and B. Beneš. Simulation of plant development using strands model. In T. Plachetka, editor, *In Proceedings of 11th Spring Conference on Computer Graphics*, pages 30–35, May 1995. URL `http://citeseer.ist.psu.edu/86191.html`.

C. Soler, F. Sillion, F. Blaise, and P. Dereffye. A physiological plant growth simulation engine based on accurate radiant energy transfer. Technical Report 4116, 2001. URL `http://citeseer.ist.psu.edu/soler01physiological.html`.

F. Somma, J. W. Hopmans, and V. Clausnitzer. Transient three-dimensional modeling of soil water and solute transport with simultaneous root growth, root water and nutrient uptake. *Plant and Soil, Kluwer Academic Publisher*, 202:281–293, 1998.

R. Strzodka and A. Telea. Generalized distance transforms and skeletons in graphics hardware. *The Eurographics Association*, 2004.

A. A. Suratgar, M. B. Tavakoli, and A. Hoseinabadi. Modified levenberg-marquardt method for neural networks training. *Enformatika*, 2005.

L. Szemla. 3d models generator simulating a grow of natural objects for virtual reality. In *Proceedings of the 4th Central European Seminar on Computer Graphics for students (CESCG 2000)*, May 2000. URL `http://citeseer.ist.psu.edu/501383.html`.

S. F. Thompson. Growth models for shapes, 1994. URL `http://citeseer.ist.psu.edu/thompson94growth.html`.

S. Thompson and A. Rosenfeld. Isotropic growth on a grid. *Pattern Recognition*, 28:241–253, 1995. URL `http://citeseer.ist.psu.edu/thompson95isotropic.html`.

M. T. van Genuchten. A closed form equation for predicting the hydraulic conductivity of unsaterated soils. *Soil Science Society of America Journal*, 44:892–898, 1980.

M. T. van Genuchten. A numerical model for water and solute movement in and below the root zone. *Research Report, U.S. Salinity Lab, ARS USDA, Riverside, Ca.*, (121), 1987.

J. Weickert. *Anisotropic Diffusion in Image Processing.* B. G. Teubner Stuttgart, Stuttgart, 1998.

C. Zach, S. Mantler, and K. Karner. Time-critical rendering of discrete and continuous levels of detail. *Eurographics Workshop on Rendering*, pages 1–8, 2002. URL `http://citeseer.ist.psu.edu/zach02timecritical.html`.

Curriculum Vitae

Curriculum Vitae

Name : Mr. Somporn Chuai-Aree

Birth : September 30, 1974 in Nakhon Si Thammarat, Thailand

Educational Background :

- Primary school (1987), Wat Khuankuey school, Nakhon Si Thammarat, Thailand
- Junior high school (1990), Khuankueysuttiwittaya school, Nakhon Si Thammarat, Thailand
- Senior high school (1993), Thung Song school, Nakhon Si Thammarat, Thailand
- B.Sc. (2^{nd} Class Honors) in Applied Mathematics (1996), Prince of Songkla University, Pattani campus, Thailand
- M.Sc. in Computational Science (2000), Chulalongkorn University, Bangkok, Thailand

Professional Experiences :

- Lecturer and researcher in the Faculty of Science and Technology, Prince of Songkla University (PSU), Pattani campus, Thailand, 1997-present
- Member of the International Graduate School (International Graduiertenkolleg, IGK 710), "Complex Processes: Modeling, Simulation and Optimization" at the Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR), University of Heidelberg, Germany, 2001-2004
- Scientist at the Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR), University of Heidelberg, Germany, 2001-2008

Scholarships and Awards :

- B.Sc. Scholarship from Prince of Songkla University, Thailand, 1994
- B.Sc. Scholarship from the Royal Thai Government, Thailand, 1995-1996
- TAB Foundation Scholarship, Chulalongkorn University, Thailand, 1998-1999
- TAB Foundation Awards for maximum GPAX (3.89) in M.Sc. (Computational Science), 2001
- Ph.D. Scholarship from Ministry of Science and Technology, the Royal Thai Government, Thailand, 2001-2006
- Scholarship from Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR), University of Heidelberg, Germany, 2007-2008

Journals

- **S. Chuai-Aree**, W. Jäger, H. G. Bock, and S. Siripant : Reconstruction of Branching Structures Using Region and Volume Growing Method, East-West Journal of Mathematics, Special Volume, pp. 253-267, 2005.
- W. Kanbua, and **S. Chuai-Aree** : Virtual Wave: An Algorithm for Visualization of Ocean Wave Forecast in the Gulf of Thailand, KMITL Science Journal Vol.5, No.1 Feb.2005, 140-150.
- Y. Rodkeaw, **S. Chuai-Aree**, C. Lursinsap, P. Sophatsathit, S. Siripant, and P. Chongstitvatana : Animating Plant Growth in L-System by Parametric Functional Symbols, International Journal of Intelligent Systems, Vol. 19(1-2), (2004), pp. 9-23.
- **S. Chuai-Aree**, W. Jäger, H. G. Bock, and S. Siripant : Smooth Animation for Plant Growth Using Time Embedded Component and Growth Function. East-West Journal of Mathematics, Special Volume, pp. 285-295, 2002.
- **S. Chuai-Aree**, C. Lursinsap, P. Sophatsathit, and S. Siripant : Fuzzy C-Mean: a Statistical Feature Classification of Text and Image Segmentation Method, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, Vol. 9(6), (2001), pp. 661-671, World Scientific Publishing Company.

Proceedings

- **S. Chuai-Aree**, H. G. Bock, W. Jäger, and S. Siripant : ALGM : An Algorithm for Leaf Growth Measurement Using Region Growing Method, Proc. of Intern. Conf. on High Performance Scientific Computing (HPSCHanoi 2009), March 2-6, Hanoi, Vietnam, 2009.
- A. Busaman, **S. Chuai-Aree**, R. Saelim, and W. Kanbua : Modeling, Simulation and Visualization of Water Flooding, Proc. of 14th Annual Meeting in Mathematics 2009 (AMM 2009), March 5-6, Suratthani, Thailand, 2009.
- W. Kanbua, and **S. Chuai-Aree** : The Simulation of Wave model Under North-East Monsoon Along Coastline of Thailand, Japan - Thailand Estuary Workshop 2008, Venue: Thai Meteorological Department, Bangkok, Thailand, August 19 - 20, 2008
- W. Kanbua, and **S. Chuai-Aree** : Ocean Wave Simulation Under North-East Monsoon Effect in the Gulf of Thailand, 13th National Convention on Civil Engineering, Venue: Pattaya, Chon Buri, Thailand, May 14 - 16, 2008
- W. Kanbua, and **S. Chuai-Aree** : Tsunami Propagation Prediction by SiTProS Model, Internation Symposium on the Restoration Program from Giant Earthquake and Tsunamis, Venue: Royal Phuket City Hotel, Phuket, Thailand, January 22-24, 2008
- **S. Chuai-Aree**, and W. Kanbua : SiTProS: Fast and Real-Time Simulation of Tsunami Propagation, AMS2007: Asia Modelling Symposium international conference, Phuket, Thailand, March 27-30, 2007.
- W. Kanbua, and **S. Chuai-Aree** : Understanding the Simple Model of Tsunami Propagation by SiTProS Model, The 11th Annual National Symposium on Computational Science and Engineering (ANSCSE11), Phuket, Thailand, March 28-30, 2007.
- **S. Chuai-Aree**, S. Siripant, W. Jäger, and H. G. Bock : PlantVR: Software for Simulation and Visualization of Plant Growth Model, PMA06: The Second International Symposium on Plant growth Modeling, Simulation, Visualization and Applications, Beijing (CHINA PR), November 13-17, 2006.
- **S. Chuai-Aree**, S. Siripant, W. Jäger, and H. G. Bock : An Algorithm for Simulation of Leaf Vein, Shoot and Root Growth Based on Their Environmental Factors, PMA06: The Second International Symposium on Plant growth Modeling, Simulation, Visualization and Applications, Beijing (CHINA PR), November 13-17, 2006.

- S. Siripant, and **S. Chuai-Aree** : Simulation and Visualization of Leaf Growth, Proc. of Intern. Conf. on High Performance Scientific Computing (HPSCHanoi 2006), March 6-10, Hanoi, Vietnam, 2006.
- H. G. Bock, **S. Chuai-Aree**, W. Jäger, W. Kanbua, S. Krömker, and S. Siripant : 3D Cloud and Storm Reconstruction from Satellite Image, Proc. of Intern. Conf. on High Performance Scientific Computing (HPSCHanoi 2006), March 6-10, Hanoi, Vietnam, 2006.
- H. G. Bock, **S. Chuai-Aree**, W. Jäger, and S. Siripant : Inverse Problem of Lindenmayer Systems on Branching Structures, Proc. of Intern. Conf. on High Performance Scientific Computing (HPSCHanoi 2006), March 6-10, Hanoi, Vietnam, 2006.
- **S. Chuai-Aree**, W. Jäger, H. G. Bock, and S. Siripant : Reconstruction of Branching Structures Using Region and Volume Growing Method International Conference in Mathematics and Applications (ICMA-MU 2005), Bangkok, Thailand, December 15-17, 2005, pp. 223-238.
- W. Kanbua, and **S. Chuai-Aree**: Virtual Wave: an Algorithm for Visualization of Ocean Wave Forecast in the Gulf of Thailand. The 2nd International Symposium on Mathematical, Statistical and Computer Sciences, Bangkok, Thailand, February 19-20, 2005.
- **S. Chuai-Aree**, W. Jäger, H. G. Bock, and S. Siripant : Modeling, Simulation and Visualization of Plant Root Growth and Diffusion Processes in Soil Volume. Proc. of 4TH International Workshop on Functional-Structural Plant Models, Montpellier, France, June 7-11, 2004, pp. 289-293.
- **S. Chuai-Aree**, W. Jäger, H. G. Bock, and S. Siripant : PlantVR: Software for Simulation and Visualization of a Model for Plant Growth. Proc. of 4TH International Workshop on Functional-Structural Plant Models, Montpellier, France, June 7-11, 2004, pp. 418.
- **S. Chuai-Aree**, W. Jäger, H. G. Bock, S. Siripant, and C. Lursinsap : PlantVR: An Algorithm for Generating Plant Shoot and Root Growth Using Applied Lindenmayer Systems. In B.Hu and M. Jaeger (eds.): Proc. of 2003' Intern. Symposium on Plant growth Modeling, simulation, visualization and their Applications, Tsinghua University Press - Springer Verlag, October 13-16, Beijing, China, pp. 55-68, 2003.
- **S. Chuai-Aree**, W. Jäger, H. G. Bock, and S. Siripant : Simulation and Visualization of Plant Growth using Lindenmayer Systems, Proc. of Intern. Conf. on High Performance Scientific Computing (HPSC), March 10-14, Hanoi, Vietnam, 2003.
- **S. Chuai-Aree**, W. Jäger, H. G. Bock, and S. Siripant : Smooth Animation for Plant Growth Using Time Embedded Component and Growth Function. In N. V. Sanh, Y. Lenbury, Y. H. Wu and B. Wiwatanapataphee (eds.): Proc. of Intern. Conf. on Computational Mathematics and Modeling, May 22-24, Thailand, in East-West Journal of Mathematics, special volume, pp. 285-295, 2002.
- **S. Chuai-Aree**, S. Siripant, and C. Lursinsap : Animating Plant Growth in L-system by Parametric Functional Symbols, Proc. of Intern. Conf. on Intelligent Technology 2000, December 13-15, Assumption University Bangkok, Thailand, pp. 135-143, 2000.
- **S. Chuai-Aree**, C. Lursinsap, P. Sophatsathit, and S. Siripant : Fuzzy C-Mean: a Statistical Feature Classification of Text and Image Segmentation Method, Proc. of Intern. Conf. on Intelligent Technology 2000, December 13-15, Assumption University Bangkok, Thailand, pp. 279-284, 2000.

Poster

- **S. Chuai-Aree**, W. Jäger, H. G. Bock, and S. Siripant : Modeling, Simulation and Visualization of Plant Root Growth and Diffusion Processes in Soil Volume. Proc. of 4TH International Workshop on Functional-Structural Plant Models, Montpellier, France, June 7-11, 2004, pp. 289-293.

Software Demonstration in Conference

- **S. Chuai-Aree**, W. Jäger, H. G. Bock, and S. Siripant : PlantVR: Software for Simulation and Visualization of a Model for Plant Growth. Proc. of 4TH International Workshop on Functional-Structural Plant Models, Montpellier, France, June 7-11, 2004, pp. 418.

Index