Modeling Vegetation with L-systems Using an Image

Srinidhi Hegde Swati Rathi

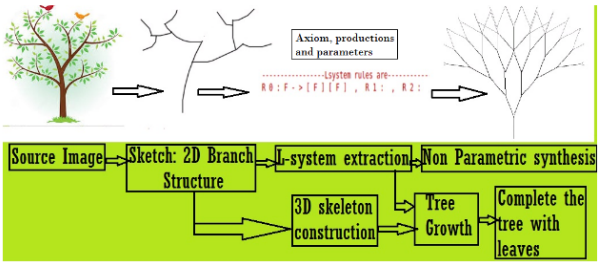

Figure 1: *Proposed Pipeline for developing 3D model of tree from its 2D image*

Abstract

Realistic models of vegetations are very essential piece of immersive virtual environment simulations. We are developing a novel technique to render models of visually similar trees from a single reference image of a tree that is to be modelled. We use a user guided method to extract out the 2D skeleton. The technique also encompasses the usage of a procedural vegetation generation technique - the L-systems. In this report we present a modelling pipeline for developing a 3D tree model of tree starting from a single image of the tree.

Keywords: procedural vegetation, l-systems, vegetation modelling

1 Introduction

3D models of botanical trees are important in geographical landscape simulation, cityscape design, virtual reality and other fields of 3D graphics. However, designing tree models is challenging because trees have enormous structural complexity. Tree modeling techniques can be classified as:

- Rule based: This requires expertise in botany and adjustment of parameter values.
- Image based: This helps in designing realistic trees but they cannot be modified.

In our approach we try to synthesize L-system rules from a single image of a tree, hence there is no need for any botany expertise . Also we use the deduced L-system rules for creating variations of the modeled tree, by tweaking its parameters, thereby generating a virtual natural landscape.

2 Literature Survey and Other Works

The growth of trees adheres to strong botanic rules and patterns. Many previous methods are rule based. Prusinkiewicz et al. [Prusinkiewicz et al. 1994] introduced a series of methods based on the idea of the generative L-system. [Weber and Penn 1995] used geometric rules to produce realistic-looking trees. de Reffye et al. [De Reffye et al. 1988] designed rules according to botanical knowledge. These techniques generate good results, but they often require expertise for effective use. The idea behind rule based methods is that the branch and leaf arrangement follow a pattern which can be predicted with a set of rules and parameters. However, these rules and parameters are nontrivial to set.

In the past several years, image based methods become popular. These methods use images to recover 3D structure of the tree. Reche-Martinez et al. [Reche-Martinez et al. 2004] recovered the opacity and color of each cell of a volume containing the tree. Although realistic results can be produced by this method, its volumetric representation makes editing and animation almost impossible.

Our approach is based on the method proposed by Ruoxi Sun et.al [Sun et al. 2009] which combines rule-based and image-based techniques and is able to intelligently model lightweight trees. In this approach, 3D skeleton of a tree trunk and the branching structure are reconstructed from images using binocular vision methods, and the parametric L-system is extracted from the reconstructed 3D skeleton and branching structure.

3 Procedural Generation of Vegetation: L-Systems Overview

Lindenmayer systems or L-systems for short were conceived as a mathematical theory of plant development. In 1968 a biologist, Aristid Lindenmayer, introduced a new type of string-rewriting mechanism, subsequently termed L-systems. The essential difference between Chomsky grammars and L-systems lies in the method of applying productions. In Chomsky grammars productions are applied sequentially, whereas in L-systems they are applied in parallel and simultaneously replace all letters in a given word. This difference reflects the biological motivation of L-systems. Productions are intended to capture cell divisions in multicellular organisms, where many divisions may occur at the same time. In order to model branching structures in trees an extension of L-Systems is used which is the Bracketed L-system. Thus trees can be represented using strings with bracket. Two new symbols are introduced to delimit a branch. They are:

- '[' 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.

4 Modelling Pipeline

The pipeline that we have proposed is not tested completely, that is , some of the modules of the pipeline are yet to be tested. Others

have been verified to produce satisfactory results. In the following subsection we present details about some of the modules that are tested successfully.

4.1 Prerequisites for Image

The prerequisites for the image that will be used for reference, also called reference image, are really simple. Firstly, the silhouette of the tree should be clearly demarcated from the background and secondly, the number visible branches, that are not occluded by leaves or other objects in the image, should be maximum possible. These prerequisites are not really essential for determining the 2D tree skeleton as in the next module we allowing the users to draw out the relevant visible branches of the tree. So the prerequisites will help in improving the user experience.

4.2 Sketching 2D Branch Strokes

While modelling a tree the initial step is to capture the branching structure of the tree. We must be able to determine the branching structure accurately as it forms an essential component of the tree structure. For this reason we allow the user to sketch the visible branches in the image. Each stroke drawn by a user represents an individual branch and it is drawn as a polyline to represent curves in a branch. The strokes drawn by user have been approximated using the Douglas Peukar algorithm. Douglas Peukar algorithm is used for simplifying the curves by reducing the number of points on the curve. The output of this algorithm is an approximation of the original curve drawn by the user. The algorithm initially includes the starting and the ending point on the curve. It then joins the two end points and determines the point that is farthest from this line. It includes the farthest point only if its orthogonal distance from the line segment is greater than a threshold e. As the value of e reduces the curve becomes smoother since the number of points used for approximating the curve increases.

4.3 Deducing L-Systems and Physical Tree Parameters

Once the skeleton is obtained from the image we deduce L-System rules from the skeleton. The deduced L-system rules help in growing the portion of tree branches which is occluded by the foliage and hence was not drawn by the user. L-system grammar is specified by an axiom(the starting point), set of production rules and a set of parameters. The logical structure of tree branches is decided by axiom and productions. Tree parameters like the branch lengths, branch widths and branching angles helps deciding the tree shape. The extraction of axiom and production from the 2D skeleton follows 2 Steps: Scan the branching structure of the 2D skeleton first bottom-up then left-right to build a initial production set P. Let the root branch r to be the axiom. Find the similar substructures in the branching structure of the 2D skeleton, record them in a table R. Reduce P according to R. The extraction algorithm will be illustrated by the sample skeleton in Figure 2.

Before extracting productions, we need to define two categories of branches first: the branches with no children are terminal, the others are internal. As shown in Figure 2, branches with dashed lines are terminal, the others are internal. Each internal branch is associated with a unique ID. Terminal branches have a common ID t because they have the same structure. Numbers (or letter) beside the branches in Figure 2 are their IDs. A production has the form of predecessor successor, where predecessor represents the ID of a parent branch and successor is a string composed of IDs of children branches belonging to predecessor. The branching structure of the tree trunk can be fully translated into a set of productions P.

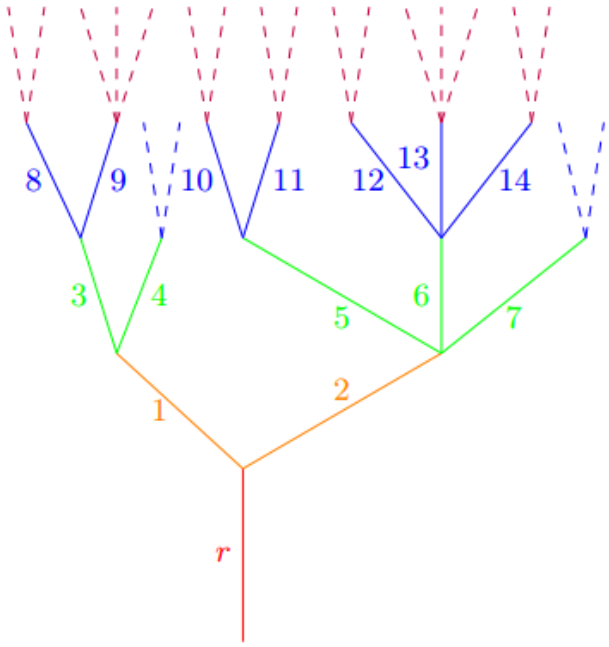

Figure 2: *The illustration of a branching structure. Dashed lines represent terminal branches, others are internal branches. IDs of internal branches are marked out and the ID t of terminal branches is omitted.*

All productions in P are sorted in the specified order, first bottom-up, then left right.. In addition, we define equivalent productions as ones with the same successors. Equivalent productions follow a property: if two productions P[i] and P[j] are equivalent, P[j].p appearing in the successor of a production can be replaced by P[i].p. In this case, P[j] can be deleted from P for simplification. This deletion can be repeated until there are no equivalent productions in P. The process of deletion is called pre-reduction. Table 2 also lists the result after pre-reduction.

Next work is to analyze the rest productions to find those productions who have similar structures. Similar structures mean those branches who have the same sequel-context in the tree skeleton until one of the branches reaches to a terminal branch. Judging if two branches are similar is a recursive process. For instance, in Figure 2, branch 3 and r are similar since 3s children 8 and 9 are similar to rs children 1 and 2 respectively. Judging process stops when branch 3 sequel-context reaches to 8s two children and 9s three children which are all terminal branch t. At the same time, rs sequel-context reaches to 1s two children 3, 4 and 2s three children 5, 6, 7. This process judges the similarity between not only 3 and r, but also all the descendants during the whole process. They are 8 to 1, 9 to 2, 8s children to 3, 4 respectively and 9s children to 5, 6, 7 respectively. Then the reduction table R is used to record all the similar relationship among all the branches. R is organized as a two-column table, the first column is the base ID and the second one is all the IDs similar to the ID in the first column and in the lower level in the skeleton. After building R, P is reduced according to R. The reduction process is replacing the ID appearing in P(ID) with R(ID). If more than one elements exist in R(ID), select the one who has the lowest level in the tree skeleton and at the most left. For instance, 14 is replaced by 3. The whole replacing process need to follow the bottom-up order when traverse R. By replacing IDs in R,

terminal branch can be replaced by a similar internal branch in our algorithm. However, if a terminal branch does not have the similar structure in the lower level, it can be reduced by the root branch r. As a result, the productions in P obtained the recursiveness, which allows the infinite growth ability of P. P after reduction is listed in Table 2.

ID	$\mathscr{P}(\mathbf{ID}).s$
r	$\{1,2\}$
1	$\{3,4\}$
2	$\{5,6,7\}$
3	$\{8,9\}$
4	$\{t,t\}$
5	$\{10,11\}$
6	$\{12,13,14\}$
7	$\{t,t\}$
8	$\{t,t\}$
9	$\{t,t,t\}$
10	$\{t,t\}$
11	$\{t,t\}$
12	$\{t,t\}$
13	$\{t,t,t\}$
14	$\{t,t\}$

ID	$\mathscr{P}(\mathbf{ID}).s$
r	$\{1,2\}$
1	$\{3,14\}$
2	$\{5,6,14\}$
3	$\{14,13\}$
5	$\{14,14\}$
6	$\{14,13,14\}$
13	$\{t,t,t\}$
14	$\{t,t\}$

Figure 3: *Left: initial P; Right: after pre-reduction.*

ID	$\mathscr{R}(\mathbf{ID})$
3	$\{r\}$
5	$\{1\}$
6	$\{2\}$
13	$\{6\}$
14	$\{3,5\}$

ID	$\mathscr{P}(\mathbf{ID}).s$
r	$\{1,2\}$
1	$\{r,r\}$
2	$\{1,2,r\}$

Table 2: *Left: R; Right: P after reduction.*

Figure 4: *Left: R; Right: P after reduction.*

4.4 Conversion of 2D Tree Skeleton to 3D

This module focussed on generation of 3D skeleton from a 2D skeleton without applying L-system production rules. The first of this task requires us to segregate and identify each branch. For this a line segment in the skeleton or a set of continuous line segment strips, also known as polylines, were to be identified separately from the skeleton. The convention that is followed in our case while drawing 2D strokes is that the first stroke will always represent the tree trunk. All these branches and trunk are indexed (0-based indexing) according to the sequence in which they were drawn. The start and terminal node location are stored along with the parent and child branch for each branch in the tree skeleton.

The crucial part of the 3D skeleton construction is branch placement in 3D space. According to the method described in [Okabe et al. 2007], the aim of that method was an optimal placement of branches that are maximally angled. For this purpose, we first place branches one by one and calculate the distance transform or the distance map for the top view of the 2D skeleton of tree every time we place a new branch and then compute the regions of local maximas in the obtained distance map. Among these areas of local maximas a region is selected that ensures maximal angled branch placement when we add a new branch in the next iteration.

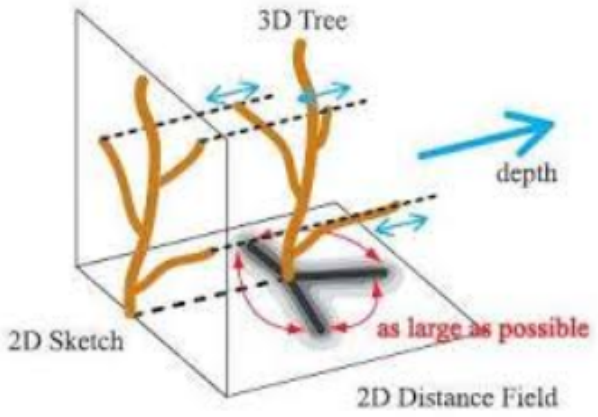

Figure 5: *Creation of distance maps.*

In our implementation, for branch placement, we resorted to uniform angular distribution in all directions. Furthermore, we added random perturbations to the angles to make the models more realistic. Thus the angle, θ, is calculated using the following equation,

$$\theta = (2 * \pi/n) + \delta \tag{1}$$

where δ is the random perturbation parameter, n is the number of immediate children of tree trunk.

The challenging part of this module was to ensure that the 3D skeleton, so created, should preserve the visual details from the perspective of the reference image. For this we employed rotation with scaling to deal with this challenge. We rotated the branch, that is to be placed, by an angle of θ and then we scaled each node of the branch to preserve the branch length as per the reference image.

There were some limitations regarding the approach we used. Although this technique perfectly captures and preserves the reference image details but it fails to look realistic from other angles due to the fact that the new z-coordinates of the nodes are linearly interpolated from start to end of the branch.

5 Future Work

In the following subsections we mention some of the unfinished tasks that we will take up in the future. These are the untested modules of the proposed pipeline.

5.1 Creating 3D Tree Model

In the previous module of the pipeline we obtained a 3D tree skeleton of the required tree. This module of the pipeline will use this 3D skeleton as input to produce a 3D model of the tree with accordance

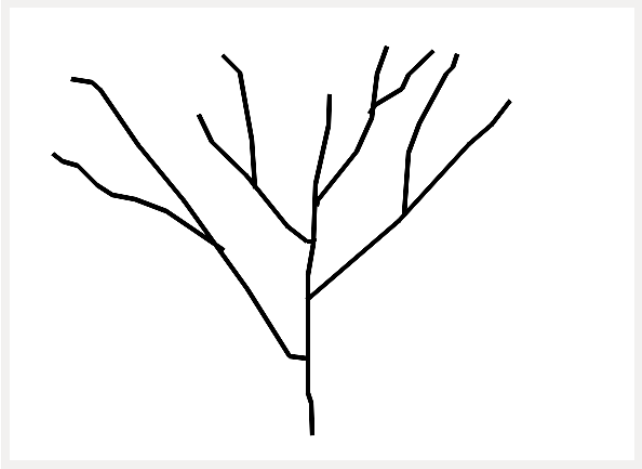

Figure 6: *Sketch of 2D skeleton by user.*

with the deduced physical parameters of the tree from the reference image.

The approach that we are planning to use to create a 3D model is to use generalized cylinders as mentioned in [Mech et al. 1997]. For creating an effect of smooth transitions at bending of the tree branches we will include a simple spherical primitive which has its radius equal to the radii of the ends, between which sphere is to be included, of the 2 generalized cylinders which are to be connected at this junction.

5.2 Fine Tuning and Further Beautification

As a part of our future work proposal we look into the beautification of tree models by enhancing the Level of Details (LoD) to the model developed from the above mentioned pipeline. Some of the features that we plan to implement are integrating leaf models to the tree model structure and texturing the above models - both of trees and leaves.

Some of the features that are planned as far as leaf models are considered are, firstly, random and evenly distributed venation patterns on the leaf surface and, secondly, the introduction of cuts in leaf silhouette and thirdly, the feature of decaying leaves among normal leaf pattern on tree.

For the texturing portion of beautification of tree model, we are planning to texture the tree branches, trunks and leaves. For this purpose we will require two different texture images - one for the tree bark texture and the other for leaf textures.

References

DE REFFYE, P., EDELIN, C., FRANÇON, J., JAEGER, M., AND PUECH, C. 1988. *Plant models faithful to botanical structure and development*, vol. 22. ACM.

MECH, R., PRUSINKIEWICZ, P., AND HANAN, J. 1997. Extensions to the graphical interpretation of l-systems based on turtle geometry.

OKABE, M., OWADA, S., IGARASHI, T., AND IGARASHI, T. 2007. Interactive design of botanical trees using freehand sketches and example-based editing. In *ACM SIGGRAPH 2007 courses*, ACM, 26.

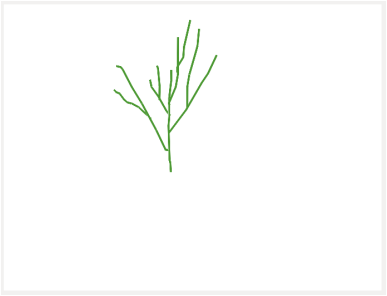

Figure 7: *Original view of model generated.*

Figure 8: *Back-view of model generated.*

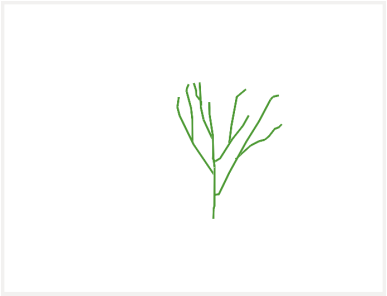

Figure 9: *Side-view of model generated*

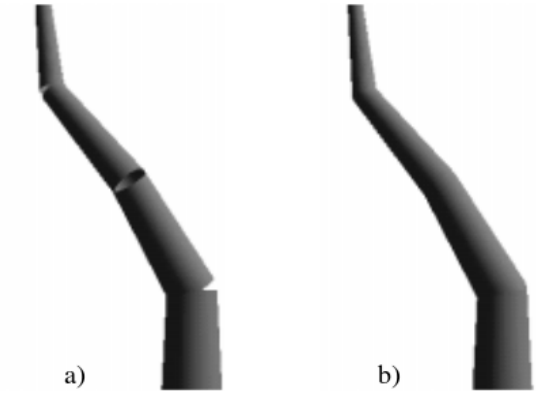

Figure 10: *Generalized cylinders (a)without spherical primitives.(b) with spherical primitives.*

PRUSINKIEWICZ, P., JAMES, M., AND MĚCH, R. 1994. Synthetic topiary. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM, 351–358.

RECHE-MARTINEZ, A., MARTIN, I., AND DRETTAKIS, G. 2004. Volumetric reconstruction and interactive rendering of trees from photographs. In *ACM transactions on graphics (ToG)*, vol. 23, ACM, 720–727.

SUN, R., JIA, J., LI, H., AND JAEGER, M. 2009. Image-based lightweight tree modeling. In *Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry*, ACM, 17–22.