6-6-2001

Recognition of plants using a stochastic L-system model

Ashok Samal
University of Nebraska - Lincoln, asamal1@unl.edu

Brian Peterson
University of Nebraska - Lincoln

David J. Holliday
University of Nebraska - Lincoln

Journal of Electronic Imaging 11(1), 50–58 (January 2002).

Recognition of plants using a stochastic L-system model

Ashok Samal
Brian Peterson
David J. Holliday
University of Nebraska
Dept. of Computer Science and Engineering
115 Ferguson Hall
Lincoln, Nebraska 68588-0115
E-mail: samal@cse.unl.edu

Abstract. *Recognition of natural shapes like leaves, plants, and trees, has proven to be a challenging problem in computer vision. The members of a class of natural objects are not identical to each other. They are similar, have similar features, but are not exactly the same. Most existing techniques have not succeeded in effectively recognizing these objects. One of the main reasons is that the models used to represent them are inadequate themselves. In this research we use a fractal model, which has been very effective in modeling natural shapes, to represent and then guide the recognition of a class of natural objects, namely plants. Variation in plants is accommodated by using the stochastic L-systems. A learning system is then used to generate a decision tree that can be used for classification. Results show that the approach is successful for a large class of synthetic plants and provides the basis for further research into recognition of natural plants.* © *2002 SPIE and IS&T.*
[DOI: 10.1117/1.1426081]

1 Introduction

The success of computer vision techniques to identify complex and diverse shapes, like natural objects such as trees and flowers has been limited for a variety of reasons. A central issue in the recognition process is a suitable representation, or model, for that class of objects. Although a model is a simplified representation of a set of entities, it still must account for their inherent complexity and variation. Models used in many approaches are not suitable to represent the natural shapes. The complexity and variability of plants, in particular, are inadequately modeled. Fractals provide new hope that this goal can be accomplished systematically and accurately.

Fractals have very compact representations but at the same time are able to generate very complex shapes that accurately represent natural objects. They have been used extensively and successfully in computer generated imagery for realistic scenes. Figure 1 shows a natural tree and a corresponding fractal model.

L-system (named after its inventor, Aristid Lindenmayer) is a class of fractals that has been used to model plants at different stages of their growth. In this research we present an approach to recognize plants using the L-system as the underlying model. This paper builds on earlier research using fractal models for recognition to include the variability in shapes of trees resulting from different growth patterns.[1,2] The approach has been implemented and tested thoroughly. A decision tree is used to aid the recognition process. To simplify the derivation of the decision tree from a large number of samples, a learning system is used. The results show the approach has a very high recognition rate for synthetic plants. We present the results in this paper and propose a detailed study of usefulness of this approach to the recognition of natural plants for future research.

1.1 *Fractals and Natural Objects*

A model is a simplified representation of some or all of the features of an object. One of its purposes is to facilitate the visualization of the structure of the object. It should allow convenient experimentation whereby some of its parameters can be altered and the resulting object can be displayed quickly.[3] More importantly, it should facilitate the understanding of these shapes. Determination of a suitable model for a class of natural objects may be difficult because of the complexity and variation found in them. Any model used to represent a particular class of natural objects needs to address these difficulties. In particular, a model chosen to represent a class of natural objects should be able to generate objects with comparable complexity to the objects found in the domain. Fractals have been used to successfully model large classes realistic objects.[4–9]

1.1.1 *Fractals*

The term ''fractal'' was first coined by Mandelbrot to describe a model for objects that lack a concise description in terms of traditional Euclidean geometry.[5] Objects generated from a fractal can be thought of as lying somewhere between the normal two and three dimensions of Euclidean geometry. Fractals generally lack a clear, precise math-

Paper 20031 received Apr. 17, 2000; accepted for publication June 6, 2001.
1017-9909/2002/$15.00 © 2002 SPIE and IS&T.

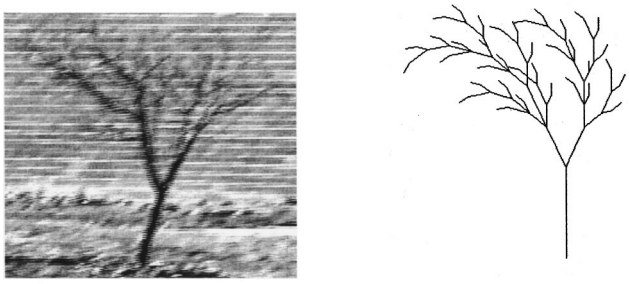

Fig. 1 A naturally occurring tree and its L-system fractal model.

ematical formula for a description, but instead are described by the kinds of objects that are generated by them.[10]

Although there are many different types of fractals, they all have several important properties in common. These properties are really what define the types of objects that can be considered fractals. One such property, and arguably the most important, is that the objects possess some kind of *self-similarity*. That is, the object is relatively invariant under magnification and change of scale.[6]

Another striking feature of fractal objects is their complexity. Objects generated from fractals may have, for example, boundaries that are too complex to describe in traditional geometric terms. Even though fractals are very complex, they often have quite simple representations. A fractal object is typically represented algorithmically, rather than by a formula. That is, a fractal object consists of an algorithm or set of rules governing the generation of the object.

The L-system is one type of fractal that has been used to successfully model a class of natural objects, namely higher plant structures such as leaves, branches, and trees. It can model complex natural objects easily and has a compact representation.

1.1.2 *Recognition*

A model is also central to recognition of objects in an image. That is, in order to recognize if a given object is in a particular class, one must have a model for the class. The problem of recognizing an object then becomes one of classification. There are many techniques available for this purpose.[11] A feature-based approach is used in this work for recognition, which starts with feature extraction. A feature can be defined very broadly as any extractable measurement.[12]

There are several problems when using a feature-based approach to recognition. One problem is that of feature selection. One must select features that make the classification process easy and accurate. Feature extraction is also a major problem since natural objects are often quite complex, making it difficult to accurately and efficiently extract them. In addition, a good model for description of a class of objects may be so complicated that derivation of useful features for classification may not be feasible. Thus, selection and computation of features are nontrivial operations.

1.2 *Modeling in Two Dimensions*

The context-free and stochastic L-systems presented here generate and represent objects that are two dimensional. Of course, natural trees and plants are three dimensional, so a two-dimensional model may not be seen as very appropriate. However, it can be argued that working in two dimensions only is not really a problem when attempting to model trees and plants. Naturally occurring trees have several properties which validate the use of a two-dimensional model.

First, trees generally look similar from all directions. Very similar branching patterns and structure can be seen in a tree by viewing one from a variety of directions. When viewed from long distances a tree tends towards appearing two dimensional. Another important consideration is that trees can be recognized from their two-dimensional projections. The information available from three dimensions is somewhat redundant since branching patterns and structure can be inferred from a two-dimensional view of the tree or plant. It should be noted that three-dimensional L-system models do exist.[4] These L-systems are very similar to the two-dimensional L-systems, but with a few additions.

1.2.1 *Previous work*

Fractals have already been used in computer vision to model and recognize natural shapes,[13–18] terrain modeling and target detection[19,20] and texture analysis.[21–24] However, the use of fractal models for recognition of single objects, such as plants and flowers, has been relatively unexplored. Samal and Holliday[1,2] used a simpler L-system model to recognize plants. In this paper, a more complex and a more realistic model of L-system, namely, stochastic L-system, has been used. It is hoped that success in the use of this class of fractals will provide a basis for other classes of fractals to be seen as a viable choice for modeling, as well as for recognizing naturally occurring objects.

2 L-System Fractals

The L-system is a class of fractals which has been used as a mathematical model to describe the growth and interaction of cells within plant structures.[25] It has since been extensively used to model plants and trees at a macro level.[7,9,26] The L-system approach is similar to different types of string and tree grammars that have been proposed to describe shapes in computer vision.[11] Unlike the string and tree grammars, L-systems have been designed specifically for modeling plants and trees as well as their growth patterns at various stages of development.

An L-system specification includes rules that govern placement of branches in a tree. These rules generate objects that contain branches forming a hierarchical structure. There are many classes of L-systems and each class is capable of generating trees with varying amount of realism and accuracy.[4] In general, more complex L-system classes produce more realistic approximations of trees and plants.

2.1 *Turtle Geometry*

Since L-systems are used to model plants and trees, a mechanism to derive the graphical interpretations of the generated objects must be provided. The most straightforward method to generate an object from an L-system specification is through the use of turtle graphics.[27] An entity, called a turtle, is used to draw an object. It can perform several different functions: it can move forward a certain

Table 1 Symbols used in L-system productions.

Symbol	Action	Turtle state	Drawing
F	Move forward d units	$(x+d\cos\alpha, y+d\sin\alpha, \alpha)$	Yes
f	Move forward d units	$(x+d\cos\alpha, y+d\sin\alpha, \alpha)$	No
+	Turn left $\delta°$	$(x+y, \alpha+\delta)$	No
−	Turn right $\delta°$	$(x, y, \alpha-\delta)$	No
[	Start a branch	Push state onto a stack	No
]	End a branch	Pop state from a stack	No

distance, draw a line while moving, and turn (adjust its heading, or orientation). The key idea is that the turtle performs the operations in small steps.

The turtle has an associated state defined as a 3 tuple (x,y,α), where x is the current horizontal location in the coordinate system, y is the current vertical location in the coordinate system, and α is the turtle's current orientation, or heading. Each part of the specification of an L-system determines how the turtle's state will be manipulated in order to generate an object.

The generation of an object from an L-system is a two-step process. The first step is to generate a string of turtle commands from the specification. The string of commands is then used to send instructions to a turtle, which draws line segments to generate the object.

2.2 *L-System Grammars*

L-systems are similar to formal grammars introduced by Chomsky[28] with two important differences. First, there is no distinction between terminals and nonterminals. Symbols that are used as nonterminals in the application of productions may be treated as terminals when the image corresponding to the L-system is actually generated. Second, all nonterminal symbols in an L-system are replaced in parallel when a production is applied. That is, no choice can be made about which nonterminal in a string may be replaced.[29] There are several different types of L-systems, e.g., context-free, context-sensitive, stochastic, etc. We will briefly describe the context-free L-system, which is the simplest form of L-system grammar, and the stochastic L-system, which is used in this research.

A context-free L-system is formally defined as a 4-tuple $G=\langle S,P,d,\delta\rangle$, where S is the starting symbol (often called the axiom), P is the set of production rules, d is the unit length, and δ is the unit angle.

The production rules describe the process of object generation. Table 1 lists the symbols that may appear in a production in an L-system. A brief description of the symbol is given along with how a turtle's state is manipulated by the symbol. The turtle's state is assumed to be (x,y,α) before application of the symbol.

2.2.1 *An example*

A simple L-system grammar and the graphical representation of the objects derived from it at several iterations are shown in Fig. 2. In the first iteration, the axiom F is replaced by F[+F]F[−F] [F]. In the next iteration each F in F[+F]F[−F] [F] is replaced by F[+F]F[−F] [F] producing the string: F[+F]F[−F] [F] [+F[+F]F[−F] [F]] F[+F]F[−F] [F] [−F[+F]F[−F][F]] [F[+F]F[−F] [F]]. Future iterations proceed in a similar manner.

Derivation of a plant structure is much like the generation of a string from a traditional grammar. It begins with the axiom S. At any point in the derivation, strings from the right-hand side of a matching production replace the nonterminals in a string. The main difference in this case is that *all* nonterminals in a string are replaced in parallel.

Once a string has been generated from an L-system grammar, it can be interpreted graphically to create the corresponding tree. The string is scanned from left to right one symbol at a time. The turtle movement and drawing pattern sketch the tree.

2.2.2 *Stochastic L-systems*

Due to the deterministic nature of context-free L-systems, a given grammar can generate a fixed set of plants. Hence, they cannot represent the variations among various members of a given tree class. While maintaining the central pattern of growth, individual plants have certain randomness to their growth. Stochastic L-systems overcome this drawback of context-free L-systems by associating probabilities to various patterns of growth (production rules).

Stochastic L-systems are almost identical to the context-free L-systems. Unlike the latter, it can have multiple productions associated with the same nonterminal on the left-hand side. In the case of plants, it signifies different patterns of growth. The frequency of application of a given production rule is determined by a pre-defined probability. For example, one production may be applied 50% of the time, another production 30%, and a third one the rest (20%) of the time. Figure 3 shows a stochastic L-system grammar and four instances of objects generated using it, all at five iterations. Due to their probabilistic nature, this

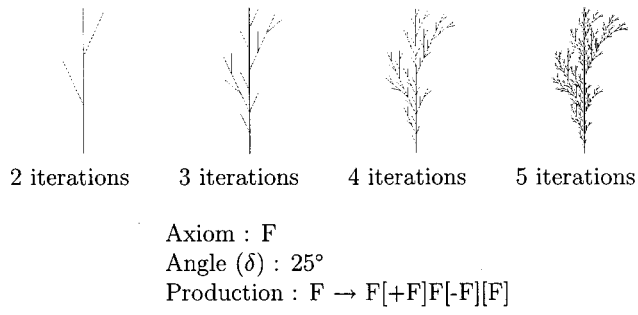

Fig. 2 An L-system grammar and some objects generated by it.

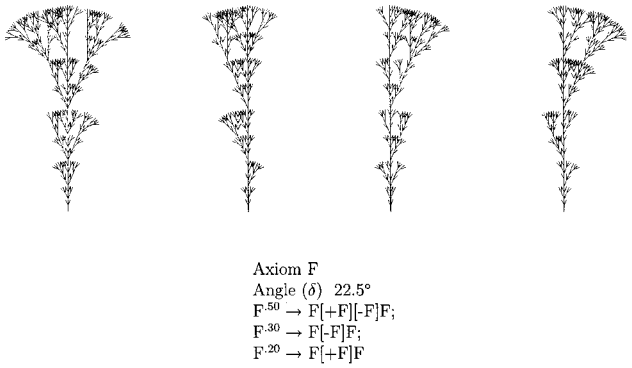

Fig. 3 Three objects generated using a stochastic L-system.

type of L-system has an advantage over context-free L-systems when modeling natural trees. Since different productions may be used at different times, it is possible to generate many different variations of the plant at the same iteration. This allows for the creation of plant classes, all sharing similar overall characteristics, but differing in details. For instance, the plants in Fig. 3 all appear similar, but are quite distinct from each other.

3 Tree Features

In a feature-based approach the selection and computation of features is critical. Two categories of features are used in this work: (a) shape features and (b) branch features and their derivatives. Many general purpose features for object recognition have been proposed in literature[11] and we use a set that is relevant for tree-like objects. Furthermore, due to the special nature of the domain, we have defined a new set of features that is used for the recognition of tree and tree-like objects. Features relating to branches, which are examples of linear features, are used extensively for classification. Trunks and branches are linear features relevant in the domain of plants and trees. It seems quite natural to describe a plant or tree in terms of its branches. These features are originally defined in Ref. 1 and are also used in Ref. 2.

3.1 *Definitions*

Several definitions related to the structure of L-system objects form the basis for the features that are used for classification and are summarized below.

- *Root:* It is the point in the object with the minimum value of the *y* coordinate.
- *Height:* It is the vertical distance between the root and the point with the highest *y* coordinate.
- *Backbone:* It is the longest vertical line starting at the root in the tree.
- *Branch:* A maximal line segment in a tree is called a branch. Branches are connected either end to end or a branch diverges from another branch in the middle.
- *Branch point:* This is the point on the backbone or a branch where one branch connects to another branch (or the backbone).
- *Trunk:* The part of the backbone starting from the root to the first branch point is called the trunk.

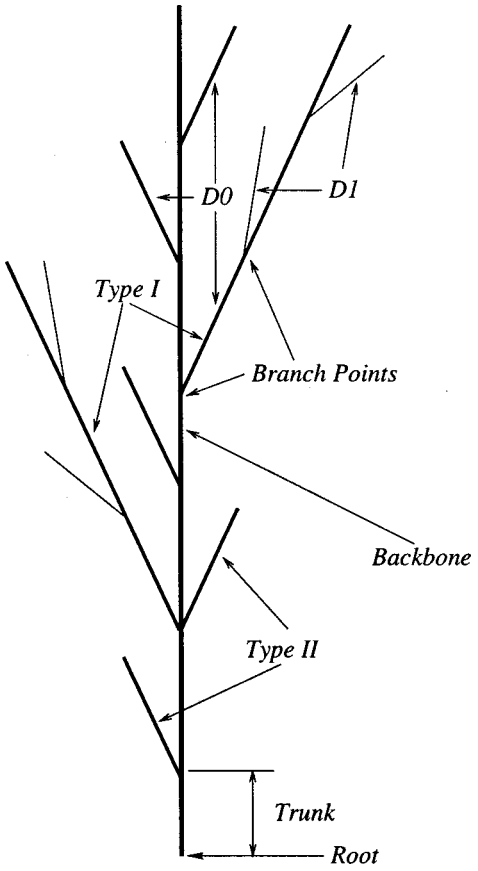

Fig. 4 An L-system object shown with its branch properties.

- *Convex hull:* It is defined to be the smallest convex set that contains all points in the tree excluding the trunk.
- *Distance:* A branch's distance is defined as the number of intermediate branch points between it and the backbone. Branches that start at the backbone are defined as distance zero branches.
- *Type:* For a given distance, branches that are the longest for are called *Type I* branches. Similarly, the second longest branches are called *Type II* branches.

Many of the above definitions are illustrated in Fig. 4. The distance zero and distance one branches are labeled as $D0$ and $D1$, respectively. This particular object does not have any branches with distance greater than one. The backbone is shown as a thick solid line. The distance zero branches are shown with a smaller thickness than the backbone and distance one branches are thinner than the distance zero branches. It should be noted that there are no type II branches at distance one in this object. In general, the branches of the same type and distance, but for different sides, do not have the same length. In this example, the type I branches that have distance zero happen to be the same length on both sides. Branches with distance greater than one and branches that are shorter than type II branches have little impact on the structure of tree and hence are not used for shape analysis. In summary, the branch features can be described in terms of its *distance* (0, or 1), *type* (I or II), and *side* (left or right). For example, a branch might be

described as being on the left side, with distance zero, and type I or on the right side, with distance one, and type II.

3.1.1 *Shape features*

The first feature that is determined is the height of the object since it plays a central role in computing many other shape features. Two features, *backbone ratio* and *trunk ratio*, relate directly to the height of an object and are good discriminators. The list of shape features used for this research is briefly described below.

- *Backbone ratio:* It is defined as the ratio of the length of the backbone to the height of the tree.
- *Trunk ratio:* It is defined as the ratio of the length of the trunk to the height of the tree.
- *Moments:* A set of moments based on the convex hull is used. The central moment of order $(i+j)$ is defined as

$$\mu_{ij}=\sum_x \sum_y (x-\overline{x})^i (y-\overline{y})^j, \tag{1}$$

  where $(\overline{x},\overline{y})$ is the centroid of the convex hull. Only the central moments of order three and less are computed.
- *Orientation:* The orientation of an object is an approximation to the location of its axis of symmetry. This feature is very useful for determining whether a shape is approximately symmetric or not. Symmetric or nearly symmetric shapes have orientations very close to zero. Since only plants and trees are being dealt with, symmetry about a vertical axis (backbone) is of interest. The orientation, θ, is then approximated by

$$\theta=\frac{1}{2}\tan^{-1}\left(\frac{2\mu_{11}}{\mu_{20}-\mu_{02}}\right). \tag{2}$$

3.1.2 *Branch features*

Branch features are very useful for distinguishing between different types of plants. Spatial positions are often unique to a class of plants and hence can be used to discriminate between different classes. In addition, different types of plants have branches that may grow longer than other types of plants. This again is a very good discriminator. The list of branch features that are used is given below:

- *Lengths:* Lengths of branches of distance zero and one are used. Also, only Type I and II branches are used.
- *Starting points:* Starting points of the branches with respect to the backbone are computed.
- *Separations:* The distances between the starting points of adjacent branches are also useful features.

4 Decision Trees for Plant Recognition

A variety of mechanisms are available to classify objects by matching features, including template matching, clustering, and minimum distance classifiers.[11] The approach taken here is to use a decision tree classifier. By traversing the tree from its root, the possible classes to which an unknown object can belong is narrowed down. This is done until the class the object belongs to is uniquely determined.

Formally, a binary decision tree is a binary tree, where each leaf node represents a specific class. The nonleaf nodes represent decision points. Each decision is in the form of a Boolean expression. In our case, a decision relates to the value of a particular feature. Each nonleaf node has two branches, one labeled true/yes and the other labeled false/no. If the decision evaluates to true the ''true/yes'' branch is traversed and vice versa.

Figure 5 illustrates a simple decision tree classifier to recognize five different L-system classes. The features used here are for illustration purposes only and do not correspond to actual features used in the full implementation of the classifier. It should be noted that some classes require more decision points than others and that the decision tree is not unique.

4.1 *Determination of the Decision Tree*

Before an optimal decision tree for classification can be derived, useful features for recognition must be determined and their values computed for each model. Only then can a decision tree be obtained for classification. These steps are accomplished in a *training* procedure.

In our previous research,[1] the decision tree was created manually with some aid from the KBVision Constraint System, a commercial computer vision package. In this system, histograms of each feature can be observed, and the best splitting value for the decision tree can be found. Suppose it is observed that half the plant classes have values less than 4.2 for feature f_i, while the other half clearly have greater values. This feature can then be used to form a node in a decision tree, with the test "$f_i<4.2$" used to split the training cases into two groups. This process is repeated until all training cases have been classified.

The manual construction of a decision tree classifier poses several problems. First, as more classes are used in the decision tree, it becomes very tedious to determine good splitting values for the decision tree. Previous research used several dozen classes for classification. Developing a decision tree with many more classes, say, several hundred, is a much more difficult task. Finally, manually derivation of decision trees is prone to human error.

Objects generated by stochastic L-systems present an even greater problem. Plants generated from a stochastic L-system do not have significant differences. As a result, a specific feature will not have the same value for all instances of objects. It makes this process even more difficult.

To address these problems, a learning system is used for this research. C4.5 is a series of programs that construct classification models by discovering and analyzing patterns found in records of information.[30] Data from a set of discrete classes are given to C4.5 as a training set. This training set is analyzed and a decision tree is derived by the system in a systematic manner based on information-theoretic approach to minimize the chances of misclassification.

5 Implementation and Results

The approach proposed here has been implemented and extensively tested. Specifically, stochastic L-system grammars were used to generate 150 different types of plants. The training set consisted of 50 instances of each plant, for

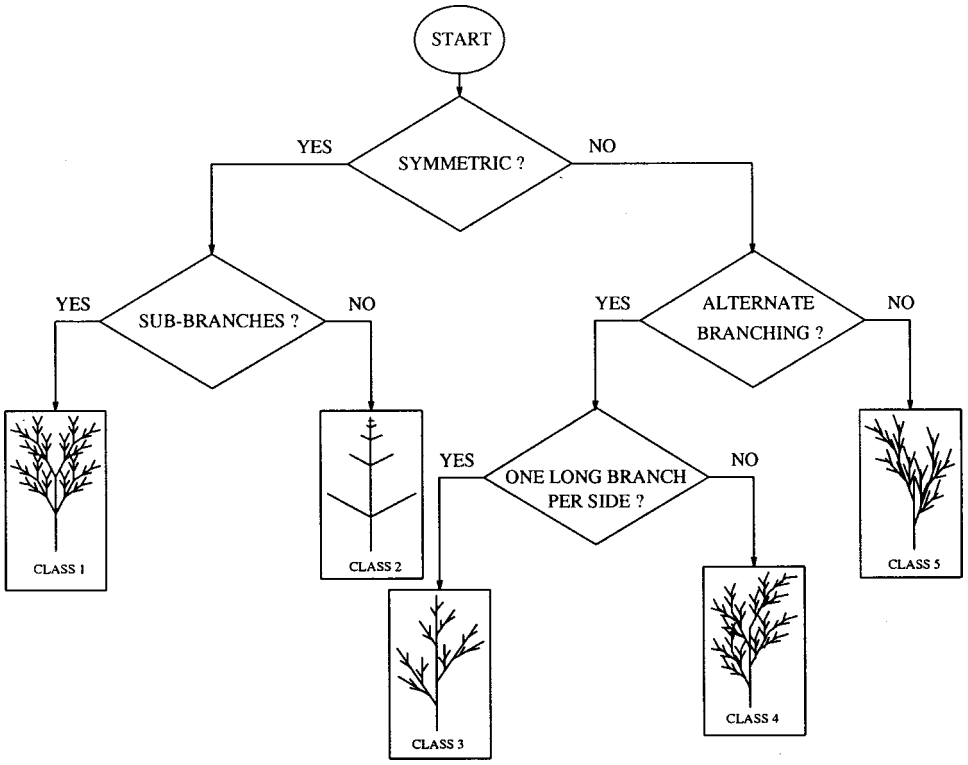

Fig. 5 A simple decision tree classifier.

a total of 7500 plants. The test set contained twice as many plant instances for each class, for a total of 15 000 plants. The test set was kept entirely disjoint from the training set. It was decided that data sets containing three-to-five-iteration plants would be used. The features computed from the plants are given to the learning system that generates the best decision tree.

5.1 *Computation of Feature Values*

Since many of the features are based on branches, the extraction of straight-line segments is a crucial step. It is a complex operation involving several vision tasks. The recovery of all branches is not trivial, particularly for smaller branches. Because of this, only the two longest types of branches are used. Since smaller branches are ignored, recovery of the longer branches and the backbone is relatively robust and insensitive to noise. The steps in feature computation are given below and are summarized in Fig. 6.

First, magnitude and orientation of edge points are obtained using a standard edge detector. In KBVision system each edge point is treated as a token, i.e., a unit line segment whose angle is given by the edge orientation. A histogram is then obtained using the angles of these tokens. Large peaks in the histogram correspond to the main branches in the tree. We use only the distance zero and distance one branches. Hence, only the tokens with orientation corresponding to the five major angles (one for the trunk and two each: left and right, for the two types of branches) are used to grow branches from the edges. The convex hull for the tree is then computed from the distance zero and distance one branches. Finally, the shape features and branch features are computed. The backbone and the trunk are computed directly (see Fig. 4). The lengths of

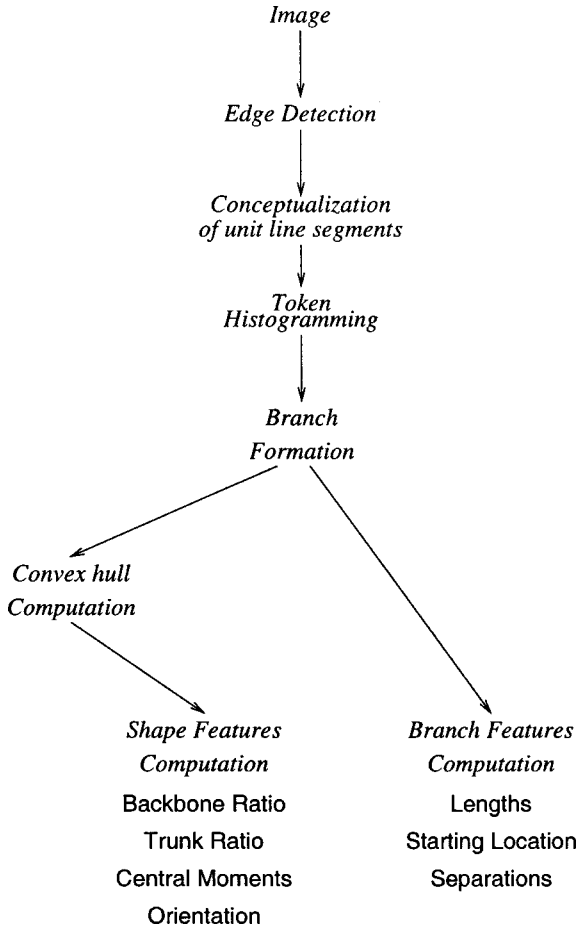

Fig. 6 Summary of preprocessing steps for recognition.

```
plant1, plant2, plant3, plant4.

Aspect Ratio:     continuous.
Backbone Ratio:   continuous.
Trunk Ratio:      continuous.
Orientation:      continuous.
```

Fig. 7 A sample C4.5 name file.

```
Decision Tree:

Backbone Ratio <= 0.13 : plant3 (3.0)
Backbone Ratio > 0.13 :
|   Backbone Ratio > 0.688 : plant1 (3.0)
|   Backbone Ratio <= 0.688 :
|   |   Trunk Ratio <= 0.13 : plant2 (3.0)
|   |   Trunk Ratio > 0.13 : plant4 (3.0)
```

Fig. 9 A C4.5 decision tree.

main branches and the separation between them are computed by following the backbone and distance zero branches. Once the branches are recovered, the rest of the features are computed.

5.2 *Implementation*

The system to recognize the plants and trees using the L-system model was implemented on a SUN Sparcstation using the KBVision system and the C4.5 learning system. The system has been tested comprehensively.

After preprocessing steps are performed, the learning system, C4.5, was employed to determine a decision tree based on feature values computed from a training set of images. The C4.5 system classifies unknown objects by using several different types of information. A ''name'' file contains a list of classes and a list of attributes used for classification. Figure 7 shows an example name file. The attributes shown are all continuous real numbers, but attributes with discrete values are also allowed. In addition, a ''data'' file containing the actual feature values for each instance in the training set is also provided to C4.5. A sample data file is given in Fig. 8.

An output file, showing the structure of the decision tree, is produced by C4.5. Figure 9 shows a sample output. A series of if-then-else statements could be created almost directly from the output file. Figure 10 shows the corresponding decision tree pictorially.

5.3 *Results*

The images generated by stochastic L-system grammars were tested using the decision tree produced by the C4.5 learning system. One hundred instances each of 150 classes of plants are used during testing. This set is different from the one used for training (building the decision tree). Figure 11 shows several instances of some classes of plants used by the system. Overall, our system correctly classified the test cases with approximately 93% accuracy using shape features only. When both shape and branch information are used the accuracy of the system increases only marginally to 94%. Table 2 shows the summary of the results.

```
3.471, 1.000, 0.250,  -6.343, plant1
2.145, 0.416, 0.104, -20.926, plant2
1.475, 0.130, 0.130,  20.233, plant3
3.371, 0.688, 0.344,  -0.102, plant4
3.105, 1.000, 0.125,  -8.468, plant1
2.034, 0.381, 0.048, -22.460, plant2
0.703, 0.104, 0.104,   0.334, plant3
2.145, 0.587, 0.294,  -1.547, plant4
2.878, 1.000, 0.062,   6.646, plant1
1.405, 0.397, 0.025,  27.697, plant2
0.915, 0.071, 0.071, -19.293, plant3
1.326, 0.380, 0.190,  -0.473, plant4
```

Fig. 8 A sample C4.5 data file.

Many interesting conclusions can be made from our results. First, plants generated from stochastic L-systems can be accurately classified, even though wide variations exist between individual members of a class. Furthermore, a system based on shape features only proved to be almost as good as a system based on both shape and branch features. It was expected that the branch statistics alone would not be sufficient due to the randomness of the branch lengths, locations, and frequencies. However, it was not expected that the shape features alone could be used to accurately classify the plants.

Classifying objects generated using stochastic L-system grammars accurately with only shape features implies that the recognition of natural plants and trees with only shape information may be feasible. Since the branches in natural plants are difficult to determine, shape features may prove to be adequate for recognition. However, some shape characteristics, such as the backbone and trunk lengths, moments of inertia, and orientation, require finding major branches first.

6 Summary and Future Research

A novel approach for recognition of plants and trees is presented here. The centerpiece of the methodology is the L-system fractals, which are used for modeling and recognizing plants. We have extended the previous use of context-free L-systems to stochastic L-systems that accurately model the variation among instances in a class of plants. We have defined a set of features that are effective in classification of plants. A decision-tree approach is used for the final classification process. The system automatically determines the decision tree using a learning system. Results show that the approach is very efficient and effective. In addition, the results here show that it may be possible to classify natural plants and trees using their shape features only. This could prove advantageous to future research, because branches are difficult to extract from natural plant images.

There are several ways in which this research may be extended. The obvious extension is to take the next step in the recognition of natural plants obtained from real scenes. We believe the soundness of the approach and the potential of success have been identified in this research. The second extension is to use even more accurate models to represent plants and trees. Initially, only context-free L-system models were considered. We have made the extension to sto-

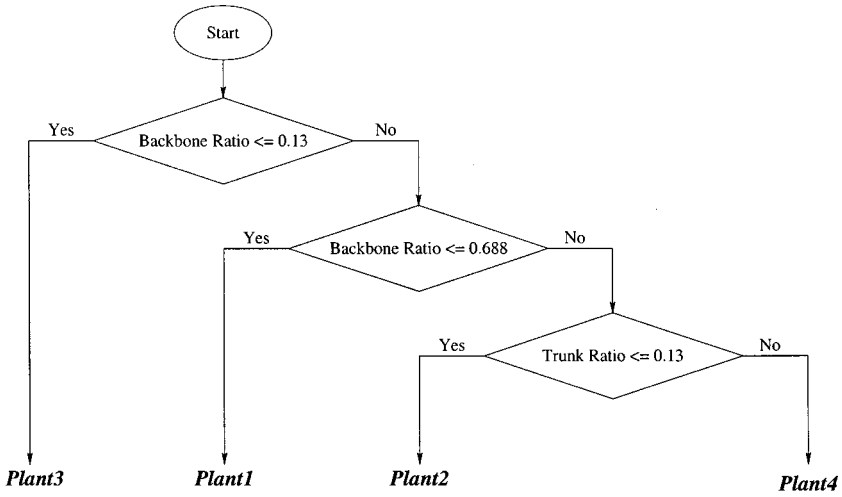

Fig. 10 The decision tree corresponding to the example in Fig. 9.

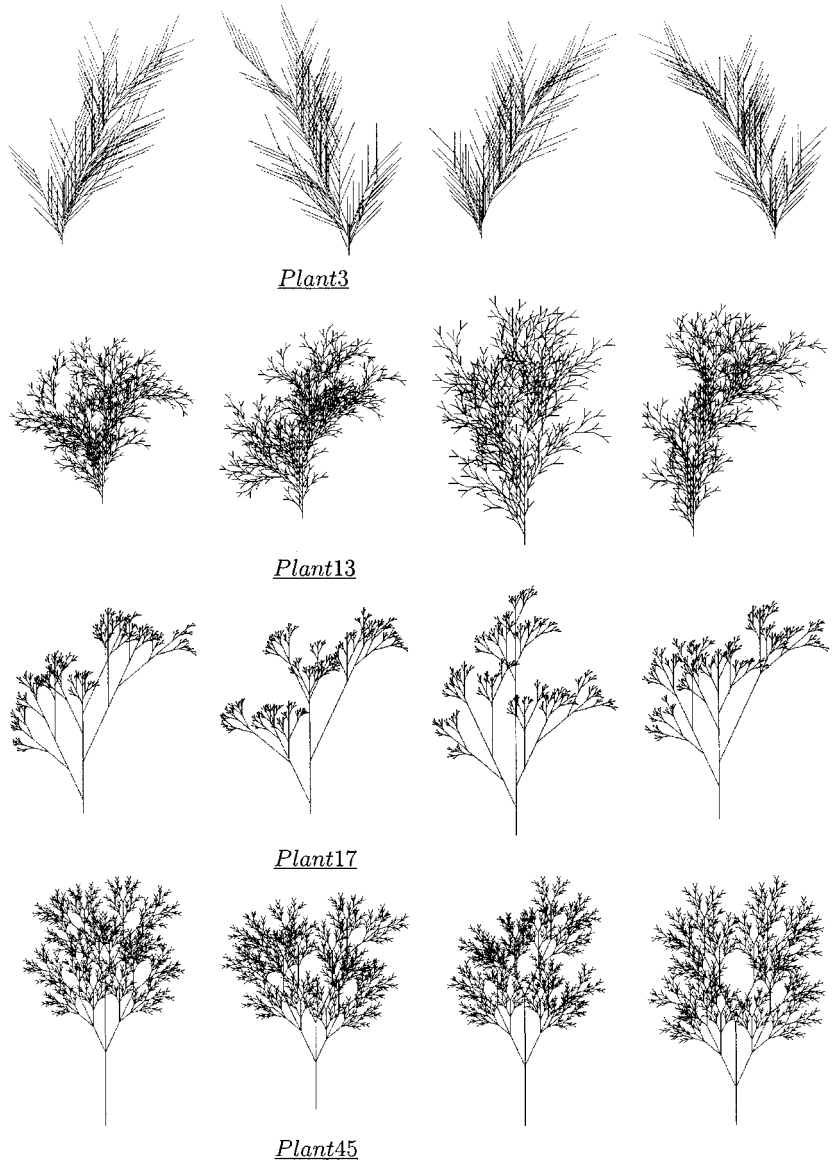

Fig. 11 Examples of plants used for testing.

Table 2 Accuracy of C4.5 decision trees.

Iterations	Feature type	Training cases	Test cases	Error rate
3, 4, and 5	Shape	7500	15 000	7.3%
3, 4, and 5	Branch	7500	15 000	20.7%
3, 4, and 5	All	7500	15 000	6.4%

chastic L-system models. The model may be improved further by considering three-dimensional L-system models.

L-systems are certainly not the only type of fractals that may be used for recognition. Different types of fractals model different types of natural objects. For instance, one particular type of fractal known as *iterated function system* is a good model for clouds and other irregularly shaped objects.[10] Such objects may be recognized by making suitable use of the appropriate type of fractals.

Acknowledgment

This work is supported in part by NSF Grant Nos. CDA-9022445 and USE-9152764.

References

1. D. J. Holliday, ''Object recognition using fractals,'' Master's thesis, University of Nebraska-Lincoln (1993).
2. D. J. Holliday and A. Samal, ''Object recognition using L-system fractals,'' *Pattern Recogn. Lett.* **16**, 33–42 (1995).
3. J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*, 2nd ed., Addison–Wesley, Reading, MA (1990).
4. M. Aono and T. L. Kunii, ''Botanical tree image generation,'' *IEEE Comput. Graphics Appl.* **4**(5), 10–34 (1984).
5. B. B. Mandelbrot, *The Fractal Geometry of Nature*, W. H. Freeman, San Francisco (1982).
6. H. Peitgen and D. Saupe, Eds., *The Science of Fractal Images*, Springer, New York (1988).
7. P. Prusinkiewicz, ''Graphical applications of L-systems,'' *Proc. Graphics Interface 1986- Vision Interface*, pp. 247–253 (1986).
8. P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*, Springer, New York (1990).
9. A. R. Smith, ''Plants, fractals, and formal languages,'' *Comput. Graphics* **18**(3), 1–10 (1984).
10. M. Barnsley, *Fractals Everywhere*, Academic, New York (1988).
11. D. H. Ballard and M. Brown, *Computer Vision*, Prentice-Hall, New York (1982).
12. R. J. Schalkoff, *Pattern Recognition: Statistical, Structural, and Neural Approaches*, Wiley, New York (1992).
13. S. S. Chen, J. M. Keller, and R. M. Crownover, ''Shape from fractal geometry,'' *Artif. Intel.* **43**, 199–218 (1990).
14. S. S. Chen, J. M. Keller, and R. M. Crownover, ''On the calculation of fractal features from images,'' *IEEE Trans. Pattern Anal. Mach. Intell.* **15**, 1087–1090 (1993).
15. J. G. Jones, R. W. Thomas, P. G. Earwicker, and S. Addison, ''Multi-resolution statistical analysis of computer-generated fractal imagery,'' *CVGIP: Graph. Models Image Process.* **53**, 349–363 (1991).
16. J. M. Keller, R. M. Crownover, and R. Y. Chen, ''Characteristics of natural scenes related to the fractal dimension,'' *IEEE Trans. Pattern Anal. Mach. Intell.* **9**, 621–627 (1987).
17. A. P. Pentland, ''Fractal-based description of natural scenes,'' *IEEE Trans. Pattern Anal. Mach. Intell.* **6**, 661–674 (1984).
18. A. P. Pentland, ''Perceptual organization and the representation of natural form'' *Artif. Intel.* **28**, 293–331 (1986).
19. H. M. Berenyi, V. F. Leavers, and R. E. Burge, ''Automatic detection of targets against cluttered backgrounds using a fractal-oriented statistical analysis and radon transform,'' *Pattern Recogn. Lett.* **13**, 869–872 (1992).
20. N. Yokoya, K. Yamamoto, and N. Funakubo, ''Fractal-based analysis of 3D natural surface shapes and their application to terrain modeling,'' *Comput. Vis. Graph. Image Process.* **46**, 284–302 (1989).
21. N. Dodd, ''Multispectral texture synthesis using fractal concepts,'' *IEEE Trans. Pattern Anal. Mach. Intell.* **9**, 703–707 (1987).
22. J. Garding, ''Properties of fractal intensity surfaces,'' *Pattern Recogn. Lett.* **8**, 319–324 (1988).
23. J. M. Keller, S. Chen, and R. M. Crownover, ''Texture description and segmentation through fractal geometry,'' *Comput. Vis. Graph. Image Process.* **45**, 150–166 (1989).
24. J. Levy Vehel, ''Fractal probability functions. An application to image analysis,'' in *Proceedings of the IEEE Internation Conference on Computer Vision and Pattern Recog.* pp. 378–383 (1990).
25. A. Lindenmayer, ''Mathematical models for cellular interaction in development, parts I and II,'' *J. Theor. Biol.* **18**, 280–315 (1968).
26. P. Prusinkiewicz, A. Lindemayer, and J. Hanan, ''Developmental models of herbaceous plants for computer imagery purposes,'' *Comput. Graphics* **22**(4), (1988).
27. D. D. Thornburg, *Beyond Turtle Graphics: Further Explorations of Logo*, Addison–Wesley, Reading, MA (1986).
28. N. Chomsky, ''Three models for the description of language,'' *IRE Trans. Inf. Theory* **2**(3), 113–124 (1956).
29. J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison–Wesley, Reading, MA (1979).
30. J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufman, Los Altos, CA (1993).

Ashok Samal received his bachelor of technology degree in computer science from the Indian Institute of Technology at Kanpur, India, in 1983. He received his PhD in computer science from the University of Utah in 1988. Since the fall of 1988 he has been with the Department of Computer Science and Engineering at the University of Nebraska-Lincoln where he is currently an associate professor. His research interests are image understanding, document analysis, and geospatial information analysis, and distributed computation.

Brian Peterson received his BS from NW Missouri State University in 1996. This research was conducted during a summer program for undergraduate research in digital image processing and computer vision as a part of research experience for undergraduates, sponsored by NSF.

David J. Holliday received his BS and MS in computer science from the University of Nebraska-Lincoln in 1991 and 1993, respectively.