Automated Tree Generation Using Grammar & Particle System

Aryamaan Jain
IIIT
Hyderabad, India
aryamaan.jain@research.iiit.ac.in

Jyoti Sunkara
IIIT
Hyderabad, India
jyoti.sunkara@students.iiit.ac.in

Ishaan Shah
IIIT
Hyderabad, India
ishaan.shah@research.iiit.ac.in

Avinash Sharma
IIIT
Hyderabad, India
asharma@iiit.ac.in

K S Rajan
IIIT
Hyderabad, India
rajan@iiit.ac.in

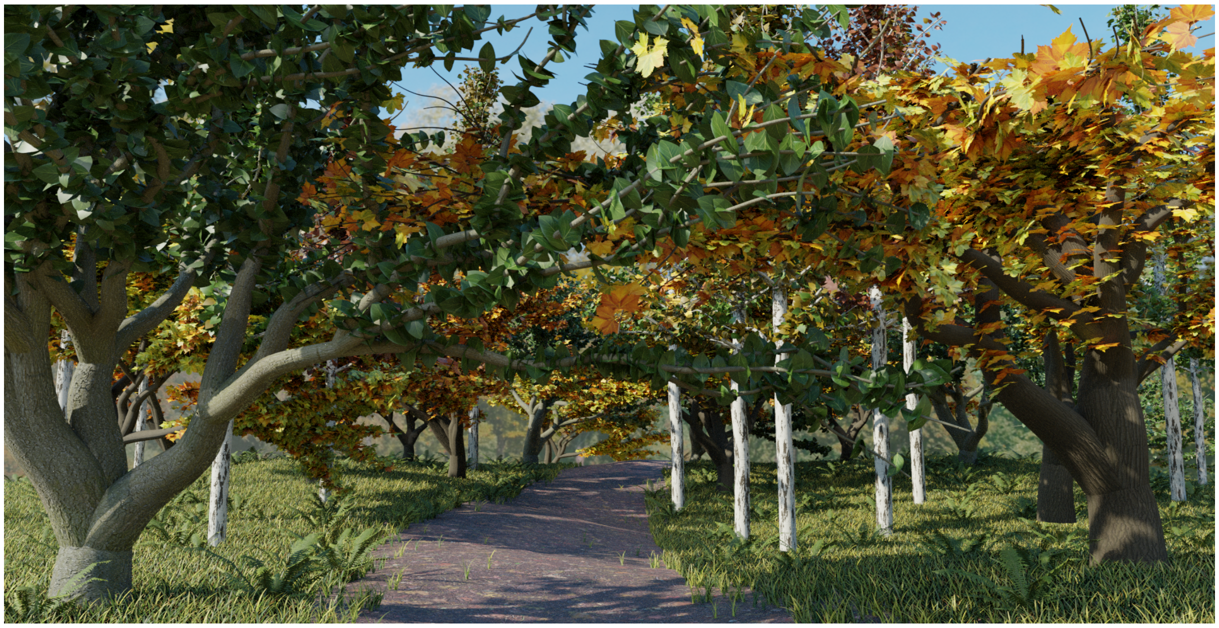


Figure 1: A forest rendered using the proposed model.

ABSTRACT

Trees are an integral part of many outdoor scenes and are rendered in a wide variety of computer applications like computer games, movies, simulations, architectural models, AR and VR. This has led to increasing demand for realistic, intuitive, lightweight and easy to produce computer-generated trees. The current approaches at 3D tree generation using a library of trees lack variations in structure and are repetitive. This paper presents an extended grammar-based automated solution for 3D tree generation that can model a wide range of species, both Western and Indian. For the foliage, we adopt a particle system approach that models the leaf, its size, orientation and changes. The proposed solution additionally allows control for individual trees, thus modelling the tree growth variations, changes in foliage across seasons, and leaf structure. This enables the generation of virtual forests with different tree compositions. In addition, a Blender add-on has been developed for use and will be released.

CCS CONCEPTS

• **Computing methodologies** → **Shape modeling**; • **Theory of computation** → *Grammars and context-free languages.*

KEYWORDS

Grammar based tree generation, tree modelling, shape grammar

ACM Reference Format:
Aryamaan Jain, Jyoti Sunkara, Ishaan Shah, Avinash Sharma, and K S Rajan. 2021. Automated Tree Generation Using Grammar & Particle System. In *Proceedings of 12th Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP'21)*, Chetan Arora, Parag Chaudhuri, and Subhransu Maji (Eds.). ACM, New York, NY, USA, Article 28, 9 pages. https://doi.org/10.1145/3490035.3490285

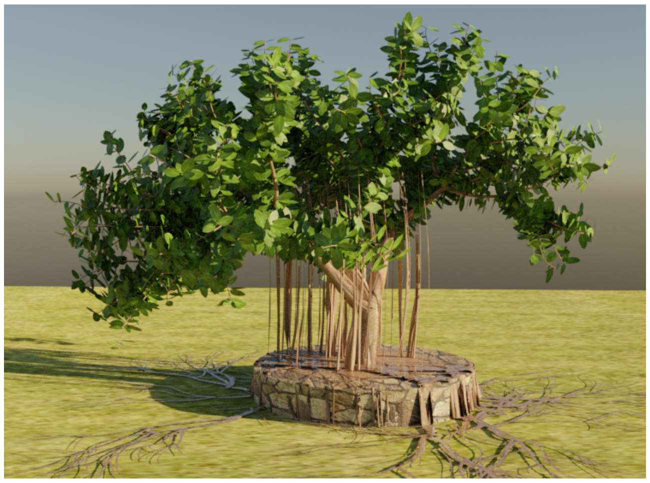


Figure 2: Banyan tree generated by our method.

1 INTRODUCTION

Virtual world generation is an important goal for computer graphics applications aiming at realistic experience for gaming or virtual reality audience. Modelling realistic *Flora* (plant life) is an important aspect of generating outdoor virtual scenes that are abundant in graphics applications. Traditionally, a set of 3D tree models designed by expert artists are repeatedly used to populate a virtual forest scene. This approach suffers from a lack of variation in tree structures, as observed in real world as the same tree models are being replicated. Hence, this approach fails to scale while generating virtual scenes with a large number of trees, making the view monotonous and repetitive. Thus, an automated approach to generating multiple trees of varying structures and appearances become important. This is a challenging task that involves modelling a large number of plant species at multiple growth stages and in different environmental conditions like lighting and wind flows.

Interestingly, trees exist as self-similar structures or fractals, where each stem is very similar to its parent stem with subtle changes in shape and width. This self-similarity can be modelled using recursive grammar and the variation in structure is further modelled using geometric specifications. Thus, each stem can be modelled as some geometric transformation relative to its parent. Examples of such geometric transformations are scaling where child is smaller than its parent, rotation where child is present at some angle relative to its parent, etc. These observations were used by [7] in the popular L-system model to generate trees. [16] later extended earlier work to make the interpretation of grammar easier. However, they used global geometry parameters, which made it difficult to model trees like Pine that has a lot of local geometric variations.

In this paper, we propose to extend the L-system grammar-based automation solution for 3D tree generation that overcomes the limitation of existing methods and generalize to a large variety of tree species with varying topologies like Banyan, Mango, Pine or Palm trees. The proposed method can also be used to model related vegetation such as flowers, shrubs or grasses like bamboo. Our method uses stochastic rules and can therefore produce structural variations from the same set of rules. More importantly, the proposed method overcomes the limitations of previous methods [16] by proposing to extend the geometric variations locally instead of using global parameters. For modelling foliage, we have adopted the particle system approach. This allows to model foliage unique to each tree in terms of leaf type and variations based on season, height and orientation. Subsequently, we perform texture mapping to give a realistic appearance to both stem and leaves.

Figure 1 shows the forest scene generated by our method where we can observe the variations within multiple instances of the same species and differences between similar species consisting Pine, Aspen and Fir. It is important to note that the proposed method is intuitive and easy to understand and thus a novice user can also easily understand and edit the grammar and geometric parameters to produce trees of their specification. We also intend to release a blender plugin implementation of our method. Additionally, the majority of the existing datasets and automation based tree generation methods have focused on western tree species. We aimed at developing a library of trees focused on Indian subcontinent (e.g., Banyan Tree shown in Figure 2).

2 LITERATURE SURVEY

There exist many classes of methods to model trees. Some of the classes are parametric methods, image-based methods, modelling through 3D reconstruction, learning-based methods, grammar-based methods, etc.

Parametric methods are often much harder to understand and work with. Weber and Penn [19] introduced a system to model trees using a parametric approach. Their method is implemented in blender as a famous plugin *Sapling Tree Gen.* [19] also proposed additional use cases in their work such as pruning of trees, wind sway, vertical attraction and tropism in trees, etc.

Image-based modelling techniques [4, 9, 18] are used to model the most realistic trees but cannot fit many use cases because of lack of scalability. Some methods such as [13] lie at an intersection of image based approaches and grammar based approaches to model trees.

[18] proposed a model using an image-based approach. Their method used structure from motion and other post-processing techniques to model trees. Other methods exist such as single image approaches [4] and multiple image approaches [12], [9]. Methods of 3D reconstruction such as [20] use an exemplar database consisting of real trees reconstructed from scanned 3D data. This approach similarly lacks scalability due to the difficulty in the acquisition of data. Additionally, in use cases like modelling forests, these methods will not work because a single set of 20 to 30 images produces a single tree and these models lack stochasticity to produce variations in those trees. Producing a forest may require a big database of images and computational needs can be too expensive. More importantly, the stem structure information might be incomplete due to occlusion of leaves and hence it is not easy to animate such reconstructed trees.

Learning-based methods have been proposed to model trees. One such strategy is inverse procedural modelling [21], [11], [15], [14] in which the parameters of grammar-based models or parametric models are estimated given the data such as tree models. Other learning-based approaches such as [2] take images as input but mainly focus on 2D trees, though they have a limited extension to 3D. Further work on Interactive procedural modelling like [8] provided for manual interaction for the 3D artist reducing their design process. Whereas [3], [5] optimise procedural modelling approaches at scale to efficiently generate forest for virtual world creation at reduced time and memory requirements. On the other hand, [6] presented a visually optimised method to make tree generation and rendering more suitable for applications in VR/AR.

Grammar-based approaches, such as the proposed work is based on the interpretation of the grammar provided by the user. Different methods of interpretation vary the expressibility and the number of species that can be modelled. L-system and its extensions [7] are popular grammar-based method to model trees. In [7], the grammar is first expanded and then interpreted to give results. Another grammar-based approach was proposed by [16]. Their model is easy for a user to understand thereby giving the user bigger flexibility to model trees but is not able to model a wide variety of trees. The limitation on the variety of trees is because all of their geometric parameters are set globally, i.e., it is the same for all types of stems. This makes it harder to model local variations present in trees. The proposed model overcomes this limitation by allowing the user to specify local variations in the tree structure.

3 TREE GENERATION

The tree generation primarily requires modelling the structure of the tree which posses self-similar structures, albeit with large variation in size/shape within same and across various species. Additionally, the foliage is also a key component in tree modelling and poses many unique challenges like variations in size, colour, orientation and position of leaves and they also appear in large numbers as compare to stems. In this section, we discuss in detail our approach for these two key tasks.

3.1 Tree Structure Generation

Regarding the first task, we propose a grammar-based approach to model tree stem structures. This approach can be viewed as an *automata*, which consists of one start state $\mathcal{S}$, a set of stop states $\mathcal{T}$ and multiple transient states. Each state transition draws a stem as a spline and is controlled by a grammar $\mathcal{G}$ and the geometry $\mathcal{H}$ associated with the grammar. Additionally, each of these transitions from one state to another state of the automata is stochastic in nature and thus can lead to one of the many next states defined as part of the grammar. This enables modelling the structural variations associated with specific tree species. Another important aspect of modelling structural variations is associated with geometrical rules defining the geometry of the stem to be drawn for that transition. Each geometric rule also has random variations within it to further increase the stochasticity within the species and also across species. This is by providing parameters such as length or angle as uniform random variables rather than a constant. Figure 3 shows how variations in grammar and geometry specification produce variations in trees. More specifically, trees in Figure 3a and Figure 3b share the same grammar and geometry, the variation produced are purely due to stochasticity in grammar and geometry. On the other hand, trees shown in Figure 3a and Figure 3c share the same grammar but have different geometries. This is shown in their similar underlying structure. Finally, trees rendered in Figure 3a and Figure 3d have different grammar as well as different geometry. In the current approach, the stems that are generated are assumed to be devoid of self-intersecting properties and is not explicitly modelled.

3.1.1 Grammar ($\mathcal{G}$).

Grammar describes the underlying structure of the tree. It holds structural information such as the number of stems a stem splits into, which can be stochastic. A grammar $\mathcal{G}$ is defined as a set of state transition rules, i.e., $\mathcal{G} = \{g_1, g_2, g_3, \ldots\}$ with each transition rule g_i defining the probabilities (or likelihood) of choosing the next (output) set of states starting from current (input) state. Therefore, each element of $\mathcal{G}$ defines a stochastic input-output rule system where the input is a single state α and output can be a set of states e.g., $\{\beta, \gamma, \delta, \ldots\}$. This means that a single input state can transition into multiple output states. This is necessary for modelling trees and can be thought of as a single stem splitting into a set of multiple child stems in a stochastic manner.

Interestingly, during such transitions from input to output states, each transition $\alpha \longrightarrow \beta$ and $\alpha \longrightarrow \gamma$ has a different set of geometries associated with them. Nevertheless, even if input and output states are the same, the stems will still be different due to stochasticity in the geometrical parameters as explained below. Some examples of tree specific grammars are provided in table 1.

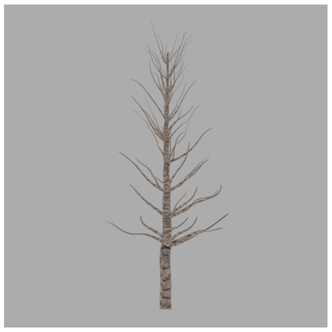
Another important parameter associated with the expansion of grammar is the maximum branching depth $\mathcal{D}$. It can also be thought of as the maximum recursion depth. More specifically, it controls how many levels of branching the tree can have. Thus, it is used to control the size of the tree and hence consequently limits the resource usage of the computer by not letting the code run indefinitely in case the stop state is not encountered.

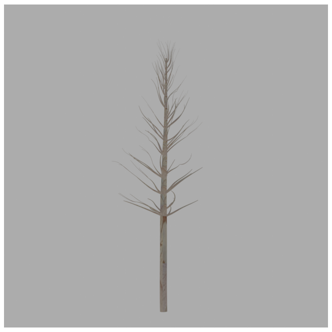
3.1.2 Geometry ($\mathcal{H}$).

Each tree has a set of geometric rules where each rule $h_i \in \mathcal{H}$ is associated with a every single state transition in the grammar $\mathcal{G}$. Each of these geometric rules has intuitive and simple parameters listed here:

- **Base Angle** is specified as 3 uniform random variables representing the 3 Euler angles. The range of these variables is specified by the user. The angle gives the relative change of orientation of the child stem with respect to its parent which is further modified to give the true angle.
- **Reduce angle** is used to reduce the range of angles a stem can make from its parent. It is observed that the stems that are higher up in a tree tend to make smaller angles from their parent stem than those lower. This may be attributed to external factors such as gravity which over time increases the angle between parent and child stem. It is specified as a vector containing the reduction for the 3 Euler angles. Reduce angle r_a modifies the base angle b_a to give the true angle t_a as a function of the current branching level c, given by relation $t_a = b_a \times r_a^c$.
- **Base length** of the stem is specified as a uniform random variable. This gives a relative measure of the length of the

(a) Pine


(b) Pine


(c) Aspen

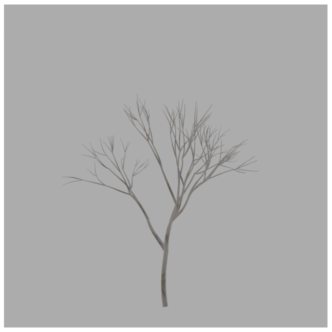

(d) Guava

Figure 3: Variations in grammar and geometry produce variations in trees structure.

Table 1: Condensed grammar for selected trees. Grammar is given as $input \rightarrow output$, with s always representing the start state and t end state where applicable. For the case of Guava, we can see that the grammar has two elements. For the second element, a can transition into aaa or aa, which can go on recursively. Each transition $a \rightarrow a$ has a geometry associated with it.

Tree	Grammar
Guava	$s \rightarrow (a); a \rightarrow (aaa/aa)$
Fir	$s \rightarrow (a); a \rightarrow (ab_1b_1b_2b_2/ab_1b_1b_2/ab_1b_1b_1); b_1 \rightarrow (t); b_2 \rightarrow (c_1c_2); c_1 \rightarrow (t); c_2 \rightarrow (t)$
Palm	$s \rightarrow (a); a \rightarrow (bb \dots b); b \rightarrow (t)$
Bamboo	$s \rightarrow (aa \dots a); a \rightarrow (a)$

stem associated with the state transition. The true length of the stem is dependent on the base length as given below.

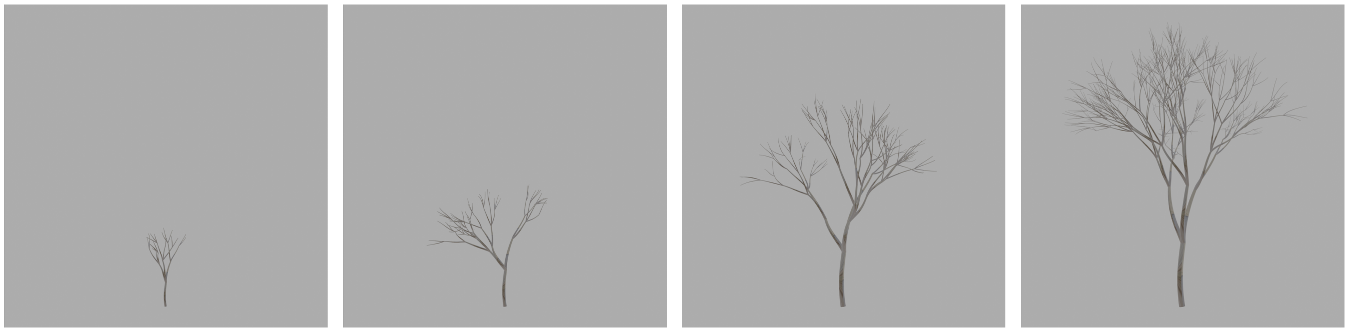
- **Reduce length** is specified as a scalar and it modifies the base length. True length of the stem t_l, is a function of the base length b_l, reduce length r_l and the current branching level c given by the relation $t_l = b_l \times r_l^c$. This gives an approximation of stem length as stem length reduces at higher branching levels of the tree.
- **Reduce width begin** (r_{wb}) and **reduce width end** (r_{we}) are specified as scalars. They control the widths of the endpoints of the stem and interpolate the intermediate control point widths based on the endpoint widths. Let w be the width of the adjoining endpoint of the parent stem, then the widths of the endpoints of the child stem will be $w \times r_{wb}$ and $w \times r_{we}$. Additional global parameter $\mathcal{W}$ represents the initial width of the base stem i.e., the trunk of the tree. It is a uniform random variable between ranges provided by the user. This rule assumes that the width of the child stem will always be less than or equal to its parent stem. This is true in most situations and should be a good approximation for modelling trees.
- **Curve** is a uniform random variable that specifies the curviness of the stem. Noise functions like Perlin noise can be used. The noise vector displaces the control points of the spline by an amount proportional to the curve parameter. This helps make the tree more realistic by adding noise at the structural level. Models using cylinders instead of splines for drawing stems cannot use this parameter.

3.2 Populating Leaves

The key challenge associated with modelling leaves is dealing with a large system of geometry. To tackle this issue, particle systems [10] have been a popular approach in the past. Particles are pieces of geometry emitted from mesh objects, typically in the thousands. Each particle can be a point of light or a mesh, and be joined or dynamic. They may react to many different influences and forces, and have the notion of a lifespan. Particle systems once created are a placeholder for the particle geometry to be placed.

We have adopted the particle system approach to model foliage unique to each tree in terms of leaf type and variations seen based on season, height and orientation. In this approach, we use hair type particles [1] which are a subset of regular particles. Hair systems form curves that can represent hair and fur in addition to leaves. The simulation and modelling of human hair is a process whose computational complexity is very large, this is due to the large number of factors that must be calculated to give a realistic appearance. Generally, the method used in the film industry to simulate hair is based on particle handling graphics. In this paper, a simpler approximation of modelling hair type particles provided in Blender [17] is used. This approach towards modelling hair type particles is a common one in the field of computer graphics.

We create a ParticleSystem in the Blender, an object parameterized to control the appearance and behaviour of individual Particle objects over time. Particles, which are born from a ParticleEmitter, have a position and type. The primary approach we take to affect the visual output is to include maximum and minimum attributes. For every variable with a maximum and minimum input, the actual value for that variable on the particle will be randomly assigned to



(a) timestep 1 **(b) timestep 2** **(c) timestep 3** **(d) timestep 4**

Figure 4: The growth stages of trees. This is achieved by controlling the maximum branching depth of the tree.

be between the maximum and minimum input and stay statically at that value for the entire life of the particle.

When we transition from an input state to output states, then each transition has a different set of geometries associated with them. With each execution of expanding the grammar, leaves are generated as particle system for the spline. The particles system takes as input the appropriate leaf texture and distributes the leaves onto the spline that is in consideration.

Leaves are represented as planes. Leaf texture is specified as an image of the leaf. Different species have different vernacular patterns that can be specified most accurately using images of real leaves. Colour variations are observed that the stems that are higher up in a tree tend to dry out faster than the ones below. Hence the provision for multiple textures to be supplied to the tree is available. The lighter textures are seen on the higher branches and vice versa. This is achieved by calling upon the lighter textures for the particle systems on the top branches, a mix of lighter and darker textures on the intermediate branches and darker textures on the lower branches.

The leaves are placed at different rotation angles within a user set range. A randomness value ensures that different values are chosen within that range. The leaves are placed along the tangent to the spline. It is observed that the stems that are higher up in a tree tend to have smaller newer leaves whereas the lower branches have larger older leaves. Hence, the leaves are scaled to different sizes within a user set range. A randomness value ensures that different values are chosen within that range.

4 RESULTS

We were able to generate trees of high diversity using the proposed method. This includes various western tree species (like Aspen, Fir, Maple, Pine, etc.) as well as Indian tree species (like Tulsi, Banyan, Mango, Neem, Gulmohar, Kadamb, Sal, Ashoka, Peepal, etc.) that are shown in Figures 9 and 10. The dataset including multiple instances of various tree species as listed in Table 2 is available at our website[1].

Additionally, our method allows adding new species or updating the grammar for existing species with minimal user editing. The code for our method as a Blender package, an user-friendly interface based automated tree generation, as shown in Figure 7 is available at our website.

Our method can produce good variation in tree structures, within and across species, due to the stochastic nature of grammar and geometries as shown in Figure 3. In addition, our method can also be used to generate the growth pattern of a given tree species, as shown in Figure 4.

In regard to comparison with other relevant state-of-the-art methods, we were able to overcome shortcomings of most relevant work in [16] by proposing the usage of local rather than global parameters, which allowed us to model a much wider range of species. This is demonstrated by generating a Pine tree as shown in Figure 8 where our method generates a more realistic tree as compared to the method from [16].

Regarding modelling of leaves, Figure 5 shows variations across seasons as depicted with colours of leaves in the maple tree. Similarly, variation in leaf appearance, size and density across seasons is shown in Figure 6. Video results in our website shows the movement of leaves and trunks owing to external factors like strong wind currents which were produced by attaching armatures to the leaves and trunk and then animating it. We can also model additional geometrical objects like fruits, flowers by using the Particle System based approach, as shown for Mango Tree in Figure 10(d).

The time to produce a single tree using our method ranged between approximately 10ms for trees with simple open structures like Sal to approximately 3000ms for complex and dense trees like Banayan.

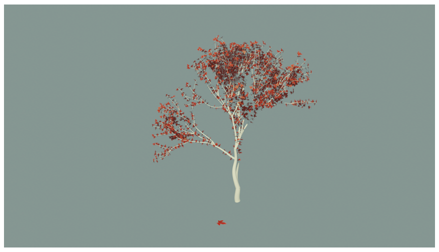
5 CONCLUSION & FUTURE WORK

We propose a novel grammar based approach for generating wide variety of tree species. The proposed grammar is easy to understand and can be edited by a novice user which overcomes the complexities of parametric models. The proposed method also gives higher control to the user by setting the geometries locally. Our method is scalable due to its inherent stochasticity to produce structurally varying trees.

Some straightforward extensions of this model can be to incorporate the features such as those proposed in [19] like pruning, wind sway, vertical attraction and tropism, degradation at range.

[1]https://cvit.iiit.ac.in/research/projects/cvit-projects/automated-tree-generation-using-grammar-particle-system/

(a) Green leaves


(b) Red leaves

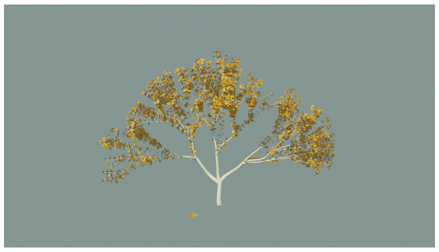

(c) Yellow leaves

Figure 5: The variation in leaves color across seasons for maple tree.

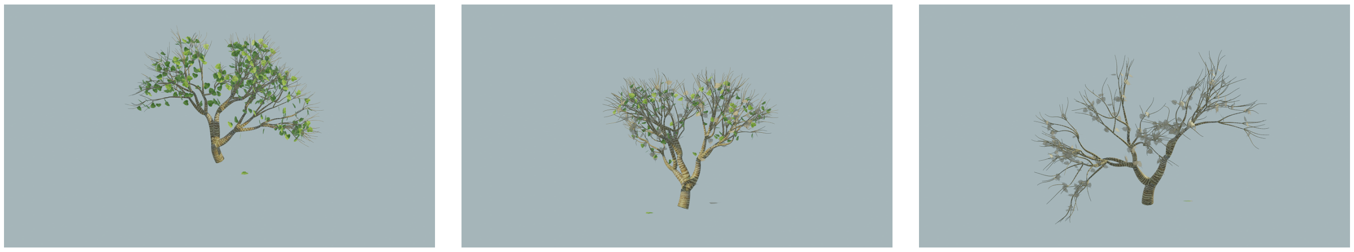
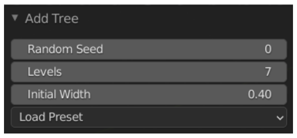


Figure 6: Different stages of tree foliage across seasons.


Add Tree

Random Seed	0
Levels	7
Initial Width	0.40
Load Preset	

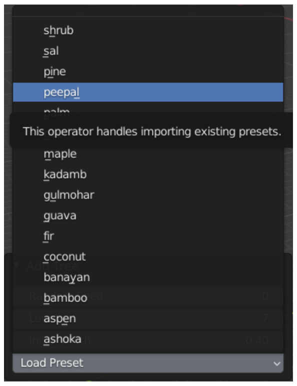
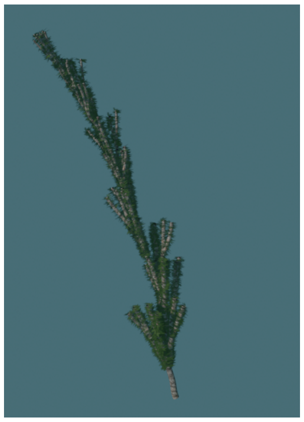

shrub
sal
pine
peepal
This operator handles importing existing presets.
maple
kadamb
gulmohar
guava
fir
coconut
banayan
bamboo
aspen
ashoka
Load Preset

Figure 7: Snapshot of our Blender add-on for easy and intuitive user editing.

Table 2: A list of tree species and number of trees for each species in our generated dataset.

Tree	Sample	Tree	Sample
Ashoka	1000	Lemon	1000
Aspen	1000	Mango	500
Bakul	800	Maple	500
Bamboo	2000	Neem	1500
Banyan	400	Palm	2000
Coconut	1500	Peepal	400
Fir	1000	Pine	1000
Guava	1500	Sal	1500
Gulmohar	500	Shrub	800
Kadamb	1000	Tulsi	800


(a) Result of Sun et al. [16]

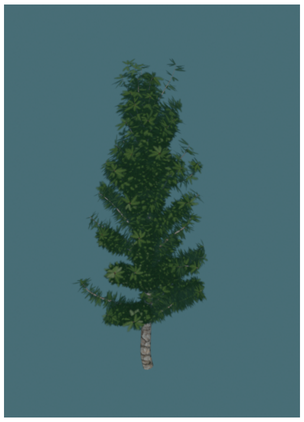
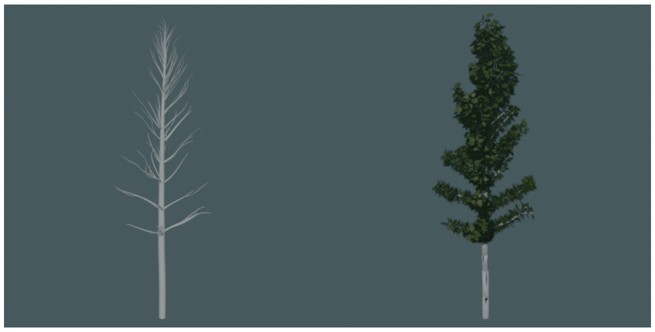
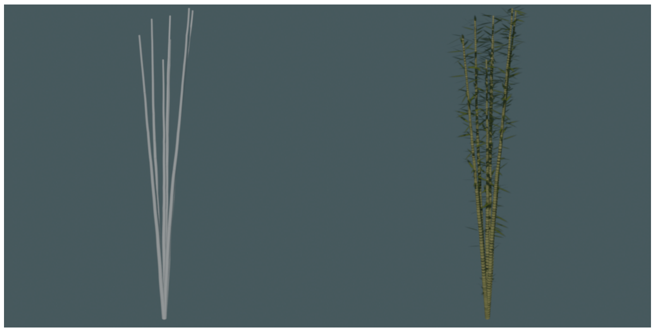
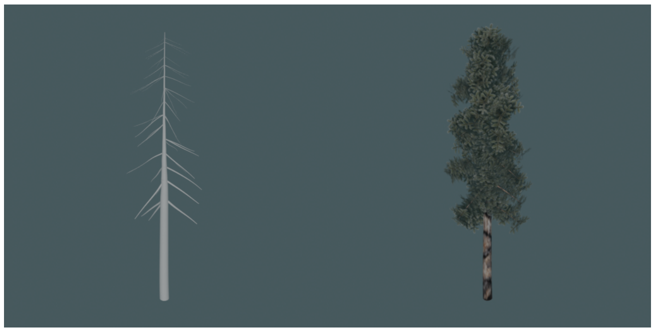

(b) Result of our method

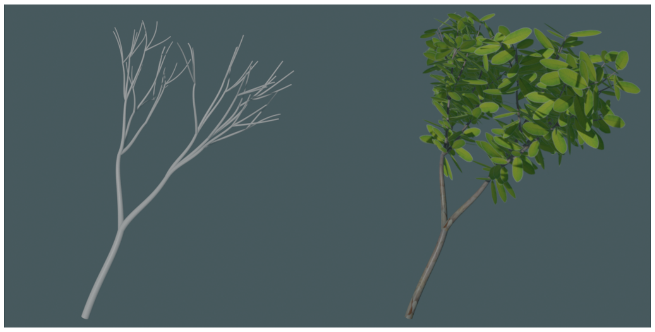
Figure 8: Qualitative comparison with [16]. (a) Keeping single global parameters for all kinds of curves [16] fails in the case of pine. (b) The proposed model specify separate local parameters for different types of curves to successfully generate a realistic Pine tree.

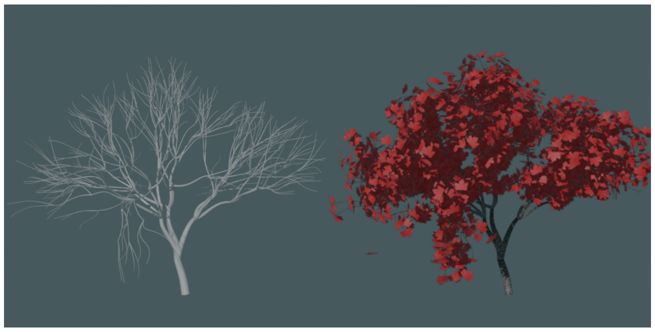
There is also a need to propose a good metric for formal comparison between various tree generation algorithms, which currently does not exist. Further its scalability for generation of vast forest stretches may need some optimisation. Learning-based approaches such as inverse procedural modelling can be integrated with this model to regress to real-world trees.

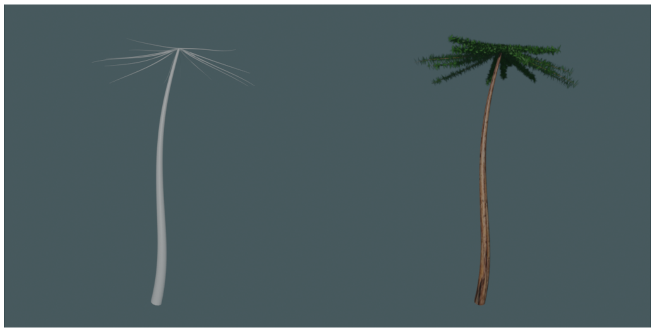

(a) Aspen

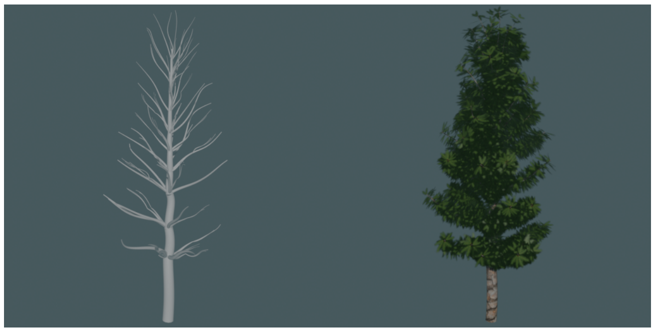

(b) Bamboo


(c) Fir


(d) Guava


(e) Maple


(f) Palm


(g) Pine

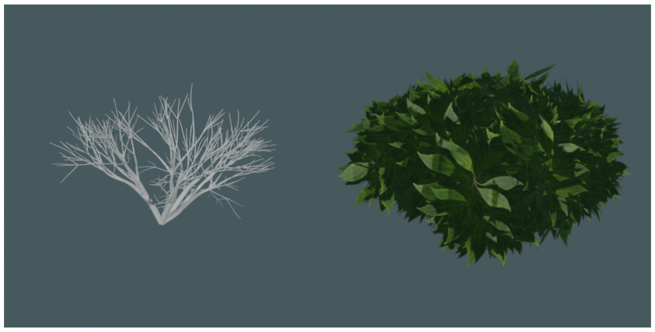
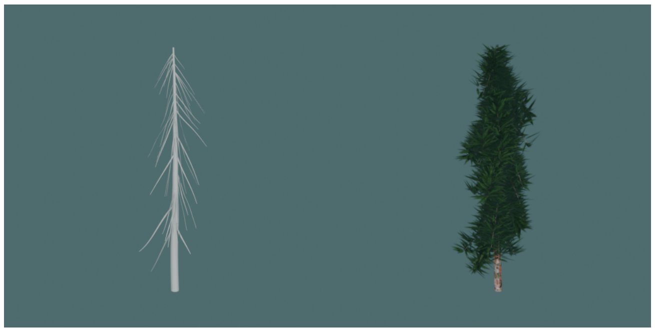
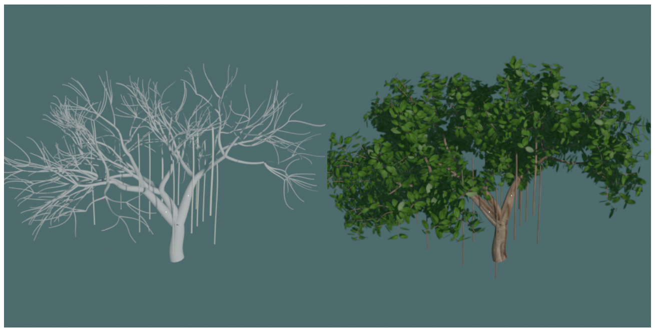
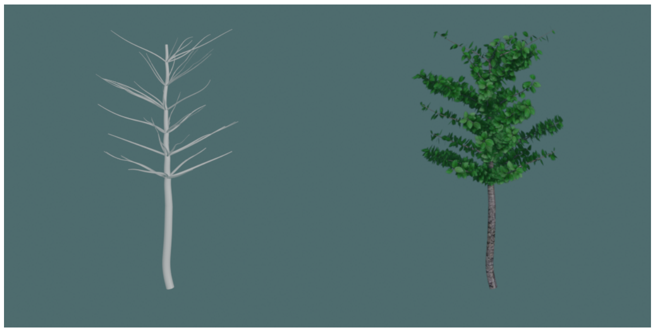
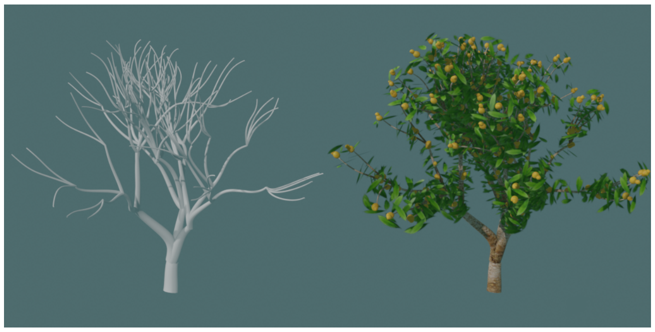

(h) Shrub

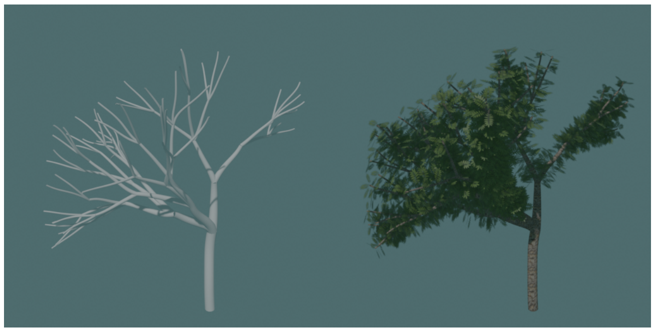
Figure 9: A subset of trees found in the western world created using the proposed model.

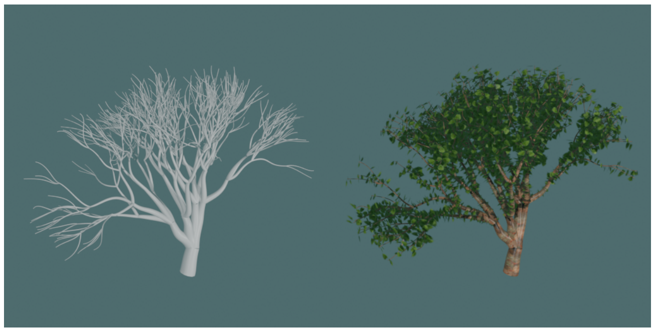

(a) Ashoka

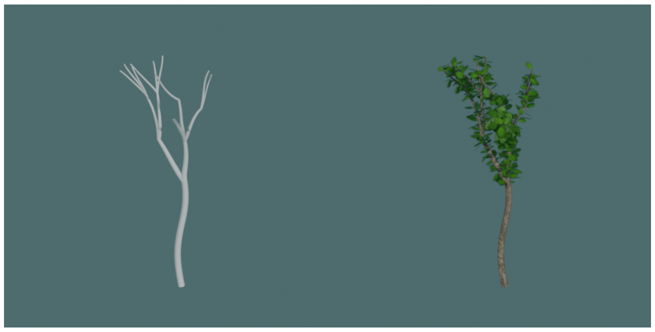

(b) Banyan


(c) Kadamb


(d) Mango


(e) Neem


(f) Peepal


(g) Sal

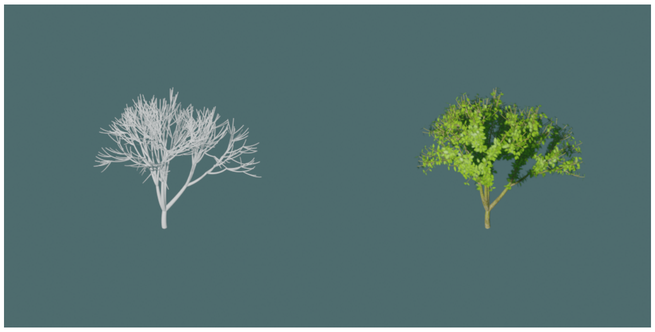

(h) Tulsi

Figure 10: A subset of trees found in the tropical Indian subcontinent generated using the proposed model.

REFERENCES

[1] Jesús Antonio Alvarez-Cedillo, Roberto Almanza-Nieto, and Juan Carlos Herrera-Lozada. 2010. Three dimensional hair model by means particles using Blender. *3D Research* 1, 3 (2010), 1–5.

[2] Oscar Argudo, Carlos Andújar, and Antoni Chica. 2020. Image-Based Tree Variations. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 174–184.

[3] Oliver Deussen, Pat Hanrahan, Bernd Lintermann, Radomír Měch, Matt Pharr, and Przemyslaw Prusinkiewicz. 1998. Realistic modeling and rendering of plant ecosystems. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 275–286.

[4] Feng Han and Song-Chun Zhu. 2003. Bayesian reconstruction of 3d shapes and scenes from a single image. In *First IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis, 2003. HLK 2003.* IEEE, 12–20.

[5] Julian Kenwood, James Gain, and Patrick Marais. 2014. Efficient Procedural Generation of Forests. (2014).

[6] Jinmo Kim. 2016. Modeling and optimization of a tree based on virtual reality for immersive virtual landscape generation. *Symmetry* 8, 9 (2016), 93.

[7] Aristid Lindenmayer. 1968. Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology* 18, 3 (1968), 300–315. https://doi.org/10.1016/0022-5193(68)90080-5

[8] Steven Longay, Adam Runions, Frédéric Boudon, and Przemyslaw Prusinkiewicz. 2012. TreeSketch: Interactive Procedural Modeling of Trees on a Tablet.. In *SBIM@ Expressive*. Citeseer, 107–120.

[9] Long Quan, Ping Tan, Gang Zeng, Lu Yuan, Jingdong Wang, and Sing Bing Kang. 2006. Image-based plant modeling. In *ACM SIGGRAPH 2006 Papers*. 599–604.

[10] William T Reeves. 1983. Particle systems—a technique for modeling a class of fuzzy objects. *ACM Transactions On Graphics (TOG)* 2, 2 (1983), 91–108.

[11] Daniel Ritchie, Ben Mildenhall, Noah D Goodman, and Pat Hanrahan. 2015. Controlling procedural modeling programs with stochastically-ordered sequential monte carlo. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–11.

[12] Tatsumi Sakaguchi. 1998. Botanical tree structure modeling based on real image set. In *ACM SIGGRAPH 98 Conference abstracts and applications*. 272.

[13] Ilya Shlyakhter, Max Rozenoer, Julie Dorsey, and Seth Teller. 2001. Reconstructing 3D tree models from instrumented photographs. *IEEE Computer Graphics and Applications* 21, 3 (2001), 53–61.

[14] Ondrej Šťava, Bedrich Beneš, Radomir Měch, Daniel G Aliaga, and Peter Krištof. 2010. Inverse procedural modeling by automatic generation of L-systems. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 665–674.

[15] Ondrej Stava, Sören Pirk, Julian Kratt, Baoquan Chen, Radomír Měch, Oliver Deussen, and Bedrich Benes. 2014. Inverse procedural modelling of trees. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 118–131.

[16] Ruoxi Sun, Jinyuan Jia, and Marc Jaeger. 2009. Intelligent tree modeling based on L-system. In *2009 IEEE 10th International Conference on Computer-Aided Industrial Design & Conceptual Design*. IEEE, 1096–1100.

[17] Tuukka M Takala, Meeri Mäkäräinen, and Perttu Hämäläinen. 2013. Immersive 3D modeling with Blender and off-the-shelf hardware. In *2013 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, 191–192.

[18] Ping Tan, Gang Zeng, Jingdong Wang, Sing Bing Kang, and Long Quan. 2007. Image-based tree modeling. In *ACM SIGGRAPH 2007 papers*. 87–es.

[19] Jason Weber and Joseph Penn. 1995. Creation and rendering of realistic trees. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 119–128.

[20] Ke Xie, Feilong Yan, Andrei Sharf, Oliver Deussen, Hui Huang, and Baoquan Chen. 2015. Tree modeling with real tree-parts examples. *IEEE transactions on visualization and computer graphics* 22, 12 (2015), 2608–2618.

[21] Mehmet Ersin Yumer, Paul Asente, Radomir Mech, and Levent Burak Kara. 2015. Procedural modeling using autoencoder networks. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 109–118.