Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Real-time Physically Plausible Simulation of Forest

Paper ID ???

ARTICLE INFO

Article history:
Received February 2, 2021

Keywords: Large-scale Forest Simulation, FEM, Look-Up-Table

ABSTRACT

In this paper, we propose a Lookup table method to simulate a physically plausible large-scale forest animation. Our method is rooted in the FEM simulator to simulate plant deformation, and establish a table through pre-computed dynamic equation data. Based on this table, an efficient physical attribute similarity algorithm for retrieval tables is proposed. Real-time search for the vertices displacement of the model from the table, and constantly update the current gesture of the model. The forest can react to the environment at an interactive rate by calculating the effects of random wind and gravity. Experiments show that our method can simulate a single tree with high efficiency, and can simulate a physically plausible forest with 1000 trees at a frame rate of 25(fps).

1. Introduction

Botanical environments occupies a large proportion in real life, therefore, a large number of plant simulations play an important role in computer graphics. Since a forest can consist of an extensive number of highly complex geometric models, real-time forest simulating is still a challenge. Most of methods simulate plant animation using physical simulation techniques that need to solve the equations of motion. The methods proposed in [1, 2] generates the real-time motion of a single tree through finite element analysis. By setting material properties, such as stiffness matrix, conform to biological characteristics, the plant animation is extremely realistic. However, due to excessive computing resources, thus the finite element algorithm cannot simulate thousands of trees at the same time. To ensure the real-time simulation of the forest, Ota el al. [3] uses a combination of stochastic and kinematic methods. Although it can reach a small scale, the model simulation depends on the spring mass model and random noise. This simulation method is easy to introduce simulation errors.

In this paper, we propose a real-time plausible simulation method of a single tree, which is physically-based. This simulation method has a high frame rate for a single tree, so it can be extended to a large-scale forest scene. Our basic idea is to convert the solution of the dynamic equation into a lookup operation on the table, which is pre-calculated using FEM in the preprocessing stage. Each entry in the table records a one-to-one correspondence between physical information and displacement increments. Considering the time and space efficiency of reconstruction work, we propose an optimized sampling data structure to store the displacement increment and physical information. The vertices displacement of a single tree deformed from the initial gesture to the current gesture can be obtained by accumulating the displacement increment continuously generated by the lookup operation on the table.

Moreover, since our animation is a free combination of all the displacement increments sampled by FEM, we can reconstruct the deformed gesture of the tree that has not been sampled. In our experiment, this method can simulate 200 trees at the same time, but for a forest, there may be only a few types of trees and the motion gesture of the same classifications of trees. Therefore, in the process of forest creation, it is essential to instantiate more trees by simple rotation and translation.

The remainder of the paper is organized as follows. In Sec.2, describes the related work. In Sec.3, we give an overview of our algorithm. In Sect.4, we propose our algorithm, including the

*Only capitalize first word and proper nouns in the title.

sampling data structure, the construction and search of process of the Look-Up-Table. The experimental results are described in Sec.6. And finally, conclusion and future work are given in Sec.7.

2. Related Work

Real-time animation of tree motion for large-scale forest scenes is challenging because of the large number of node data for a single plant. Many methods of plant animation have been developed. The representative methods are described as follows.

Morphing. Morphing[4, 5] is a common technique to generate animations of deformable characters by spline interpolation. Although realistic, this approach cannot generate extra motion. Zhang et al. [6] use motion graphs combined with stochastic motion. Barbič and Marco et al. [[7]] present a fast space-time optimization method to author physically based deformable object simulations that conform to keyframes.

Physics-based simulation. Physics-based simulation builds physical models of objects to simulate the movement of objects in real scenes. Unlike Morphing, physically-based simulation can provide dynamics, secondary motion and easier runtime control. Sakaguchi et al. [8] describe the tree model as a set of branches, twigs and leaves. Twigs grow from branches, leaves are attached to branch twigs. Tree animation is described as the rotational movements of segments. Ota et al. [3] simulation method based on the spring model that use $1/f^{\beta}$ noise provides natural motion to simulation trees swaying in wind fields. A hundred tree simulated in real-time using elastic body physics on CUDA[9]. Tree branches modeled by using an 1D Euler-Bernoulli beam model [10] or three-prism structures [11]. Using position-based simulation(PBD)[12, 13] or Finite Element Method(FEM)[14] can deform the whole plant. Our work is based on Finite Element Method to produce a reasonable deformation. We develop a search algorithm to achieve the movement of trees in the forest scene in real-time while preserving the kinematic characteristics of objects.

The Finite Element Method is one of the most popular approach to simulate solid model to produce large deformation. With discretizing the triangular mesh model into tetrahedral meshes[15] or hexahedron meshes[16, 17], we can artificially set the overall constitutive relationship of the model and the material properties between each voxel to simply control the model deformation. Perez et al.[18]set the constitutive equation use limiting elastic energy to model extremely nonlinear elasticity within the linear co-rotational formulation. Wang et al.[2] propose a solver can achieve realistic plant motion effects by setting the stiffness, mass density and damping properties based on the botanical system of individual tree[19]. FEM is an extremely time-consuming simulation, each time step needs to calculate the large sparse matrix in equations of motion. The equation has to be solved by several time-stepping schemes like implicit Euler integration[20] or implicit Newmark integrator[21, 22]. In the following years, multiple acceleration methods are proposed, which make real-time interaction possible. Barbič[23, 24] and Metaxas et al.[25] use model reduction of nonlinear systems to speed up solving equations of motion. Twigg and KačićAlesić et al.[26] simulate object consisted of multiple subparts with only limited inter-part interaction. Zhao et al.[1] extended this method to plant deformation. Their strategy can simulate a few plants in real-time, but it is not designed for large scene forest. Our method is similar to the Morphing, according to the linear calculation relationship of implicit Euler integration [20], we store data such as model stiffness matrix state and corresponding relative displacement through pre-calculation. The calculation process of each time step of the sparse matrix is converted into a special search model vertex relative deformation process to achieve real-time dynamic deformation of large-scale forests.

3. Algorithm Overview

As we mentioned, though FEM-based methods [2, 22, 24] provide an accurate physical simulation for a single tree, their computation overhead is too high for real-time application. In order to a chieve a real-time physically plausible simulation for a large-scale forest, the following two issues have to be addressed:

- *How to achieve a high performance physically plausible simulation for a single tree?*
- *How to extend the above tree simulation method to a large-scale forest?*

In order to address the first issue, we propose a real-time physically plausible simulation method for a single tree. The basic idea is to convert the physical simulation problem into a set of lookup operations from a FEM-based precomputed table. Thus all the expensive FEM computations are conducted in a precomputation stage, and the simulation stage only contains cheap lookup and reconstruction operations.

According to FEM, the vertices displacement Δu_{i+1} of a volumetric mesh at time step $i+1$ can be computed from Eq. 1:

$$\Delta u_{i+1} = f(K_i, v_i, f_i^{int}, f_i^{ext}) \tag{1}$$

where $K_i, v_i, f_t^{int}, f_i^{ext}$ are the global stiffness matrix, velocity matrix, internal force matrix and external force matrix at time step i respectively. And the quadruple $\mathbf{S} = (K, v, f^{int}, f^{ext})$ can be used to represent a physical state of a tree.

And the tree gesture Γ_i at time step i can be computed by adding the movement Δu_i at time step i to the previous Γ_{i-1}:

$$\Gamma_i = \Gamma_{i-1} + \Delta u_i \tag{2}$$

As illustrated in Fig. 1, in order to avoid conducting expensive FEM computation in the simulation stage, we precompute a table $\mathbb{T}$, each entry of which records a one-to-one correspondence from a physical state $\mathbf{S_i} = (K_i, v_i, f_i^{int}, f_i^{ext})$ to displacement increment Δu_{i+1}. Based on Eq. 1 and Eq. 2, $\mathbf{S_i}$ triggers Δu_{i+1}, and the previous gesture adds Δu_{i+1} to the next gesture. Due to the continuity of the data in the table, the physical state corresponding to the updated gesture of the tree is $\mathbf{S_{i+1}} = (K_{i+1}, v_{i+1}, f_{i+1}^{int}, f_{i+1}^{ext})$ (refer to Sec. 4.3 for more details). And in

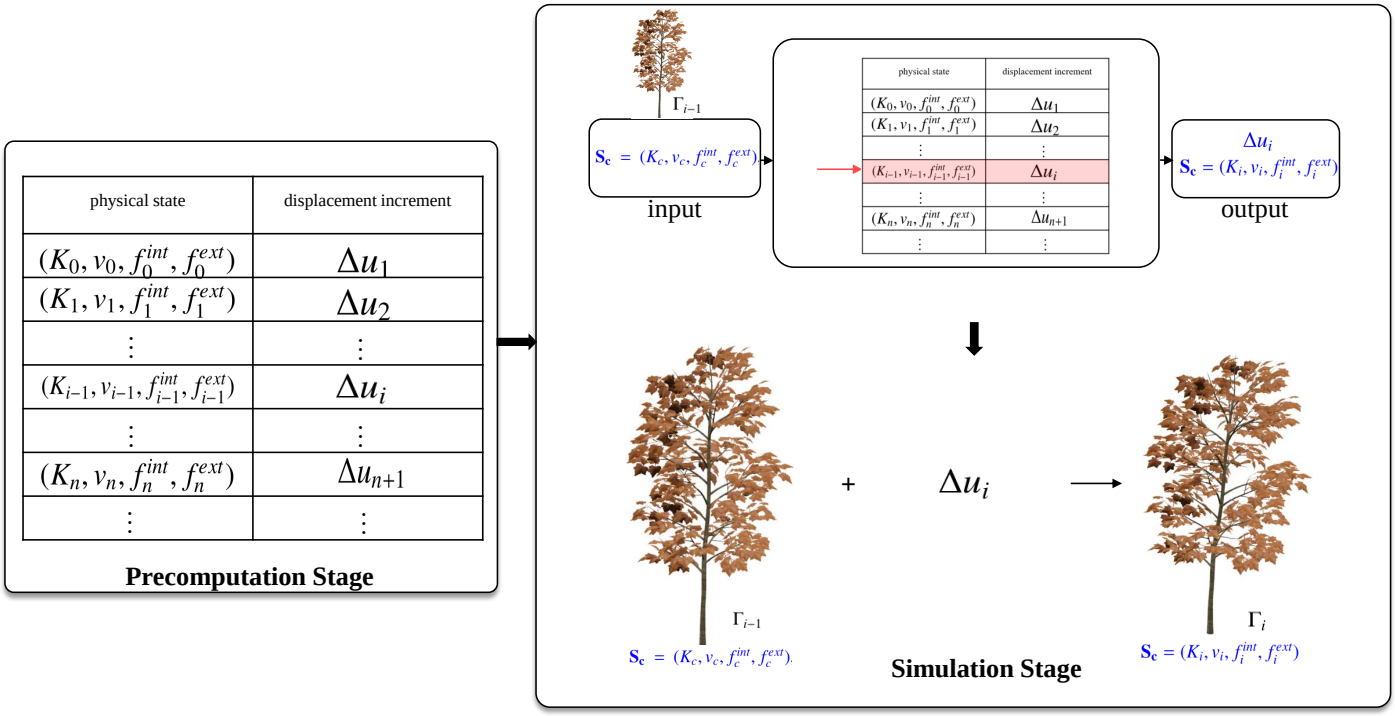


Fig. 1: Overview of the simulation method of a single tree. In the precomputation stage, a lookup table is established based on FEM sampling, then lookup the entry closest to the input physical state form the table, and finally update the current gesture and physical state of the tree by the output of the lookup table.

the simulation stage, we find the closest $(K, v, f^{int}, f^{ext}) \in \mathbb{T}$ to the current state of tree $\mathbf{S_c} = (K_c, v_c, f_c^{int}, f_c^{ext})$, and update the tree gesture according to the corresponding Δu by Eq. 2 (see Sec.4.4 for more details).

However, it is not practical to directly extend the above method to the simulation of a large-scale forest because the search table operation is time-consuming and the expensive storage memory cost to store an independent Lookup table for each tree. We use a skeleton algorithm to reduce the dimension of (K, v, f^{int}), reduce the storage capacity of the table, and accelerate the search efficiency of the table. In addition, build an independent Lookup table for every type of tree, instead of each tree, because of several types of trees occupy the main component of the forest. Even so, it is still impossible for us to simulate all the trees in a forest, and a type of tree in a forest has only a few ways to move. By randomly rotating and scaling the common motions of these representative trees that we simulate in real-time, many tree instances are created to form a forest.

However, in order to extend the above method to the simulation of a large-scale forest, the following two issues have to be addressed:

- *The problems caused by the high dimensionality of quadruple* $\mathbf{S} = (K, v, f^{int}, f^{ext})$: Sampling high dimensional $\mathbf{S}$ requires a huge number of samples, which results in a huge table $\mathbb{T}$. Additionally, the high dimensionality of $\mathbf{S}$ also suggests a time-consuming operation to find the closest (K, v, f^{int}, f^{ext}) in $\mathbb{T}$ to the current tree state $\mathbf{S_c}$. In order to reduce the dimensionality of $\mathbf{S}$, we propose to ...
- *The problem caused by the large number of trees in a forest:* It is still impractical to use an independent $\mathbb{T}$ for each tree because the memory requirement of storing all $\mathbb{T}$s for a large number of trees is too high. Our method only precomputes $\mathbb{T}$s for a few representative trees, and we think the forest is composed of a large number of tree instances by randomly rotating and scaling these representative trees.

4. Simulation of a Single Tree

How can one define meaningful physically-based dynamic for a large-scale forest in real-time. Only a tree is physically-based, the forest can maintain physical properties. In this section, we present a physically-based method to build a large-scale forest in real-time. We firstly derive the relationship between the displacement increment Δu_{i+1} and the physical state $\mathbf{S}_i$. Based on this derivation, we can sample the displacement increment and the corresponding physical state to construct a Lookup table. And then a physical state similarity algorithm is proposed to compare the state of the current gesture $\mathbf{S} = (K, v, f^{int}, f^{ext})$ with all $(K, v, f^{int}, f^{ext}) \in \mathbb{T}$. The animation of a large number of trees can be build based on the lookup of the table, finally, the simulation of a single tree can be extended to a forest by our optimization method.

4.1. Background: FEM Simulation

We use FEM discretizes the complex structure into a large number of elements, the deformable body can be described

as a volumetric mesh[18], which is consisted of s hexahedral meshes.

For a volumetric mesh with n vertices, $u \in \mathbb{R}^{3n}$, the displacement of the model relative to the static state, which contains displacements in the three directions of x,y,z. Buliding the equilibrium equation of each element based on the analysis of stress and strain in each element, then the dynamic equation of the whole model is established by the element equilibrium equation. The numerical solution of each element is obtained by boundary conditions and other constraints.

The motion of a deformable solid can be described by the Euler-Lagrange equation [27], which is a second order system of ordinary differential equations used to solve the unknown displacement vector u.

$$M\ddot{u} + D(u, \dot{u}) + f^{int}(u) = f^{ext} \tag{3}$$

where $M \in \mathbb{R}^{3n\times3n}$ is the mass matrix, $D(u, \dot{u}) \in \mathbb{R}^{3n\times1}$ is the damping matrix, $f^{int}(u) \in \mathbb{R}^{3n\times1}$ are internal deformation forces, $f^{ext} \in \mathbb{R}^{3n\times1}$ are external forces(e.g..,wind, pull forces). Internal forces are related to the displacement u:

$$f^{int}(u) = K(u)u \tag{4}$$

where $K \in \mathbb{R}^{3n\times3n}$ is the global stiffness matrix of volumetric mesh, which consists of element stiffness matrix:

$$K = \sum_{i=1}^{s} K_i^e(u) \tag{5}$$

where $K^e \in \mathbb{R}^{3n\times3n}$ is the element stiffness matrix.

In this paper, $D(u, \dot{u})$ is represented by the Rayleigh damping model[23], as shown in Eq. 6:

$$D(u, \dot{u}) = (\alpha M + \beta K(u)) \tag{6}$$

where α and β is the low-frequency and high-frequency component during deformation respectively.

4.2. Gesture Control Function

In the finite element method, the displacement increment of the model is affected by factors such as the internal properties of the model and external forces. We choose implicit Euler Algorithm 2 to solve Eq. 3, the velocity of the model at time step $i + 1$ can be calculated by Eq. 7.

$$v_{i+1} = \frac{(\Delta t)^2 K(u_i) + \Delta t D(u_i, \dot{u}_i) + M}{\Delta t\left(-(\Delta t K(u_i) + D(u_i, \dot{u}_i))\, v_i + f^{ext}(t) - f^{int}(u_i)\right)} \tag{7}$$

And Δu_{i+1} can be derived by Eq. 8.

$$\Delta u_{i+1} = v_{i+1}\Delta t \tag{8}$$

Based on Eq. 7 and 8, the displacement increment at time step $i + 1$ can be represented as:

$$\begin{aligned}\Delta u_{i+1} &= \frac{(\Delta t)^2 K(u_i) + \Delta t D(u_i, \dot{u}_i) + M}{-(\Delta t K(u_i) + D(u_i, \dot{u}_i))\, v_i + f^{ext}(t) - f^{int}(u_i)} \\ &= f(K_i, v_i, f_i^{int}, f_i^{ext})\end{aligned} \tag{9}$$

where M is a constant mass matrix, $D(u, \dot{u})$ is linearly related to the K matrix, M and $D(u, \dot{u})$ can be omitted. (K_i, v_i, f_i^{int}) are the global stiffness matrix, velocity matrix and internal force matrix at time step i, they can reflect the internal physical state of the model. f_i^{ext} is an external force matrix at time step i, an external factor that affects the state of the model. Therefore, the current physical state of the model can be represented by the quadruple $\mathbf{S} = (K, v, f^{int}, f^{ext})$.

The animation U of the tree consists of a set of continuous gestures $\{\Gamma_0, \Gamma_1, \ldots, \Gamma_t\}$, each can be computed by accumulating all previous displacement increments to the initial gesture Γ_0. Based on Eq. 2 and Eq. 9 , the gesture Γ_i at time step i can be calculated by the physical state $\mathbf{S_{i-1}} = (K_{i-1}, v_{i-1}, f_{i-1}^{int}, f_{i-1}^{ext})$ and the gesture Γ_{i-1} at time step $i - 1$:

$$\begin{aligned}\Gamma_{i+1} &= \Gamma_i + \Delta u_{i+1} \\ &= \Gamma_i + f(K_i, v_i, f_i^{int}, f_i^{ext}) \\ &= \Gamma_0 + \sum_{0}^{i} f(K_i, v_i, f_i^{int}, f_i^{ext})\end{aligned} \tag{10}$$

Each item in the physical state is a large matrix, the process of FEM iterative calculation is complicated and time-consuming. Our idea is to convert the calculation into a lookup operation on the table.

4.3. Design a Lookup Table

Construct the structure of the Lookup table. Based on Eq. 9, we decided to sample the physical state $\mathbf{S} = (K_i, v_i, f_i^{int}, f_i^{ext})$ and corresponding displacement increment Δu_{i+1} per frame. Sampling each frame of the continuous animation of different motion modes of the model to construct a Lookup table of $key(K_{i-1}, v_{i-1}, f_{i-1}^{int}, f_{i-1}^{ext}) \rightarrow value(\Delta u_i)$ pairs. The table consists of a series of physical state and corresponding displacement increments, $T = \left\{\left\{(K_0, v_0, {f_0}^{int}, f_0^{ext}) \rightarrow \Delta u_1\right\}, \ldots, \left\{(K_{z-1}, v_{z-1}, f_{z-1}^{int}, f_{z-1}^{ext}) \rightarrow \Delta u_z\right\}_z\right\}$.

Based on Eq. 9, the deformation Δu_i at time step i simulated by FEM can be calculated by the physical state $\mathbf{S}_{i-1}$. In other word, $\mathbf{S}_{i-1}$ causes Δu_i. And according to Eq. 4, Eq. 5, Eq. 7 and user-defined external force, the physical state $\mathbf{S}_i$ is update by Δu_i. Fem-based sampling is continuous, when we find the closet $(K_i, v_i, f_i^{int}, f_i^{ext}) \in \mathbb{T}$ to the state of current gesture $\mathbf{S}_c = (K_c, v_c, f_c^{int}, f_c^{ext})$, and updating the current gesture with the corresponding Δu_{i+1}, the internal physical state (K_c, v_c, f_c^{int}) of the current gesture is updated to $(K_{i+1}, v_{i+1}, f_{i+1}^{int})$. The external force $(f_{ext})_c$ corresponding to the current gesture is custom updated.

Physical State Similarity Algorithm. According to Eq. 10, the model animation consists of a series of displacement increments accumulated to the initial gesture. To search for a suitable displacement increment Δu needs to evaluate the total errors between the state of the current gesture $\mathbf{S_c}$ and the state of each entry in the table, we propose an effective approximation algorithm.

The current physical state $\mathbf{S_c} = (K_c, v_c, {f_c}^{int}, f_c^{ext})$ of the model is composed of internal state $(K_c, v_c, {f_c}^{int})$ and external state (f_c^{ext}). Thus, the total error can be divided into internal error and external error.

We use Eq. 11 to calculate the error E_i^{ext} of the external physical state:

$$E_i^{ext} = f_c^{ext} - f_i^{ext} \tag{11}$$

where f_c^{ext} is the external force input in real time, f_i^{ext} is the external force of the i-th record stored in the table.

For the velocity matrix $v_c \in \mathbb{R}^{3q}$, the velocity of q points maybe quiet different, Eq. 12 assign an error range for each point.

$$ErrorRange_{v_i^m} \cdot xyz = \frac{|v_i^m \cdot xyz|}{20}, m \in (1, q) \tag{12}$$

where v_i^m is the velocity of the m-th point of the velocity matrix in the i-th record in table. Then, computing the error of the velocity matrix based on q points:

$$if\left(\left|v_i^m - v_c^m\right| < ErrorRange_{v_i^m}\right) \quad E_{v_i} += 1 \tag{13}$$

where v_c^m is the velocity of the m-th point of the velocity matrix of the current gesture, E_{v_i} is the velocity matrix error. The error calculation of internal force f^{int} and global stiffness matrix K is the same as velocity v. The internal state error of the i-th record can be calculated as:

$$E_i^{int} = E_{K_i} + E_{v_i} + E_{f_i^{int}} \tag{14}$$

where E_{K_i} is the error of the global stiffness matrix, E_{v_i} is the error of velocity and $E_{f_i^{int}}$ is the error of internal force. Normalized internal error:

$$E_i^{int} = \frac{E_i^{int}}{3q} \tag{15}$$

We use the Gaussian method to normalize the external state error, set a weight for each physical state item calculate the total error. The total error of the physical state $E_{\mathbf{S_i}}$ of the record i is calculated using Eq. 16:

$$E_{\mathbf{S_i}} = (1 - w) * E_i^{int} + w * gaussian\left(E_i^{ext}\right) \tag{16}$$

where w is the weight of external state, the larger the weight, the more likely it is to cause sudden changes in the model. In order to make the simulation results more accurate, we need to manipulate the weights specially. When the next record found in the table is adjacent to the previous record, increase the value of the weight w.

4.4. Reconstruction of a Single Tree

The external forces are user-defined, such as the wave function of the simulated wind. At initialization, all matrices in the internal state (K, v, f^{int}) to zero matrix, we only need to compare the external error by Eq. 11 and search for the displacement increment Δu corresponding to the record with the smallest error as the first displacement of the animation.

After this step, add the displacement increment to the previous gesture to update the current gesture of the model. Simultaneously, update the internal state $(K, v, f^{int})_c$ of the current gesture with the physical state in the next record, and use the wave function to update the external state (f_c^{ext}). In order to lookup the next deformation increment of the model in the table, we need to compare the total error of the current state by Eq. 16.

The pseudocode for the simulation of a single tree is shown in Alg. 1.

Algorithm 1: One frame segment search algorithm

Input: Pre-calculated table
$T = \left\{\{\mathbf{S_0} \rightarrow \Delta u_1\}, \ldots, \{\mathbf{S}_{z-1} \rightarrow \Delta u_z\}_z\right\}$
Output: Δu

```
f_c^ext ← f^ext,
if the first step then // is the initial search
    for each S_i in 𝕋 do
        if min(E_i^ext) then
            Index = i
else
    for each S_i in 𝕋 do
        if min(E_Si) then
            Index = i
K_c = K_{i+1}
f_c^int = f_{i+1}^int
v_c = v_{i+1}
return Index;
```

5. Simulation of a Large-scale Forest

Although we use sampling to replace the calculation of the physical state of each frame, but for $K \in \mathbb{R}^{3n\times3n}$, $v \in \mathbb{R}^{3n\times1}$, $f^{int} \in \mathbb{R}^{3n\times1}$, $f^{ext} \in \mathbb{R}^{3n\times1}$, we can only store a few table entries. We used the following method to reduce sampling data, thereby optimizing storage space and search algorithm.

Extract skeleton elements: A deformation body contains a lot of elements, and the physical state of all elements constitutes the overall physical state. Based on Eq. 5, since the global stiffness matrix of the model is linearly related to the element stiffness matrix. The global stiffness matrix of an element is the sum of the stiffness matrices of all elements adjacent to the element. Similarly, the global velocity matrix and the global internal force matrix are also composed of the velocity matrix and the internal matrix of the elements respectively. We can use the physical state of some elements to approximate the overall physical state to reduce the dimensionality of $\mathbf{S}$.

We use L1-media skeleton construction algorithm [28] to extract global skeleton of deformable solid, and calculate all the elements surrounding the skeleton.

Supposing that the extracted skeleton is composed of M elements, and the elements contains $q(q << n)$ vertices. Reducing the global stiffness matrix $K \in \mathbb{R}^{3n\times3n}$ to $\tilde{K} \in \mathbb{R}^{3q\times3q}$, also, the velocity matrix and the internal force matrix are reduced from $\left\{v, f^{int}\right\} \in \mathbb{R}^{3n\times1}$ to $\left\{\tilde{v}, \tilde{f}^{int},\right\} \in \mathbb{R}^{3q\times1}$. The structure of the table changed from $(K, v, f^{int}, f^{ext}) \rightarrow \Delta u$ to $(\tilde{K}, \tilde{v}, \tilde{f}^{int}, f^{ext}) \rightarrow \Delta u$.

6. Result

In this section, we present several examples of trees and forests simulation using our simulation method. All the results we show are on NVIDIA GeForce RTX 2060 GPU and Intel Core i7-8700K CPU with 16GB of memory. We use the Vega [22] running process for sampling, which depends on the anisotropic stiffness matrix and the implicit backward Euler algorithm. Our experiment is mainly compared with FEM. We evaluate our method in simulating and rendering a large number of various types of trees, and analyze the overall performance and time cost of the rendering time.

The deformation of the tree is determined by the external force point and the external forces, such as wind and gravity. The force point and gravity are usually constant, the main influence on the deformation of the object is the magnitude and direction of the wind. Our wind is superimposed by various types of cosine waves to simulate spatially varying and controllably turbulence in the wind field. Other wind fields such as Perlin[1, 29] noise ,$1/f^{\beta}$[3] noise and Navier-Stokes equations [30] can also be used.

6.1. Simulation of A Single Tree

In this part of the results, we use a maple tree for experiment. We use FEM to sample its movement in the pre-computation stage. FEM discretizes the apricot tree into a volumetric mesh with 5710 elements, but we only extracted 35 skeleton elements for sampling.

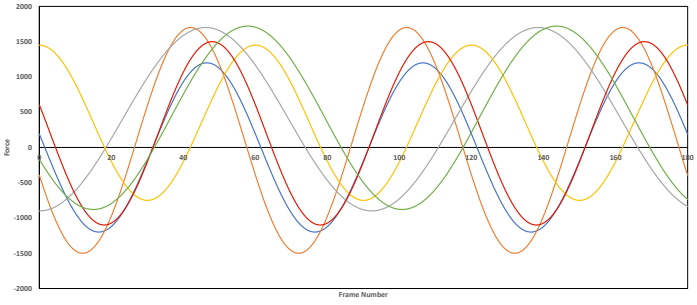
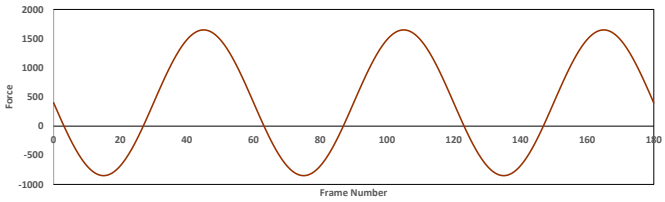
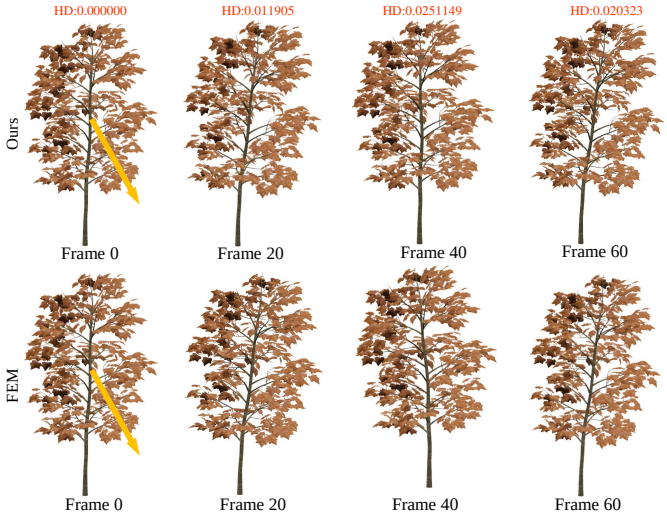


Fig. 2: Several wind field for sampling. Each kind of curve represents the change law of the wind force in the wind field with the frame number. We sample the movement of maple trees in each wind field.

As shown in Fig. 2, we use the cosine waves to represent each type of wind field, there are six types of wind field, the angle between these wind fields and the initial tree is 60°, and the direction will not change over time. Since our wind field is represented by a cosine wave, after a period time, the motion of the tree will repeated. For the maple tree movement under each wind field, we only sample the physical information and the corresponding displacement increment of the previous 180 frames, and formed a Lookup table using the above sampled data. In the simulation stage, by constantly looking up this table, the trajectory of the maple tree in the target wind field shown in Fig. 3a is generated.



(a) Wind field function



(b) Gesture and corresponding Hausdorff distance error

Fig. 3: The realism of our method for single tree simulation. (a) is a function of generating wind field. Generate the motion in (b) through the wind field in (a), top:simulation with FEM, bottom:simulation with our method. The value of HDE represents the similarity between our method and FEM in the current frame. The yellow arrow indicates the the angle between the wind field and the initial tree direction is 60°.

Number of elements(polar)	Frame rate(fps)
35	1133
135	1098
235	836
335	602
435	487
Number of elements(maple)	**Frame rate(fps)**
35	1133
135	1098
235	836
335	602
435	487

Table 1: Comparison of frame rate under different number of sampling elements. Polar:. Maple:

The direction of the target wind field we simulated also intersects 60° with the initial tree, we compared the gesture of our simulation method and FEM in the wind field at frames 20, 40 and 60 (Fig. 3b). Hausdorff Distance is an algorithm to measures how far two subsets of a metric space are from each other. We use Hausdorff Distance HD measure the similarity between

our frame and the real frame. The first maximum deformation is achieved at frame 22, where the error is only 0.011905. At frame 40, the maple tree may be affected by wind or damping force and start to move in the opposite direction. Our simulation method is to use the sampled continuous displacement increment to simulate, and the simulation error does not always accumulate. Through the HD value or directly observing the gesture in the Fig. 3b, we can see that our simulation error has dropped at frame 60.

The optimization method of extracting elements on the skeleton mentioned in Sec. 4.3, the number of sampling elements it extracts is the decisive factor to achieve a high performance simulation for a single tree. Therefore, in this experiment, we tested the influence of the number of sampled elements on the frame rate of a single tree. We still sample the movement of the maple tree in the six wind fields with constant directions shown in Fig. 2, but the number of elements sampled is different, the results are as shown in Table. 1. It can be seen that the frame rate will decrease as the number of sampling elements increases. Normally, the more sampling information, the more accurate the trajectory of the tree, but at the same time the simulation speed will also decrease.

6.2. Large-scale Forest Simulation

The purpose of our paper is to plausible simulate a forest in real-time, so the simulation performance of a single tree must have a high frame rate. And all the above experimental results also prove that when the number of elements sampled is 35, the frame rate and simulation effect of a single tree can be accepted, so our forest simulation experiment will also use 35 elements for sampling.

Fig. 4: Three kinds of trees used in our forest simulation. The left is poplar, the two on the right are maple and apricot.

For a forest scene simulation, the type and number of trees are essential. We want to simulate an autumn forest scene, which contains several common trees in this season. Fig.4 shows the three representative trees we selected for this experiment, namely apricot, maple and apricot. The sampling method of each tree is the same, and then we use poplar tree to explain in detail. As illustrated in Fig. 5, the trajectory of the tree in a wind field with the same wind strength and different directions is different. Considering the diversity of the movement of trees in the forest, it is important to sample the movement of a type of tree in all directions. We sampled the motion data of each tree in different wind fields in six directions($0°, 60°, 120°...300°$) shown in the Fig. 5. Similar to the first experiment, we still only sample the previous 180 frames of data for each trajectory. Maple and apricot trees are also sampled in the same way, and an independent Lookup table is established for the sampling data of each tree.

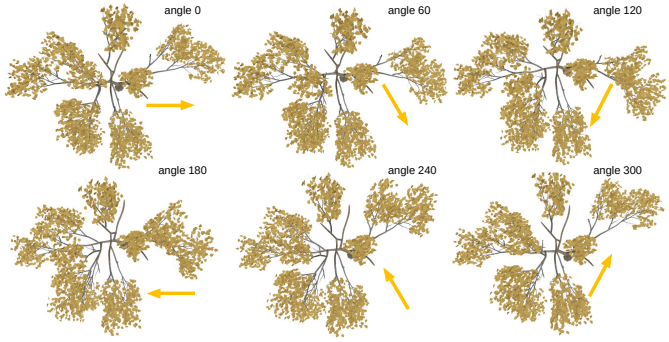


Fig. 5: Gestures at frame 20 of six wind field. Poplar moves in six wind fields respectively, the directions of these wind field are different, but the wind strength in each direction is the same, which is represented by cosine waves. The yellow arrow indicates the direction of the wind field.

We first full simulate the movement of multiple trees in a small scene, the scene contains the above three types of trees, specify the number of each type of tree. Apply a wind field to the scene, establish a table lookup operation for each tree in the scene, and reconstruct the movement of each tree in parallel according to the wind field. Our simulation performance will decrease as the number of trees increase, Table . shows the effect of the number of trees on the frame rate. The result of this experiment fully simulating 50 trees is shown in Fig. .

Tree numbers	20	30	40
Frame rate(fps)	100	80	70

*******+ small scene full simulation result********

For the full simulation method of the experiment, we can only simulate 50 trees in real-time. However, the number of trees in a forest is hundreds of thousands, so it is impossible to simulate all the trees in the forest. In order to realize the simulation of a large-scale forest scene, we plane to simulate a small part of the trees in real-time, by randomly rotating and scaling these trees to instantiate more trees to form a forest. Fig. ? is our simulated forest with 500 trees. In this experiment, we simulated ? poplar trees, ? maple trees and ? almond trees in real time, and all other trees were obtained by instantiation. The simulation FPS of this forest is ? FPS.

******+ a large-scale forest scene simulation result*****

For the aim to beautify the effect of the forest simulation, we added lighting and shadows to the scene. Physically based rendering was chosen as our lighting model, and we use shadow mapping technology to render simple shadows. Undoubtedly, these additional operations will reduce the performance of the simulation. The number of trees we fully simulated is the same as above, but the number of instantiated trees is reduced to 400,

the rendering FPS is 20 FPS. The experiment effect can be seen in Fig. 6.

Fig. 6: Forest in the wind tree. Our large-scale forest scene with lighting, shadows and terrain.

7. Conclusion and Future Work

7.1. Conclusion

This paper proposes a high performance method to simulate a single tree by efficiently updating the gesture of the tree from a precomputed table that records one-to-one mapping relationship between a physical state and a displacement increment. And finally we extend this method to the simulation of a forest scene, which can physically simulate large-scale trees in real-time. Our method is based on physical theory, and the experiment results also demonstrate that our method can obtain plausible simulation results in real-time.

However, our method relies on precomputing tables for a few representative trees, which limits the ability to simulate a large-scale species-rich forest.

7.2. Future Work

Domain decomposition is a future direction worth exploring. When the rigidity of the trunk part is high, an external force is applied, the twigs and leaves will shake more violently, but branches almost stationary. The main idea of model decomposition is to no longer store the global displacement vector and physical information of the model at the same time, but to decompose the model into multiple sub-domains and store the information of each sub-domains separately(Fig. 7). The model can be decomposed into k sub-domains C, $C \in \{C_1, C_2, ..., C_k\}$. After the model is decomposed, the small fluctuation part of the tree trunk does not need to be stored repeatedly, which reduces the redundancy of data and saves a lot of storage space.

Fig. 7: Domain decomposition. Tree model(left) is decomposed into individual leaves, twigs and branches(middle left; middle right; right).

Acknowledgments

Acknowledgments should be inserted at the end of the paper, before the references, not as a footnote to the title. Use the unnumbered Acknowledgements Head style for the Acknowledgments heading.

Appendix A. Implicit Backward Euler Integration

Algorithm 2: One step of implicite Backward Euler integration

Input: values of $u, \dot{u}$ at timestep i,external force f_{ext} at timestep i; max number of iterations per step j_{max}(implicit backward euler solver: $j_{max} = 1$); tolerance TOL to avoid unnecessary iterative steps; timestep size $\triangle t$.

Output: values of $u, \dot{u}$ at timestep $i + 1$

$u_{i+1} \leftarrow u_i$;

for $j = 1 : j_{max}$ **do** `// perform a implicite backward euler interation`

  Evaluate internal forces $R(u_{i+1})$

  Evaluate stiffness matrix $K(u_{i+1})$

  Evaluate the local damping matrix

  $C = \alpha M + \beta K(u_{i+1})$

  Form the system matrix $A = (\triangle t)^2 K(u_{i+1}) + \triangle t C + M$

  $residual \leftarrow \triangle t(-(\triangle t K(u_{i+1}) + C)v_t + f_{ext} - R(u_{i+1})))$

if $(\|residual\|_2 < TOL)$ **then**

  break out of for loop;

solve the $n \times n$ dense symmetric linear system:

  $A(\triangle \dot{u}) = residual$

$\dot{u}_{i+1} = \dot{u}_i + \triangle \dot{u}$ `// update velocities`

$u_{i+1} = \triangle u + \dot{u} \triangle t$ `// update accelerations`

return $u_{i+1}, \dot{u}_{i+1}$

References

[1] Zhao, Y, Barbič, J. Interactive authoring of simulation-ready plants. ACM Transactions on Graphics (TOG) 2013;32(4):1–12.

[2] Wang, B, Zhao, Y, Barbič, J. Botanical materials based on biomechanics. ACM Transactions on Graphics (TOG) 2017;36(4):1–13.

[3] Ota, S, Tamura, M, Fujimoto, T, Muraoka, K, Chiba, N. A hybrid method for real-time animation of trees swaying in wind fields. The Visual Computer 2004;20(10):613–623.

[4] Alexa, M. Recent advances in mesh morphing. In: Computer graphics forum; vol. 21. Wiley Online Library; 2002, p. 173–198.

[5] Lamouret, A, van de Panne, M. Motion synthesis by example. In: Computer Animation and Simulation'96. Springer; 1996, p. 199–212.

[6] Zhang, L, Zhang, Y, Chen, W, Peng, Q. Real-time simulation of large-scale dynamic forest with gpu. In: APCCAS 2008-2008 IEEE Asia Pacific Conference on Circuits and Systems. IEEE; 2008, p. 614–617.

[7] Barbič, J, da Silva, M, Popović, J. Deformable object animation using reduced optimal control. In: ACM SIGGRAPH 2009 papers. 2009, p. 1–9.

[8] Sakaguchi, T, Ohya, J. Modeling and animation of botanical trees for interactive virtual environments. In: Proceedings of the ACM symposium on Virtual reality software and technology. 1999, p. 139–146.

[9] Oliapuram, NJ, Kumar, S. Realtime forest animation in wind. In: Proceedings of the Seventh Indian Conference on Computer Vision, Graphics and Image Processing. 2010, p. 197–204.

[10] Habel, R, Kusternig, A, Wimmer, M. Physically guided animation of trees. In: Computer Graphics Forum; vol. 28. Wiley Online Library; 2009, p. 523–532.

[11] Yang, M, Huang, MC, Yang, G, Wu, EH. Physically-based animation for realistic interactions between tree branches and raindrops. In: Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology. 2010, p. 83–86.

[12] Müller, M, Chentanez, N. Solid simulation with oriented particles. In: ACM SIGGRAPH 2011 papers. 2011, p. 1–10.

[13] Bender, J, Müller, M, Macklin, M. Position-based simulation methods in computer graphics. In: Eurographics (tutorials). 2015, p. 8.

[14] Shabana, AA, Ling, FF. Vibration of discrete and continuous systems. Springer; 1997.

[15] Mueller, M, Teschner, M. Volumetric meshes for real-time medical simulations. In: Bildverarbeitung für die Medizin 2003. Springer; 2003, p. 279–283.

[16] Hahs Jr, CA, Carroll, ED. Method and apparatus for generating a hexahedron mesh of a modeled structure. 1998. US Patent 5,731,817.

[17] Nieser, M, Reitebuch, U, Polthier, K. Cubecover–parameterization of 3d volumes. In: Computer graphics forum; vol. 30. Wiley Online Library; 2011, p. 1397–1406.

[18] Perez, J, Perez, AG, Otaduy, MA. Simulation of hyperelastic materials using energy constraints. Proc of CEIG 2013;.

[19] Makowski, M, Hädrich, T, Scheffczyk, J, Michels, DL, Pirk, S, Pałubicki, W. Synthetic silviculture: multi-scale modeling of plant ecosystems. ACM Transactions on Graphics (TOG) 2019;38(4):1–14.

[20] Müller, M, Gross, MH. Interactive virtual materials. In: Graphics interface; vol. 2004. 2004, p. 239–246.

[21] Wriggers, P, Zavarise, G. Computational contact mechanics. Encyclopedia of computational mechanics 2004;.

[22] Sin, FS, Schroeder, D, Barbič, J. Vega: non-linear fem deformable object simulator. In: Computer Graphics Forum; vol. 32. Wiley Online Library; 2013, p. 36–48.

[23] Barbič, J, James, DL. Real-time subspace integration for st. venant-kirchhoff deformable models. ACM transactions on graphics (TOG) 2005;24(3):982–990.

[24] Barbič, J, Zhao, Y. Real-time large-deformation substructuring. ACM transactions on graphics (TOG) 2011;30(4):1–8.

[25] Metaxas, D, Terzopoulos, D. Dynamic deformation of solid primitives with constraints. In: Proceedings of the 19th annual conference on Computer graphics and interactive techniques. 1992, p. 309–312.

[26] Twigg, CD, Kacic-Alesic, Z. Point cloud glue: Constraining simulations using the procrustes transform. In: Symposium on Computer Animation. 2010, p. 45–53.

[27] Rao, SS. Vibration of continuous systems; vol. 464. Wiley Online Library; 2007.

[28] Huang, H, Wu, S, Cohen-Or, D, Gong, M, Zhang, H, Li, G, et al. L1-medial skeleton of point cloud. ACM Trans Graph 2013;32(4):65–1.

[29] Perlin, K. Improving noise. In: Proceedings of the 29th annual conference on Computer graphics and interactive techniques. 2002, p. 681–682.

[30] Pirk, S, Niese, T, Hädrich, T, Benes, B, Deussen, O. Windy trees: computing stress response for developmental tree models. ACM Transactions on Graphics (TOG) 2014;33(6):1–11.