This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2017.2661308, IEEE Transactions on Visualization and Computer Graphics

Real-time Interactive Tree Animation

Ed Quigley, Yue Yu, Jingwei Huang, Winnie Lin, Ronald Fedkiw

Abstract—We present a novel method for posing and animating botanical tree models interactively in real time. Unlike other state of the art methods which tend to produce trees that are overly flexible, bending and deforming as if they were underwater plants, our approach allows for arbitrarily high stiffness while still maintaining real-time frame rates without spurious artifacts, even on quite large trees with over ten thousand branches. This is accomplished by using an articulated rigid body model with as-stiff-as-desired rotational springs in conjunction with our newly proposed simulation technique, which is motivated both by position based dynamics and the typical $O(N)$ algorithms for articulated rigid bodies. The efficiency of our algorithm allows us to pose and animate trees with millions of branches or alternatively simulate a small forest comprised of many highly detailed trees. Even using only a single CPU core, we can simulate ten thousand branches in real time while still maintaining quite crisp user interactivity. This has allowed us to incorporate our framework into a commodity game engine to run interactively even on a low-budget tablet. We show that our method is amenable to the incorporation of a large variety of desirable effects such as wind, leaves, fictitious forces, collisions, fracture, etc.

Index Terms—Computer Graphics, Physically-based Modeling, Botanical Tree

1 Introduction

The subtle movement of trees is a prevalent yet mesmerizing natural phenomenon. When walking down a street in autumn, one can find the fluttering of leaves and the swaying of branches no less varied or poetic than the flames in a fireplace or ocean waves on the beach. In this work, we strive to capture the complex natural motions exhibited by trees.

Motivated by position based dynamics [1], as well as the $O(N)$-style algorithms for articulated rigid body simulation proposed by [2] and others, we designed a new method for efficiently solving articulated rigid body systems where joints can be arbitrarily stiff as they are in the case of trees, especially those with thick trunks and branches. This is accomplished in part by using the analytic solutions to the spring dynamics equations in order to model rotations about joints. Our simulation method is significantly faster than Featherstone-style approaches since these quite often require either small time steps or expensive implicit time integration, whereas we leverage the simplicity of a position based dynamics approach. However, we do not require repeated iteration as is common in the Gauss-Seidel-style approach to position based dynamics. Our solver is significantly faster than both of these approaches with the drawbacks of neither when it comes to stiff-joint systems. However, in order to obtain this increase in performance, we sacrifice the ability to accurately model joints that are readily flexible, such as the links of a chain. Fortunately, this aggressive approximation does not hinder the simulation of trees since their angular displacements are relatively small on a per-joint basis and large-scale bending is achieved from the cumulative effect of many joints.

Specifically, we propose a new approach for the simulation of trees as systems of rigid bodies articulated by stiff joints via approximations that permit an analytic (and thereby robust and efficient) treatment of joint springs. We discuss how various factors such as fictitious forces, collisions, and wind can be integrated into such a solver. We demonstrate this method's utility in parameterizing and efficiently simulating tree models with many degrees of freedom.

2 Previous Work

There are many approaches one might take in order to simulate trees, and many of these have not yet been fully explored or even considered. For example, there is an interesting body of work on hair simulation, see e.g. [3], [4], [5], and it would be interesting to consider extending this work to situations with branching structures as in [6], [7]. The most physically accurate way to simulate trees would be to use many small tetrahedra and a finite element method. However, this is infeasible in practice since it would require a vast number of tetrahedral elements as well as biomechanically accurate constitutive models. One could perform faster simulations by embedding small thin branches into a coarser tetrahedral cage mesh or by approximating the deformations with a reduced deformable model [8]. However, a reduced deformable has global modes and cannot be used to model varying local stiffnesses. This can be partially addressed by articulating multiple reduced deformable domains [9]. Even so, the reduced deformable domains still contain their own global modes making it difficult to set varying stiffnesses locally. This typically leads to trees with twigs that are too stiff or a main trunk that is too soft bending severely in a manner that resembles plants with green, flexible stems, whereas our proposed method is geared towards stiff, woody trees.

When objects are stiff, one typically considers rigid body approximations, and well-known approaches for evolving articulated rigid bodies in linear time exist for both generalized [2] and maximal coordinates [10]. A tree can be modeled as an articulated rigid body (see e.g. [11]) as long as one provides some notion of an elastic stiffness for the joints. The main drawback with this approach is that the stiff springs utilized to limit bending in the thicker regions of the tree require either small time steps or implicit integration, both of which can be rather costly. Therefore, when

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2017.2661308, IEEE Transactions on Visualization and Computer Graphics

Fig. 1: A park containing many trees. The willow tree contains 970,748 articulated rigid bodies, the cherry blossom tree contains 246,979 articulated rigid bodies, each apple tree has 42,212 articulated rigid bodies, and each maple tree has 138,965 articulated rigid bodies. The most complex tree in the scene contains 3,267,482 articulated rigid bodies.

computational efficiency is required, either for real-time or large scale applications, one is forced to resort to either softer springs or potentially error-prone approximations of the linear solver. Both approaches produce undesirable artifacts and/or excessive bending similar to that exhibited by flexible underwater plants. To combat these issues, [12] proposed a modification to Baraff's Lagrange multiplier method that integrates stiff forces implicitly for open-chain articulated rigid bodies, and [13] proposed the use of exponential integrators for solving stiff problems in computer graphics. The general methodology that we pursue—that of improving stability by finding an approximate subproblem that can be solved analytically and thus robustly—dates back some time; see for example [14] and the discussions therein.

Work on the geometric modeling of trees considers L-systems [15], rules based on geometry [16], environmental factors [17], botany [18], and various combinations of these, e.g. L-system models that interact with their environment [19]. Researchers have also developed capture and reconstruction techniques that take in data formats such as point clouds [20], videos [21], [22], images [23], [24], [25], and sketches [26], [27]. Wind-tree interaction has been simulated in [28], [29], [30] and animated in real time in [31], [32], [33]. [34] modeled tree motion via a stochastic treatment that obviated the need to simulate a full wind field. [35] proposed a growth model and used SPH combined with sensor particles to achieve two-way coupled effects in real time. The aforementioned [9] presented a Featherstone-style method for tree simulation that operates on articulated reduced deformable domains, but for shallow hierarchies and few branches. This method was extended by [36] to incorporate fracture, wind using Perlin noise, and rest state editing using inverse kinematics; it was further extended by [37] to incorporate automatic assignment of stiffness and damping parameters. [38] proposed a technique for setting anisotropic stiffnesses for materials such as stems and leaves. [39] presented a linear time algorithm for simulating flexible trees, and [40] detailed an image-based modeling approach as well as a sweeping method for simulation that propagates forces and branch displacements. Trees have also been simulated to interactively respond to rain [41]. [42] simulated the change in shape of drying leaves. [43] used a particle-based approach to simulate underwater plants.

3 Geometric Representation

We model the trunk, branches, and twigs as a series of conical frustums obtaining a piecewise linear approximation of the curved branch. A leaf is represented by a thin triangular prism allowing us to approximate it as a triangle for interactions with the wind. Fruits are treated as spheres in order to utilize efficient collision detection techniques. Each articulated rigid body of the tree maintains a rigid state vector $(m, \mathbf{x}, \mathbf{v}, \mathbf{a}, I, R, \boldsymbol{\omega}, \boldsymbol{\alpha})$ in maximal coordinates as well as information about its parent, its children, and the forces being applied to it. The root rigid body has zero parent rigid bodies and all other rigid bodies have exactly one parent, defined by sweeping upward from the root.

All joints are assumed to be point joints with zero translational and three rotational degrees of freedom. The rotation vector $\boldsymbol{\theta} = (\theta_x, \theta_y, \theta_z)^T$ is the rotation from a parent rigid body's joint frame to the frame of the child rigid body. Regardless of the rest pose for the child rigid body's orientation, we may set $\boldsymbol{\theta} = \mathbf{0}$ by choosing an appropriate orientation for the parent rigid body's joint frame. To each joint we attach three one-dimensional rotational springs,

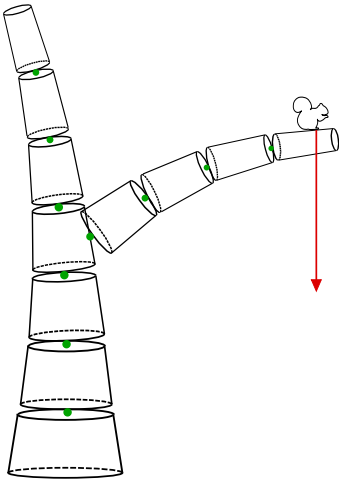


Fig. 2: A tree is modeled as a collection of articulated rigid bodies. Green circles represent the point joints between bodies, and the red arrow illustrates a force applied to the rightmost rigid body due to the weight of a small creature.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2017.2661308, IEEE Transactions on Visualization and Computer Graphics

Fig. 3: A tree made up of 9.5k articulated rigid bodies is simulated in real time at 86 frames/sec.[1]

each resisting the angular displacement θ_x, θ_y, or θ_z about the corresponding axis of the joint frame.

Given a quaternion or equivalent rotation matrix for the rotation from a child's world space frame to its parent's joint frame, we convert the rotation into a rotation vector representation $\boldsymbol{\theta}$ in the standard fashion. Then, after updating $\boldsymbol{\theta}$ in time using our newly proposed algorithm, we transform back to the quaternion or rotation matrix formulation. Although one might worry about temporal consistency issues in the spring dynamics equations emanating from the well-known issues with Euler angles, we have not observed any problematic phenomena. In fact, the spurious artifacts caused by picking a particular sequence of rotation axes can be alleviated by interleaving infinitesimal rotations about the x, y, and z axes. In the limit, interleaving infinitely many triples of infinitesimally small fractional angles, the differential rotations become asymptotically commutative and the order of rotations becomes inconsequential. See e.g. [44], [45]. A diagram of our model is given in Figure 2.

4 Internal Tree Dynamics

It is well-known that branching articulated rigid bodies can be evolved using $O(N)$-style algorithms, see e.g. [2]. We propose an $O(N)$-style algorithm as well, but we make several aggressive assumptions that alleviate the need for small time steps or costly iterative solvers without restricting our ability to emulate tree dynamics. Most importantly, our algorithm allows for the use of analytic solutions to the spring dynamics equations in order to alleviate any time step restriction whatsoever either for accuracy or stability reasons. In fact, our method robustly and efficiently takes only a single time step per rendered frame ($1/30$ sec), allowing us perform simulations in real time (see Figure 3).

During each time step of our evolution algorithm, we evolve the tree's rigid bodies in the following four stages:

External force computation: For each rigid body we compute external forces and torques due to gravity, user mouse input, collision penalty forces (Section 6), wind forces (Section 7), etc. The world space net external force and torque on rigid body p are given by $\mathbf{f}_p = \sum_i \mathbf{f}_{p,i}$ and $\boldsymbol{\tau}_p = \sum_i \mathbf{r}^*_{p,i}\mathbf{f}_{p,i}$. Here, $\mathbf{r}_{p,i}$ extends from the point at which rigid body p is articulated with its parent to the point at which $\mathbf{f}_{p,i}$ is applied, and $\mathbf{r}^*_{p,i}$ denotes the cross product matrix of $\mathbf{r}_{p,i}$. This stage is entirely parallelizable and exhibits $O(N/T)$ time complexity for T threads.

Composite body update: Our algorithm makes use of composite rigid bodies, which are defined for each rigid body by rigidifying the entire outboard subtree emanating from that body inclusive of the body itself. For each composite rigid body, we compute the mass, the world space inertia tensor, and the total external force and torque applied to the composite body evaluated about its parent joint. We perform these calculations starting from the leaf-level rigid bodies and working backward towards the root of the tree. For each rigid body p, we first compute its contribution to its composite body (i.e. to $\hat{m}_p$, $\hat{I}^{\mathrm{w}}_p$, $\hat{\mathbf{f}}_p$, $\hat{\boldsymbol{\tau}}_p$, where $\hat{\cdot}$ denotes a composite body value), then sum over its children's composite bodies adding in their contributions:

$$\hat{m}_p = m_p + \sum_{c\in C_p} \hat{m}_c \tag{1}$$

$$\hat{I}^{\mathrm{w}}_p = I^{\mathrm{w}}_p - m_p\mathbf{p}^{m*}_p\mathbf{p}^{m*}_p + \sum_{c\in C_p} (\hat{I}^{\mathrm{w}}_c - \hat{m}_c\hat{\mathbf{p}}^{m*}_c\hat{\mathbf{p}}^{m*}_c) \tag{2}$$

$$\hat{\mathbf{f}}_p = \mathbf{f}_p + \sum_{c\in C_p} \hat{\mathbf{f}}_c \tag{3}$$

$$\hat{\boldsymbol{\tau}}_p = \boldsymbol{\tau}_p + \sum_{c\in C_p} (-\hat{\mathbf{p}}^{a*}_c\hat{\mathbf{f}}_c + \hat{\boldsymbol{\tau}}_c) \tag{4}$$

where C_p is the set of children of body p, I^{w} is the world space inertia tensor, $\mathbf{p}^m_p$ is the center of mass of rigid body p relative to the center of mass of rigid body p's composite body, $\hat{\mathbf{p}}^m_c$ is the center of mass of rigid body c's composite body relative to the center of mass of rigid body p's composite body, and $\hat{\mathbf{p}}^a_c$ points from the location where rigid body c connects with rigid body p to the location where rigid body p attaches to its parent (see Figure 4).

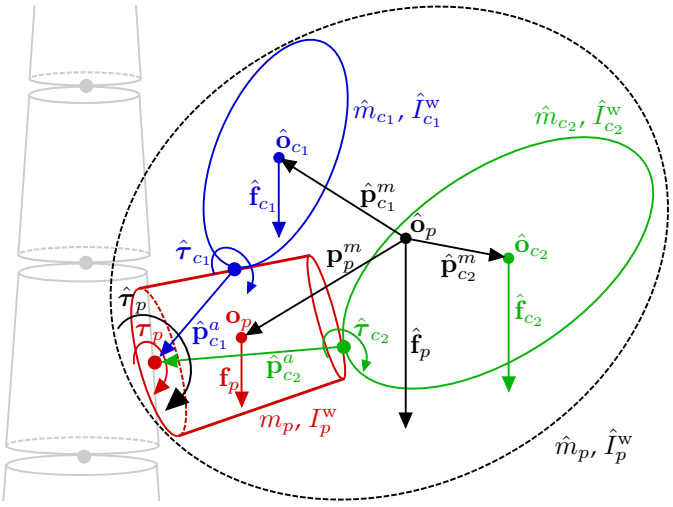


Fig. 4: The red rigid body has two children, whose composite rigid bodies are shown as blue and green ellipses. The center of mass of each composite rigid body is denoted by a point $\hat{\mathbf{o}}$. The black, dashed ellipse denotes the composite rigid body corresponding to the red rigid body. Other labeled values correspond to the quantities in Equations 1–4.

1. We record all timing information from simulations run with 15 threads.

Fig. 5: A willow tree sways freely in the wind. Both the trunk and the individual branches are articulated using many rigid bodies. While the thin branches are flexible enough to sway in the wind, the thick trunk does not bend (971k rigid bodies, 1.6 sec/frame).

This sweep has the dependency structure of the rigid body hierarchy but is still amenable to parallelization. For example, one could maintain a list of all childless rigid bodies in the tree and begin the sweep by assigning each thread to the next available childless rigid body. Each thread then processes rigid bodies from the leaf towards the root. Whenever a thread encounters a rigid body with multiple children where at least one child has not yet been processed, the thread stops and moves on to the next available childless body. Once no childless rigid bodies are available to be processed, the sweep ceases to be entirely parallel. At this point the worst case performance occurs if another thread happens to be starting at a childless rigid body which is the maximum depth d away from the root of the tree. Thus, the time complexity of this sweep is $O(N/T + d)$.

Analytic spring evolution: We evaluate our analytic springs in joint space where we assume the joint is at the origin. The inertia tensor of the composite body $\hat{I}^{\mathrm{w}}$ is rotated into joint space via $R^j_{\mathrm{w}}\hat{I}^{\mathrm{w}}R^{\mathrm{w}}_j$, and then the inertia tensor about the joint is computed as $\tilde{I} = -\hat{m}(R^j_{\mathrm{w}}\hat{\mathbf{p}})^*(R^j_{\mathrm{w}}\hat{\mathbf{p}})^* + R^j_{\mathrm{w}}\hat{I}^{\mathrm{w}}R^{\mathrm{w}}_j$ where $\hat{\mathbf{p}}$ is the center of mass of the composite body relative to the point at which the composite body is attached to its parent. Similarly, we move the external torque into the joint space via $\tilde{\boldsymbol{\tau}} = R^j_{\mathrm{w}}\hat{\boldsymbol{\tau}}$.

We define K as a 3×3 diagonal matrix whose entries correspond to the stiffnesses of the three one-dimensional springs governing the joint. Using Rayleigh damping, the analytic equation of motion for the spring is

$$\tilde{I}\ddot{\boldsymbol{\theta}} + (\alpha\tilde{I} + \beta K)\dot{\boldsymbol{\theta}} + K\boldsymbol{\theta} = \tilde{\boldsymbol{\tau}} \tag{5}$$

where we expose β as a user parameter, but α is set identically to 0 since our analytic approach provides robustness and stability even when the time step is equal to the frame duration (i.e. not requiring any so-called "ether drag"). We transform $\tilde{I}$ and K into a pair of diagonal matrices by solving a generalized eigenvalue problem, i.e. using the Cholesky factorization $\tilde{I} = LL^T$ to write the eigendecomposition $L^{-1}KL^{-T}X = X\Lambda$. Setting $U = L^{-T}X$, we use the identities $U^T\tilde{I}U = \delta_{3\times3}$ and $U^TKU = \Lambda$ in order to diagonalize Equation 5 into three independent second order linear ordinary differential equations

$$\ddot{\boldsymbol{\theta}}' + (\alpha\delta_{3\times3} + \beta\Lambda)\dot{\boldsymbol{\theta}}' + \Lambda\boldsymbol{\theta}' = U^T\tilde{\boldsymbol{\tau}} \tag{6}$$

where $\boldsymbol{\theta}' = U^{-1}\boldsymbol{\theta}$. Each row of Equation 6 takes the form $\ddot{\theta} + b\dot{\theta} + k\theta = f$. Depending on the roots of the characteristic equation $r^2 + br + k = 0$, the analytic solutions are of the form $\theta(t) = c_1e^{r_1t} + c_2e^{r_2t} + f/k$, $\theta(t) = (c_1 + c_2t)e^{rt} + f/k$, or $\theta(t) = c_1e^{\gamma t}\cos(\mu t) + c_2e^{\gamma t}\sin(\mu t) + f/k$. Thus, given the current joint condition, we may analytically integrate the state $\boldsymbol{\theta}$ robustly and stably forward for all time, albeit we simply use the analytic solution $1/30$ sec later. Note that $\boldsymbol{\theta}$ can be clamped in order to enforce joint limits. Since we evolve each joint independently of all other joints, this stage exhibits $O(N/T)$ time complexity when parallelized.

Note that Equation 5 is technically not the analytic solution to the exact problem since in reality external torques and composite body configurations change throughout the time step. Instead, Equation 5 is the analytic equation for a "frozen coefficient" version of the problem. However, all discretizations of ordinary differential equations, both explicit and implicit, use approximations "frozen" throughout the time step. Our analytic equation is therefore not without errors; however, it is quite stable and robust, and it admits well-behaved, visually pleasing solutions.

Rigid body state update: Given $\boldsymbol{\theta}$ for every joint, we traverse the tree upwards from the root in order to compute the final maximal coordinates state for each rigid body. For a rigid body c that shares joint j with its parent p, we first compute R^j_c from $\boldsymbol{\theta}_c$. Then, R^j_c and the analytic derivatives of $\boldsymbol{\theta}_c$ can be used to obtain

$$R^{\mathrm{w},n+1}_c = R^{\mathrm{w},n+1}_p R^p_j R^j_c \tag{7}$$

$$\boldsymbol{\omega}^{n+1}_c = \boldsymbol{\omega}^{n+1}_p + R^{\mathrm{w},n+1}_p R^p_j\dot{\boldsymbol{\theta}}_c \tag{8}$$

$$\boldsymbol{\alpha}^{n+1}_c = \boldsymbol{\alpha}^{n+1}_p + R^{\mathrm{w},n+1}_p R^p_j\ddot{\boldsymbol{\theta}}_c + (\boldsymbol{\omega}^{n+1}_p)^*\boldsymbol{\omega}^{n+1}_c. \tag{9}$$

Finally, we update the center of mass state of rigid body c to be consistent with its parent joint configuration via

$$\mathbf{x}_c = \mathbf{x}_p + R^{\mathrm{w}}_p\mathbf{d}^p_j - R^{\mathrm{w}}_c\mathbf{d}^c_j \tag{10}$$

$$\mathbf{v}_c = \mathbf{v}_p + \boldsymbol{\omega}^*_pR^{\mathrm{w}}_p\mathbf{d}^p_j - \boldsymbol{\omega}^*_cR^{\mathrm{w}}_c\mathbf{d}^c_j \tag{11}$$

$$\mathbf{a}_c = \mathbf{a}_p + (\boldsymbol{\alpha}^*_p + \boldsymbol{\omega}^*_p\boldsymbol{\omega}^*_p)R^{\mathrm{w}}_p\mathbf{d}^p_j - (\boldsymbol{\alpha}^*_c + \boldsymbol{\omega}^*_c\boldsymbol{\omega}^*_c)R^{\mathrm{w}}_c\mathbf{d}^c_j \tag{12}$$

where $\mathbf{d}^p_j$ and $\mathbf{d}^c_j$ are the time-invariant position of joint j in rigid body p's and rigid body c's frame, respectively, and all other quantities are taken to be at time t^{n+1}. These equations satisfy the joint constraints by construction. The dependency structure of this sweep is the exact opposite of the composite body update sweep, and therefore the time complexity of this sweep is also $O(N/T + d)$.

5 Fictitious Forces

In the world frame, the Newton-Euler equations of motion about the center of mass of rigid body c's composite body are

$$\hat{\mathbf{f}}^{\mathrm{total}}_c = \hat{m}_c\hat{\mathbf{a}}_c \tag{13}$$

$$\hat{\boldsymbol{\tau}}^{\mathrm{total}}_{c,m} = \hat{I}^{\mathrm{w}}_c\boldsymbol{\alpha}_c + \boldsymbol{\omega}^*_c\hat{I}^{\mathrm{w}}_c\boldsymbol{\omega}_c. \tag{14}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2017.2661308, IEEE Transactions on Visualization and Computer Graphics

Fig. 6: A tree's root is translated left and right. (Left) Without fictitious forces, the tree remains vertical. (Right) With fictitious forces, the tree bends in response to the root's acceleration.

Here, $\hat{\mathbf{f}}_c^{\text{total}}$ is the total internal and external force on the composite body and $\hat{\boldsymbol{\tau}}_{c,m}^{\text{total}}$ is the total internal and external torque on the composite body evaluated at its center of mass. Note that our use of Equations 8 and 9 implies that $\hat{\boldsymbol{\alpha}}_c = \boldsymbol{\alpha}_c$ and $\hat{\boldsymbol{\omega}}_c = \boldsymbol{\omega}_c$, i.e. the joint angle of rigid body c's composite body inherits its value from the joint angle of rigid body c. The analog of Equation 12 for rigid body c's composite body is

$$\hat{\mathbf{a}}_c = \mathbf{a}_j + (\boldsymbol{\alpha}_c^* + \boldsymbol{\omega}_c^* \boldsymbol{\omega}_c^*)\hat{\mathbf{p}}_c \tag{15}$$

where $\mathbf{a}_j$ is the pointwise linear acceleration of rigid body c's parent joint. Recall that $\hat{\mathbf{p}}_c$ is the center of mass of rigid body c's composite body relative to its parent joint position. The total torque on rigid body c's composite body evaluated about its parent joint

$$\hat{\boldsymbol{\tau}}_c^{\text{total}} = \hat{\mathbf{p}}_c^* \hat{\mathbf{f}}_c^{\text{total}} + \hat{\boldsymbol{\tau}}_{c,m}^{\text{total}} \tag{16}$$

(with Equations 13–15 substituted in) can be rewritten as

$$\hat{\boldsymbol{\tau}}_c^{\text{total}} = \hat{m}_c \hat{\mathbf{p}}_c^* \mathbf{a}_j + (\hat{I}_c^{\text{w}} - \hat{m}_c \hat{\mathbf{p}}_c^* \hat{\mathbf{p}}_c^*)\boldsymbol{\alpha}_c + \boldsymbol{\omega}_c^* (\hat{I}_c^{\text{w}} - \hat{m}_c \hat{\mathbf{p}}_c^* \hat{\mathbf{p}}_c^*)\boldsymbol{\omega}_c. \tag{17}$$

Rotating Equations 17 and 9 into rigid body c's parent joint frame gives

$$\hat{m}_c \tilde{\mathbf{p}}_c^* \tilde{\mathbf{a}}_j + \tilde{I}_c \tilde{\boldsymbol{\alpha}}_c + \tilde{\boldsymbol{\omega}}_c^* \tilde{I}_c \tilde{\boldsymbol{\omega}}_c = \tilde{\boldsymbol{\tau}}_c^{\text{int}} + \tilde{\boldsymbol{\tau}}_c \tag{18}$$

$$\tilde{\boldsymbol{\alpha}}_c = \tilde{\boldsymbol{\alpha}}_p + \ddot{\boldsymbol{\theta}}_c + \tilde{\boldsymbol{\omega}}_p^* \tilde{\boldsymbol{\omega}}_c \tag{19}$$

where the internal and external forces of $\tilde{\boldsymbol{\tau}}_c^{\text{total}}$ become $\tilde{\boldsymbol{\tau}}_c^{\text{int}}$ and $\tilde{\boldsymbol{\tau}}_c$, respectively. Substituting Equation 19 into Equation 18 allows us to revise Equation 5 to include fictitious forces on the right hand side, i.e. replacing $\tilde{\boldsymbol{\tau}}_c$ with $\tilde{\boldsymbol{\tau}}_c + \tilde{\boldsymbol{\tau}}_c^{\text{fict}}$ where

$$\tilde{\boldsymbol{\tau}}_c^{\text{fict}} = -\hat{m}_c \tilde{\mathbf{p}}_c^* \tilde{\mathbf{a}}_j - \tilde{I}_c(\tilde{\boldsymbol{\alpha}}_p + \tilde{\boldsymbol{\omega}}_p^* \tilde{\boldsymbol{\omega}}_c) - \tilde{\boldsymbol{\omega}}_c^* \tilde{I}_c \tilde{\boldsymbol{\omega}}_c. \tag{20}$$

Note that the fictitious forces do not participate in the recursive summations in Equations 3 and 4 which are for the external force and torque only.

One way to observe the effect of the fictitious force terms is to move a tree's base kinematically, as in Figure 6. However, fictitious forces are always active to some degree as long as the terms in Equation 20 are not identically zero. Note that using our analytic spring evolution with a time step corresponding to 30Hz can give inaccurate results when including the fictitious forces. Thus, our real-time system allows for a multiplier on the various terms in $\tilde{\boldsymbol{\tau}}_c^{\text{fict}}$ in order to obtain the desirable effects of fictitious forces without undesirable artifacts from taking large time steps.

6 Collisions

In addition to the rigid bodies that comprise a tree, other "free" rigid bodies may exist independently in our environment. For example, when the torque on a fruit's joint exceeds a threshold, we break the joint and allow the fruit to fall away from its parent branch as a non-articulated rigid body. To maintain visual plausibility it is important to handle collisions between these free rigid bodies and the articulated rigid bodies that are still part of the tree. See Figures 7 and 8.

At the beginning of each frame, we update each free rigid body using typical rigid body evolution. Then we detect collisions between these bodies and the tree's articulated rigid bodies. For each collision, we apply a penalty force of the form $\mathbf{f}_{coll} = \zeta(\mathbf{v}_{rel} \cdot \mathbf{n})\mathbf{n}$ to the intersecting tree rigid body where $\mathbf{v}_{rel}$ is the pointwise velocity of the free rigid body relative to the pointwise velocity of the tree and ζ is a user parameter. We then evolve the tree as usual. After updating the tree, we compute the impulse that makes the velocity of the free rigid body match the velocity of the tree, and only apply this impulse when it pushes the free rigid body away from the tree. That is, the impulse is not applied when it would cause the free rigid body to stick to the tree. Finally, if the free rigid body is interpenetrating the tree, we apply a small "push out" displacement to help alleviate this intersection, see e.g. [46]. Although our internal tree dynamics algorithm runs quite well at 30Hz, this penalty-based approach to collisions may require a smaller time step especially when penalty forces become large.

We speed up collision detection by building an acceleration structure based on the reachable workspace (see e.g. [47]). The reachable workspace of any given subtree with root body p is the sphere swept out by finding the longest straight path within the subtree starting from p's parent joint and rotating it about p's parent joint. We precompute the radius of every subtree's reachable workspace starting from the leaves and traveling inwards. Let η_p be the length of rigid body p, and $\hat{\mathbf{p}}_c^a$ be the vector that points from rigid body c's parent joint to rigid body p's parent joint. Then we precompute the radius of the reachable workspace of body p's subtree $\hat{\eta}_p$ recursively via $\hat{\eta}_p = \max(\eta_p, \max_{c \in C_p}(||\hat{\mathbf{p}}_c^a|| + \hat{\eta}_c))$ where $\hat{\eta}_c$ is the radius of the

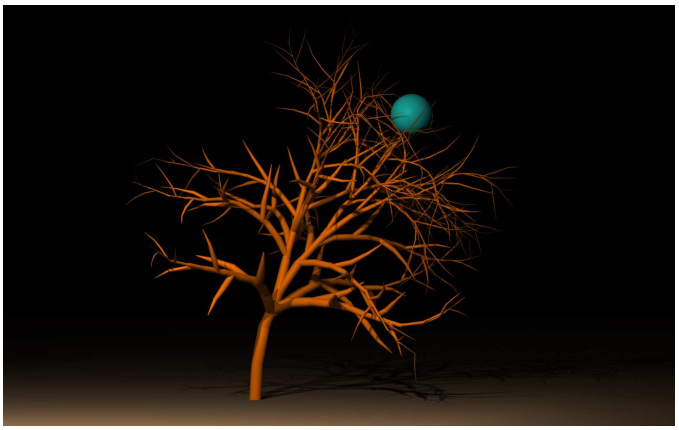


Fig. 7: We use penalty forces to handle collisions with free rigid bodies. Here, a ball falls onto a small tree colliding with and bending its branches.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2017.2661308, IEEE Transactions on Visualization and Computer Graphics

Fig. 8: Apples break away from their parent branches and collide with other branches as they fall to the ground (42k rigid bodies, 10 sub-timesteps/frame, 13 substeps/sec).

Fig. 9: (Top) A maple tree bends in a strong wind. (Bottom) The leaves of the maple tree flutter in the breeze. Some break away from their branches swirling and tumbling as they fall to the ground (139k rigid bodies, 0.2 sec/frame).

reachable workspace of body c's subtree. During simulation, we test for collisions by traversing the tree from the root outwards pruning a subtree if its reachable workspace does not intersect with the free rigid body.

7 Wind

Most interesting tree motion is created by the influence of wind. In our simulation we immerse a tree in a spatially varying wind velocity field by orthographically projecting a two-dimensional uniform square grid that encompasses the tree and its surroundings. The wind velocity is defined outside of the orthographic projection via constant extrapolation. Each grid cell contains a spatially constant wind velocity with its own number of octaves, persistence of amplitudes in higher octaves, and starting phase angle. To add more visually interesting perturbations, we rotate the wind back and forth about the up vector $\hat{\mathbf{j}}$ using an oscillating spline function of the form $h(||\mathbf{x}-(\mathbf{x}\cdot\hat{\mathbf{j}})\hat{\mathbf{j}}||)$ where the root of the tree is at the world origin so that the oscillations occur on concentric cylinders resulting in a back-and-forth wiggling of the wind. That is, given a parametric wind velocity $\bar{\mathbf{v}}_{wind}(\mathbf{x})$, we normalize $\bar{\mathbf{v}}_{wind}(\mathbf{x})\times\hat{\mathbf{j}}$ obtaining a unit vector $\mathbf{s}(\mathbf{x})$ and modify the wind to $\bar{\mathbf{v}}_{wind}(\mathbf{x})+h(||\mathbf{x}-(\mathbf{x}\cdot\hat{\mathbf{j}})\hat{\mathbf{j}}||)\mathbf{s}(\mathbf{x})$ before rescaling to preserve the original magnitude.

We incorporate the effects of wind on tree branches using a pressure-based force as in [48], [49]. First, we compute $\mathbf{v}_{rel} = \mathbf{v}_{wind} - \mathbf{v}_{body}$ where both $\mathbf{v}_{wind}$ and $\mathbf{v}_{body}$ are evaluated at the rigid body's center of mass. Since we only consider the component of wind not along the direction of the branch, we compute $\mathbf{v}_{rel,n} = \mathbf{v}_{rel} - (\mathbf{v}_{rel}\cdot\mathbf{t})\mathbf{t}$ where $\mathbf{t}$ is the axial direction of the branch rigid body. Then the wind force we apply to each branch rigid body is

$$\mathbf{f}_{wind} = \kappa\rho A_c||\mathbf{v}_{rel,n}||\mathbf{v}_{rel,n} \tag{21}$$

where A_c is the longitudinal cross-sectional area of the conical frustum, ρ is the density of air, and κ is a scaling parameter. For fruits, there is no preferred direction, so we instead use

$$\mathbf{f}_{wind} = \kappa\rho A_s||\mathbf{v}_{rel}||\mathbf{v}_{rel} \tag{22}$$

where A_s is the circular cross-sectional area of the fruit's proxy sphere.

8 Leaves

Leaves are included among the tree's articulated rigid bodies, as opposed to being one-way coupled, since the accumulated force and torque (due to gravity, wind, and potentially collisions) over all the leaves can have a significant impact on branch motion. We allow leaves (or petals) to fall by detaching them from their parent branches when the torque on the parent joint exceeds a threshold (see Figures 9 and 10).

Leaves exhibit interesting fluttering, tumbling, and spiraling motion, as well as chaotic motion in between these phases as discussed in [50], [51]. The phenomenological approach of [52] estimates the force and torque applied to a thin disk using Kutta-Joukowski's theorem to compute the lift force and moment while adding friction forces for the normal and tangential wind directions. They solve their

Fig. 10: Petals fall from a cherry blossom tree and are carried away by the wind (247k rigid bodies, 0.3 sec/frame).

ordinary differential equations using a fourth-order Runge-Kutta method and obtain in-plane fluttering and tumbling motion; however, they are not able to model spiraling since their method is restricted to two spatial dimensions. A subsequent comment [53], a corrigendum [54], and further revisions to the model [55], [56] (the latter backed by experiments) indicate that accurately modeling falling thin disks or squares may still be unsettled science. Alternatively, one could take a data-driven approach as in [57] or even randomly sample prescribed spline curves as in [58]. Motivated by these aforementioned works and the restrictions imposed by real-time constraints as well as the inaccuracies incurred by taking forward Euler time steps at 30Hz, we propose a related technique that seems to work well in practice.

Let the normal and tangential components of the wind velocity relative to a leaf be $\mathbf{v}_{rel,n} = (\mathbf{v}_{rel} \cdot \mathbf{n})\mathbf{n}$ and $\mathbf{v}_{rel,t} = \mathbf{v}_{rel} - \mathbf{v}_{rel,n}$ where $\mathbf{n}$ is the leaf's normal. Then, we propose a wind force and torque given by

$$\mathbf{f}_{wind} = \kappa_l \rho A_t ||\mathbf{v}_{rel}|| \mathbf{v}_{rel,n} + \nu m(\mathbf{v}_{rel,n} + \mathbf{v}_{rel,t}/\xi) \quad (23)$$

$$\boldsymbol{\tau}_{wind} = \kappa_l \rho \frac{A_t}{2} \sqrt{\frac{A_t}{\pi}} (\mathbf{v}_{rel} \cdot \mathbf{n}) \mathbf{n}^* (\mathbf{v}_{rel} \cos\varphi + \mathbf{n}^* \mathbf{v}_{rel} \sin\varphi) - \nu I \boldsymbol{\omega} \quad (24)$$

where κ_l is a scale factor, A_t is the area of the leaf's proxy triangle, m is the leaf's mass, ν is an air resistance multiplier, and ξ is the ratio of the normal drag coefficient to the tangential drag coefficient. The second terms on the right hand sides of Equations 23 and 24 are taken directly from [52]. The first term on the right hand side of Equation 23 is similar in spirit to that in [52] but more along the lines of Equation 21 except that we use $||\mathbf{v}_{rel}||$ instead of $||\mathbf{v}_{rel,n}||$ since the former seems to give more visually rich fluttering motion. The first term in Equation 24 reduces to that in [52] when $\varphi = 0$. Nonzero values of the parameter φ give an azimuthal offset angle for the torque direction that allows for three-dimensional spiraling motion.

9 Real-time Animation and Posing System

Physical simulation has been prevalent for decades as a tool for realistically modeling natural phenomena such as cloth [59], [60], smoke [61], explosions [62], water [63], etc. However, control of such phenomena often remains problematic. One of the major benefits of real-time simulation, see e.g. [64], is the ability to quickly, iteratively tune parameters in order to obtain desirable motions for the many degrees of freedom. Our tree modeling algorithm lends itself well to such an approach. In fact, our algorithm scales so well that we incorporated it into the Unity game engine and ran it interactively on a low-budget tablet (see Figure 11).

The ability to tune a tree's internal dynamics is particularly important, since the stiffness and damping of the joint springs are essential for achieving realistic motion. Thus, we created an interactive interface that allows for the manipulation of spline curves via control points in order to set the stiffness and damping values for the parent joint of a tree branch based on the branch's radius. Editing these splines allows one to fine-tune the extent of bending and vibration in response to gravity, wind, collisions, etc. In order to further fine-tune local regions of the tree, we also allow the user to interactively add splines to any branch and subsequently use those splines to override the stiffness/damping values of that rigid body's entire subtree. See Figure 12.

In addition to the controls for interactively tuning the parameters of the wind grid discussed in Section 7, our user interface exposes controls for gravity, branch and fruit density, multipliers for the global stiffness and damping of branches and leaves, etc. The user can also select and pull on rigid bodies in the tree. In fact, we allow the user to apply forces to edit the tree and change its rest state. For leaves on branches, we can readily set different stiffnesses

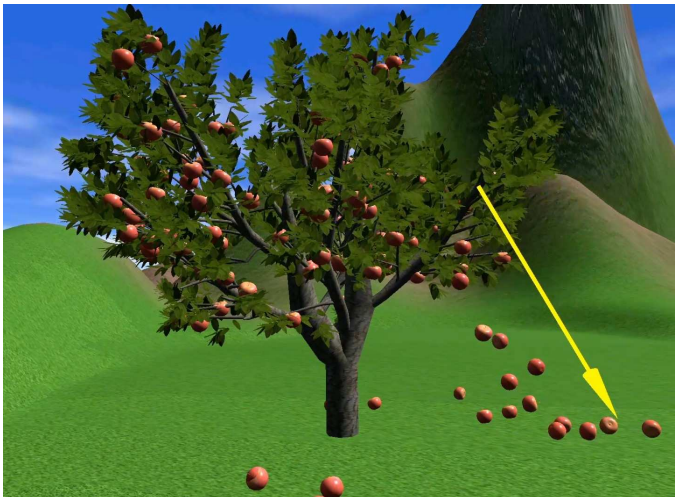


Fig. 11: Our method runs interactively on a low-budget tablet. Here, a user exerts a force on a branch via touch (visualized by the yellow arrow).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2017.2661308, IEEE Transactions on Visualization and Computer Graphics

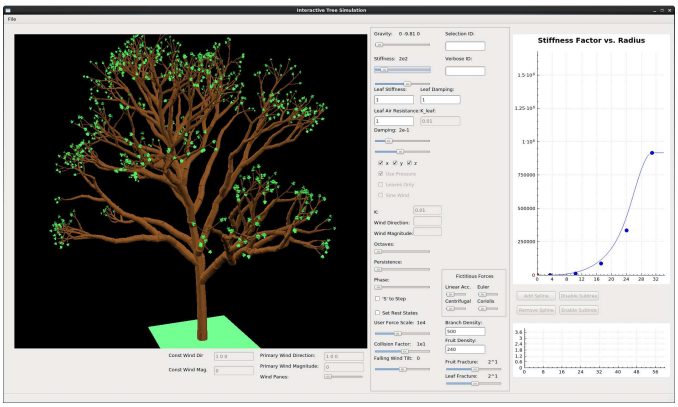


Fig. 12: Our interactive posing and animation interface can be used to add, edit, and remove splines for stiffness and damping to achieve a desired tree motion. The interface also exposes controls for gravity, wind, fracture thresholds, branch density, etc.

for their flapping, swaying, and twisting motions, since we orient each leaf's width, normal, and length directions to be aligned with the x, y, and z-axes of its parent joint frame in its rest pose simply by carefully choosing the orientation of the parent joint frame. We also expose the parameters discussed in Section 8 for tuning the motion of falling leaves.

While the interface can run in real time for trees with ten thousand branches, our system is also useful for the interactive tuning of trees that are too big to be simulated in real time. The interface supports the tuning of large trees by exposing controls to selectively simulate only a subset of the branches. In practice, this has allowed us to interactively tune trees with over three million articulated rigid bodies, which subsequently run offline at speeds of around two seconds per frame. See Figure 13.

10 Examples

Data for tree models can be obtained in various ways. Procedural modeling techniques [16], [27] can be used to generate an articulated rigid body structure directly. Dedicated modeling tools such as SpeedTree, Xfrog, and GrowFX[2] can be used to produce articulated rigid bodies, e.g. by converting a set of splines and thicknesses representing a tree into a set of connected rigid bodies. While one could model a tree using only a single rigid body per tree branch, in practice we segment branch lengths into multiple rigid bodies to enable sub-branch bending. For rendering, we construct a skin mesh of a tree's articulated rigid body such that adjacent rigid bodies do not appear disjoint and texture coordinates vary smoothly.

Our algorithm is fast enough to support interactive tree simulation on a mobile device. We demonstrate this by integrating the algorithm with the Unity game engine (see Figure 11). With this integration, trees can be tuned interactively using the interface described in Section 9, and then simulated on a tablet once the desired parameters have been found. More complex trees are simulated offline. Some of our larger trees are shown in Figures 5, 9, 10, and

2. `http://speedtree.com`, `http://xfrog.com`, and `https://exlevel.com`

Fig. 13: Our largest tree is composed of over 3 million articulated rigid bodies and acts as a stress test running at 2.3 sec/frame.

13. Figure 1 shows a larger scene containing many of our simulated trees. Within that scene, a squirrel runs along the branches between two trees (see Figure 15), applying a force that bends the branches.

We compare our method to the implementation of Featherstone's articulated-body algorithm provided by RBDL [65] with forward Euler time integration. While the explicit method can take frame-rate time steps for a small number of bodies, increasingly more substeps are required as the number of bodies and/or the stiffness of the bodies' joints increases. As expected, our algorithm does not produce the same motion as the explicit algorithm for the same set of stiffness and damping values; the strength of our method is that it can be used to tune parameters to produce a desired motion while still taking frame-rate time steps regardless of the number of rigid bodies or the magnitude of their joint stiffnesses. See Figure 14.

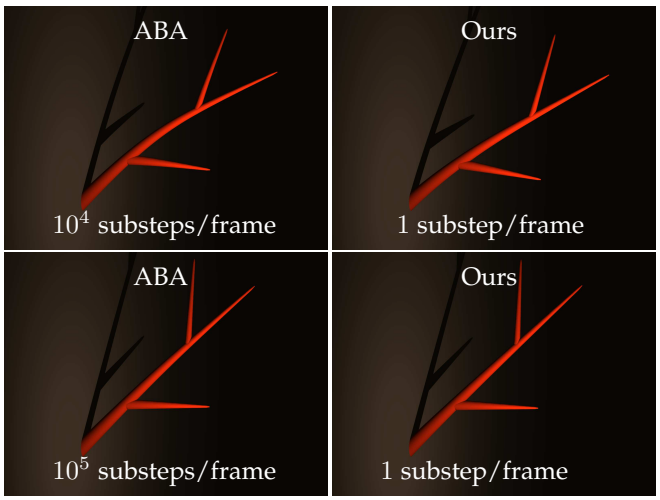


Fig. 14: (Top) A branch sags under gravity using Featherstone's articulated-body algorithm (ABA) and our method. (Bottom) The test is repeated with greater stiffnesses, requiring a greater number of substeps per frame.

Fig. 15: A squirrel runs along the branch of a tree, leaps, and lands on the branch of another tree, gently perturbing both.

11 Limitations and Future Work

There are several avenues along which our proposed method can be extended and improved. In our opinion, the primary drawback of the method is that it does not consider two-way coupling between wind and trees. As such the method can produce quite realistic global motions, but the trees can sometimes fall into the uncanny valley when observing inter-branch motion. We feel that improving the wind model to include branch opacity and tree porosity as well as two-way coupling would yield the most marked gains in realism. It is also important to consider how various trees may shield each other from the wind.

Our stiffness model might be improved by taking additional factors into consideration, e.g. increasing the stiffness of joints based on their topology so that trees are stiffer in areas near bifurcations. Our force-based interaction model could be further expanded to incorporate effects such as rain hitting leaves or snow accumulating on branches. Various level of detail considerations could be taken into account in order to simulate large numbers of trees in a scene, e.g. by taking larger-than-framerate time steps or by evolving only a subset of the springs in each tree using an approach like that of [66]. Collision handling could be improved by considering an impulse-based treatment of the articulated rigid bodies that compose a tree. This would require one to ascertain an impulse factor that models the response of our tree evolution algorithm.

The other primary limitation to obtaining realism is that real trees appear much more visually rich than any of the models we have been able to create. Thus, it would be interesting to use data capture technologies and computer vision techniques to model real trees for use in our simulations. Progress has already been made in this direction via the techniques of [67], [68] and the aforementioned [20]. One might obtain parameters for the real tree using an approach like that of [69] to observe the vibration of branches in response to the wind. The combination of highly realistic models obtained in such ways with our simulation algorithm would be an intriguing line of future work.

Acknowledgements

Research supported in part by ONR N00014-13-1-0346, ONR N00014-11-1-0707, ONR N-00014-11-1-0027, and ARL AHPCRC W911NF-07-0027. E.Q. was supported in part by an NDSEGF. Y.Y. was supported in part by a Stanford Graduate Fellowship. Computing resources were provided in part by ONR N00014-05-1-0479.

References

[1] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff, "Position based dynamics," *J. Visual Comm. and Image Repr.*, vol. 18, no. 2, pp. 109–118, 2007.

[2] R. Featherstone, *Robot Dynamics Algorithms*. Kluwer Academic Publishers, Boston/Dordrecht/Lancaster, 1987.

[3] D. K. Pai, "Strands: Interactive simulation of thin solids using cosserat models," in *Proc. of Eurographics*, ser. Comput. Graph. Forum, vol. 21. Eurographics Assoc., 2002, pp. 347–352.

[4] F. Bertails, B. Audoly, M.-P. Cani, B. Querleux, F. Leroy, and J.-L. Lévêque, "Super-helices for predicting the dynamics of natural hair," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1180–1187, 2006.

[5] A. McAdams, A. Selle, K. Ward, E. Sifakis, and J. Teran, "Detail preserving continuum simulation of straight hair," in *Proc. SIGGRAPH 2009*, 2009, pp. 385–392.

[6] F. Bertails, "Linear time super-helices," in *Computer graphics forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 417–426.

[7] S. Hadap, "Oriented strands: dynamics of stiff multi-body system," in *Proc. of the 2006 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.* Eurographics Association, 2006, pp. 91–100.

[8] J. Barbič and D. James, "Real-time subspace integration of St. Venant-Kirchhoff deformable models," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 24, no. 3, pp. 982–990, 2005.

[9] J. Barbič and Y. Zhao, "Real-time large-deformation substructuring," *ACM TOG (SIGGRAPH 2011)*, vol. 30, no. 4, pp. 91:1–91:7, 2011.

[10] D. Baraff, "Linear-time dynamics using Lagrange multipliers," in *Proc. of ACM SIGGRAPH 1996*, 1996, pp. 137–146.

[11] R. Weinstein, J. Teran, and R. Fedkiw, "Dynamic simulation of articulated rigid bodies with contact and collision," *IEEE TVCG*, vol. 12, no. 3, pp. 365–374, 2006.

[12] F. Hernández, C. Garre, R. Casillas, and M. A. Otaduy, "Linear-time dynamics of characters with stiff joints," in *V Ibero-American Symp. in Comput. Graph. (SIACG 2011)*, 2011.

[13] D. L. Michels, G. A. Sobottka, and A. G. Weber, "Exponential integrators for stiff elastodynamic problems," *ACM Trans. Graph.*, vol. 33, no. 1, p. 7, 2014.

[14] D. Brydon, J. Pearson, and M. Marder, "Solving stiff differential equations with the method of patches," *J. Comput. Phys.*, vol. 144, no. 2, pp. 280–298, 1998.

[15] P. Prusinkiewicz, M. Hammel, J. Hanan, and R. Měch, "Visual models of plant development," in *Handbook of formal languages*. Springer, 1997, pp. 535–597.

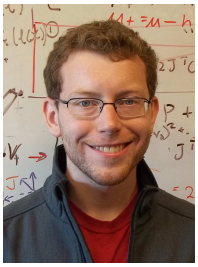
[16] J. Weber and J. Penn, "Creation and rendering of realistic trees," in *Proc. 22nd Ann. Conf. Comput. Graph. Int. Tech.* ACM, 1995, pp. 119–128.

[17] S. Pirk, O. Stava, J. Kratt, M. A. M. Said, B. Neubert, R. Měch, B. Benes, and O. Deussen, "Plastic trees: Interactive self-adapting botanical tree models," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 50:1–50:10, 2012.

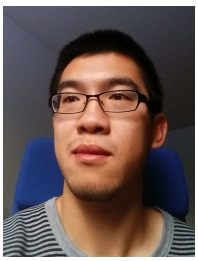
[18] P. de Reffye, C. Edelin, J. Françon, M. Jaeger, and C. Puech, "Plant models faithful to botanical structure and development," in *Proc. of the 15th Ann. Conf. Comput. Graph. Int. Tech.*, ser. SIGGRAPH '88. ACM, 1988, pp. 151–158.

[19] R. Měch and P. Prusinkiewicz, "Visual models of plants interacting with their environment," in *Proc. of the 23rd Annual Conf. on Comput. Graph. and Interactive Techniques*. ACM, 1996, pp. 397–410.

[20] Y. Livny, F. Yan, M. Olson, B. Chen, H. Zhang, and J. El-sana, "Automatic reconstruction of tree skeletal structures from point clouds," *Proc. SIGGRAPH Asia 2010*, vol. 29, pp. 151:1–151:8, 2010.
[21] J. Diener, L. Reveret, and E. Fiume, "Hierarchical retargetting of 2d motion fields to the animation of 3d plant models," in *Proc. of the 2006 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, ser. SCA '06, 2006, pp. 187–195.
[22] C. Li, O. Deussen, Y.-Z. Song, P. Willis, and P. Hall, "Modeling and generating moving trees from video," in *Proc. of SIGGRAPH Asia 2011*, 2011, pp. 127:1–127:12.
[23] B. Neubert, T. Franken, and O. Deussen, "Approximate image-based tree-modeling using particle flows," *ACM Trans. Graph.*, vol. 26, no. 3, p. 88, 2007.
[24] P. Tan, T. Fang, J. Xiao, P. Zhao, and L. Quan, "Single image tree modeling," in *ACM SIGGRAPH Asia 2008 Papers*, 2008, pp. 108:1–108:7.
[25] D. Bradley, D. Nowrouzezahrai, and P. Beardsley, "Image-based reconstruction and synthesis of dense foliage," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 74:1–74:10, 2013.
[26] X. Chen, B. Neubert, Y.-Q. Xu, O. Deussen, and S. B. Kang, "Sketch-based tree modeling using Markov random field," in *ACM SIGGRAPH Asia 2008 Papers*, 2008, pp. 109:1–109:9.
[27] S. Longay, A. Runions, F. Boudon, and P. Prusinkiewicz, "Treesketch: Interactive procedural modeling of trees on a tablet," in *Proc. Int. Symp. Sketch-Based Interf. Modeling*, 2012, pp. 107–120.
[28] Y. Akagi and K. Kitajima, "Computer animation of swaying trees based on physical simulation," *Computers & Graphics*, vol. 30, no. 4, pp. 529–539, 2006.
[29] X.-y. Hu, W.-m. Tao, and Y.-m. Guo, "Using FEM to predict tree motion in a wind field," *J. Zhejiang Univ. Sci. A*, vol. 9, no. 7, pp. 907–915, 2008.
[30] M. Shinya and A. Fournier, "Stochastic motion—motion under the influence of wind," in *Comput. Graph. Forum*, vol. 11, no. 3. Wiley Online Library, 1992, pp. 119–128.
[31] R. Habel, A. Kusternig, and M. Wimmer, "Physically guided animation of trees," in *Comput. Graph. Forum*, vol. 28, no. 2, 2009, pp. 523–532.
[32] J. Diener, M. Rodriguez, L. Baboud, and L. Reveret, "Wind projection basis for real-time animation of trees," in *Comput. Graph. Forum*, vol. 28, no. 2, 2009, pp. 533–540.
[33] M. Yang, M.-c. Huang, and E.-h. Wu, "Physically-based tree animation and leaf deformation using CUDA in real-time," in *Trans. Edutainment VI*, ser. LNCS, 2011, vol. 6758, pp. 27–39.
[34] J. Stam, "Stochastic dynamics: Simulating the effects of turbulence on flexible structures," in *Comput. Graph. Forum*, vol. 16, no. s3. Wiley Online Library, 1997.
[35] S. Pirk, T. Niese, T. Hädrich, B. Benes, and O. Deussen, "Windy trees: Computing stress response for developmental tree models," *ACM Trans. Graph.*, vol. 33, no. 6, pp. 204:1–204:11, 2014.
[36] Y. Zhao and J. Barbič, "Interactive authoring of simulation-ready plants," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 84:1–84:12, 2013.
[37] Y. Zhao, "Plant substructuring and real-time simulation using model reduction," Ph.D. dissertation, University of Southern California, 2014.
[38] Y. Li and J. Barbič, "Stable orthotropic materials," *ACM/Eurographics Symp. on Comput. Anim.*, 2014.
[39] J.-M. Aubry and X. Xian, "Fast implicit simulation of flexible trees," in *Math. Progress in Expressive Image Synthesis II*. Springer, 2015, pp. 47–61.
[40] T. Sakaguchi and J. Ohya, "Modeling and animation of botanical trees for interactive virtual environments," in *Proc. of the ACM Symp. on Virt. Reality Software and Tech.* ACM, 1999, pp. 139–146.
[41] M. Yang, L. Jiang, X. Li, Y. Liu, X. Liu, and E. Wu, "Interactive coupling between a tree and raindrops," *Comput. Anim. Virt. Worlds*, vol. 23, no. 3-4, pp. 267–277, 2012.
[42] S. Jeong, S.-H. Park, and C.-H. Kim, "Simulation of morphology changes in drying leaves," *Comput. Graph. Forum*, vol. 32, no. 1, pp. 204–215, 2013.
[43] M. Müller and N. Chentanez, "Solid simulation with oriented particles," *ACM TOG*, vol. 30, no. 4, pp. 92:1–92:10, 2011.
[44] S. Tomažič and S. Stančin, "Simultaneous orthogonal rotation angle," *Electrotech. Rev*, vol. 78, pp. 7–11, 2011.
[45] S. Stančin and S. Tomažič, "Angle estimation of simultaneous orthogonal rotations from 3d gyroscope measurements," *Sensors*, vol. 11, no. 9, pp. 8536–8549, 2011.
[46] E. Guendelman, R. Bridson, and R. Fedkiw, "Nonconvex rigid bodies with stacking," *ACM TOG*, vol. 22, no. 3, pp. 871–878, 2003.
[47] J. J. Craig, *Introduction to robotics: mechanics and control*, 3rd ed. Pearson Prentice Hall Upper Saddle River, 2005.
[48] J. Wejchert and D. Haumann, "Animation aerodynamics," *Comput. Graph.*, vol. 25, no. 4, pp. 19–22, 1991.
[49] M. Lentine, J. T. Gretarsson, C. Schroeder, A. Robinson-Mosher, and R. Fedkiw, "Creature control in a fluid environment," *IEEE TVCG*, vol. 17, no. 5, pp. 682–693, 2011.
[50] H. Zhong, S. Chen, and C. Lee, "Experimental study of freely falling thin disks: Transition from planar zigzag to spiral," *Physics of Fluids*, vol. 23, no. 1, p. 011702, 2011.
[51] F. Auguste, J. Magnaudet, and D. Fabre, "Falling styles of disks," *J. Fluid Mech.*, vol. 719, pp. 388–405, 3 2013.
[52] Y. Tanabe and K. Kaneko, "Behavior of a falling paper," *Phys. Rev. Lett.*, vol. 73, no. 10, pp. 1372–1375, 1994.
[53] L. Mahadevan, H. Aref, and S. Jones, "Comment on behavior of a falling paper," *Phys. Rev. Lett.*, vol. 75, no. 7, p. 1420, 1995.
[54] Y. Tanabe and K. Kaneko, "Tanabe and Kaneko reply," *Phys. Rev. Lett.*, vol. 75, no. 7, p. 1421, 1995.
[55] A. Belmonte, H. Eisenberg, and E. Moses, "From flutter to tumble: inertial drag and froude similarity in falling paper," *Phys. Rev. Lett.*, vol. 81, no. 2, p. 345, 1998.
[56] A. Andersen, U. Pesavento, and Z. J. Wang, "Unsteady aerodynamics of fluttering and tumbling plates," *J. Fluid Mech.*, vol. 541, pp. 65–90, 2005.
[57] T. Martin, N. Umetani, and B. Bickel, "OmniAD: Data-driven omni-directional aerodynamics," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 113:1–113:12, 2015.
[58] C. Li, J. Qian, R. Tong, J. Chang, and J. Zhang, "GPU based real-time simulation of massive falling leaves," *Comput. Visual Media*, vol. 1, no. 4, pp. 351–358, 2015.
[59] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically deformable models," *Comput. Graph. (Proc. SIGGRAPH 87)*, vol. 21, no. 4, pp. 205–214, 1987.
[60] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *Proc. SIGGRAPH 1998*, 1998, pp. 43–54.
[61] J. Stam, "Stable fluids," in *Proc. of SIGGRAPH 99*, 1999, pp. 121–128.
[62] G. Yngve, J. O'Brien, and J. Hodgins, "Animating explosions," in *Proc. of ACM SIGGRAPH 2000*, 2000, pp. 29–36.
[63] D. Enright, S. Marschner, and R. Fedkiw, "Animation and rendering of complex water surfaces," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 21, no. 3, pp. 736–744, 2002.
[64] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim, "Unified particle physics for real-time applications," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 33, no. 4, pp. 153:1–153:12, 2014.
[65] M. L. Felis, "RBDL: an efficient rigid-body dynamics library using recursive algorithms," *Autonomous Robots*, pp. 1–17, 2016.
[66] S. Redon, N. Galoppo, and M. Lin, "Adaptive dynamics of articulated bodies," *ACM TOG*, vol. 24, no. 3, pp. 936–945, 2005.
[67] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan, "Image-based tree modeling," in *ACM Trans. Graph.*, vol. 26, no. 3. ACM, 2007, p. 87.
[68] K. Xie, F. Yan, A. Sharf, O. Deussen, B. Chen, and H. Huang, "Tree modeling with real tree-parts examples," *IEEE TVCG*, vol. 22, no. 12, pp. 2608–2618, Dec 2015.
[69] A. Davis, K. L. Bouman, J. G. Chen, M. Rubinstein, F. Durand, and W. T. Freeman, "Visual vibrometry: Estimating material properties from small motions in video," in *2015 IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 5335–5343.



Ed Quigley received his B.S. in Computer Science from Grove City College in 2013. He has interned at SURVICE Engineering and Pilot AI Labs, and is currently pursuing his Ph.D. at Stanford University where he is supported by an NDSEG Fellowship. His research interests include physically-based simulation, animation, and 3D reconstruction.



Yue Yu received his B.S. in Computer Science and Mathematics from the Hong Kong University of Science and Technology in 2011. He is currently pursuing a Ph.D. at Stanford University where he was awarded a Stanford Graduate Fellowship. He was a Technology Intern at Walt Disney Animation Studios. He has coauthored four papers published in computer graphics and computational physics. His current research interest is in high order accurate cut-cell solid-fluid coupling algorithms.

Jingwei Huang received his bachelor's degree from the School of Software, Tsinghua University, Beijing, China. He is currently pursuing his Ph.D. in the Department of Computer Science at Stanford University. His research interests include physically-based simulation, image processing, and 3D reconstruction.

Winnie Lin received her B.S. in Mathematics from Stanford University in 2016, and has worked as an undergraduate research assistant in computational geometry and computer graphics. She is currently pursuing her M.S. in Computer Science, also at Stanford University.

Ron Fedkiw received his Ph.D. in Mathematics from UCLA in 1996 and did postdoctoral studies both at UCLA in Mathematics and at Caltech in Aeronautics before joining the Stanford Computer Science Department. He was awarded an Academy Award from The Academy of Motion Picture Arts and Sciences, the National Academy of Science Award for Initiatives in Research, a Packard Foundation Fellowship, a Presidential Early Career Award for Scientists and Engineers (PECASE), a Sloan Research Fellowship, the ACM Siggraph Significant New Researcher Award, an Office of Naval Research Young Investigator Program Award (ONR YIP), the Okawa Foundation Research Grant, the Robert Bosch Faculty Scholarship, the Robert N. Noyce Family Faculty Scholarship, two distinguished teaching awards, etc. He has served on the editorial board of the Journal of Computational Physics, Journal of Scientific Computing, SIAM Journal on Imaging Sciences, and Communications in Mathematical Sciences, and he participates in the reviewing process of a number of journals and funding agencies. He has published over 110 research papers in computational physics, computer graphics, and vision, as well as a book on level set methods. For the past fifteen years, he has been a consultant with Industrial Light + Magic. He received screen credits for his work on "Terminator 3: Rise of the Machines," "Star Wars: Episode III – Revenge of the Sith," "Poseidon," and "Evan Almighty."