Generating and Estimating Linear FEM Plant Simulation Models

Shaoxiong Yao Kris Hauser

***Abstract*— Developing agricultural simulation software has the potential to speed up the development of robotic systems, but it presents significant challenges due to the difficulty in simulating plants. Plants are deformable objects with complex geometries, and existing simulation software either cannot handle deformable structures or is inefficient and unstable to run. In this paper, we present a system that generates diverse plant geometries and simulates plant deformation using a linear Finite Element Method (FEM). Our simulator efficiently computes quasi-static plant deformation by leveraging a linear model. Additionally, we propose a model parameter estimation technique, which we validate through simulation data, further enhancing the system's applicability to agricultural robotics. Our simulation code is available at: `https://github.com/ShaoxiongYao/visual-tactile-simulate`**

I. Introduction

The increasing global demand for agricultural products, coupled with persistent labor shortages, motivates the need for agricultural automation using robots [1]. Robotic simulation software, which enables low-cost and efficient evaluation across various robotics applications, is promising to accelerate the development of agricultural robots. However, developing simulation software for agricultural environments presents unique challenges, particularly because plants, such as orchard trees, are deformable objects with complex geometries. Existing robotics simulation tools, such as Bullet [2] and DART [8], are commonly used for modeling rigid objects but are not well-suited for simulating deformable structures like plants. While Isaac Sim [9] provides an API for finite element method (FEM) simulations, it gets slow and unstable to simulate large tetrahedral meshes.

Plant simulation has been extensively studied in the computer graphics domain, with a primary focus on achieving visual realism, though these methods are typically slow to run [13]. While real-time simulation techniques exist, they often involve tedious setup procedures for creating plant simulation models [10]. In the context of robotics, however, we need software that can be quickly set up and executed with both efficiency and stability. To validate the performance of agricultural robots, we would like to automatically generate diverse plant simulation models. The simulator should be accessible and lightweight, with minimal dependencies to facilitate ease of use. To ensure that evaluation results align closely with real-world outcomes, we also need to calibrate the simulation model parameters using real-world data. More recently, Gazebo-based plant simulators [3] have enabled parameter estimation using real-world data, but they require a tedious manual setup process. An Isaac Gym-based simulator utilizes Bayesian learning to infer model parameters; however, this approach demands the implementation of complex learning and inference functions [7]. Yao and Hauser [12] introduced a method for efficiently estimating the stiffness of artificial plants in real-time, but it is limited to predicting tactile responses. In contrast, our approach simplifies the estimation problem by reducing it to a linear observation with respect to FEM parameters.

S. Yao and K. Hauser are with the University of Illinois at Urbana-Champaign, IL, USA.

Fig. 1: Examples of generated plant geometries and simulated deformation under controlled points displacements. The black wireframe is the rest undeformed shape and the gray triangle mesh shows the deformed shape. The red arrows indicate controlled displacement.

In summary, we present a lightweight system to simulate and estimate the plant simulation model. We generate diverse plant geometries and simulate the quasi-static linear FEM plant deformation model efficiently at 40 FPS. Our estimation system accurately calibrates FEM parameters using contact forces and vertex displacements, leading to reduced prediction errors.

II. Method

Our system automatically generates finite element method (FEM) simulation models for plants through three steps as illustrated in Fig. 2. First, it employs a parametric procedural plant generator to create the plant geometry as a triangle mesh, using the open-source package tree-gen [5]. Next, a tetrahedral meshing tool [6] is used to convert the triangle mesh into a tetrahedral mesh, allowing for FEM-based deformation simulations. Additionally, we assign material parameters to each tetrahedral element, defining its elastic behavior. Finally, the system incorporates a lightweight, pure-Python linear FEM simulator that computes plant deformations in response to external forces or displacements. This simulator has minimal dependencies, requiring only PyTorch and SciPy for basic operations.

a) Linear FEM simulation: We represent the tetrahedral mesh of the plant as $\mathcal{M}$, consisting of N vertices and M tetrahedrons. Each vertex p_i^t has a displacement given

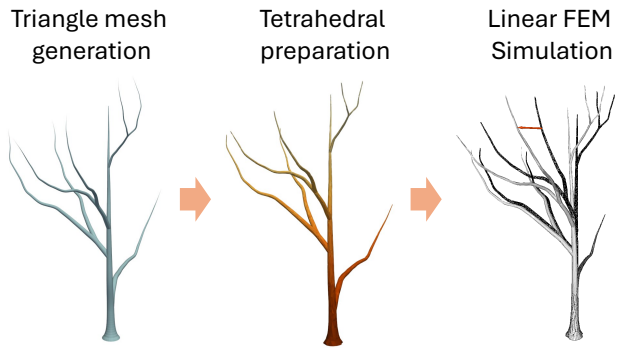

Fig. 2: Overview of the plant simulation system. The initial triangle mesh plant geometry is shown in blue and tetrahedral mesh with color shows different material parameters at different elements. The right figure shows the deformed tree shape in gray triangle mesh under the red arrow push displacement and the initial rest shape in the black wireframe.

by $u_i^t = p_i^t - p_i^0$, where p_i^0 denotes the vertex's initial position. We concatenate the displacements of all vertices as $u^t = [u_1^t, \ldots, u_N^t] \in \mathbb{R}^{3N}$. In a linear FEM model, each vertex experiences internal elastic forces $f_{i,\text{int}}^t$ and external contact forces $f_{i,\text{ext}}^t$. The internal elastic forces are related to the vertex displacements by the linear equation $f_{\text{int}}^t = Ku^t$, where K is the stiffness matrix. The stiffness matrix is precomputed using material parameters (e.g. Lamé parameters (μ_j, λ_j)) of each tetrahedron following [11].

The plant contains a set of fixed base vertices, denoted by $B \subset 1, \ldots, N$, and controlled vertices, such as those grasped by a robot gripper, which are represented by $C^t \subset 1, \ldots, N$. The controlled vertices have known displacements $u_{i,c}^t$. We assume that the remaining vertices are free to move without constraints. To compute the equilibrium deformation u^t, we first apply the known displacements to the vertices in the fixed base set B and to the controlled vertices in C^t. Then, for the freely moving vertices, we solve for zero external forces. This approach allows the quasi-static deformation to be computed by solving a linear system of equations.

b) Material parameters estimation: Our material parameter estimation assumes known contact forces at the controlled points C^t and plant deformations u^t from visual observations. The key insight is that, in a linear FEM model, the internal elastic forces are proportional to the Lamé parameters [11]. From force equilibrium, the contact forces at each point are linearly related to the Lamé parameters, i.e., $f_{i,\text{ext}}^t = W_i(u^t)\theta$, where W_i is derived from the material model, and $\theta = [\mu_1, \lambda_1, \ldots, \mu_M, \lambda_M]$ represents the concatenated Lamé parameters for each tetrahedral element. Therefore, we can use a linear estimation problem, minimizing expected force

$$\hat{\theta} = \arg\min_{\theta} \sum_t \sum_{i \in C^t} \|f_{i,ext}^t - W_i(u^t)\theta\|^2.$$

Given this quadratic programming problem, we used CVXPY [4] to find the estimated $\hat{\theta}$.

III. Experimental Results

Qualitative results We visualize several examples of generated plant deformations in Fig. 3. The procedural

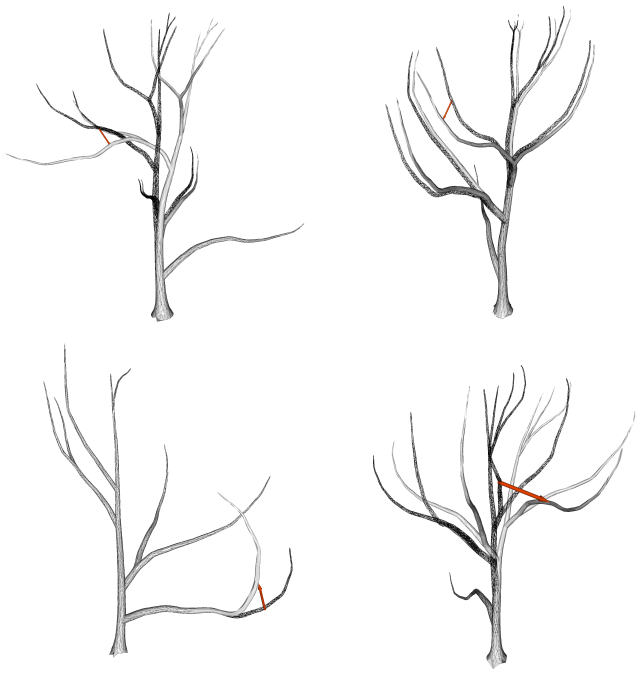

Fig. 3: Plants deformation simulation examples with different tree geometries. The color codes are the same as Fig. 1.

generator effectively produces diverse plant geometries, and our method accurately simulates their deformations under various external contacts.

Quantitative results We also quantitatively evaluate our material estimation formulation on prediction accuracy in Tab. I. First, we sample several training sequences from a test plant and apply the estimation algorithm to determine the material parameters. As the number of touch sequences increases, both the visual and tactile prediction errors decrease, indicating improved accuracy of the estimated material parameters.

Run-time discussion Generating the triangle mesh only takes 0.007s on average using the Blender plugin. Tetrahedral meshing takes 52.5s on average. Computing the stiffness matrix takes ∼10s and simulation each frame takes ∼0.025s.

TABLE I: Visual and tactile prediction error over a query set on a test tree FEM model. Here visual prediction error is the average of the deformation prediction error over all vertices in (cm) and tactile error sums the force prediction error over all contact vertices in (N).

	Visual prediction error (cm)	Tactile prediction error (N)
0-shot	1.61	13.411
1-shot	1.60	13.394
5-shot	1.08	9.330
10-shot	1.07	6.733

IV. Conclusion and Future Work

In this work, we present a novel lightweight system for generating and estimating plant simulation models. We demonstrate diverse plant simulation results and validate the effectiveness of our material parameter estimation algorithm. In the future, we aim to integrate robot models and collision checking into the simulation, as well as use real-world data to create digital twins of actual plants.

REFERENCES

[1] F. Beed, M. Taguchi, B. Telemans, R. Kahane, F. Le Bellec, J.-M. Sourisseau, E. Malézieux, M. Lesueur-Jannoyer, P. Deberdt, J.-P. Deguine *et al.*, "Fruit and vegetables. opportunities and challenges for small-scale sustainable farming," 2021.

[2] E. Coumans, "Bullet physics simulation," *ACM SIGGRAPH 2015 Courses*, p. 7, 2015. [Online]. Available: https://doi.org/10.1145/2776880.2792704

[3] J. Deng, S. Marri, J. Klein, W. Pałubicki, S. Pirk, G. Chowdhary, and D. L. Michels, "Gazebo plants: Simulating plant-robot interaction with cosserat rods," *arXiv preprint arXiv:2402.02570*, 2024.

[4] S. Diamond, E. Chu, and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization, version 0.2," http://cvxpy.org/, May 2014.

[5] C. Hewitt, "Procedural generation of tree models for use in computer graphics," https://github.com/friggog/tree-gen, 2024, accessed: 2024-08-13.

[6] Y. Hu, T. Schneider, B. Wang, D. Zorin, and D. Panozzo, "Fast tetrahedral meshing in the wild," *ACM Trans. Graph.*, vol. 39, no. 4, Jul. 2020. [Online]. Available: https://doi.org/10.1145/3386569.3392385

[7] J. Jacob, T. Bandyopadhyay, J. Williams, P. Borges, and F. Ramos, "Learning to simulate tree-branch dynamics for manipulation," *IEEE Robotics and Automation Letters*, 2024.

[8] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, "DART: Dynamic animation and robotics toolkit," *The Journal of Open Source Software*, vol. 3, no. 22, p. 500, Feb 2018. [Online]. Available: https://doi.org/10.21105/joss.00500

[9] J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox, "Gpu-accelerated robotic simulation for distributed reinforcement learning," in *Conference on Robot Learning*. PMLR, 2018, pp. 270–282.

[10] E. Quigley, Y. Yu, J. Huang, W. Lin, and R. Fedkiw, "Real-time interactive tree animation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, pp. 1717–1727, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:937460

[11] E. Sifakis and J. Barbic, "Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction," in *ACM SIGGRAPH 2012 Courses*, ser. SIGGRAPH '12. New York, NY, USA: Association for Computing Machinery, 2012. [Online]. Available: https://doi.org/10.1145/2343483.2343501

[12] S. Yao and K. Hauser, "Estimating tactile models of heterogeneous deformable objects in real time," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 12 583–12 589.

[13] Y. Zhao and J. Barbič, "Interactive authoring of simulation-ready plants," *ACM Trans. on Graphics (SIGGRAPH 2013)*, vol. 32, no. 4, pp. 84:1–84:12, 2013.