Leaf Deformation Taking Into Account Fluid Flow

Paulo Silva
The University of Tokyo

Yonghao Yue
The University of Tokyo

Bing-Yu Chen
National Taiwan University

Tomoyuki Nishita
The University of Tokyo

***Abstract*—This paper proposes a method for animating leaf structural deformation considering the leaf water content and its change due to evaporation. To achieve this we embed a mass-spring system into a mesh representing the leaf, and couple the mass-spring parameters with a simulation representing the fluid flow and evaporation occurring in the leaf. This chain of events aims to simulate the natural process that connects the leaf structural deformation to the water amount in the cells through the turgor pressure. To the best of our knowledge no existing method considers this phenomenon in the simulation of leaf wilting in computer graphics.**

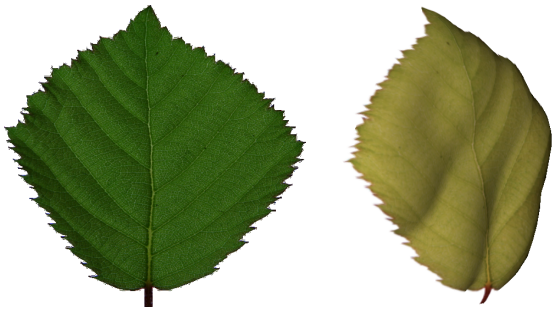
***Index Terms*—Leaf Deformation, Fluid Flow, Aging.**

I. Introduction

Leaves curling and crinkling are commonplace during the cold months of the year, as they are part of most natural sceneries. These this type of natural phenomena are useful in many applications from the movies industry to the interactive and entertainment industry. Therefore modeling leaves is an important topic in computer graphics. Apart from possible static applications, usually it is important to have a time-varying representation of natural phenomena. In this paper we focus on the deformation leaves suffer as they age, dry and die. Our objective is to provide a mechanism that is as automatic as possible by taking as base the actual natural processes occurring in the plants. We believe this to be fundamental for modeling ecosystems realistically in both interactive and non-interactive applications.

While modeling trees and plants has seen quite a lot of attention from the scientific community [1], [2], [3], [4], leaf deformation has been much less focused on. Most current research focuses on the modeling part. By contrast we assume the object is already modeled and we concentrate on animating it.

As the plant ages, changes in its metabolism and in the surrounding environment lead to structural changes in the plant itself. These changes include curling and crinkling, which vary depending on the underlying support structure and on factors such as the amount of fluid in the leaf. It is our opinion that this process needs to be simulated to faithfully capture the deformation leaves undergo. In the method proposed in this paper, we model the changes in internal fluid pressure while taking into account the underlying structural support offered by the leaf venation. We build our method on the work by [5]. In that work the authors propose a model for the fluid flow in leaves. They consider a venation, stomata and fluid maps and use these to simulate the plant transpiration cycle. We take the ideas in that paper for the fluid flow simulation and add our own deformation model on top. Our deformation model is composed by a mass-spring system embedded in a mesh representing the input leaf. This mass-spring system includes springs to deform the leaf, accounting for both shrinking and curling effects due to water loss. Then we couple this simulation with the fluid flow simulation from [5] using their output to control the parameters of our system. Those parameters include spring constants and spring rest lengths. Then by updating this system in a Newtonian physical simulation we obtain an indirect relation between the leaf shape and the fluid pressure in the leaf. This fluid pressure is called turgor pressure and is an important factor in keeping the rigidity of the leaf structure [6]. Additionally we also consider the influence of the leaf venation in the structure support, by connecting it to the spring stiffness constant. The main contribution of this paper is a method of animating leaf deformation based on implicit turgor pressure variation. This paper has the following structure. First we present present some related work in senescence simulation in computer graphics. In section III we present the general idea of our method, and detailed descriptions of the fluid flow simulation, mass-spring model and how it is embedded in the mesh, and how we couple the two simulations. Then, in section IV, we present animations generated by our method and we include some details about our experimental environment. Finally we conclude this paper with some ideas on possible future directions.



(a) (b)

Fig. 1: Leaf deformation example, (a) input image, (b) simulation result.

II. Related Work

There is a considerable amount of work modeling plants, but the amount of work related to leaf deformation is scarce. Here we focus on the works related to leaf deformation that

we find most relevant.

Hong et al. [7] proposed a method based on a venation skeleton and a surface membrane to model the aging leaves. This system can be used to interactively deform the leaf. This contrasts with our method where no user interaction is required apart from specifying the simulation input.

Lu et al. [8] presented a method to animate leaf wilting due to gravity. This method closely relates to the method by [7] as it also uses a venation skeleton to deform the leaf. However, this method only considers the effect of gravity.

Kidder et al. [9] presented a method for simulating fruit senescence, which also includes animation of the fruit skin. Like ours, this method also considers the amount of water content in the deformation of the fruit skin. Nevertheless, each method has a completely different target, as Kidder et al.'s method focuses fruits or volumetric objects, and ours focuses leaves or planar objects.

Recently Xiao and Chen [10] presented a strain based method to explain the deformation of leaves. The author then uses FEM based simulation to produce a simulation of drying leaves. On the other hand, our method considers the effect of the fluid pressure or turgor pressure, which is essential for plants to maintain rigidity [6]. To the best of our knowledge, no other work takes this into consideration in the simulation of leaf structural deformation.

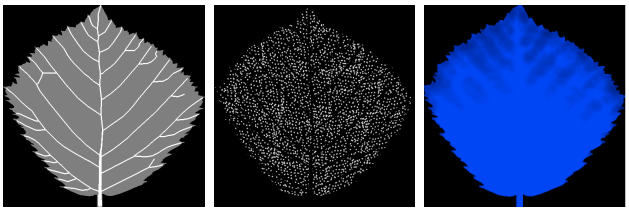
III. Leaf Deformation

Our method aims to reproduce the structural deformation visible in senescent leaves. Leaves come in a wide variety of shapes and sizes. In this work we focus on *angiospermic* and *sheath* leaf types. These leaf types tend to have a more or less flat surface, with an underlying venation that serves as both support to the leaf structure and also has transports channels for nutrients, water, and so on. The fluid flow in the leaf, together with the leaf venation are the two factors that will most influence the leaf shape in our model. To achieve the leaf deformation effect we propose two separate simulations. One simulation that accounts for the fluid flow and evaporation, and a mass-spring based simulation that actually deforms the leaf shape. Both these simulations take into account the amount of fluid and the venation distribution in the leaf. Next we explain how we couple these two simulations and we give detailed explanations on each.

A. Method Overview

The main input to our method is an image of a leaf (see Fig. 1 (a)). First we extract the leaf texture to use as a color map. Together with this map we extract the leaf boundary using computer vision techniques based on [11]. From this boundary we create a mesh representation of the leaf using [12]. This textured mesh is then used in the subsequent simulations by applying extra textures maps and embedding a mass-spring model as explained next.

We consider the leaf has a two dimensional surface on which we define three maps based on [5]. These maps are a fluid map, a stomata map and a venation map. The fluid map represents



(a) (b) (c)

Fig. 2: (a) venation map (b) stomata map, (c) fluid map.

the amount of water at each location in the leaf (see Fig. 2 (c)). Stomata are small openings located mostly on the lower surface of the leaf, which serve the purpose of transpiration, moving water molecules out of the leaf, and also respiration. The stomata map represents the distribution of these openings, and is used to remove water out of the leaf simulating the water lost by transpiration (see Fig. 2 (b)). The water flows in a network of channels called Xylem which is part of a larger vascular network or venation. In order to model the venation distribution we use a venation map. A two dimensional map that represents the venation implicitly by its flow capacity. That is, the amount of water that can flow at each location in the structure (see Fig. 2 (a)). Together with these maps we also need to consider a location where the fluid enters the system. An explanation of this simulation is presented in Sec. III-B.

Additionally to this simulation we add a mass-spring model to the the input mesh. As the fluid map is updated, the spring constants and their rest lengths also change to reflect the new water content in the leaf. More precisely, the as the water content changes at a certain location the springs at that location, if any, will use those values to calculate their parameters. This will cause the springs to contract or extend accordingly, which aims to simulate the changes in the turgor pressure. Then, when the water amount is reduced the plant will, for example, loose rigidity and wilt under its own weight, or contract in case the spring stiffness increase. We update the leaf shape by updating the mass-spring system in a Newtonian physics simulation. This simulation is described in Sec. III-C.

B. Fluid Diffusion

This component of our method uses the work by [5]. For the sake of clarity we make a short explanation of this method here. The fluid flow simulation main data structures are the venation, stomata and fluid maps (see Fig. 2). Then the fluid flow is simulated using a continuous model based on a diffusion process as described by Eq. (1).

$$\frac{\partial y(u,v)}{\partial t} = \nabla \cdot (\beta(u,v)\nabla y(u,v)) + I(u,v) - O(u,v), \quad (1)$$

where $y(u,v)$ represents the fluid map, $\beta(u,v)$ represents the venation map, $I(u,v)$ represents the sources map and $O(u,v)$ represents the stomata map. The continuous model in Eq. (1) is discretized as shown in Eq. (2).

$$-\beta_{i-1}y_{i-1}^{m+1} + (\beta_{i-1} + \beta_i + 1)y_i^{m+1} - \beta_i y_{i+1}^{m+1} = y_i^m. \quad (2)$$

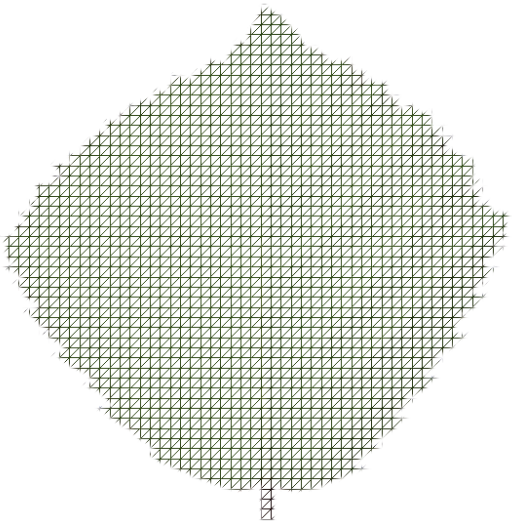


Fig. 3: Example of a leaf mesh used as input in our deformation model.

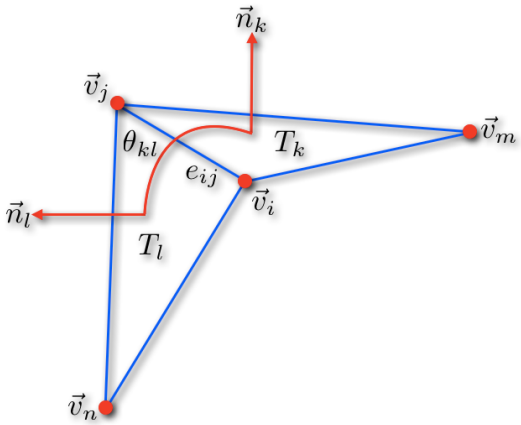


Fig. 4: Schematic of a very simple mesh.

where i is the pixel index and m is the time step index. The author then separates the simulation into three steps. A transpiration step where fluid is extracted from the system, an source step where fluid enters the system and a diffusion step where the fluid is diffused in the system. Then Eq. (2) will lead to a tridiagonal system which can be efficiently solved. Solving this system for y will consequently update the fluid map. For more details we refer the reader to the original paper.

C. Deformation Model

At this stage we assume to have an input mesh such as the one in Fig. 3. We want to deform this mesh by both contracting parts of it or just letting it yield under its own weight. To do this we make use of a mass-spring system which we embed in the mesh. Mass-spring systems are widely used in computer graphics. The interested reader might want to consult [13] for an overview. One of the phenomena that we expect to see from a drying leaf is loss of mass and size. To model this, we start by positioning a mass particle at each mesh vertex. Then we add a spring at each triangle edge. A spring s_{ij} on edge e_{ij}, which connects vertices $\vec{v}_i$ to $\vec{v}_j$ (see Fig. 4), exerts a force $\vec{F}_{ij}$ on particle p_i, located at $\vec{v}_i$, and an opposing force $-\vec{F}_{ij}$ on particle p_j located at $\vec{v}_j$. This force is defined simply by Eq. (3).

$$\vec{F}_{ij} = -k_{ij}(\Delta x_{ij} - l_{ij_0})\vec{u}_{ij}, \tag{3}$$

where k_{ij} is the spring constant, $\Delta x_{ij} = \|\vec{x}_i - \vec{x}_j\|$, $\vec{x}_i$ and $\vec{x}_j$ are the current position of the particles p_i and p_j respectively, l_{ij_0} is the initial spring rest length which is calculated as Δx_{ij} at time $t = 0$ and $\vec{u}_{ij} = (\vec{x}_i - \vec{x}_j)/\|\vec{x}_i - \vec{x}_j\|$. Changing the distance l_{ij_0} we can contract the leaf producing a localized shrinking effect. Changing the spring constant k_{ij} we can also let the leaf mesh stretch under its weight or under external forces. Apart from shrinking, the leaf also tends to curl. To model this behavior, and to lock the orientation between different triangles, we add another type of spring, this time on each shared edge e_{ij}. We call these angle springs, as they use the angle between faces sharing edge e_{ij} in computing the spring force. For this we require the mesh to be homomorfic to a disk. Or in other words, the mesh can only have edges referenced by one face, boundary edges, or edges referenced by two faces, inner edges. We also require the normal orientation to be consistent, as the angle is calculated using the direction of the triangle normal . The force these springs generate is defined by Eq. (4).

$$F_{mn} = k_{\theta_{mn}}(\theta_{mn} - \theta_{mn_0}), \tag{4}$$

where F_{mn} is a scalar which represents the force that the angular spring s_{mn} exerts on particle p_m and p_n. These two particles are located on opposite triangles which share a common edge (see Fig. 4). Additionally $k_{\theta_{mn}}$ is the angular spring constant, θ_{mn} is the current angle between triangles and θ_{mn_0} is the initial rest angle. This force is applied on each particle in the triangle normal direction. When $\theta_{mn} - \theta_{mn_0} < 0$ the the force will be in the direction opposite to the normal. This will force the spring to go back to its rest configuration. Analogously for when $\theta_{mn} - \theta_{mn_0} > 0$. Therefore, the equation has no minus sign. The angular springs are initialized according to Alg. 1. The function $trianglesReferencing$ returns the triangles that share edge e_{ij}, $vertexOpposing$

Algorithm 1 Angular Spring Placement

```
for each inner edge e_ij do
    (T_k, T_l) ← trianglesReferencing(e_ij)
    v_m ← vertexOpposing(T_k, e_ij)
    v_n ← vertexOpposing(T_l, e_ij)
    θ_mn0 = angle(T_k, T_l)
    addSpring(e_ij, v_m, v_n, θ_mn0)
end for
```

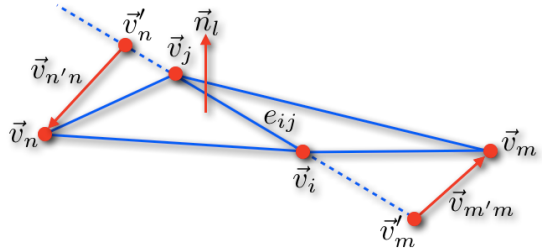


Fig. 5: We project the vertices opposite to edge e_{ij} onto the line that contains e_{ij}. The we calculate the vectors between these projected points and the original. Finally we compute the angle between these vectors. This will be the angle of the angular spring associated with edge e_{ij}.

returns the vertex from face T_k that is not part of edge e_{ij}, and $angle$ calculates the angle between faces T_k and T_l. The function $angle$ is defined in Alg. 2. The function $project$

Algorithm 2 Angular Spring Angle

$\vec{v}'_m \leftarrow \text{project}(\vec{v}_m, e_{ij})$
$\vec{v}'_n \leftarrow \text{project}(\vec{v}_n, e_{ij})$
$\vec{v}_{m'm} \leftarrow \text{normalize}(\vec{v}_m - \vec{v}'_m)$
$\vec{v}_{n'n} \leftarrow \text{normalize}(\vec{v}_n - \vec{v}'_n)$
$\cos(\theta_{mn}) = \vec{v}_{mm} \cdot \vec{v}_{nn}$
$\theta_{mn} \leftarrow \cos^{-1}(\cos(\theta_{mn}))$
$\vec{n}_l = \text{normal}(T_l)$
if $\vec{v}_{m'm} \cdot \vec{n}_l < 0$ **then**
  $\theta_{mn} \leftarrow \theta_{mn} + \pi$
end if
return θ_{mn}

returns the projection of $\vec{v}_m$ on the line defined by e_{ij}, the function $normalize$ normalizes the vector passed as argument and the function $normal$ returns the vector normal to the triangle T_l passed as argument (see Fig. 5). The rest angles of the springs are updated directly by the amount of fluid present at the center point of each spring. For example, for a spring on edge e_{ij} that would be the point halfway between vertices v_i and v_j. Analogously for the angular springs. On the odder hand, the venation map values weight the spring constants. That is, using a normalized version of this map (values $\in [0, 1]$) we can make areas with thicker venation have stiffer spring constants. Finally this mass-spring system is integrated in time, and the change in the fluid map will indirectly cause a change in the leaf structure. This aims to simulate the change in turgor pressure.

IV. Results

This section provides information about our test environment, simulation timings, example deformations and a brief discussion of our method limitations.

TABLE I: Mesh information, number of triangles, edge springs and angular springs for each input mesh.

Name	Vertices	Triangles	Edge Springs	Angular Springs
Bramble	1707	3232	4938	4758

TABLE II: Average time step in milliseconds per update step given the fluid map size. This includes fluid flow simulation and mass-spring simulation.

Name	128×128	256×256
Bramble	11	17
Name	512×512	1024×1024
Bramble	40	145

A. Performance

Our test environment was equipped with an *Intel Core 2 Duo* CPU clocked at $2.13GHz$, $2GB$ of main memory and a *NVIDIA GeForce 9400M* GPU with $256MB$ of dedicated video memory.

We tested our method with several input pictures each at several resolutions. Triangulation parameters were similar in all cases. In these tests the number of triangles, number of edge springs and number of angular springs can be seen in Tbl. I. The timings presented in Tbl. II include the average time it took for a complete update step of both the fluid flow simulation and the mass-spring simulation. The fluid flow simulation has a very big weight at higher resolutions, while for the smaller fluid map resolutions both simulations take about the same time. Note that the triangulation resolution is fixed. Even in our unoptimized implementation we still obtain real-time performance.

B. Example Deformations

Here we show the results of applying our method to a few input meshes. The data for the fluid flow simulation was mostly hand made. The mesh generation however, as with the spring placement, were both done automatically. In Fig. 6 we show a few frames of the deformation the leaf suffers while drying. However we believe our method is versatile enough to be applied in a much larger variety of plants than the few presented here.

C. Limitations

Since our method is based on the method by [5] it does suffer from the same limitations. Our method speed is mostly influenced by the resolution of the fluid map. However we found small maps, such as 256×256, still provide compelling results. The result depends on the triangulation. We found an uniform triangulation to be suited for our purposes. But we believe the triangulation can be improved using the venation map as constraints.

V. Conclusions and Future Work

We presented a method for leaf deformation by implicitly accounting for the turgor pressure. This pressure controls the

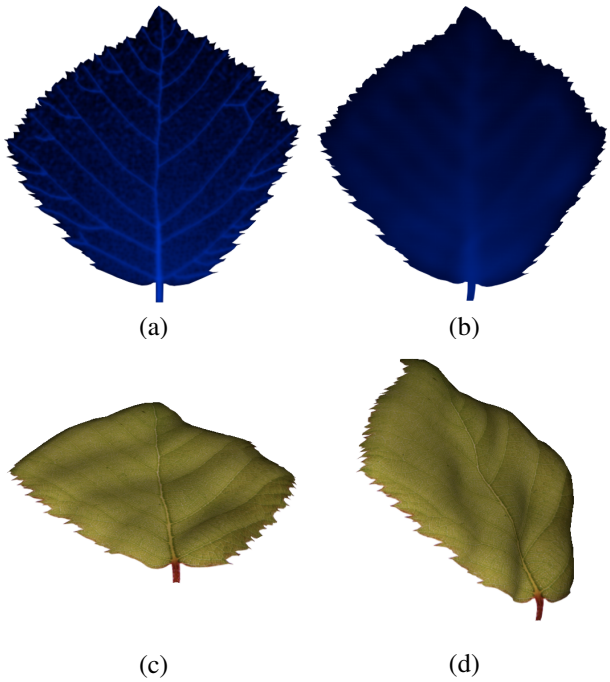


Fig. 6: Above, example of a fluid diffusion sequence for a bramble leaf at time (a) $t = T_0$ and (b) $t = T_1$. Below, the corresponding deformation (c) and (f) respectively.

rigidity in plants [6], and we think it's important to take it into consideration. To model this we used leveraged the work by citeSilva12 which accounts for the plant internal fluid flow. On top of this method we introduced a mass-spring model to deform the leaf mesh. The parameters of this mass-spring model are controlled by the fluid flow simulation. In this way, the plant natural transpiration cycle indirectly controls the leaf structure deformation. This is equivalent to indirectly accounting for the influence of the turgor pressure in controlling the leaf structure. We shown this with some results generated by our method.

We leave for future work the usage of the venation as a form of improving the triangulation result. We don't consider this essential, since the mesh might be artist created. But we aim to achieve a fully automatic system, and this step can possibly increase the quality of the generated mesh. Although we the shape is a very important indicator of the stage of senescence in which a leaf is in, other factors also contribute to a more realistic result. Color change is one such factor [14]. We believe simultaneous simulation of leaf structural and color changes is important to achieve a realistic senescence simulation, and therefore that is another important future direction. We think another interesting direction would be to extend this method to generate an animation of an entire plant. That is, to consider other plant structures such as the stalk, roots and so on.

REFERENCES

[1] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan, "Image-based tree modeling," *ACM Trans. Graph.*, vol. 26, 2007.

[2] T. Ijiri, S. Owada, M. Okabe, and T. Igarashi, "Floral diagrams and inflorescences: interactive flower modeling using botanical structural constraints," *ACM Trans. Graph.*, vol. 24, pp. 720–726, 2005.

[3] J. Weber and J. Penn, "Creation and rendering of realistic trees," SIGGRAPH '95, pp. 119–128, 1995.

[4] M. T. Wong, D. E. Zongker, and D. H. Salesin, "Computer-generated floral ornament," SIGGRAPH '98, pp. 423–434, 1998.

[5] P. Silva, Y. Yue, B.-Y. Chen, and T. Nishita, "Simulating plant color aging taking into account the sap flow in the venation," IEVC, 2012.

[6] N. A. Campbell, J. B. Reece, L. A. Urry, M. L. Cain, S. A. Wasserman, P. V. Minorsky, and R. B. Jackson, *Biology*, p. 134. 8 ed., 2008.

[7] S. M. Hong, B. Simpson, and G. V. G. Baranoski, "Interactive venation-based leaf shape modeling: Natural phenomena and special effects," *Comput. Animat. Virtual Worlds*, vol. 16, no. 3-4, pp. 415–427, 2005.

[8] S. Lu, C. Zhao, and X. Guo, "Venation skeleton-based modeling plant leaf wilting," *Int. J. Comput. Games Technol.*, vol. 2009, pp. 1:1–1:8, 2009.

[9] J. T. K. Jr., S. Raja, and N. I. Badler, "Fruit senescence and decay simulation," *Computer Graphics Forum*, vol. 30, no. 2, pp. 257–266, 2011.

[10] H. Xiao and X. Chen, "Modeling and simulation of curled dry leaves," *Soft Matter*, vol. 7, pp. 10794–10802, 2011.

[11] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[12] "CGAL, Computational Geometry Algorithms Library." http://www.cgal.org.

[13] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson, "Physically based deformable models in computer graphics," *Computer Graphics Forum*, vol. 25, no. 4, pp. 809–836, 2000.

[14] P. O. Lim, H. J. Kim, and H. Gil Nam, "Leaf senescence," *Annual Review of Plant Biology*, vol. 58, no. 1, pp. 115–136, 2007.