Robust Skin Simulation in *Incredibles 2*

Ryan Kautzman
Pixar Animation Studios

Gordon Cameron
Pixar Animation Studios

Theodore Kim
Pixar Animation Studios

Figure 1: A skin simulation was performed on this fast-moving raccoon during a key fight scene. The kinematic mesh, particularly the hind legs and the torso, contains frequent self-intersections that cause problems with previous approaches. ©Disney/Pixar

ABSTRACT

Robustly simulating the dynamics of skin sliding over a character's body is an ongoing challenge. Skin can become non-physically "snagged" in curved or creased regions, such as armpits, and create unusable results. These problems usually arise when it becomes ambiguous which kinematic surface the skin should be sliding along. We have found that many of these problems can be addressed by performing 2D ray-tracing over the surface of the mesh. The approach is fast and robust, and has been used successfully in *Incredibles 2*.

ACM Reference Format:
Ryan Kautzman, Gordon Cameron, and Theodore Kim. 2018. Robust Skin Simulation in *Incredibles 2* . In *Proceedings of SIGGRAPH '18 Talks*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3214745.3214793

1 THE PROBLEM

The dynamics of a thin layer of skin sliding along the surface of a character is an important feature of realistic anatomical motion. Many approaches model the skin as cloth, and use existing methods to constrain the cloth to the character's kinematically evolving surface [Kautzman et al. 2012; Milne et al. 2016; Saito and Yuen 2017]. Alternative parameterization-based techniques also exist [Li et al. 2013], but we chose not to take this approach because, among other considerations, it involves non-trivial pipeline changes.

SIGGRAPH '18 Talks, August 12-16, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5820-0/18/08.
https://doi.org/10.1145/3214745.3214793

One of the key ways in which skin simulation differs from cloth simulation is that the character's skin must somehow be physically constrained to the underlying character mesh (i.e. the "kinematic mesh"). While existing collision processing techniques are physically correct when simulating tight-fitting cloth [Tamstorf et al. 2015], skin is attached to the underlying muscle by an adipose layer that must be taken into account.

Inserting a zero-length penalty spring between each vertex on the skin mesh and a corresponding *anchor* position on the kinematic mesh is a natural approach, as it models the actual springiness of the adipose tissue. The main challenge comes when updating the positions of these anchors so that the skin is not over-constrained and can slide in the tangent direction. The most straightforward approach is to apply all the skin forces to the anchor positions, integrate forward in time in the full $\Re^3$ space, and then project the result to the closest point on the kinematic mesh.

We call this the "projection springs" approach. While it has been used successfully in production, for example for the character Hank in *Finding Dory* [Kautzman et al. 2016], it can still create snags that are difficult to diagnose and resolve. The core problem lies on the definition of "closest point" when projecting the anchors back to the flesh mesh. When an anchor can be projected to multiple candidate points on the kinematic mesh, the one that has the smallest geodesic distance to the previous anchor position should usually be selected. However, computing the geodesic distance for each candidate point, for every anchor, for every timestep, is computationally prohibitive.

The Euclidean distance can be used as an inexpensive proxy, or in very complex situations, the geodesic distance can be computed in a limited N-ring around the original anchor position (Saito and Yuen [2017] propose something similar). However, if the kinematic mesh is sufficiently pinched, or the overall motion is sufficiently large, both of these approaches can still fail.

2 OUR RAY-TRACING APPROACH

We instead use 2D ray tracing to "walk" each anchor over the kinematic mesh. The anchors are not allowed to leave the mesh surface at any time, so it is impossible for them to be incorrectly projected onto a face with a misleadingly small Euclidean distance that masks a large geodesic distance. The walk uses ray-tracing, which may seem expensive at first glance, but the 2D geometry involved is extremely simple, so the approach is fast in practice.

Specifically, the force integration loop hands each anchor $\mathbf{a}^0$ a candidate position, $\mathbf{a}^*$, which is usually not on the surface of the kinematic mesh. The goal is to compute a final, updated anchor position $\mathbf{a}^1$ that is close to $\mathbf{a}^0$ in terms of geodesic distance. The orthogonal projection of $\mathbf{a}^*$ onto the triangle containing $\mathbf{a}^1$ should also yield $\mathbf{a}^1$.

We denote the vector between $\mathbf{a}^0$and $\mathbf{a}^*$ as $\mathbf{d}$ (i.e. $\mathbf{d} = \mathbf{a}^* - \mathbf{a}^0$). In order to walk along the mesh, we project $\mathbf{d}$ into the plane of the triangle that contains $\mathbf{a}^0$, and then record which edge that the ray hits. In other words, we perform a 2D ray-triangle intersection test. This intersection test can be made extremely simple and robust by transforming both $\mathbf{d}$ and $\mathbf{a}^0$ into a 2D canonical space. The necessary transform takes the vertices of the triangle, $(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)$, and arranges them in a matrix $\mathbf{D}_m \in \mathfrak{R}^{3\times2}$:

$$\mathbf{D}_m = \left[\begin{array}{c|c} \mathbf{v}_1 - \mathbf{v}_0 & \mathbf{v}_2 - \mathbf{v}_0 \end{array}\right]. \tag{1}$$

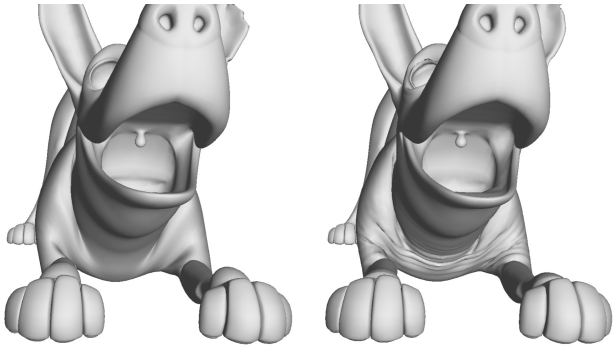
We then compute the QR decomposition $\mathbf{D}_m = \mathbf{Q}\mathbf{R}$ where $\mathbf{Q} \in \mathfrak{R}^{3\times2}$ and $\mathbf{R} \in \mathfrak{R}^{2\times2}$. The anchor and direction can now be transformed into a canonical coordinate system using $\hat{\mathbf{d}} = \mathbf{R}^{-1}\mathbf{Q}^\top\mathbf{d}$ and $\hat{\mathbf{a}} = \mathbf{R}^{-1}\mathbf{Q}^\top(\mathbf{a} - \mathbf{v}_0)$. Since $\mathbf{R}$ is only 2×2, computing the inverse is fast. The ray tracing problem is now simple. Determine if the ray $(\hat{\mathbf{a}}, \hat{\mathbf{d}})$ intersects the one of three edges: $\{(0,0),(0,1)\}$, $\{(0,0),(1,0)\}$ or $\{(1,0),(0,1)\}$. Once the hit position $\hat{\mathbf{h}}$ has been determined, it can be transformed back to world space using $\mathbf{h} = \mathbf{D}_m\hat{\mathbf{h}} + \mathbf{v}_0$.

If no edge was hit, it is because the ray terminated at point $\hat{\mathbf{a}}_1$ on the interior of the triangle, and $\mathbf{a}_1 = \mathbf{D}_m\hat{\mathbf{a}}_1 + \mathbf{v}_0$ is the new anchor position. Otherwise, we walk across the edge to the opposing face, and perform the ray trace again. If there is no opposing face, we are at the edge of the mesh, so we set $\mathbf{a}_1 = \mathbf{h}$. Care must be taken to ensure that the transformed point $\hat{\mathbf{a}}$ is always inside the canonical triangle, but we found that a simple clamp suffices. Finally, we keep track of the triangle faces we have already during the current time step. If we encounter the same face twice, the trace is terminated. In this case, it means a sharp crease was encountered, and the optimal anchor position lies along an edge instead of a face.

3 IMPLEMENTATION AND RESULTS

We found that it is not sufficient to insert an anchor at every vertex in the skin mesh, because a face can then unnaturally straddle a sharp crease in the kinematic mesh. Inserting an additional anchor at the barycenter of each face alleviated this problem.

We added the 2D formulation of the Stable Neo-Hookean constitutive model [Smith et al. 2018] as the membrane energy in all our simulations. Its area preservation term is smooth under inversion, and also helped to automatically untangle snagged configurations.



(a) Without bending energy **(b) With bending energy**

Figure 2: When a bending energy term is added, plausible skin wrinkling appears below the Dante's neck. ©Disney/Pixar

By default, bending energies are not included in the simulation, because they can cause the character's silhouette to deviate significantly from the underlying animation. Interestingly, we found during the course of experimenting that when bending energies are activated, it does in fact produce plausible skin wrinkling over character's body (Figure 2).

The raccoon in Figure 1 contained 18929 triangles, and took an average of 78.63 seconds per frame to simulate over 181 frames. The majority of the simulation time was spent in collision processing and linear system solution, because the raccoon mesh contained several badly-conditioned triangles. As can be seen in the supplemental video, without the new algorithm, the simulation collapses entirely. In all of our experiments, the ray-tracing the skin anchors never exceeded 1% of the running time, so its computation time was negligible.

REFERENCES

Ryan Kautzman, Jiayi Chong, and Patrick Coleman. 2012. Stable, Art-Directable Skin and Flesh Using Biphasic Materials. In *ACM SIGGRAPH Talks*.

Ryan Kautzman, Bill Wise, Meng Yu, Per Karlsson, Mark Hessler, and Audrey Wong. 2016. Finding Hank: Or How to Sim an Octopus. In *ACM SIGGRAPH Talks*.

Duo Li, Shinjiro Sueda, Debanga R. Neog, and Dinesh K. Pai. 2013. Thin Skin Elastodynamics. *ACM Trans. Graph.* 32, 4, Article 49 (July 2013), 10 pages.

Andy Milne, Mark McLaughlin, Rasmus Tamstorf, Alexey Stomakhin, Nicholas Burkard, Mitch Counsell, Jesus Canal, David Komorowski, and Evan Goldberg. 2016. Flesh, Flab, and Fascia Simulation on Zootopia. In *ACM SIGGRAPH Talks*. Article 34, 2 pages.

Jun Saito and Simon Yuen. 2017. Efficient and Robust Skin Slide Simulation. In *Proceedings of the ACM SIGGRAPH Digital Production Symposium*. Article 10, 6 pages.

Breannan Smith, Fernando de Goes, and Theodore Kim. 2018. Stable Neo-Hookean Flesh Simulation. *ACM Trans. Graph.* in press (2018).

Rasmus Tamstorf, Toby Jones, and Stephen F. McCormick. 2015. Smoothed Aggregation Multigrid for Cloth Simulation. *ACM Trans. Graph.* 34, 6, Article 245 (Oct. 2015), 245:1–245:13 pages.