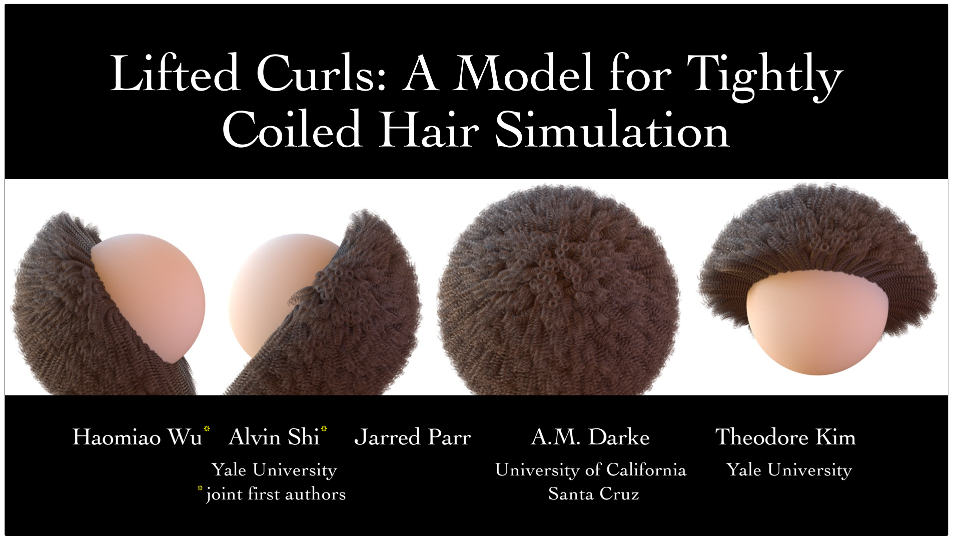


Lifted Curls: A Model for Tightly Coiled Hair Simulation

Haomiao Wu[*] Alvin Shi[*] Jarred Parr A.M. Darke Theodore Kim
Yale University University of California Santa Cruz Yale University
[*]joint first authors

Hi everybody, I'm going to wrap up this session by diving a little deeper into the algorithms.

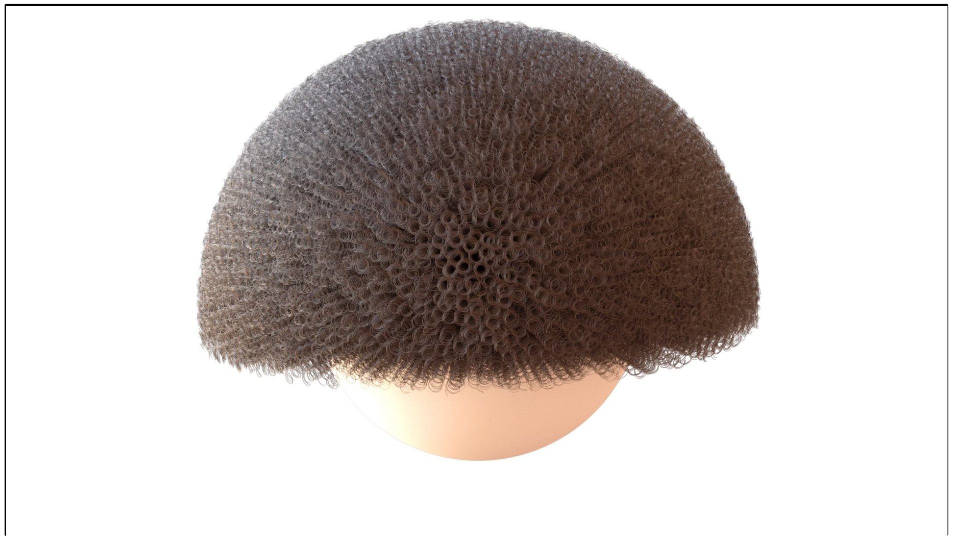
This is the Lifted Curls algorithm that my students and Professor Darke presented at the Symposium on Computer Animation, SCA last year.

What we're going to see is that past algorithms for simulating the motion of hair has an assumption baked into the math that the hair is straight, or lightly wavy.

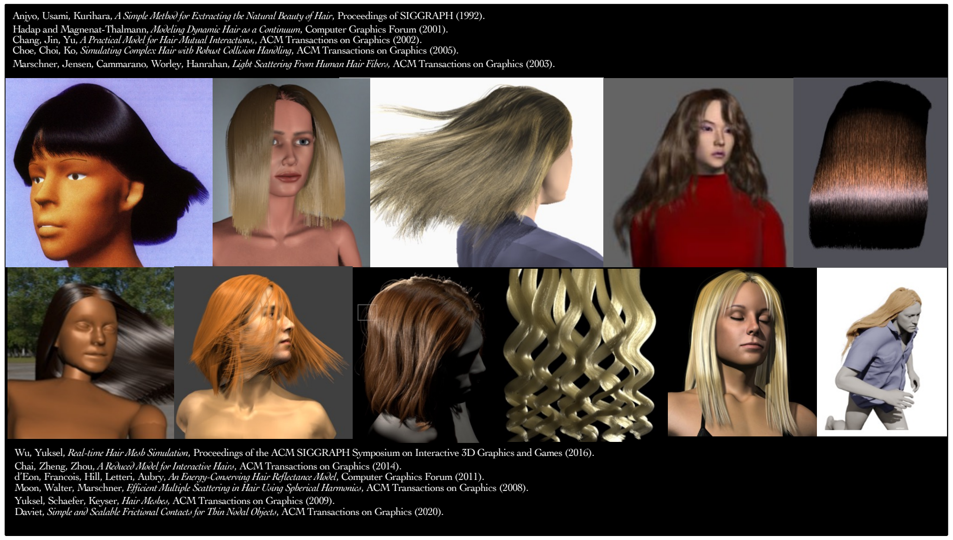
If you make a different assumption, that the hair is very curly, like Black hair, you can change the algorithm, and make it both faster and more robust.

Other researchers in the community thought this was pretty neat too. We got the Best Paper award at the conference last year.

As with everything else we've seen in this course, we're going to look at the specific case of Black hair. Not straight hair.



And here's one of our results. I'm showing this to you first, because it's not that common to see hair like this in computer graphics.



Anjyo, Usami, Kurihara, *A Simple Method for Extracting the Natural Beauty of Hair*, Proceedings of SIGGRAPH (1992).
Hadap and Magnenat-Thalmann, *Modeling Dynamic Hair as a Continuum*, Computer Graphics Forum (2001).
Chang, Jin, Yu, *A Practical Model for Hair Mutual Interactions,*, ACM Transactions on Graphics (2002).
Choe, Choi, Ko, *Simulating Complex Hair with Robust Collision Handling*, ACM Transactions on Graphics (2005).
Marschner, Jensen, Cammarano, Worley, Hanrahan, *Light Scattering From Human Hair Fibers,* ACM Transactions on Graphics (2003).

Wu, Yuksel, *Real-time Hair Mesh Simulation*, Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (2016).
Chai, Zheng, Zhou, *A Reduced Model for Interactive Hairs*, ACM Transactions on Graphics (2014).
d'Eon, Francois, Hill, Letteri, Aubry, *An Energy-Conserving Hair Reflectance Model*, Computer Graphics Forum (2011).
Moon, Walter, Marschner, *Efficient Multiple Scattering in Hair Using Spherical Harmonics*, ACM Transactions on Graphics (2008).
Yuksel, Schaefer, Keyser, *Hair Meshes,* ACM Transactions on Graphics (2009).
Daviet, *Simple and Scalable Frictional Contacts for Thin Nodal Objects*, ACM Transactions on Graphics (2020).

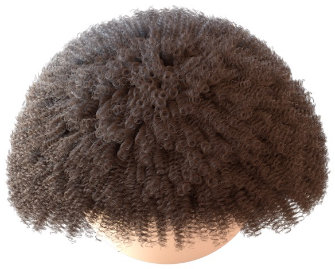
Over the last 30 years, we've instead have looked almost exclusively at the simulation of straight or wavy hair.

Artistic Simulation of Curly Hair

Hayley Iben, Mark Meyer, Lena Petrovic,
Olivier Soares, John Anderson and Andrew Witkin
Pixar Animation Studios

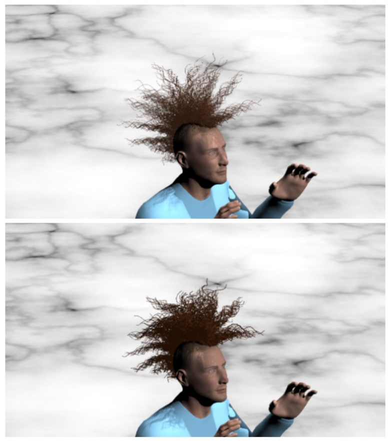
Maybe you remember this curly hair paper from Pixar, where I used to work.

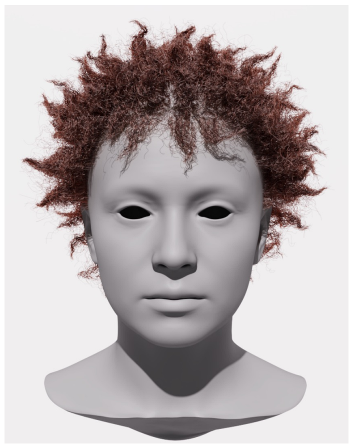
This is the Taz hair system, developed for the movie *Brave*. It was presented here at SCA exactly 10 years ago.



But put it side-by-side with our results, and it's not the same.

Yes, it's curly. We're going for curlier.


[Lee et al. 2018]


[Daviet 2023]

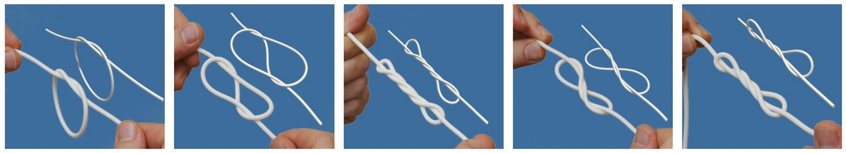
Other papers claim to do "curly" hair, but they seem to mean noise layered over straight hair. That's also not what we're going for.

[Hsu et al. 2023]

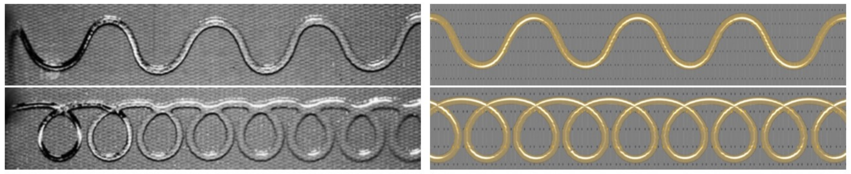
There *is* this paper this year with similar hair, which is great. This *is position-based*, whereas ours is FEM.

Some of their findings are similar to ours, where existing rod models run into trouble with highly curved strands.

If you're interested in this topic, you might give this one a look too.



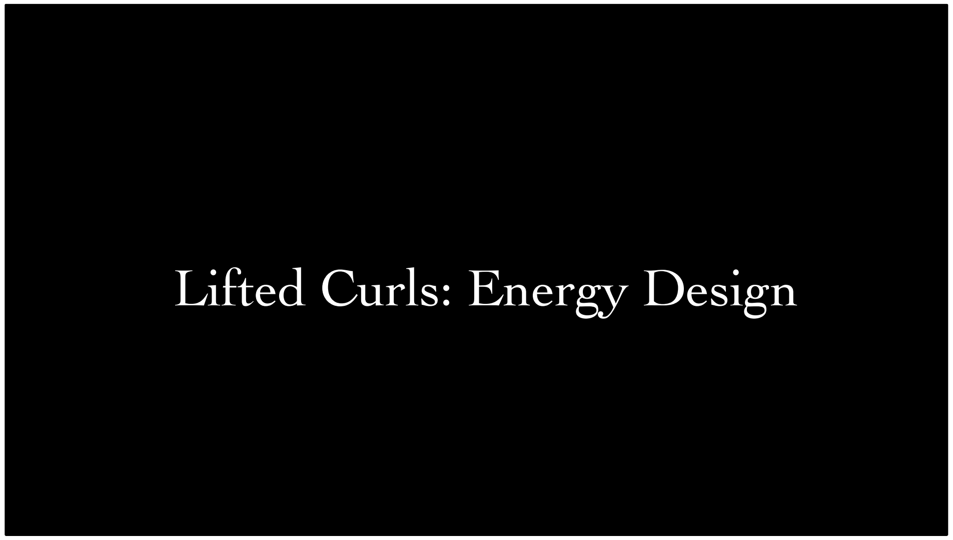
[Bergou et al. 2008]



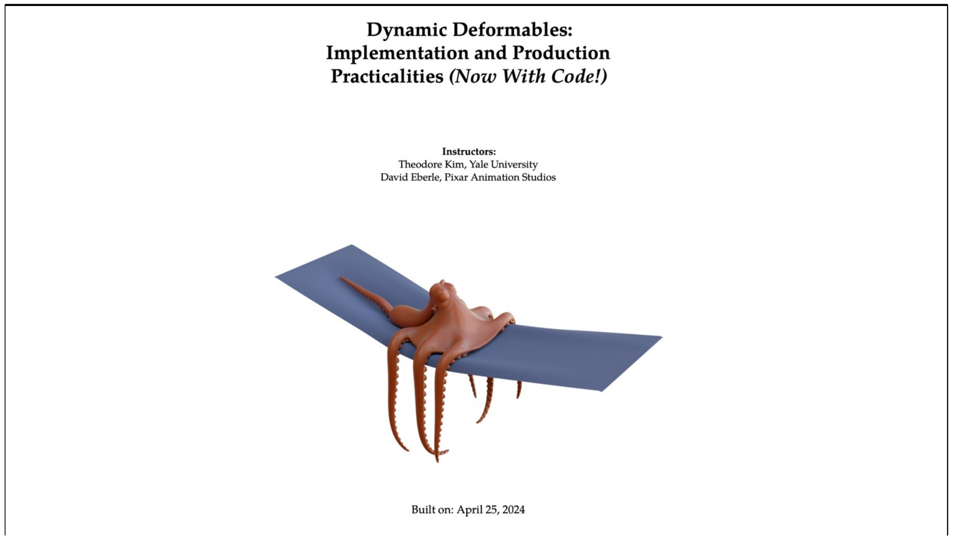
[Bergou et al. 2010]

Then there’s Discrete Elastic Rods, and its extension, Discrete Viscous Threads. We’re talk about those later.

I’m mentioning it now just to assure you that, no, I didn’t forget about them.


Lifted Curls: Energy Design

Let’s start looking at the details of our method, particularly the elastic energy.



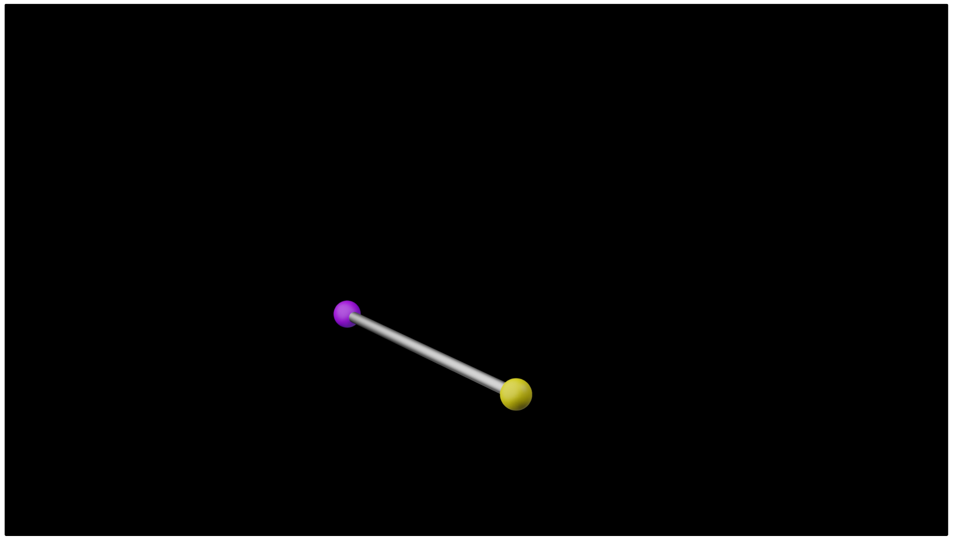
Dynamic Deformables:
Implementation and Production
Practicalities *(Now With Code!)*

Instructors:
Theodore Kim, Yale University
David Eberle, Pixar Animation Studios

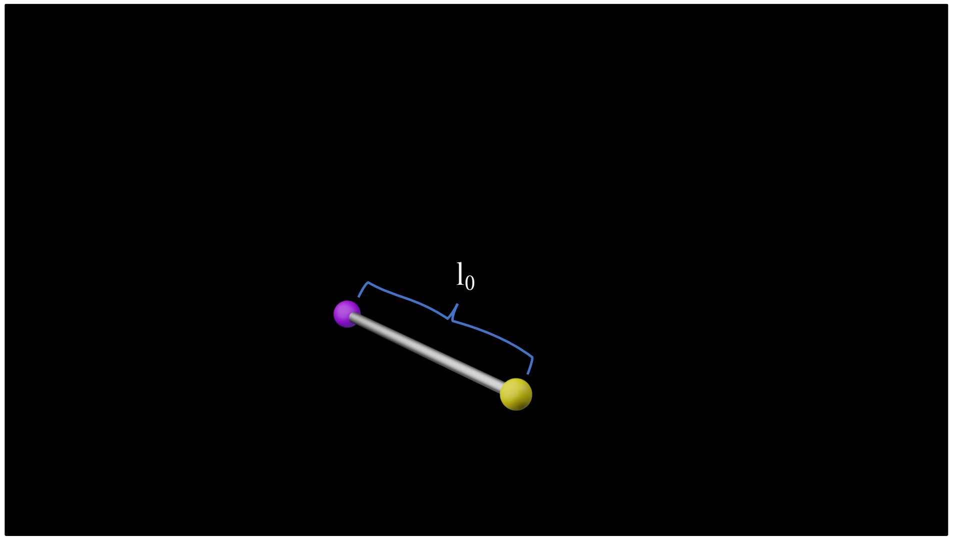
Built on: April 25, 2024

If you're not familiar with elastic energy design, I did a whole course on this with David Eberle a few years ago, where we went through things in lots of detail.

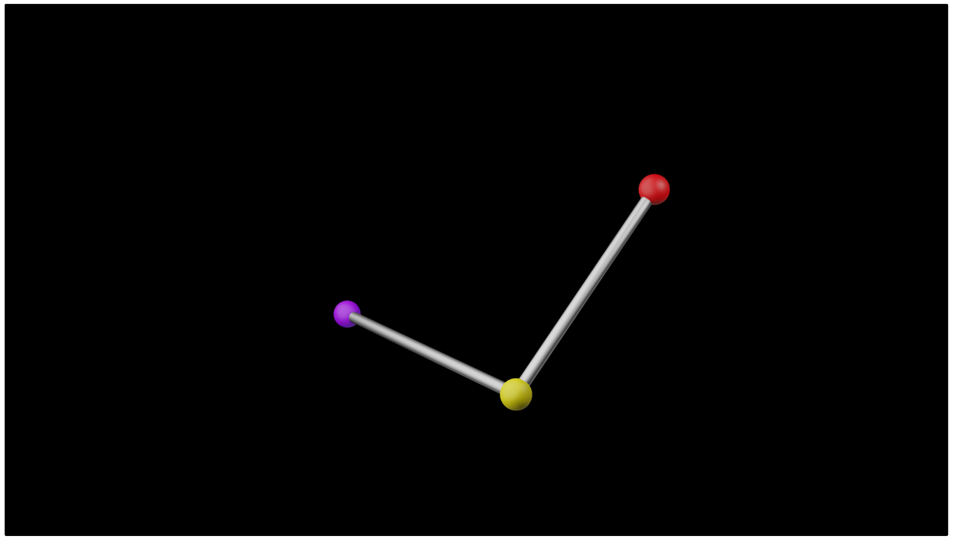
For any segment-based strand energy, you're going to need three components.



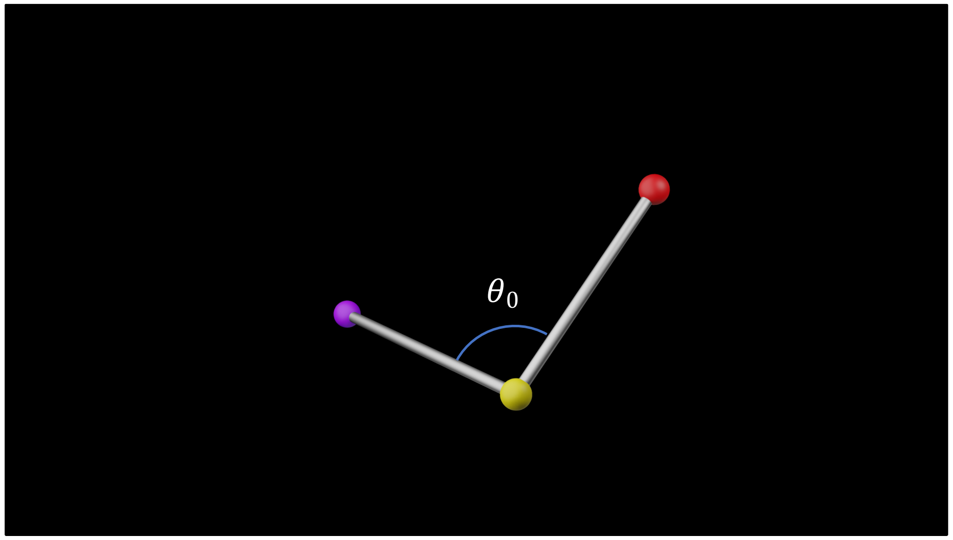
First, along each line segment



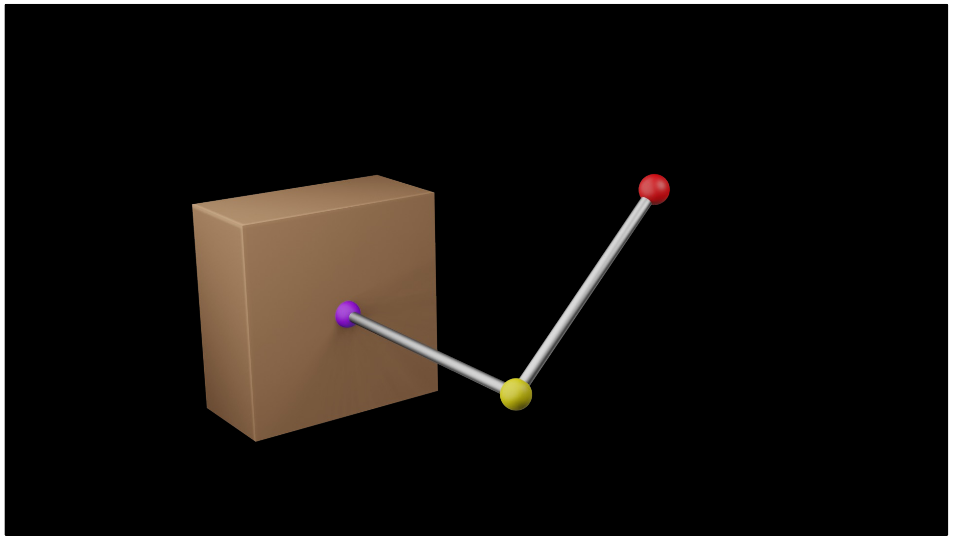
We have a stretching term that tries to preserve the original length, l0



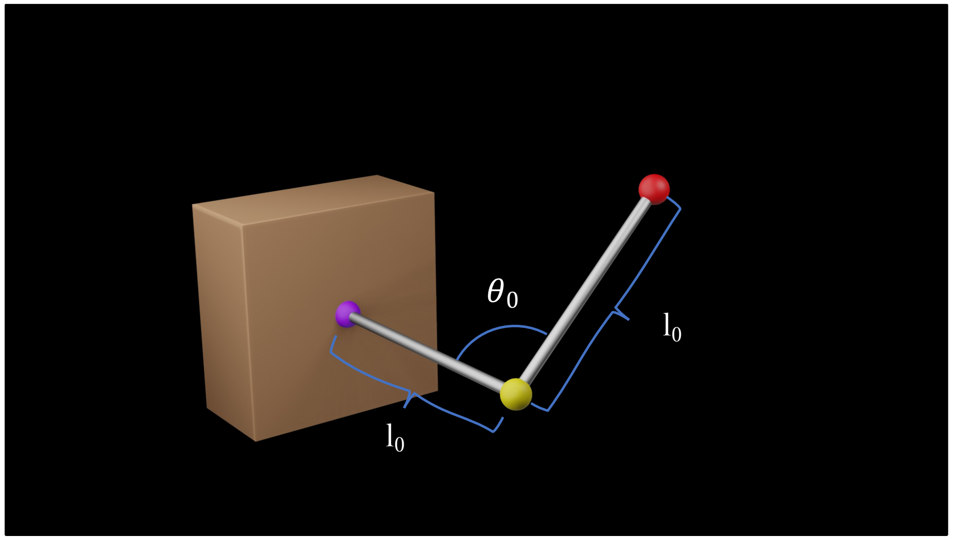
Second, between successive line segments



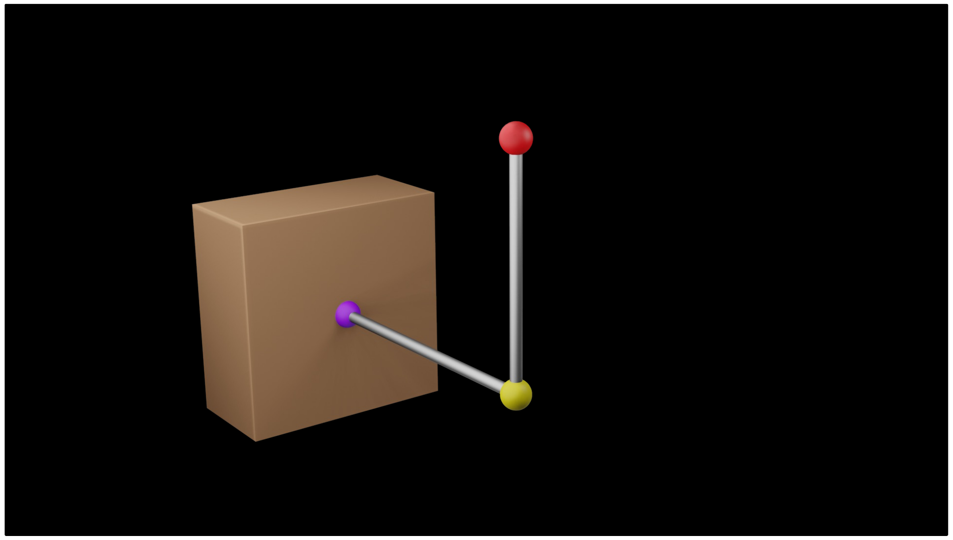
A bending energy that tries to maintain some rest angle, theta0 between the segments.



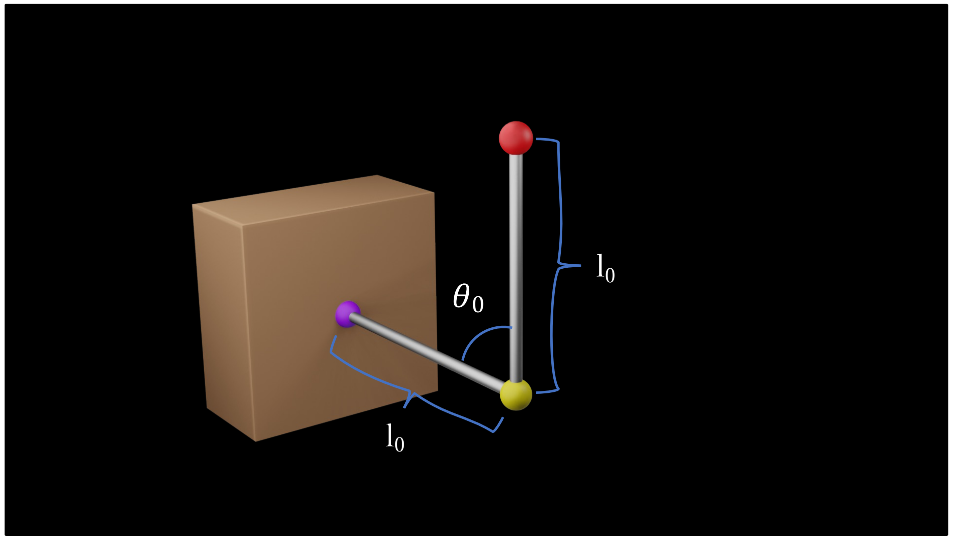
I said three components, because these two forces are not sufficient. Let's attach this strand to the wall.



And say that all of its length and angle constraints are satisfied.



You can actually rotate it

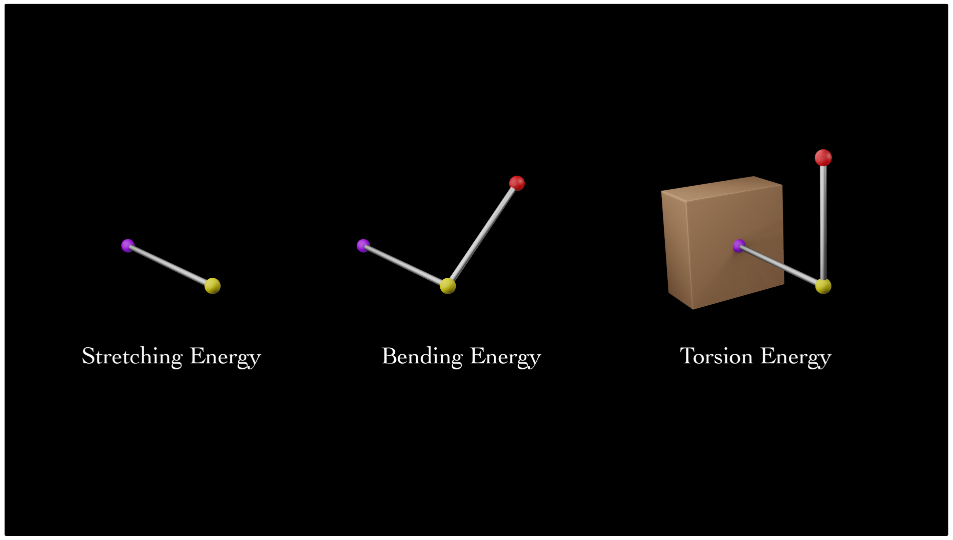


And still maintain have the same lengths and angle be the same.

That might seem right for like a ball and socket joint, but for a piece of hair? That doesn't sound right.

If you twirl a strand of hair that's attached to your scalp, it should resist a little.

So, you need a third energy. A twisting or torsion energy is needed to make this description complete.



Here are the three energies we need, a stretching energy, a bending energy, and a torsion energy.

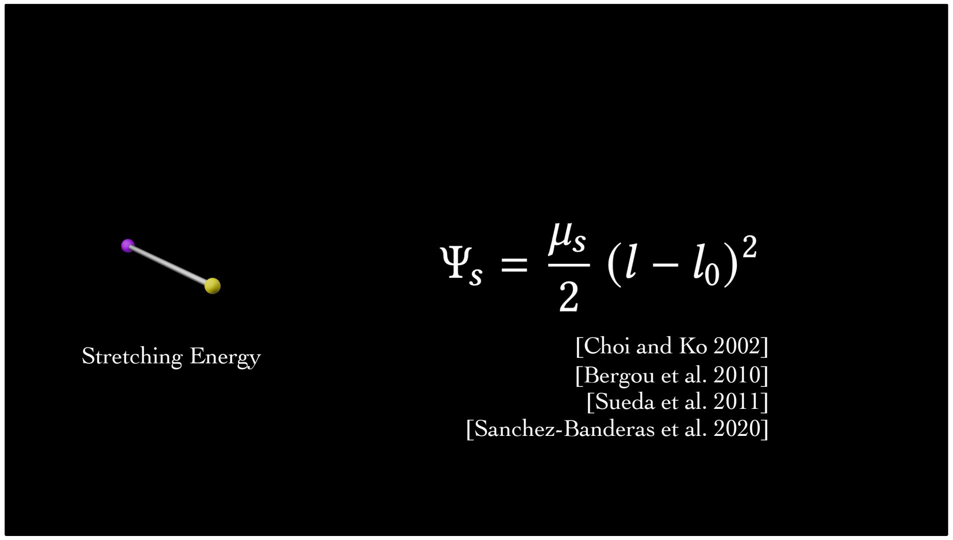
I'm going to describe each one in our model. Two will be familiar, but the third will be different.

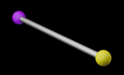
So let’s start with the stretching energy

Stretching Energy

$$\Psi_s = \frac{\mu_s}{2}\left(l - l_0\right)^2$$

We use a quadratic length penalty

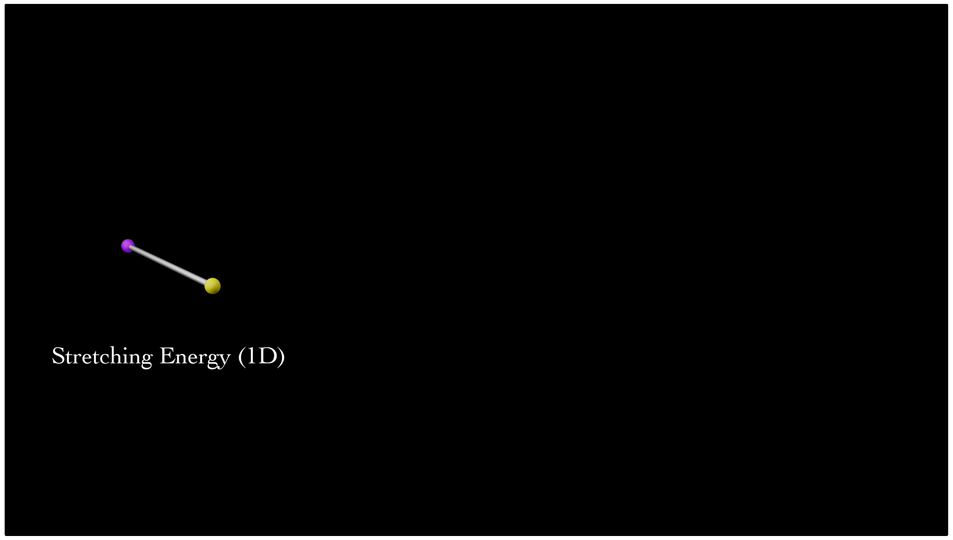




$$\Psi_s = \frac{\mu_s}{2}\,(l - l_0)^2$$

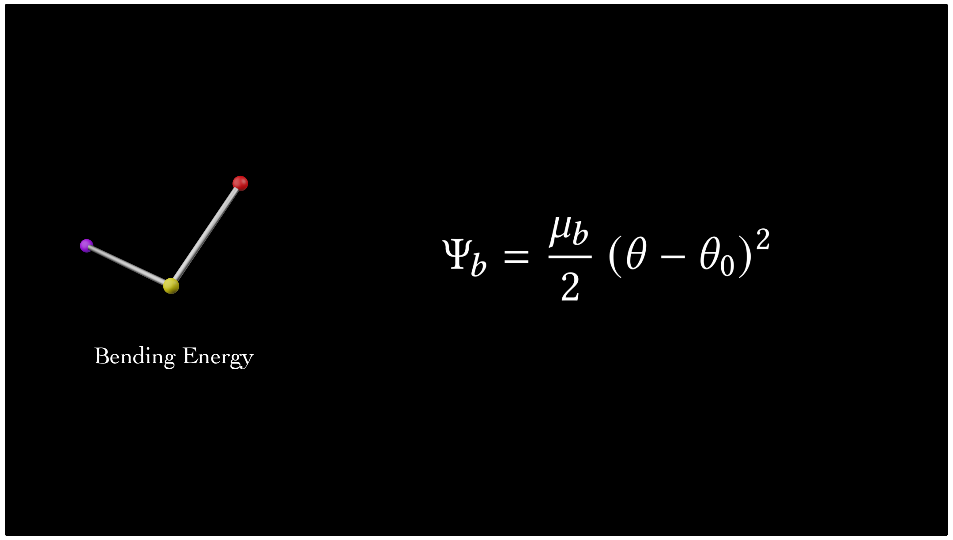
[Choi and Ko 2002]
[Bergou et al. 2010]
[Sueda et al. 2011]
[Sanchez-Banderas et al. 2020]

This is nothing special. Lots of people use this energy.



Geometrically, the stretching energy only involves two vertices, so it is intrinsically 1D. We're dealing with a one-dimensional quantity.

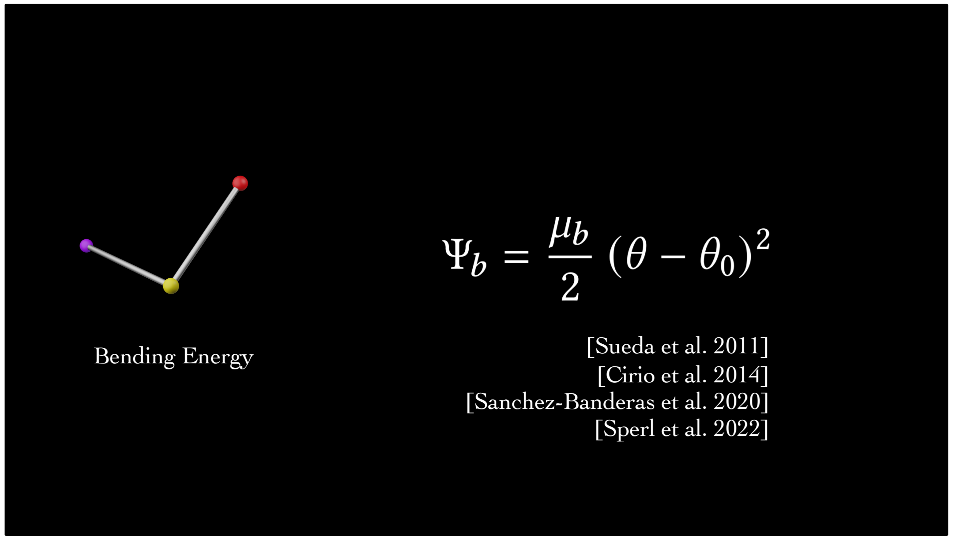
Next let's look at bending.

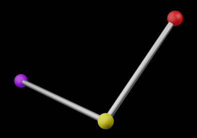


Bending Energy

$$\Psi_b = \frac{\mu_b}{2}\left(\theta - \theta_0\right)^2$$

Again, a quadratic energy





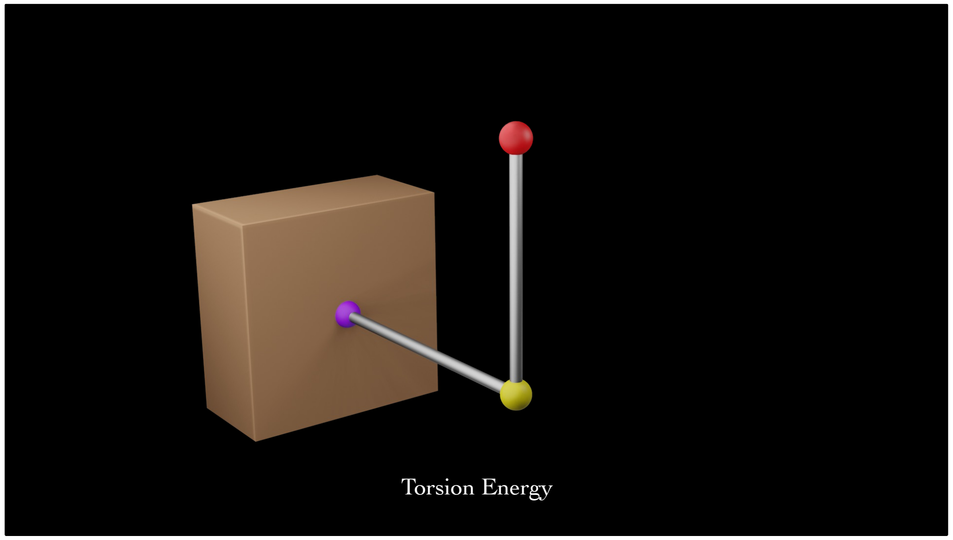
$$\Psi_b = \frac{\mu_b}{2}\left(\theta - \theta_0\right)^2$$

[Sueda et al. 2011]
[Cirio et al. 2014]
[Sanchez-Banderas et al. 2020]
[Sperl et al. 2022]

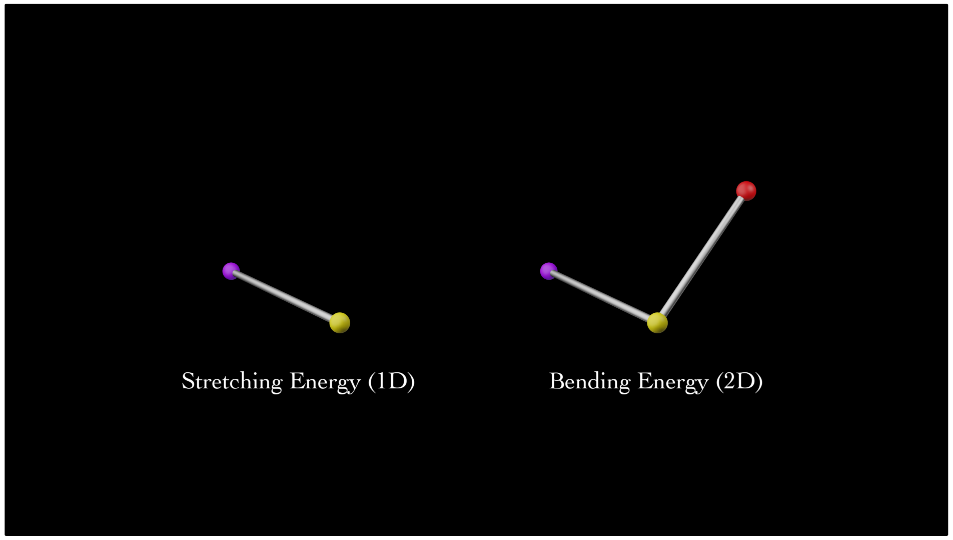
Again, lots of people use this.

Bending Energy (2D)

Geometrically, this energy involves *three* vertices, so it is intrinsically 2D. We're essentially looking at a triangle.

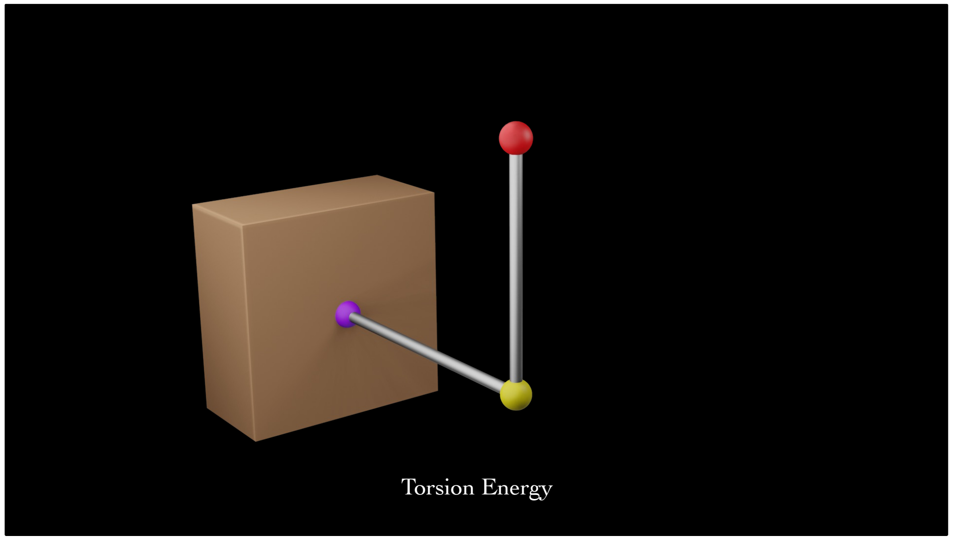


Third, the torsion energy. Here, we're going to do something a little bit different.

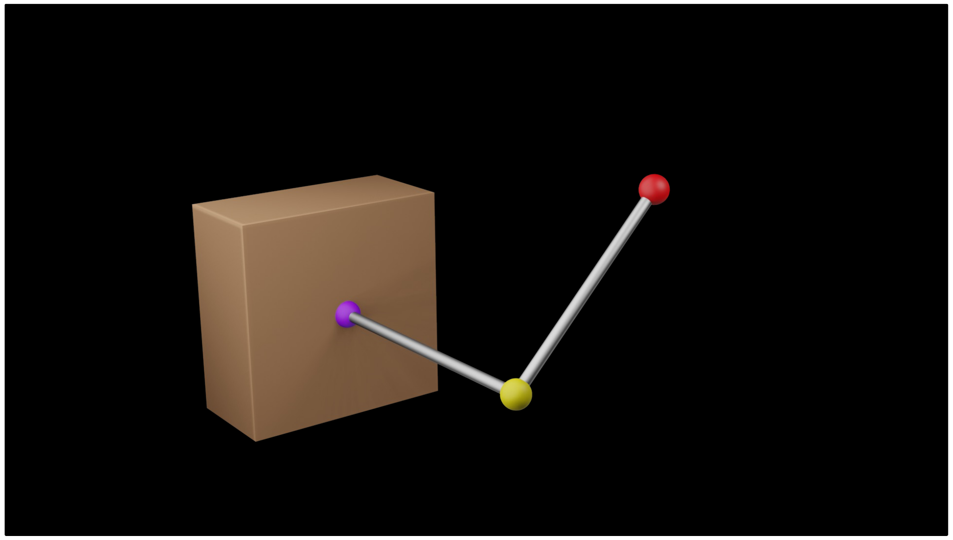


If we look at the stretching and bending energy, we see that went from 1D to 2D.

But 2D isn't sufficient, for torsion

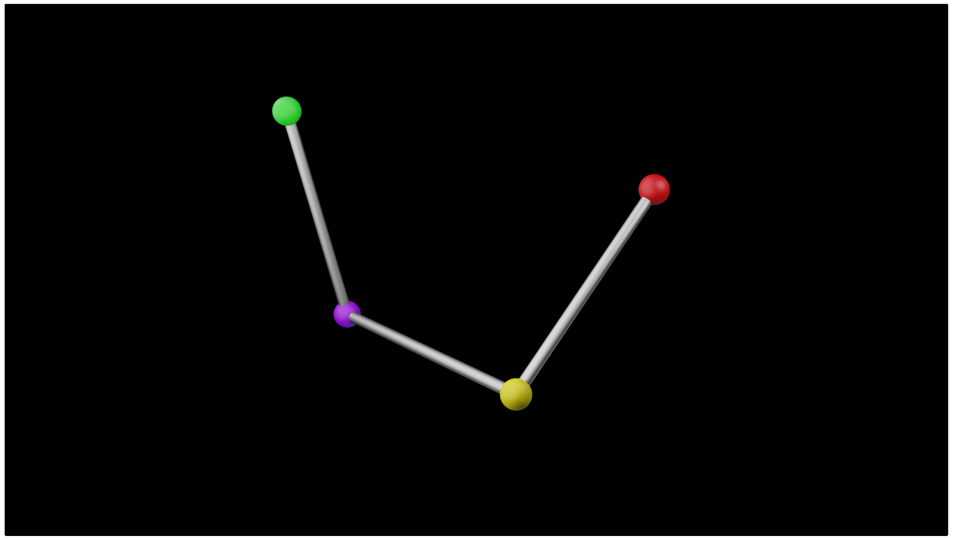


We saw under this rotation mode.



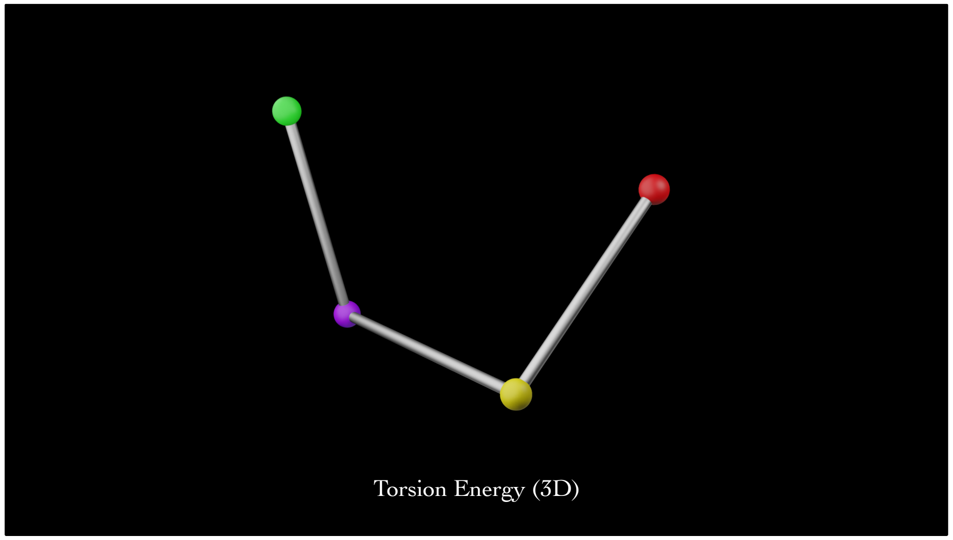
That the energy stays exactly the same, so it won't exert a force.

You know what we can add to resolve this ambiguity?

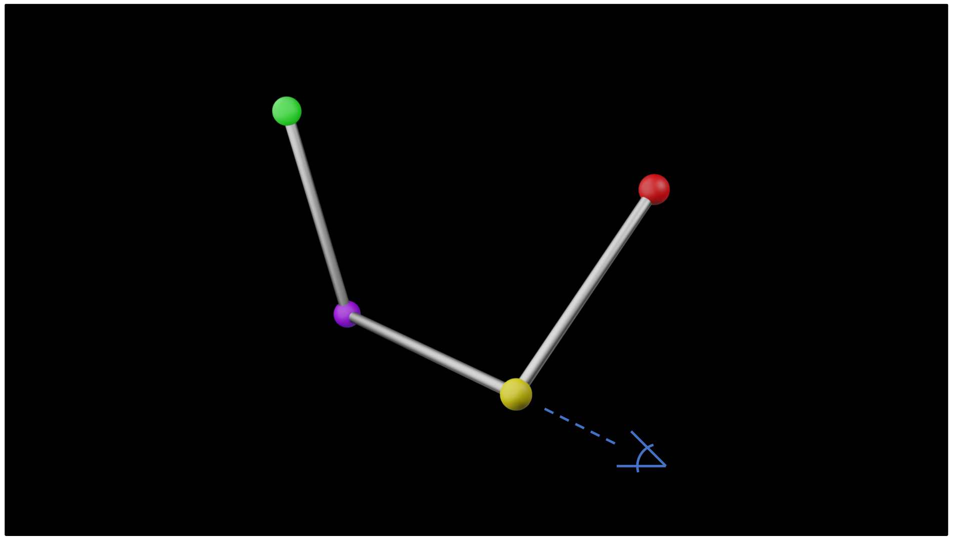


Another vertex.

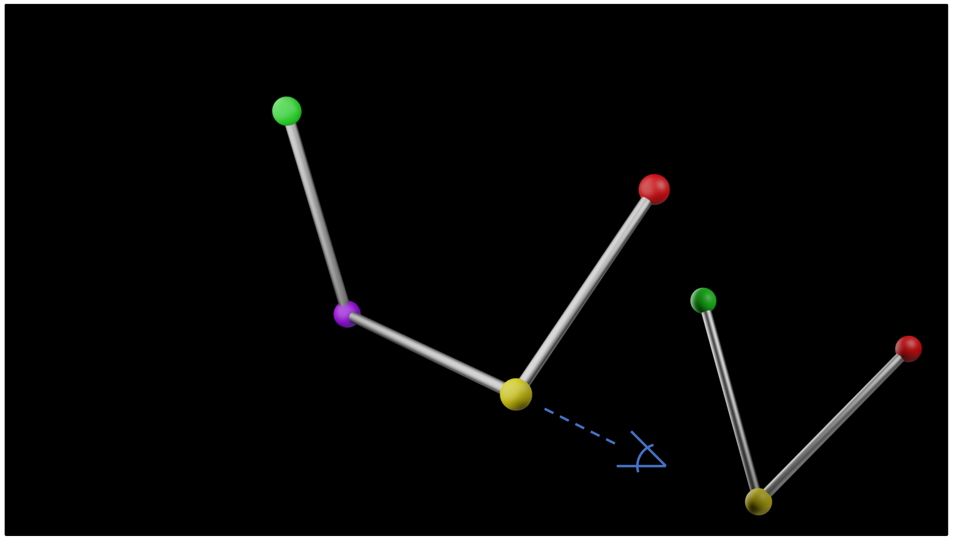
We're going to define the third component of our energy using four vertices.



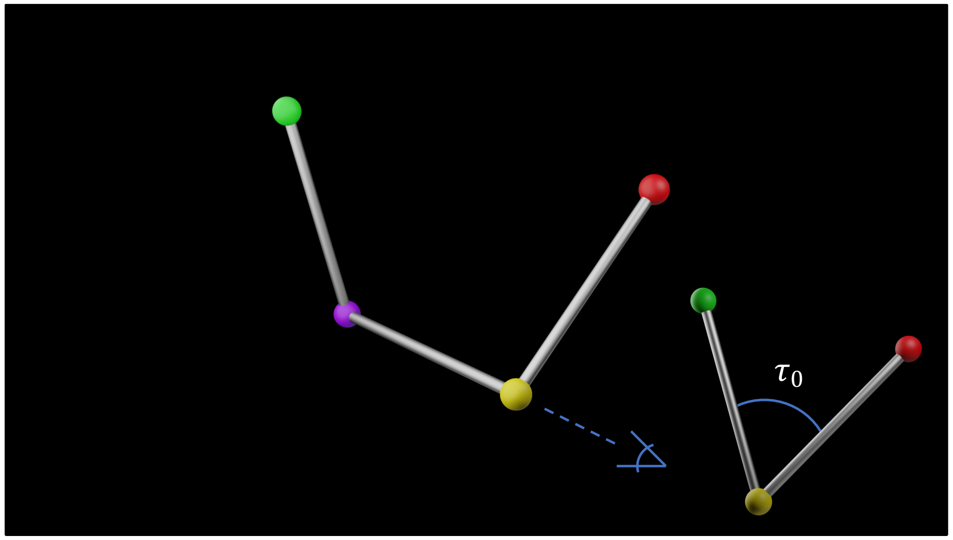
This is then intrinsically a 3D energy. After all, if you have four vertices, they form a tetrahedron.



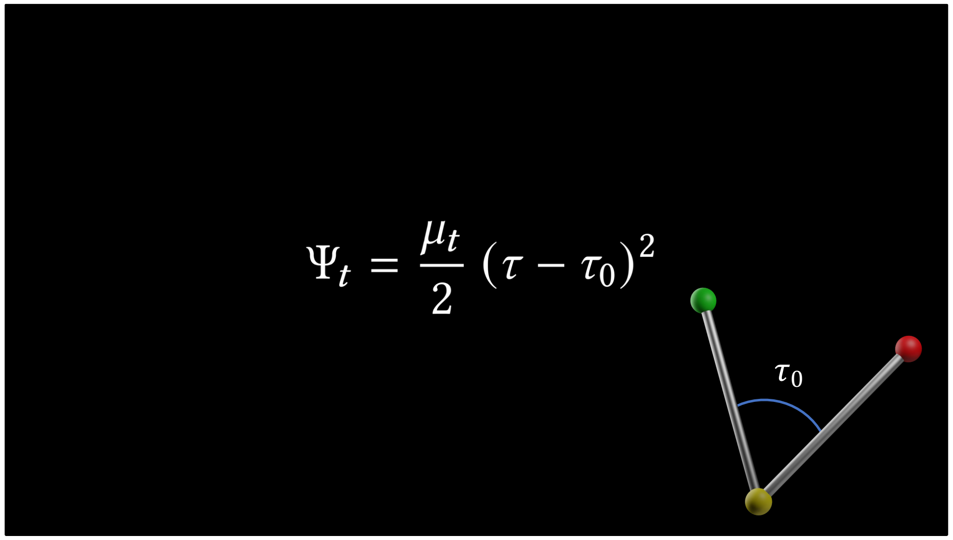
Let’s look at the projection of the tet along the center edge.



It'll look something like this.

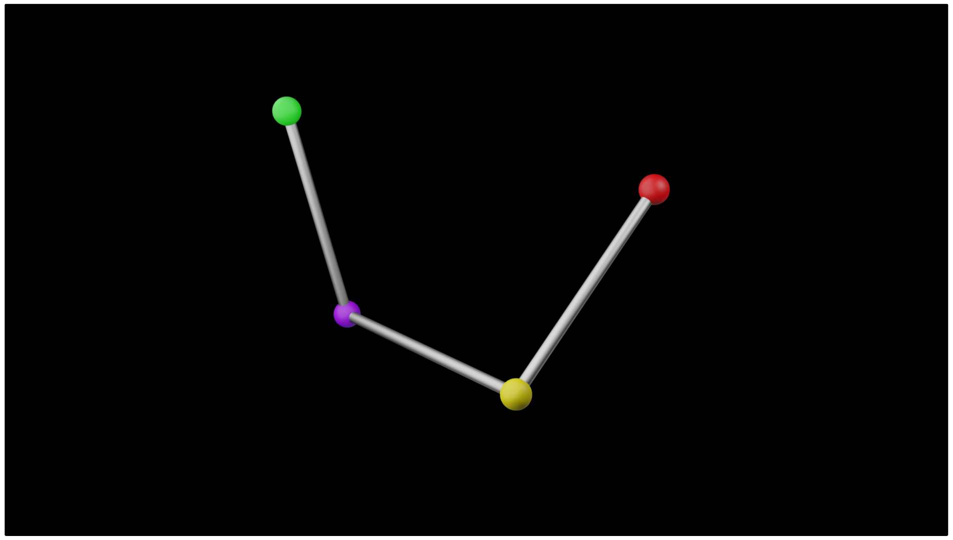


And we can measure the angle tau between these two non-adjacent segments



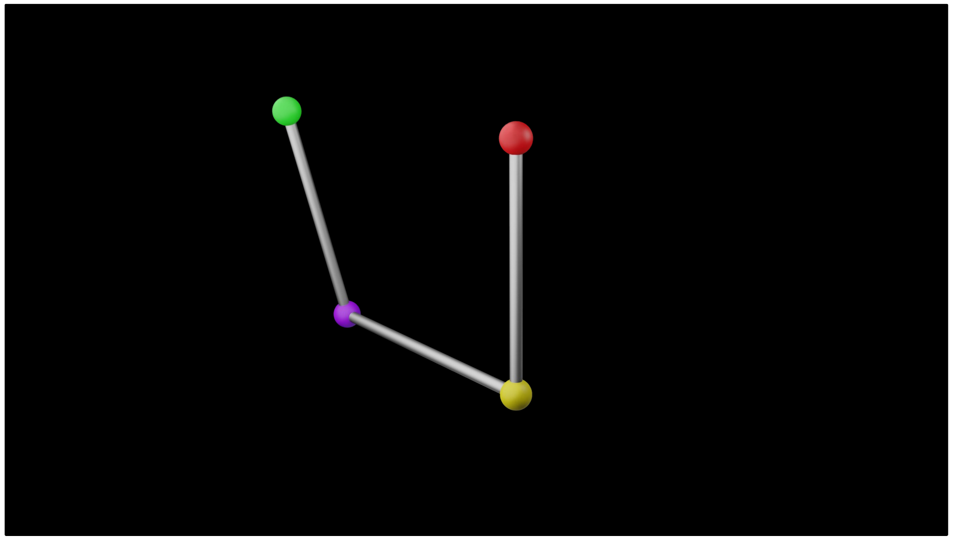
$$\Psi_t = \frac{\mu_t}{2}\left(\tau - \tau_0\right)^2$$

We formulate a new torsion energy that is quadratic with respect to tau.

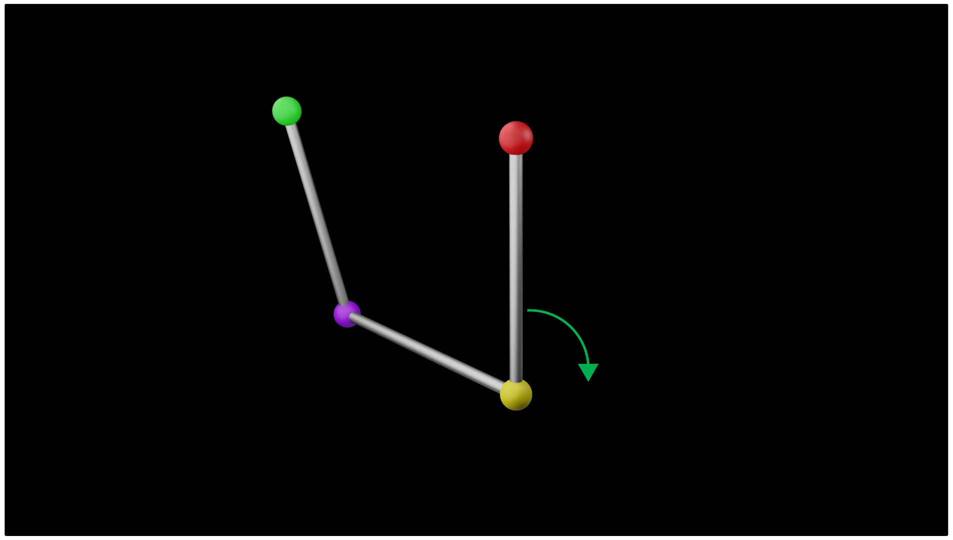


This does remove the ambiguity.

With this fourth vertex, we have a force that will try to maintain an angle with respect to that vertex.

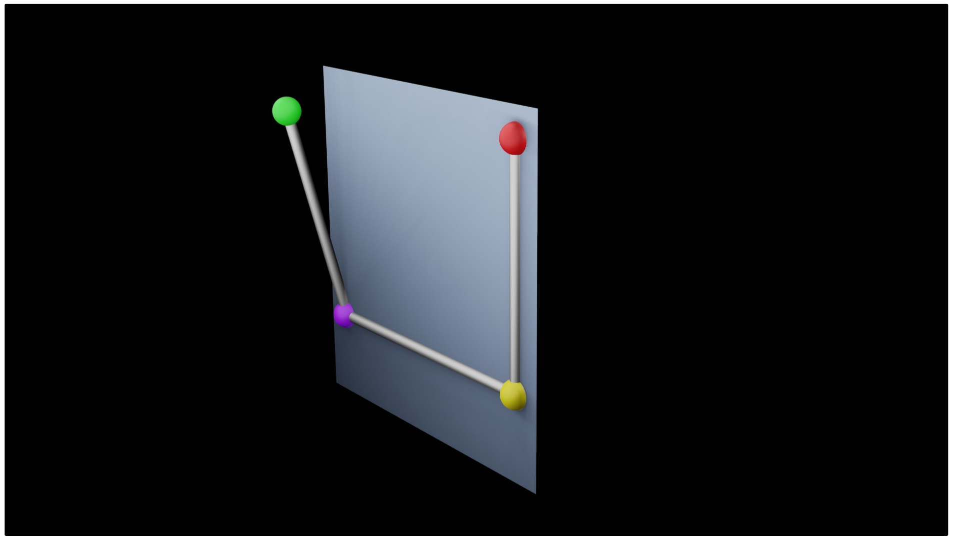


If this red vertex rotates, its angle relative to the green vertex will change

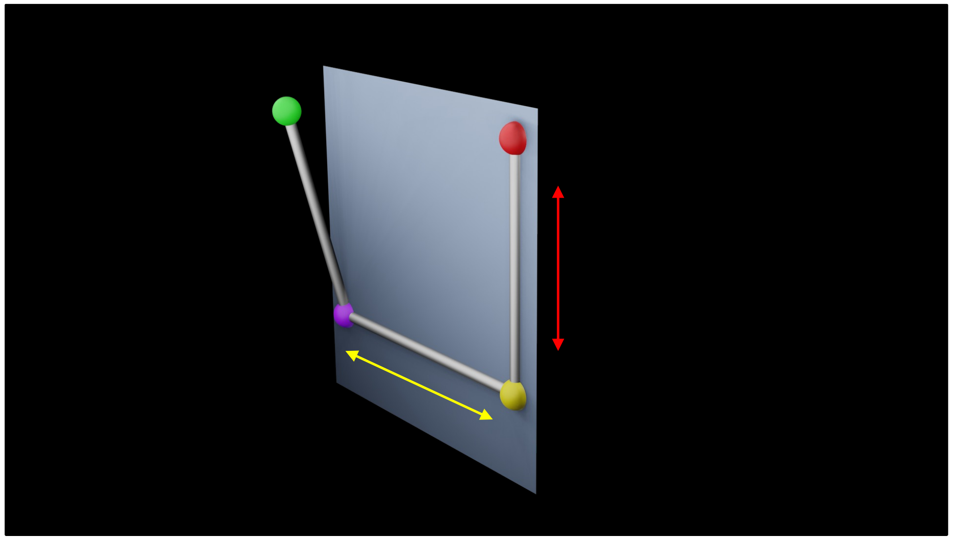


And a force will appear to rotate it back.

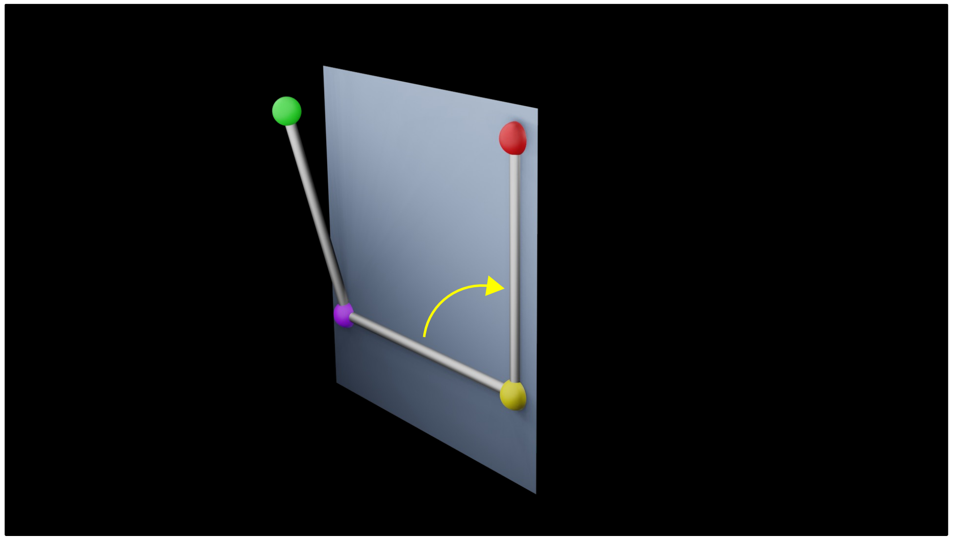
Exactly the behavior we wanted.



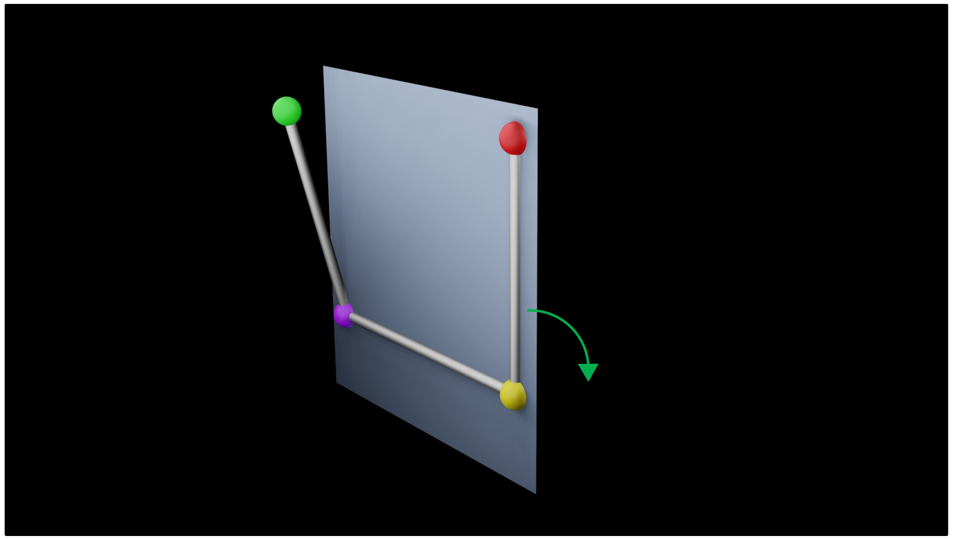
More formally, the other two energies only exert forces in a fixed plane. This was not sufficient.



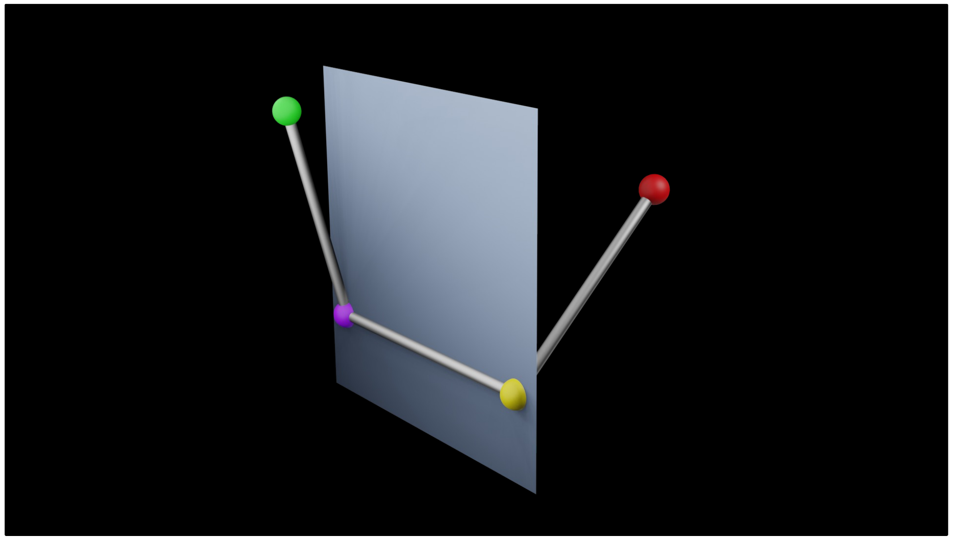
The stretching forces are constrained to the plane



And the bending forces can also only occur in the plane



It’s only by adding this out-of-plane torsion force



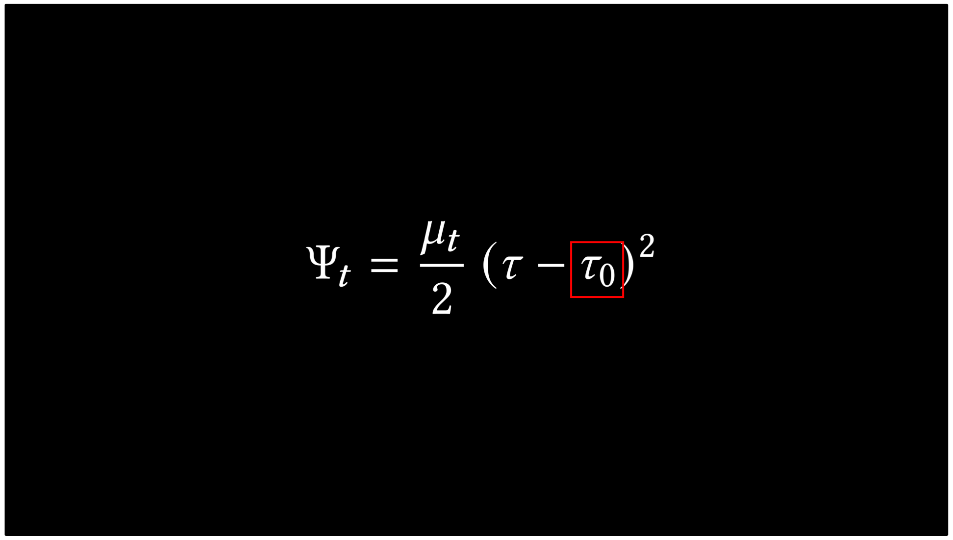
That we can get the red vertex to snap back to its original place.

We've added the missing out-of-plane force.

$$\Psi_t = \frac{\mu_t}{2}\left(\tau - \tau_0\right)^2$$

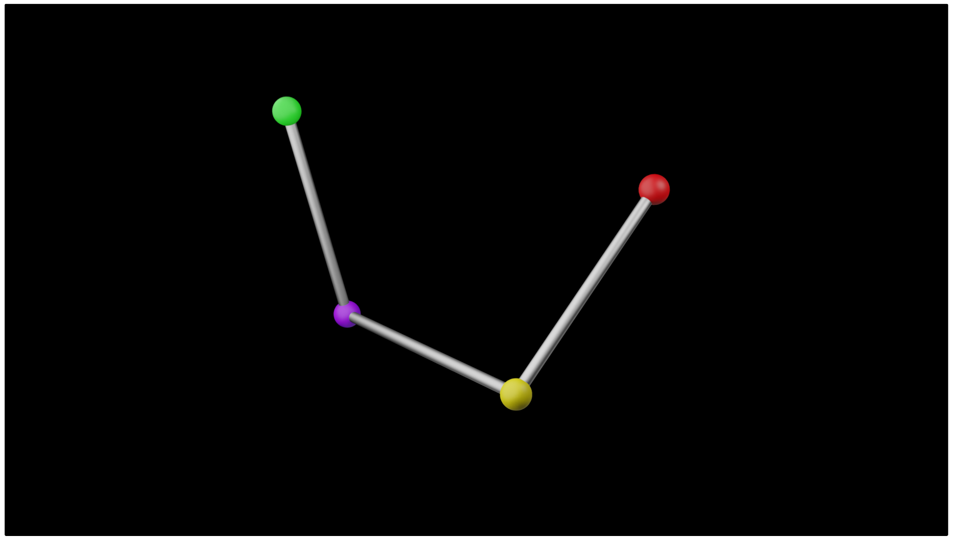
So here's the torsion energy again.

If you're paying attention, you may have an objection to this.

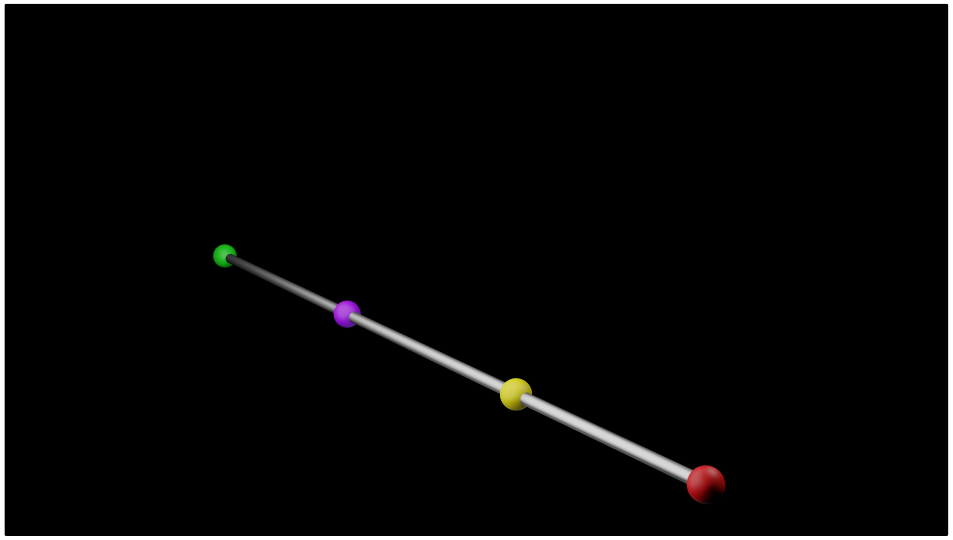


$$\Psi_t = \frac{\mu_t}{2}\left(\tau - \boxed{\tau_0}\right)^2$$

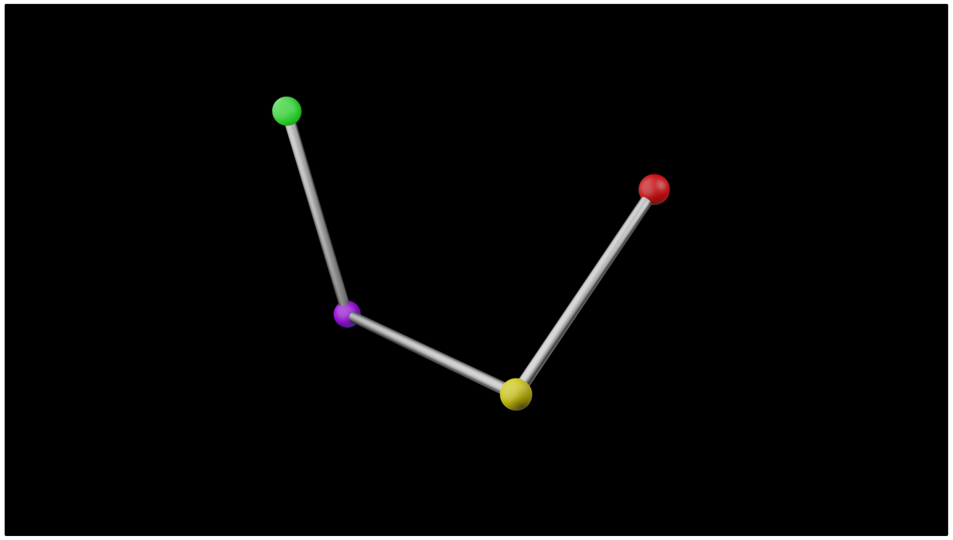
What if this angle tau0 is degenerate? What if tau0 is undefined?



That can happen.

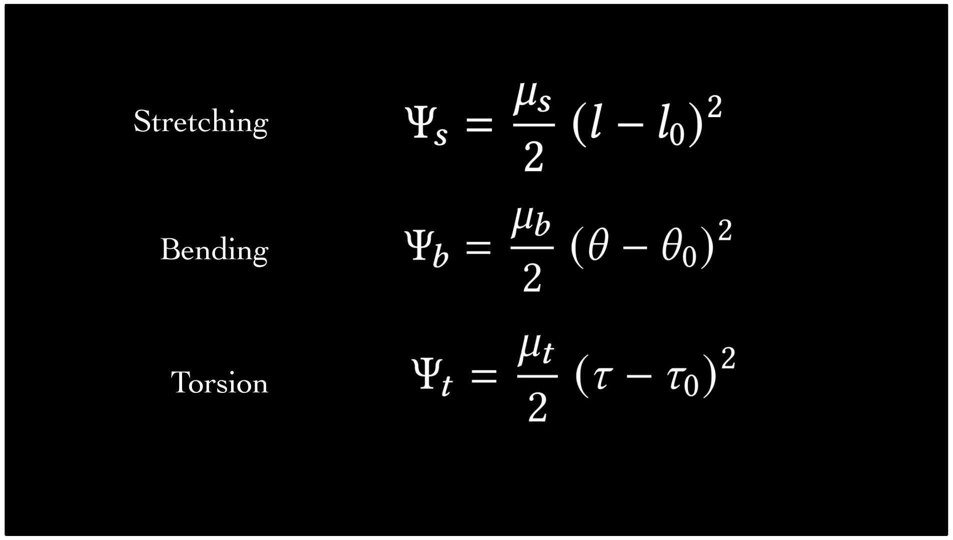


If you're simulating straight hair. We're not simulating straight hair.



We’re simulating curly hair. The angle will always be well- defined.

Just an aside – we did try it on straight hair, and it actually does fine.

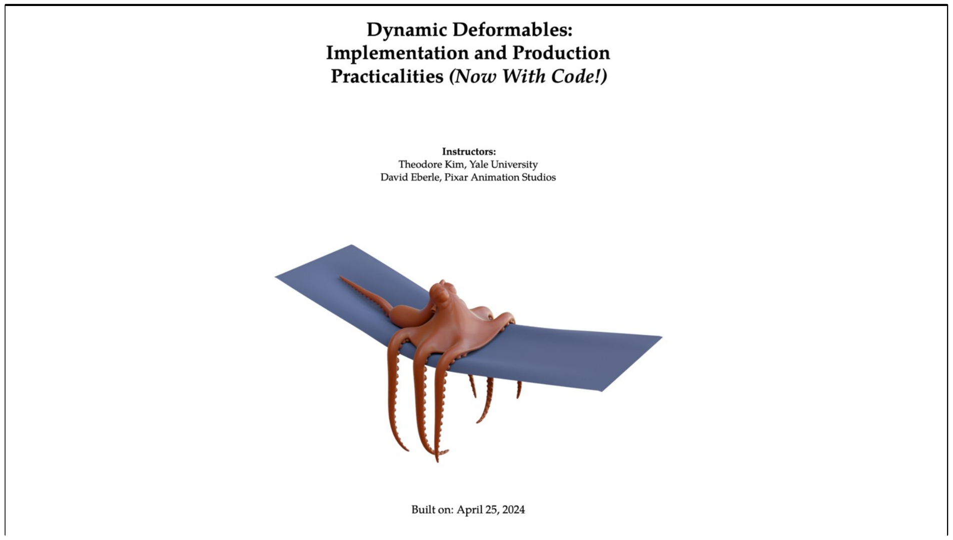


Stretching $$\Psi_s = \frac{\mu_s}{2}(l - l_0)^2$$

Bending $$\Psi_b = \frac{\mu_b}{2}(\theta - \theta_0)^2$$

Torsion $$\Psi_t = \frac{\mu_t}{2}(\tau - \tau_0)^2$$

Okay, that's the terms of our energy. All quadratic.

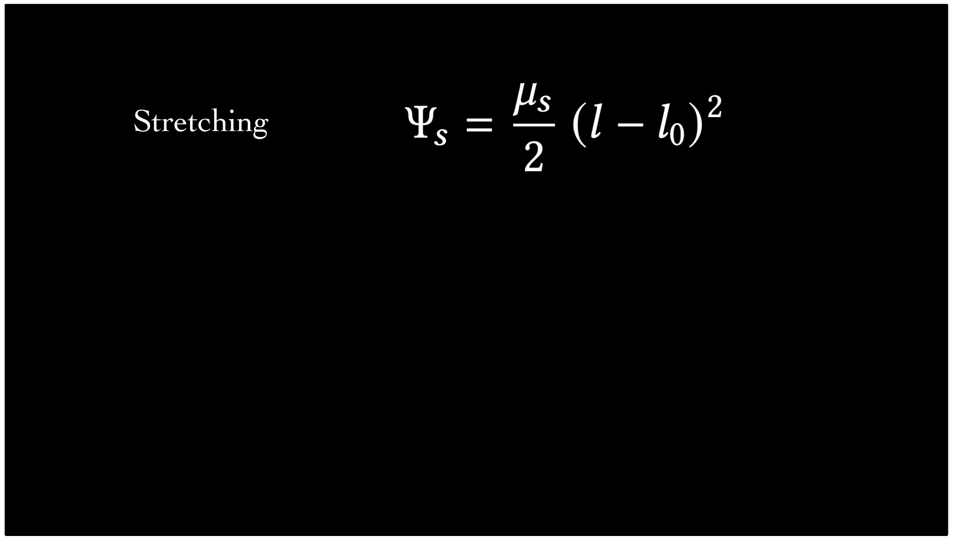


Dynamic Deformables:
Implementation and Production
Practicalities *(Now With Code!)*

Instructors:
Theodore Kim, Yale University
David Eberle, Pixar Animation Studios

Built on: April 25, 2024

Again, referring back to my course notes from two years ago, if you want to push an energy into an FEM-style solver, you need to filter its eigenvalues. Can we get expressions for them?

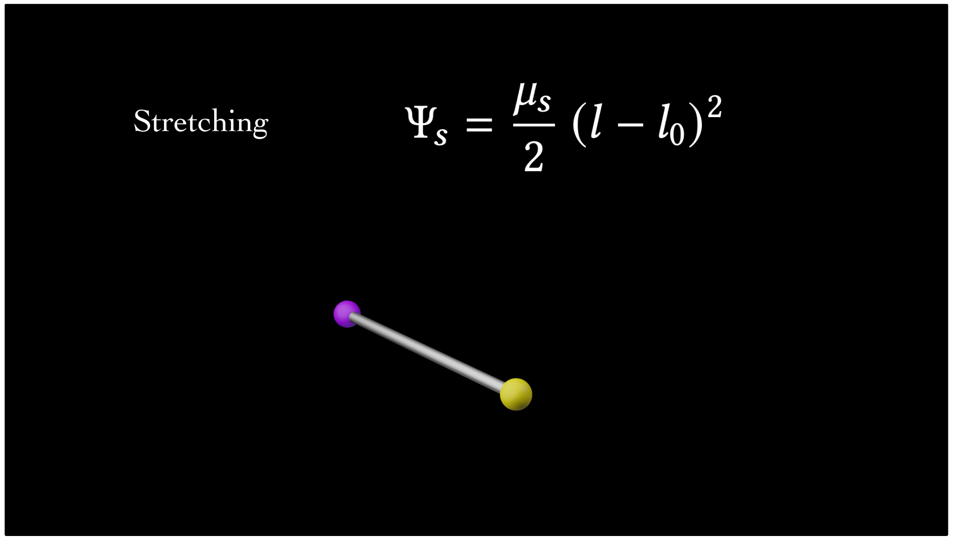


Stretching $\Psi_s = \frac{\mu_s}{2}\left(l - l_0\right)^2$

Let’s look at the stretching energy.

This is the oldest and simplest-looking term

But numerically, it’s the most important term, because with hair, it’s the stiffest energy of the system.

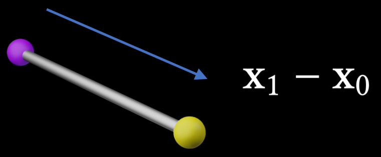

Stretching

$$\Psi_s = \frac{\mu_s}{2}\left(l - l_0\right)^2$$

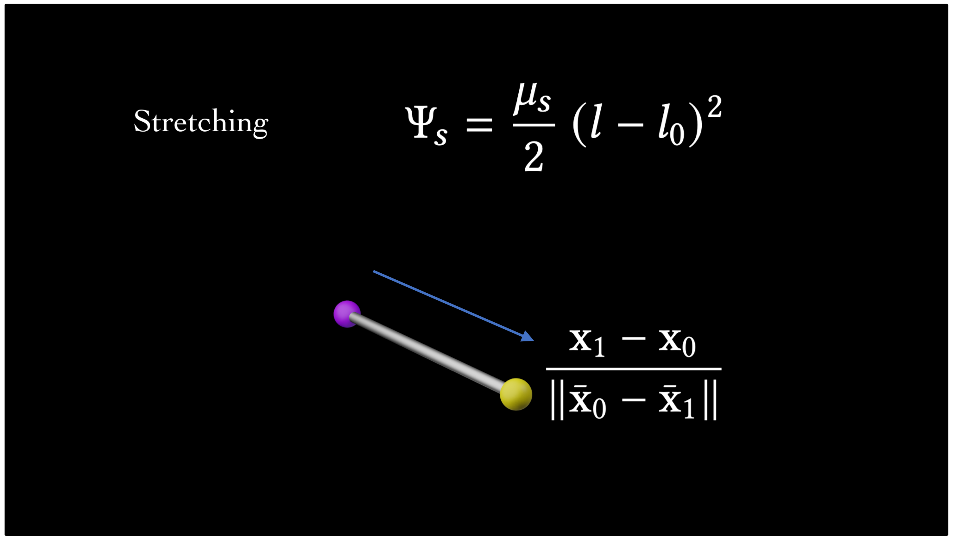
Let’s start with the edge

Stretching

$$\Psi_s = \frac{\mu_s}{2}\left(l - l_0\right)^2$$



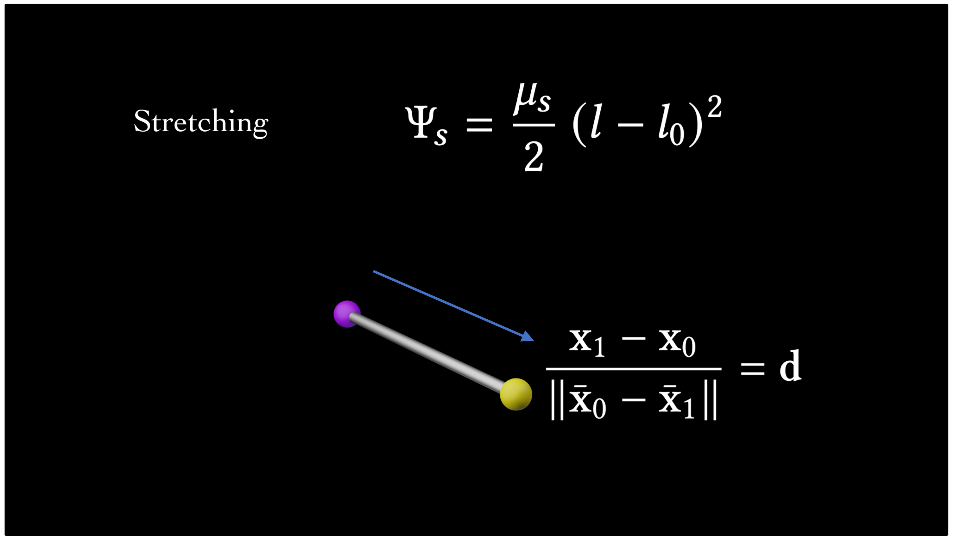
Let's look at the direction along the edge



$$\Psi_s = \frac{\mu_s}{2}(l - l_0)^2$$

$$\frac{\mathbf{x}_1 - \mathbf{x}_0}{\|\bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_1\|}$$

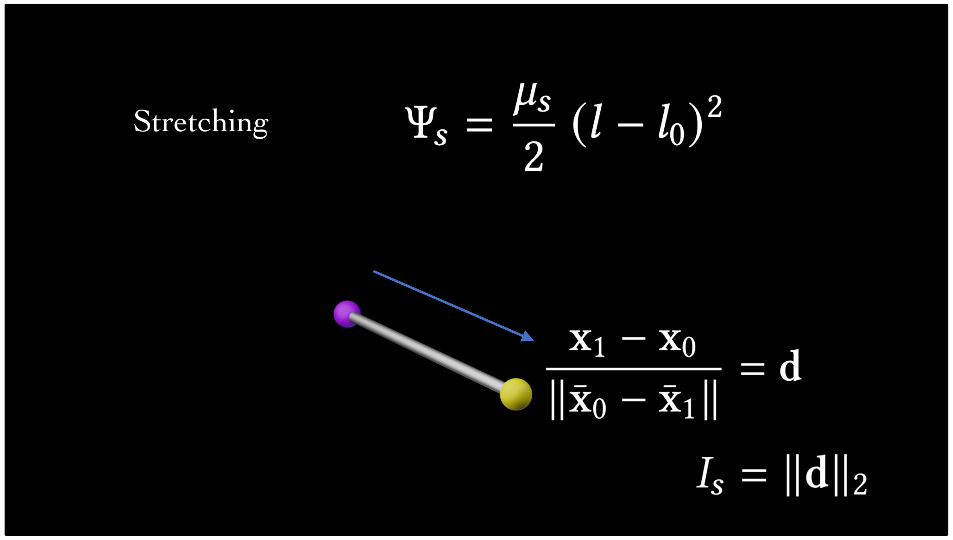
If we normalize it according to the length of the original edge.


Stretching

$$\Psi_s = \frac{\mu_s}{2}\left(l - l_0\right)^2$$

$$\frac{\mathbf{x}_1 - \mathbf{x}_0}{\|\bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_1\|} = \mathbf{d}$$

Then this is a 1D deformation gradient. Let's call it d.



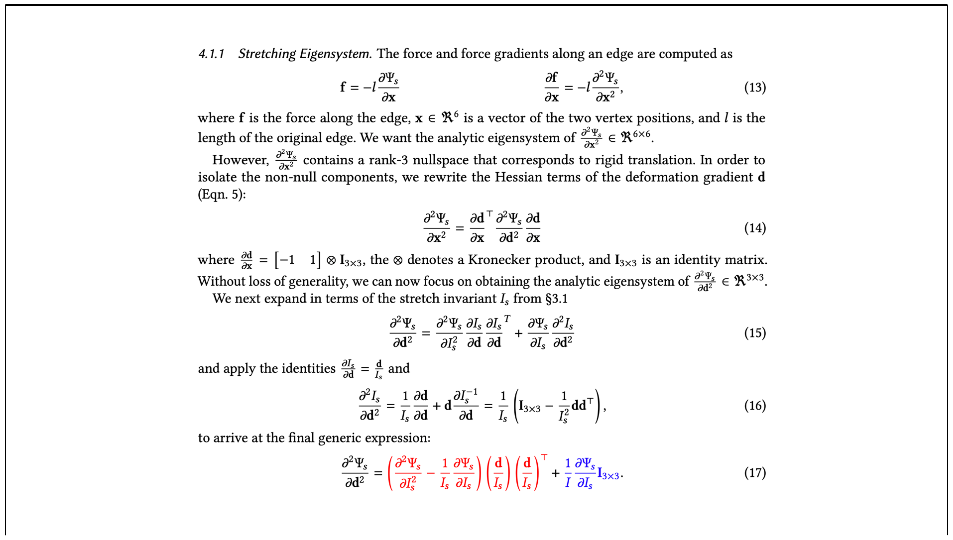
Stretching

$$\Psi_s = \frac{\mu_s}{2}(l - l_0)^2$$

$$\frac{\mathbf{x}_1 - \mathbf{x}_0}{\|\bar{\mathbf{x}}_0 - \bar{\mathbf{x}}_1\|} = \mathbf{d}$$

$$I_s = \|\mathbf{d}\|_2$$

If you then take the norm of this direction, you get a deformation invariant



4.1.1 Stretching Eigensystem. The force and force gradients along an edge are computed as

$$\mathbf{f} = -l\frac{\partial \Psi_s}{\partial \mathbf{x}} \qquad \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = -l\frac{\partial^2 \Psi_s}{\partial \mathbf{x}^2}, \tag{13}$$

where $\mathbf{f}$ is the force along the edge, $\mathbf{x} \in \Re^6$ is a vector of the two vertex positions, and l is the length of the original edge. We want the analytic eigensystem of $\frac{\partial^2 \Psi_s}{\partial \mathbf{x}^2} \in \Re^{6\times 6}$.

However, $\frac{\partial^2 \Psi_s}{\partial \mathbf{x}^2}$ contains a rank-3 nullspace that corresponds to rigid translation. In order to isolate the non-null components, we rewrite the Hessian terms of the deformation gradient $\mathbf{d}$ (Eqn. 5):

$$\frac{\partial^2 \Psi_s}{\partial \mathbf{x}^2} = \frac{\partial \mathbf{d}}{\partial \mathbf{x}}^\top \frac{\partial^2 \Psi_s}{\partial \mathbf{d}^2} \frac{\partial \mathbf{d}}{\partial \mathbf{x}} \tag{14}$$

where $\frac{\partial \mathbf{d}}{\partial \mathbf{x}} = \begin{bmatrix} -1 & 1 \end{bmatrix} \otimes \mathbf{I}_{3\times 3}$, the $\otimes$ denotes a Kronecker product, and $\mathbf{I}_{3\times 3}$ is an identity matrix. Without loss of generality, we can now focus on obtaining the analytic eigensystem of $\frac{\partial^2 \Psi_s}{\partial \mathbf{d}^2} \in \Re^{3\times 3}$.

We next expand in terms of the stretch invariant I_s from §3.1

$$\frac{\partial^2 \Psi_s}{\partial \mathbf{d}^2} = \frac{\partial^2 \Psi_s}{\partial I_s^2} \frac{\partial I_s}{\partial \mathbf{d}} \frac{\partial I_s}{\partial \mathbf{d}}^T + \frac{\partial \Psi_s}{\partial I_s} \frac{\partial^2 I_s}{\partial \mathbf{d}^2} \tag{15}$$

and apply the identities $\frac{\partial I_s}{\partial \mathbf{d}} = \frac{\mathbf{d}}{I_s}$ and

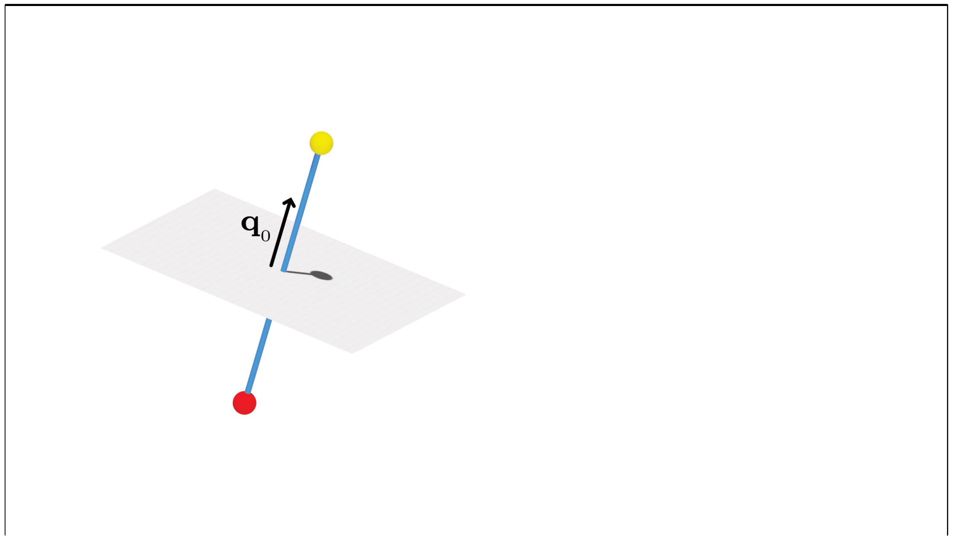
$$\frac{\partial^2 I_s}{\partial \mathbf{d}^2} = \frac{1}{I_s} \frac{\partial \mathbf{d}}{\partial \mathbf{d}} + \mathbf{d} \frac{\partial I_s^{-1}}{\partial \mathbf{d}} = \frac{1}{I_s} \left(\mathbf{I}_{3\times 3} - \frac{1}{I_s^2} \mathbf{d}\mathbf{d}^\top \right), \tag{16}$$

to arrive at the final generic expression:

$$\frac{\partial^2 \Psi_s}{\partial \mathbf{d}^2} = \left(\frac{\partial^2 \Psi_s}{\partial I_s^2} - \frac{1}{I_s} \frac{\partial \Psi_s}{\partial I_s} \right) \left(\frac{\mathbf{d}}{I_s} \right) \left(\frac{\mathbf{d}}{I_s} \right)^\top + \frac{1}{I} \frac{\partial \Psi_s}{\partial I_s} \mathbf{I}_{3\times 3}. \tag{17}$$

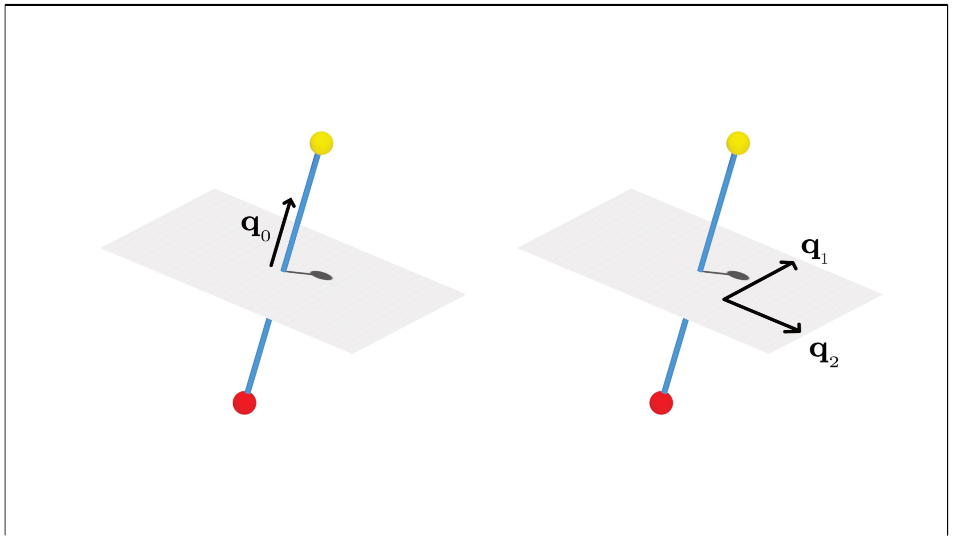
Once you have an invariant, you can do a bunch of analysis that gets you the closed-form eigensystem.

Details are in the paper. I'll just summarize the results here.



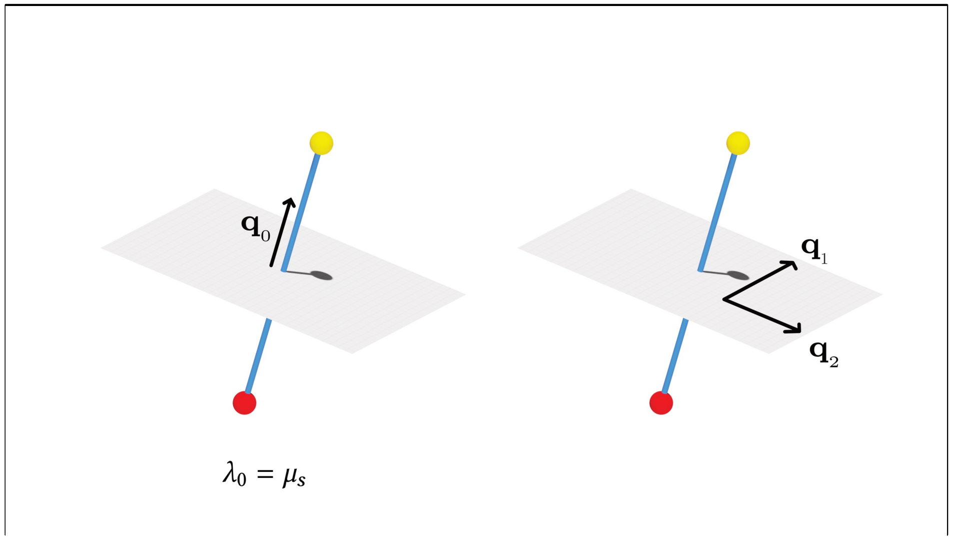
Here’s the first eigenvector.

Quite reasonably, it points along the direction of the edge.

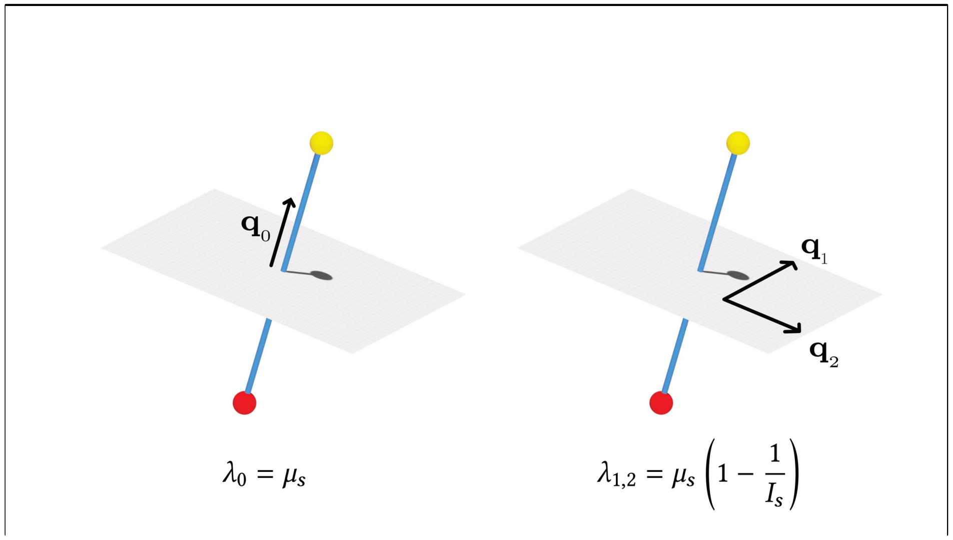


The other two eigenvectors are not unique, but span the plane orthogonal to the edge. This also seems reasonable. For an isotropic rod, there shouldn't be a preferred buckling direction.

Now the eigenvalues.

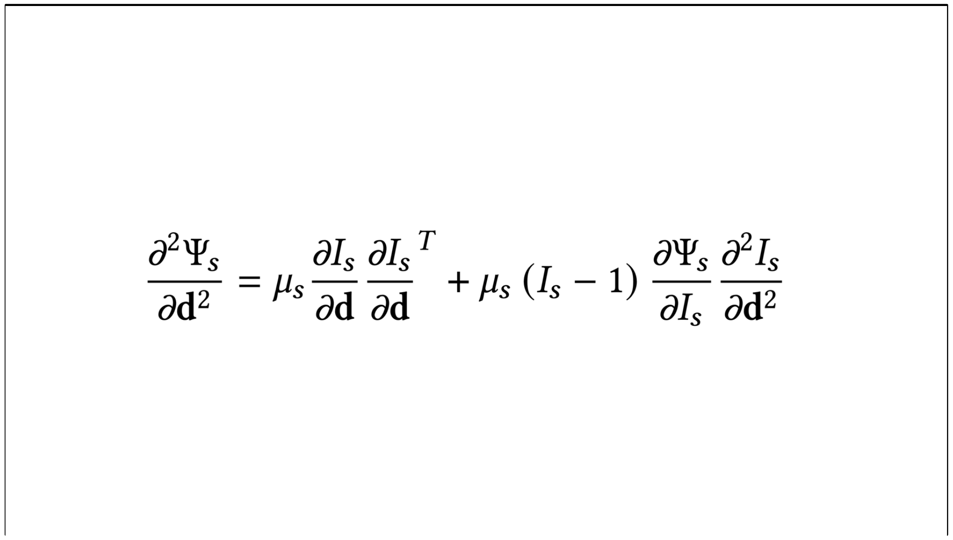


First, the eigenvalue along this edge? It can’t go negative. For our quadratic energy, it’s just a constant.



The eigenvalues for the buckling direction look like this, and only go negative under compression.

This is reasonable, because if you look at my student Haomiao Wu's paper from last year, she saw that this corresponds to the physics of buckling.

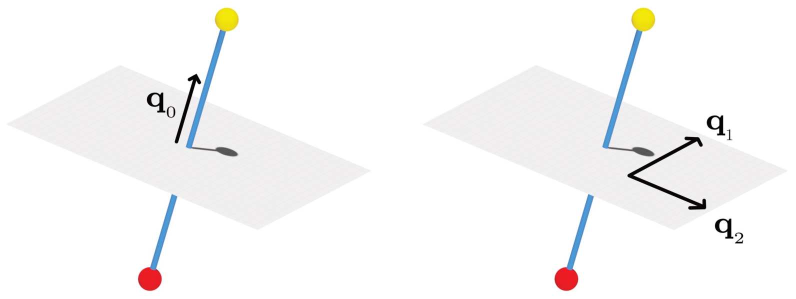


$$\frac{\partial^2 \Psi_s}{\partial \mathbf{d}^2} = \mu_s \frac{\partial I_s}{\partial \mathbf{d}} \frac{\partial I_s}{\partial \mathbf{d}}^T + \mu_s \left(I_s - 1\right) \frac{\partial \Psi_s}{\partial I_s} \frac{\partial^2 I_s}{\partial \mathbf{d}^2}$$

Here’s something interesting. There's a very common approximation that occurs for these kinds of simulations.

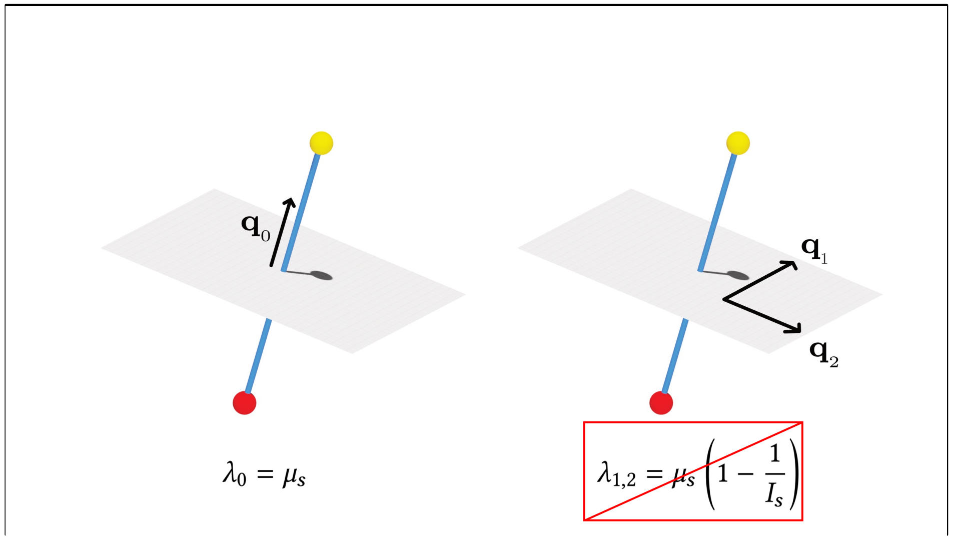
$$\frac{\partial^2 \Psi_s}{\partial \mathbf{d}^2} = \mu_s \frac{\partial I_s}{\partial \mathbf{d}} \frac{\partial I_s}{\partial \mathbf{d}}^T + \underbrace{\boxed{\cancel{\mu_s \left(I_s - 1\right) \frac{\partial \Psi_s}{\partial I_s} \frac{\partial^2 I_s}{\partial \mathbf{d}^2}}}}_{\text{Gauss-Newton}}$$

Gauss-Newton approximation, which amounts to throwing this term away.



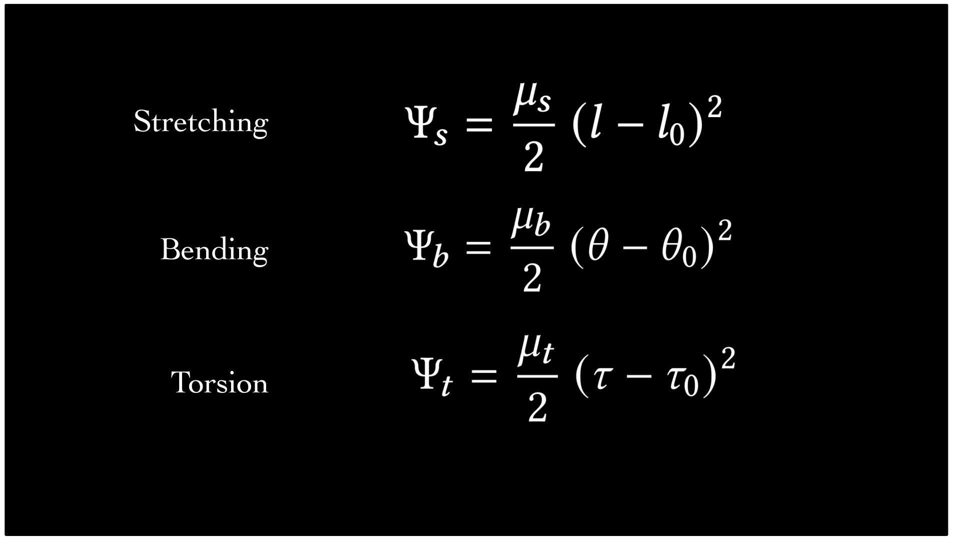
$$\lambda_0 = \mu_s \qquad \lambda_{1,2} = \mu_s\left(1 - \frac{1}{I_s}\right)$$

For the super-specific case of stretching energies,


$\mathbf{q}_0$
$\mathbf{q}_1$
$\mathbf{q}_2$
$\lambda_0 = \mu_s$
$\cancel{\lambda_{1,2} = \mu_s\left(1 - \frac{1}{I_s}\right)}$

Filtering off the negative eigenvalues under compression, exactly equals the Gauss-Newton approximation. That strategy actually *is* the exact eigenvalue filter.

But, a warning: don't use it all the time, just under compression. We'll see examples later.

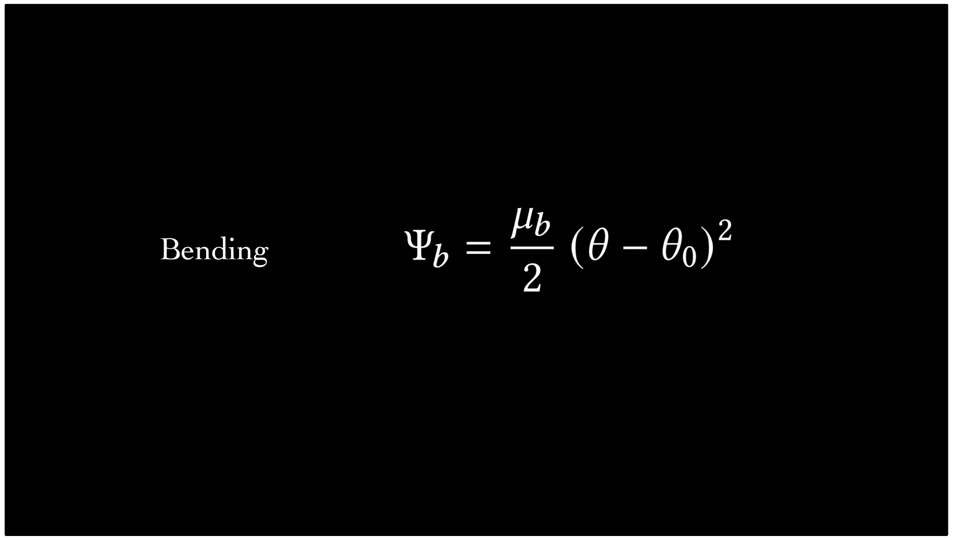


Stretching $\Psi_s = \frac{\mu_s}{2} (l - l_0)^2$

Bending $\Psi_b = \frac{\mu_b}{2} (\theta - \theta_0)^2$

Torsion $\Psi_t = \frac{\mu_t}{2} (\tau - \tau_0)^2$

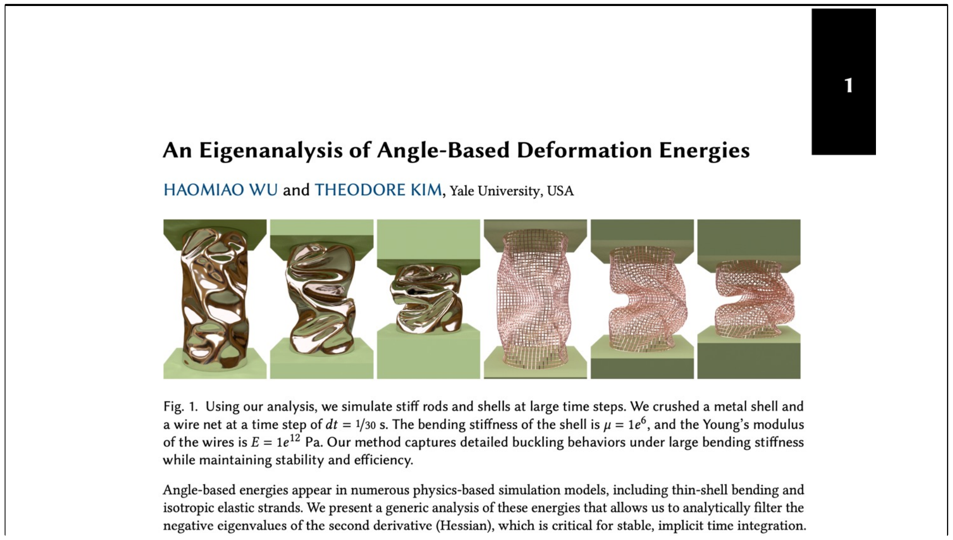
That's it for stretching



Bending

$$\Psi_b = \frac{\mu_b}{2}\left(\theta - \theta_0\right)^2$$

Let's look at bending



1

An Eigenanalysis of Angle-Based Deformation Energies

HAOMIAO WU and THEODORE KIM, Yale University, USA

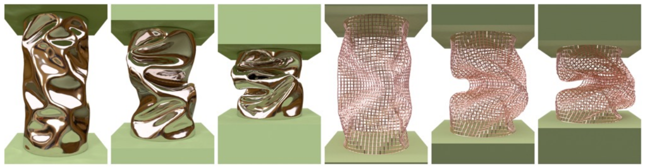
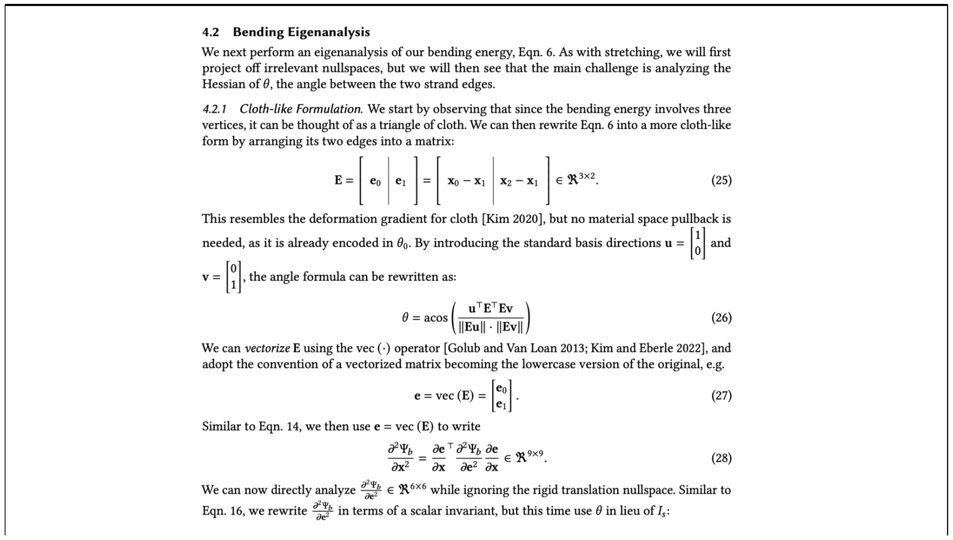


Fig. 1. Using our analysis, we simulate stiff rods and shells at large time steps. We crushed a metal shell and a wire net at a time step of $dt = 1/30$ s. The bending stiffness of the shell is $\mu = 1e^6$, and the Young's modulus of the wires is $E = 1e^{12}$ Pa. Our method captures detailed buckling behaviors under large bending stiffness while maintaining stability and efficiency.

Angle-based energies appear in numerous physics-based simulation models, including thin-shell bending and isotropic elastic strands. We present a generic analysis of these energies that allows us to analytically filter the negative eigenvalues of the second derivative (Hessian), which is critical for stable, implicit time integration.

For this, my student Haomiao Wu did an extremely in-depth analysis of the phenomena in her paper last year.



4.2 Bending Eigenanalysis

We next perform an eigenanalysis of our bending energy, Eqn. 6. As with stretching, we will first project off irrelevant nullspaces, but we will then see that the main challenge is analyzing the Hessian of θ, the angle between the two strand edges.

4.2.1 Cloth-like Formulation. We start by observing that since the bending energy involves three vertices, it can be thought of as a triangle of cloth. We can then rewrite Eqn. 6 into a more cloth-like form by arranging its two edges into a matrix:

$$\mathbf{E} = \left[\begin{array}{c|c} \mathbf{e}_0 & \mathbf{e}_1 \end{array} \right] = \left[\begin{array}{c|c} \mathbf{x}_0 - \mathbf{x}_1 & \mathbf{x}_2 - \mathbf{x}_1 \end{array} \right] \in \Re^{3\times 2}. \tag{25}$$

This resembles the deformation gradient for cloth [Kim 2020], but no material space pullback is needed, as it is already encoded in θ_0. By introducing the standard basis directions $\mathbf{u} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, the angle formula can be rewritten as:

$$\theta = \operatorname{acos}\left(\frac{\mathbf{u}^\top \mathbf{E}^\top \mathbf{E} \mathbf{v}}{\|\mathbf{E}\mathbf{u}\| \cdot \|\mathbf{E}\mathbf{v}\|} \right) \tag{26}$$

We can *vectorize* $\mathbf{E}$ using the vec $(\cdot)$ operator [Golub and Van Loan 2013; Kim and Eberle 2022], and adopt the convention of a vectorized matrix becoming the lowercase version of the original, e.g.

$$\mathbf{e} = \operatorname{vec}(\mathbf{E}) = \begin{bmatrix} \mathbf{e}_0 \\ \mathbf{e}_1 \end{bmatrix}. \tag{27}$$

Similar to Eqn. 14, we then use $\mathbf{e} = \operatorname{vec}(\mathbf{E})$ to write

$$\frac{\partial^2 \Psi_b}{\partial \mathbf{x}^2} = \frac{\partial \mathbf{e}}{\partial \mathbf{x}}^\top \frac{\partial^2 \Psi_b}{\partial \mathbf{e}^2} \frac{\partial \mathbf{e}}{\partial \mathbf{x}} \in \Re^{9\times 9}. \tag{28}$$

We can now directly analyze $\frac{\partial^2 \Psi_b}{\partial \mathbf{e}^2} \in \Re^{6\times 6}$ while ignoring the rigid translation nullspace. Similar to Eqn. 16, we rewrite $\frac{\partial^2 \Psi_b}{\partial \mathbf{e}^2}$ in terms of a scalar invariant, but this time use θ in lieu of I_s:

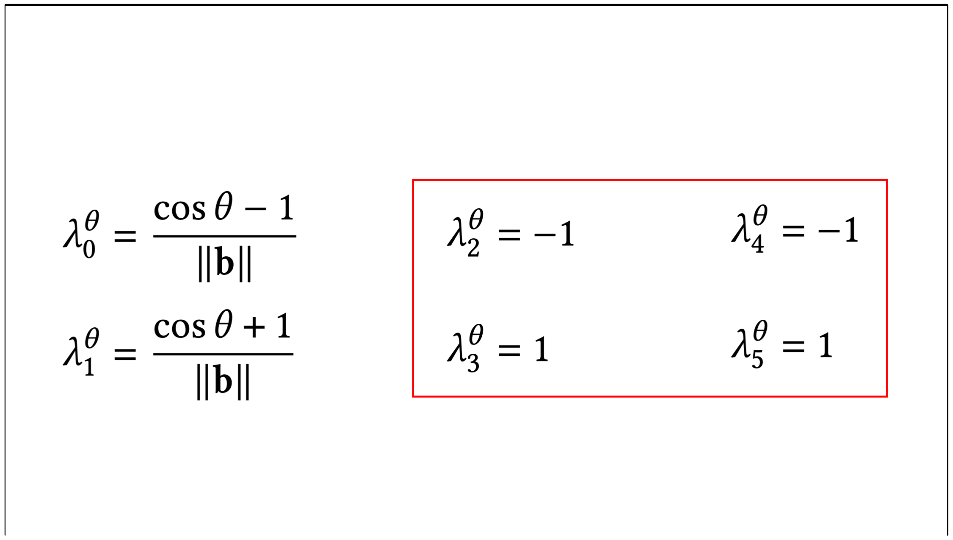
We actually present an alternative analysis in our Lifted Curls paper, based on the special case where the edges are unit length.

Under this approach, you can get some *extremely* simple results.

$$\lambda_0^\theta = \frac{\cos\theta - 1}{\|\mathbf{b}\|} \qquad \lambda_2^\theta = -1 \qquad \lambda_4^\theta = -1$$

$$\lambda_1^\theta = \frac{\cos\theta + 1}{\|\mathbf{b}\|} \qquad \lambda_3^\theta = 1 \qquad \lambda_5^\theta = 1$$

When I mean simple, I mean really simple.



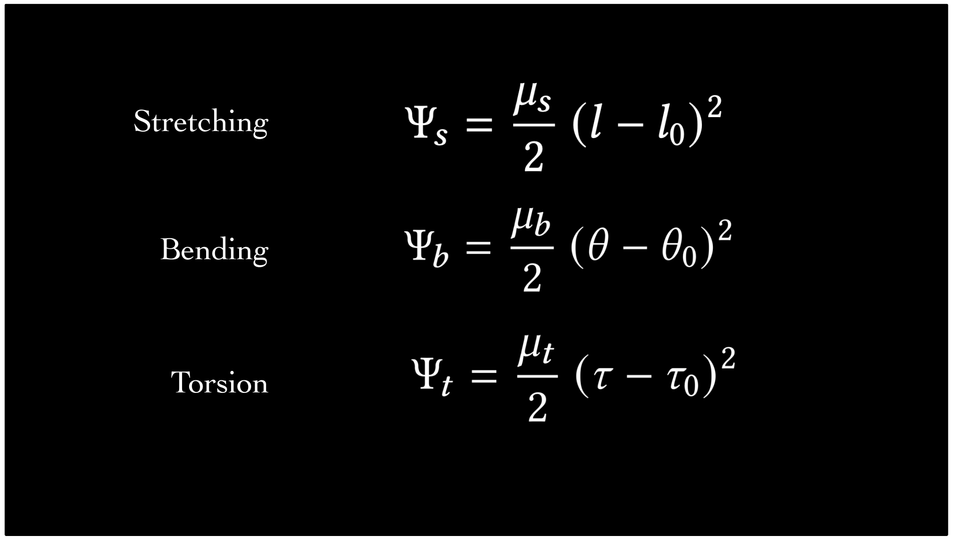
$$\lambda_0^\theta = \frac{\cos\theta - 1}{||\mathbf{b}||}$$

$$\lambda_1^\theta = \frac{\cos\theta + 1}{||\mathbf{b}||}$$

$$\boxed{\begin{array}{ll} \lambda_2^\theta = -1 & \lambda_4^\theta = -1 \\ \lambda_3^\theta = 1 & \lambda_5^\theta = 1 \end{array}}$$

Under this different coordinate system, some of the eigenvalues work out to 1 and -1. Hard to get simpler than that.

You then have to do a little work to multiply the edge lengths back in, but I refer you to the paper for details.

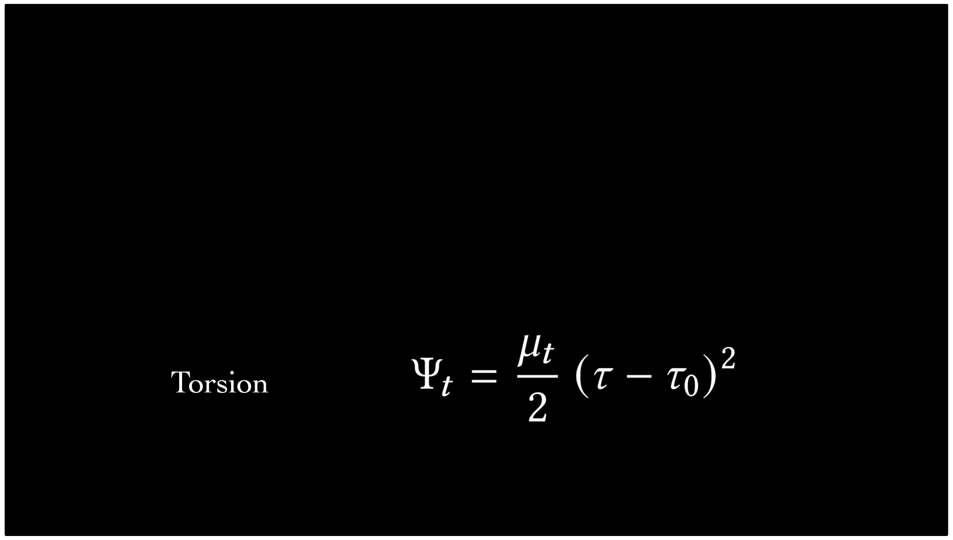


Stretching $$\Psi_s = \frac{\mu_s}{2}\left(l - l_0\right)^2$$

Bending $$\Psi_b = \frac{\mu_b}{2}\left(\theta - \theta_0\right)^2$$

Torsion $$\Psi_t = \frac{\mu_t}{2}\left(\tau - \tau_0\right)^2$$

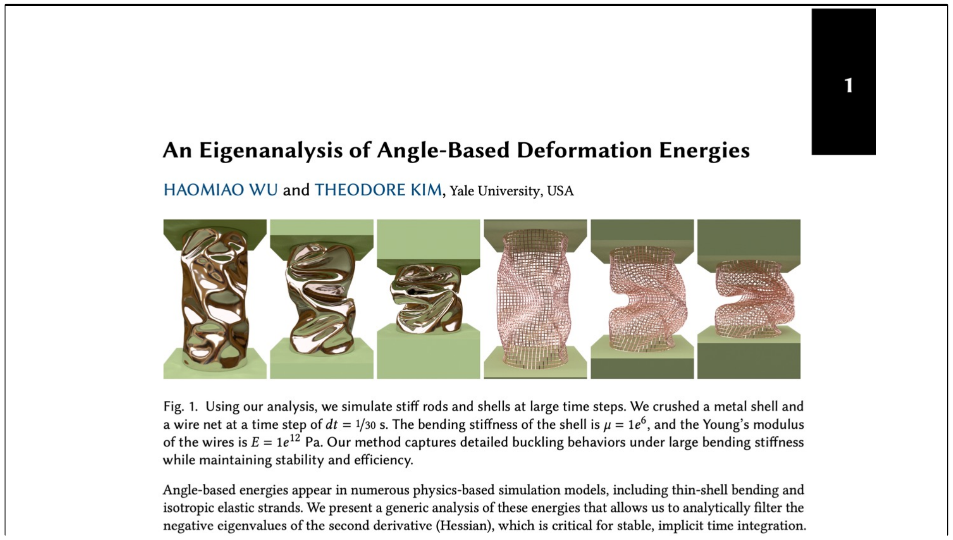
Okay, one energy left



Torsion

$$\Psi_t = \frac{\mu_t}{2}\left(\tau - \tau_0\right)^2$$

Okay, one energy left



1

An Eigenanalysis of Angle-Based Deformation Energies

HAOMIAO WU and THEODORE KIM, Yale University, USA

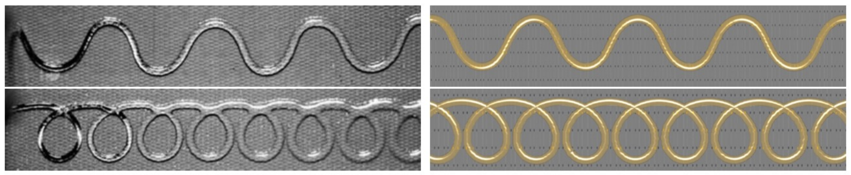
Fig. 1. Using our analysis, we simulate stiff rods and shells at large time steps. We crushed a metal shell and a wire net at a time step of $dt = 1/30$ s. The bending stiffness of the shell is $\mu = 1e^6$, and the Young's modulus of the wires is $E = 1e^{12}$ Pa. Our method captures detailed buckling behaviors under large bending stiffness while maintaining stability and efficiency.

Angle-based energies appear in numerous physics-based simulation models, including thin-shell bending and isotropic elastic strands. We present a generic analysis of these energies that allows us to analytically filter the negative eigenvalues of the second derivative (Hessian), which is critical for stable, implicit time integration.

Since it's angle based, we can again apply the analysis from Haomiao's paper, or the alternative from our current Lifted Curls paper.

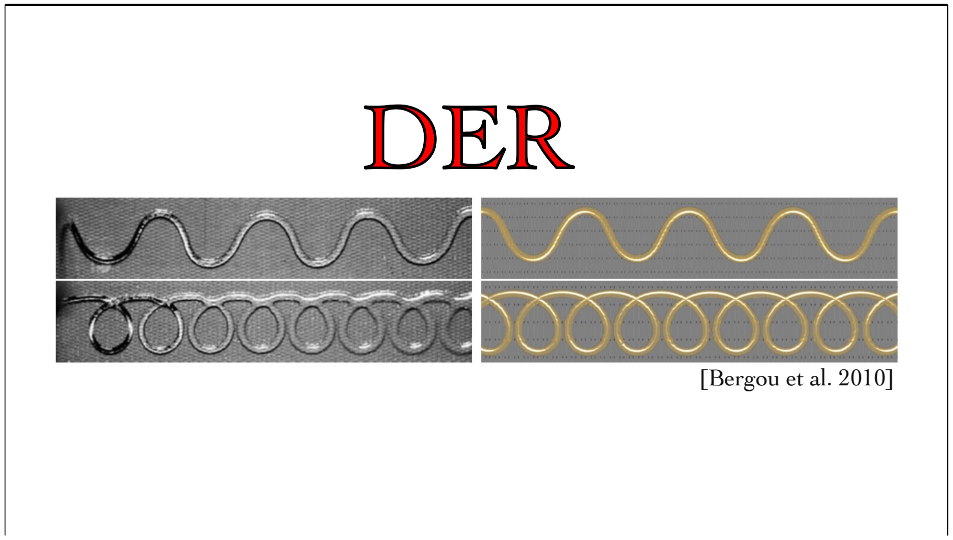
Comparisons and Results

Lets get to comparisons and results

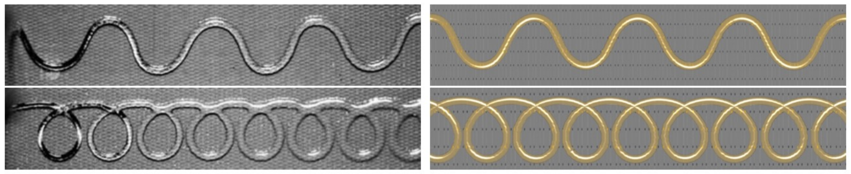

[Bergou et al. 2010]

I promised we'd get to Discrete Viscous Threads and elastic rods, and we're going to look at that now.

It's a popular model, so it's important to compare and contrast.

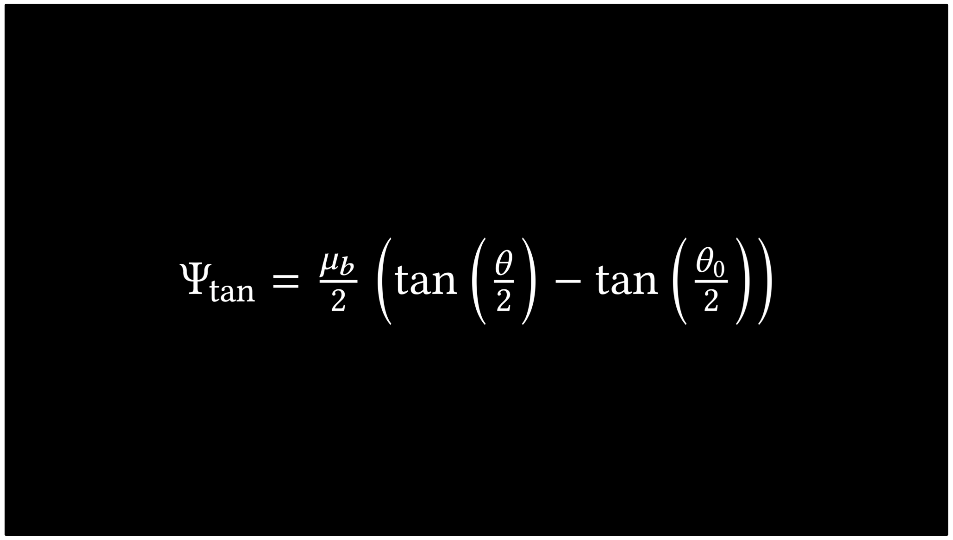


DER



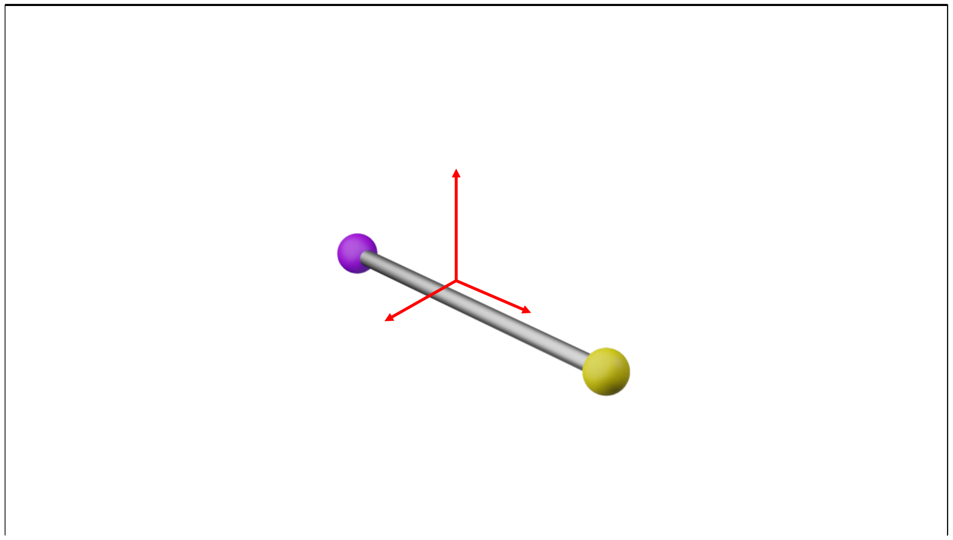
[Bergou et al. 2010]

For simplicity, I'll call it DER.

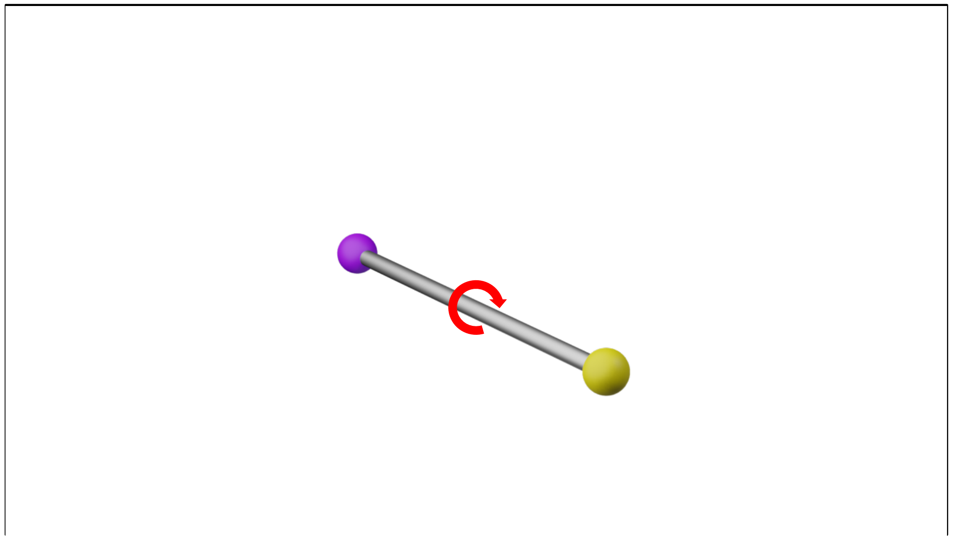


$$\Psi_{\tan} = \frac{\mu_b}{2}\left(\tan\left(\frac{\theta}{2}\right) - \tan\left(\frac{\theta_0}{2}\right)\right)$$

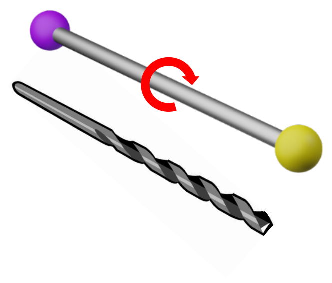
DER has a tangent-based bending energy, which goes to infinity at large angles.



DER is a Cosserat model that has a per-edge frame.



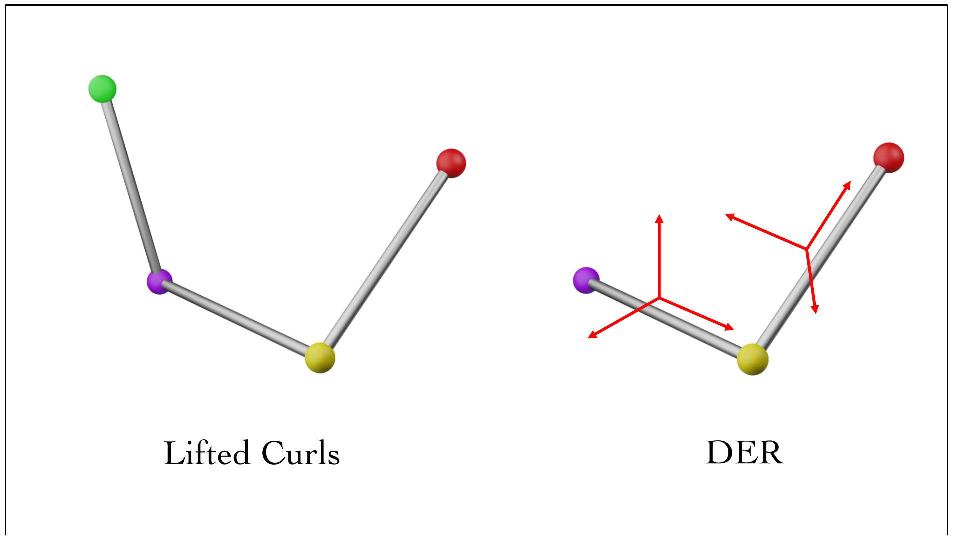
Our model doesn't have that, and the main thing we lose is subsegment twist.



If the edge twists like a drill bit, our model will miss it.

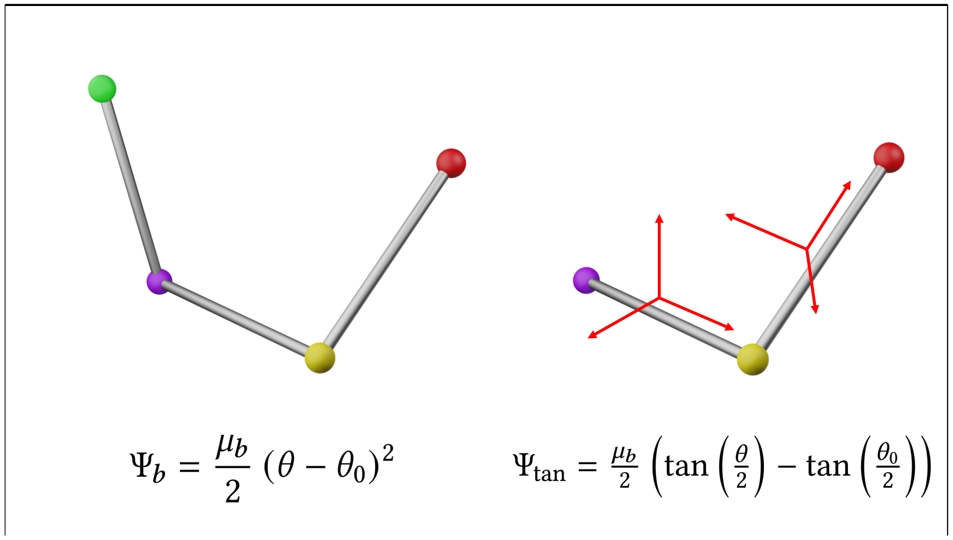
This mostly happens when strands are constrained at both ends.

But Hair is usually constrained at one end – the scalp. So, we're not too worried about this.



To summarize, Lifted Curls only uses vertex positions,

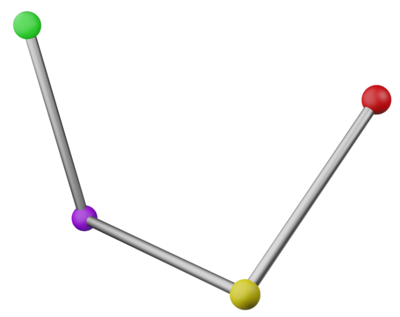
DER uses vertices and frames.

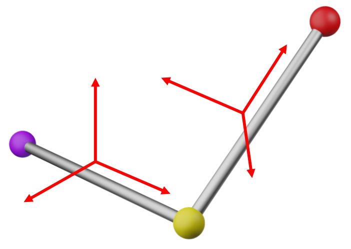


$$\Psi_b = \frac{\mu_b}{2}\left(\theta - \theta_0\right)^2$$

$$\Psi_{\tan} = \frac{\mu_b}{2}\left(\tan\left(\frac{\theta}{2}\right) - \tan\left(\frac{\theta_0}{2}\right)\right)$$

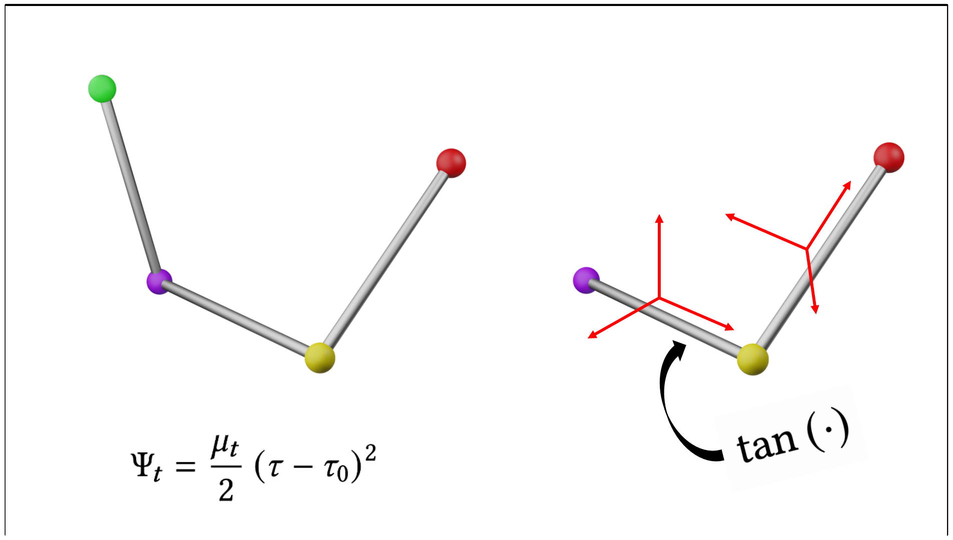
We use quadratic bending, while DER uses tangent bending.





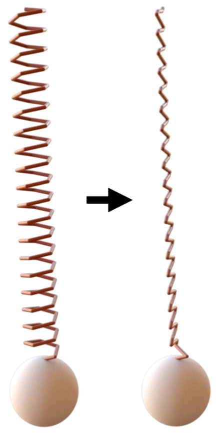
$$\Psi_t = \frac{\mu_t}{2}\left(\tau - \tau_0\right)^2$$

We use quadratic torsion

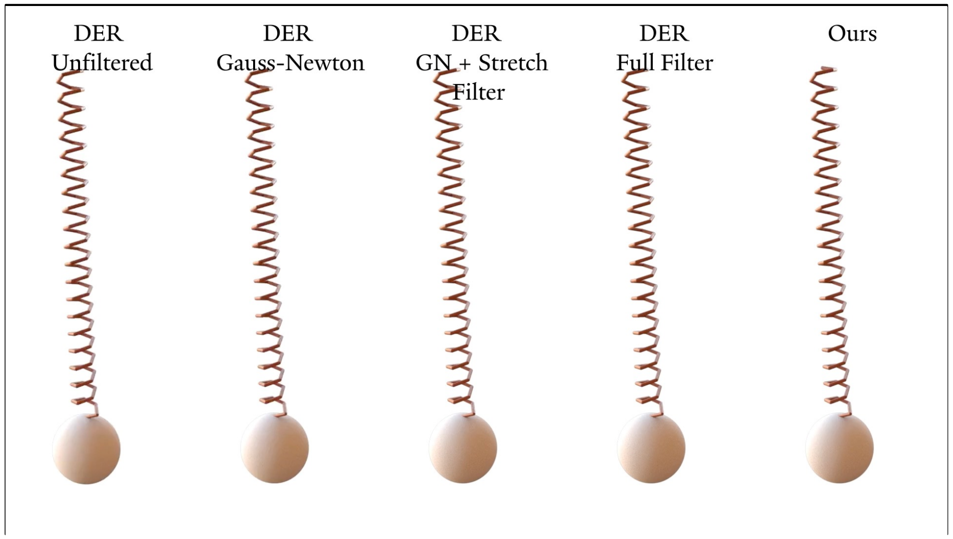


While if you dig through the math, DER's twisting energy is implicitly another tangent function.

Functions with higher non-linearity usually introduce stability challenges.



To test things out, let's see what happens when we kinematically compress a coil.

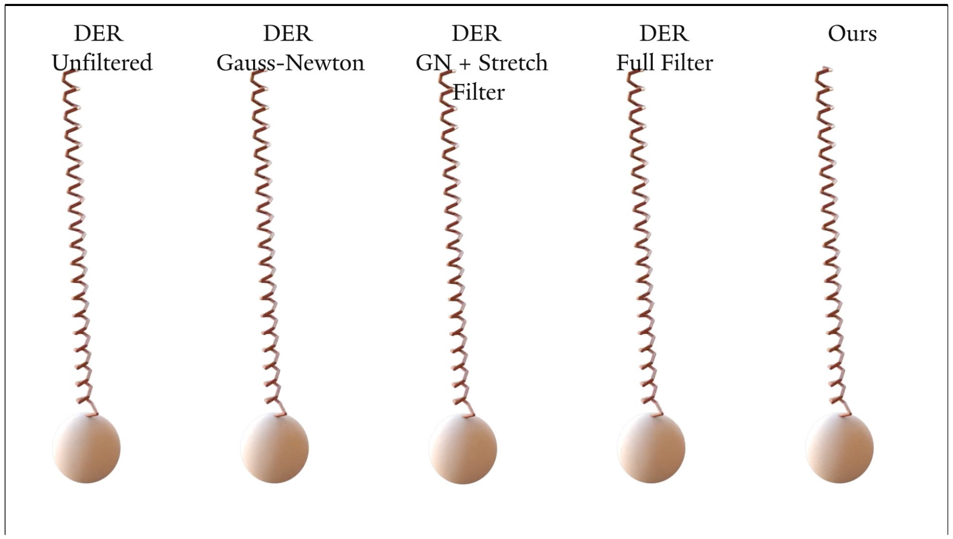


Let’s start small and compress it by 30%. You can barely see it.

For all these tests we wanted to see behavior under large timesteps, so this is 1/30 of a second, using 3 Newton iterations.

We tried a lot of different filtering strategies for DER. No filtering, Gauss-Newton, exact filtering, and then Gauss-Newton but with stretching filtered using our method. Remember, most of the stiffness lives in that term anyway.

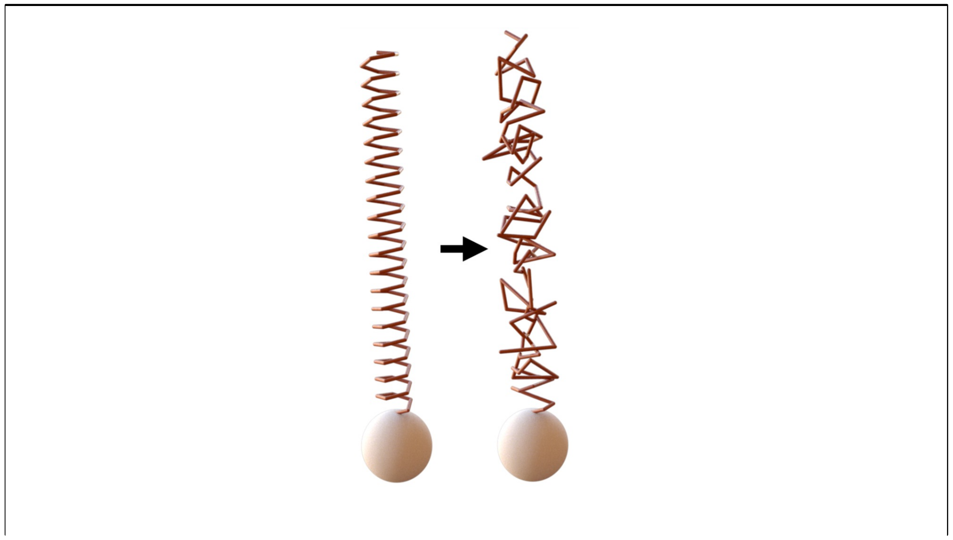
A bunch of the strategies fail, and some survive. Most importantly, ours survives.



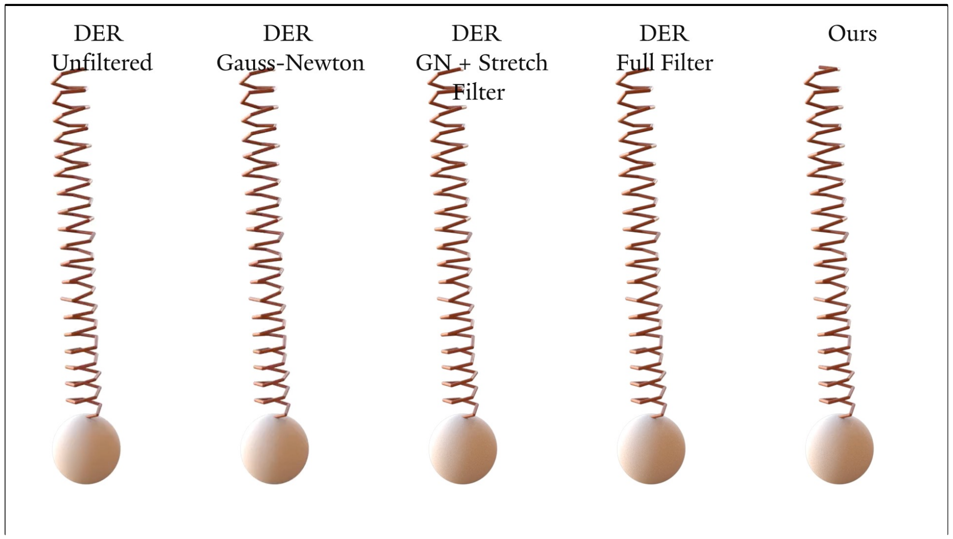
If we double it to 60% compression, all the DER strategies fail.

Just for fun, we compressed our by 99.999%. Still recovered. Quadratic energies are super-robust.

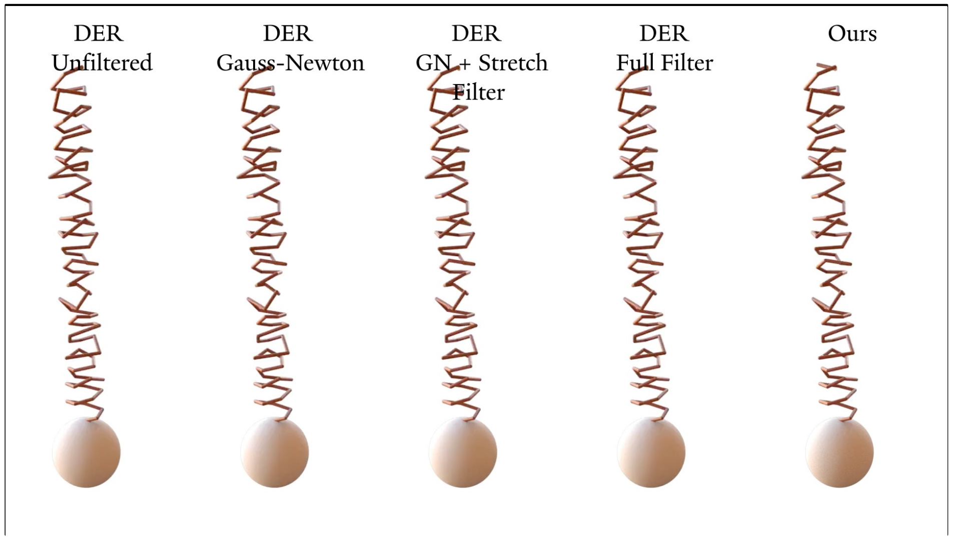
To be fair, DER does recover under smaller timesteps. But, we found you have to make it 100X to 10,000X smaller. When you're approaching the singularity in the tangent function, the solver really has to tiptoe around.



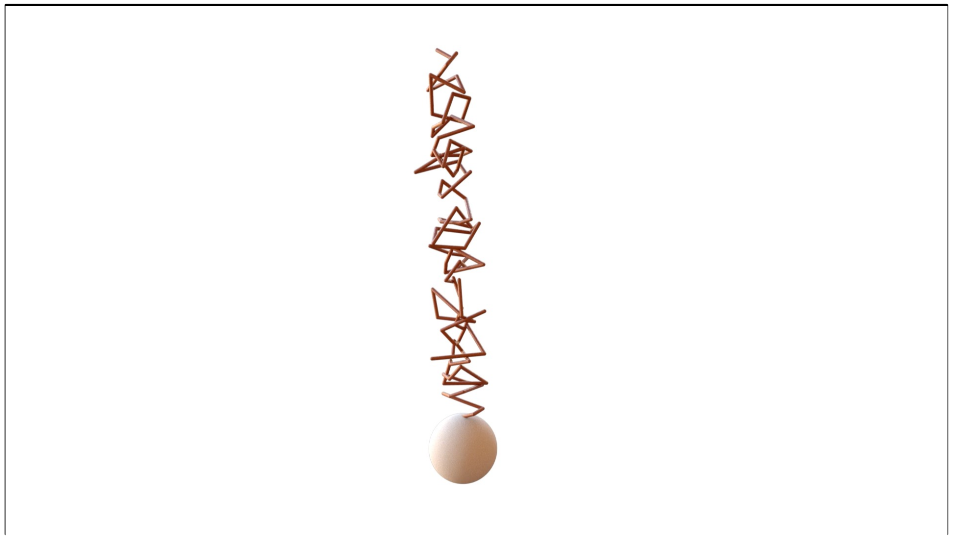
Instead of compressing a coil, let’s jitter its vertices.



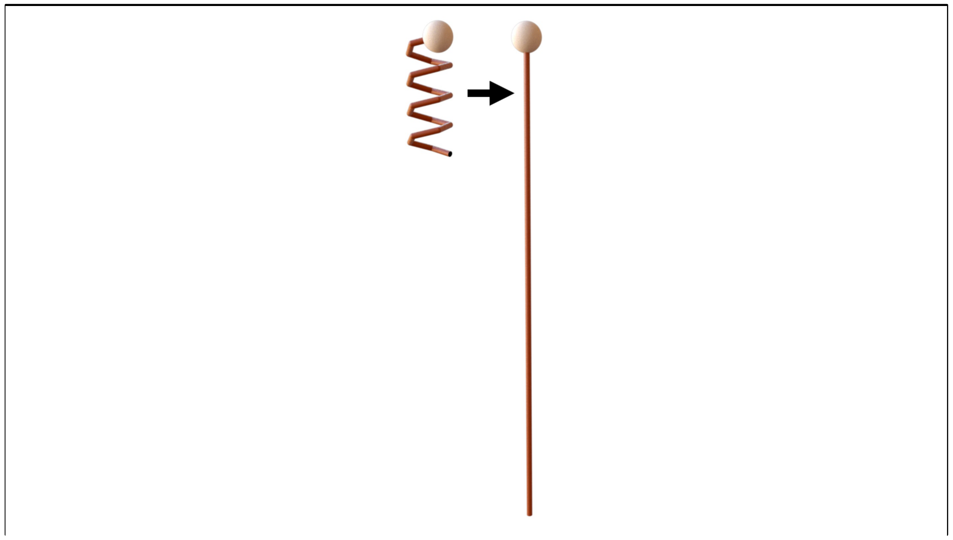
We jiggled each vertex by 0.5mm, and again, some recover, and some fail.



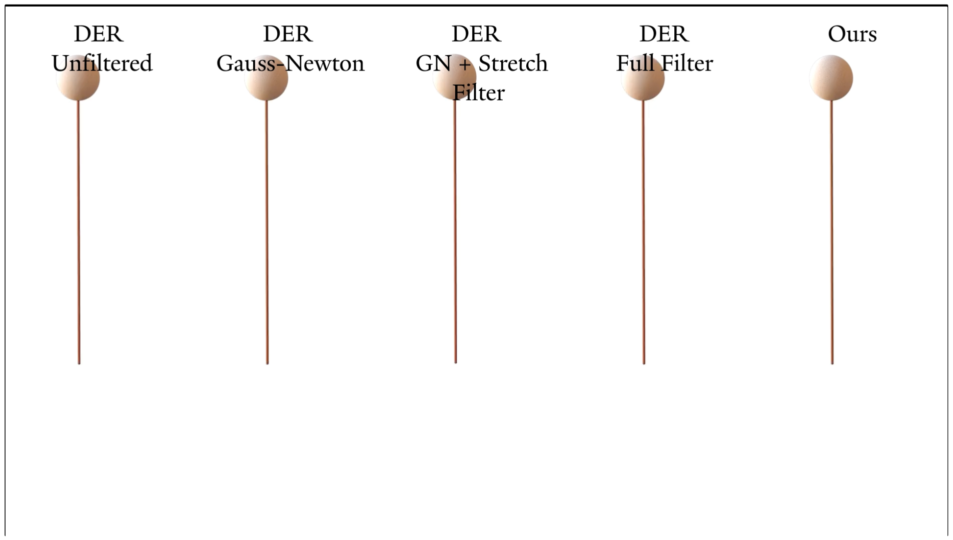
Push to 2mm, everybody fails but us. Again, you need a 100X or 100,000X smaller timestep to stabilize DER.



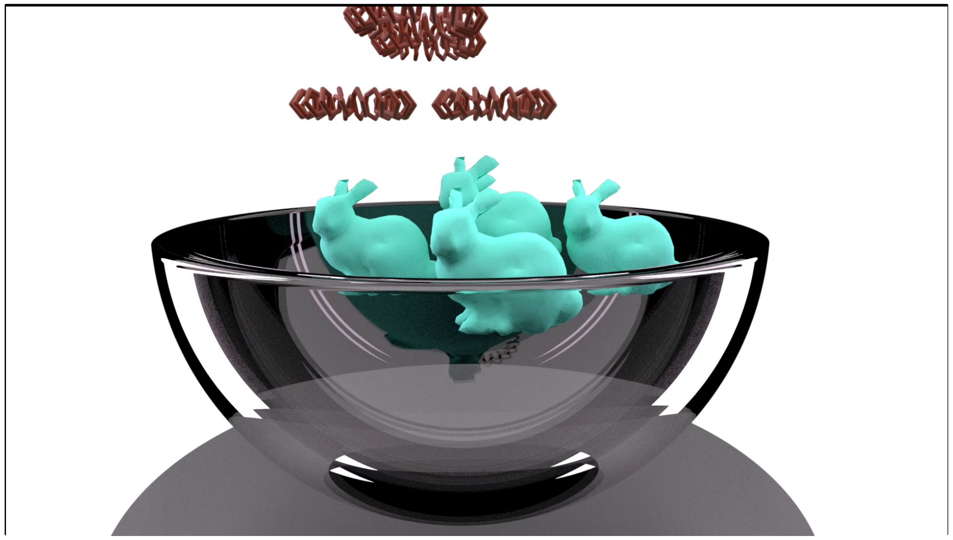
Just for fun, we jiggled ours by 5mm. It recovered fine.



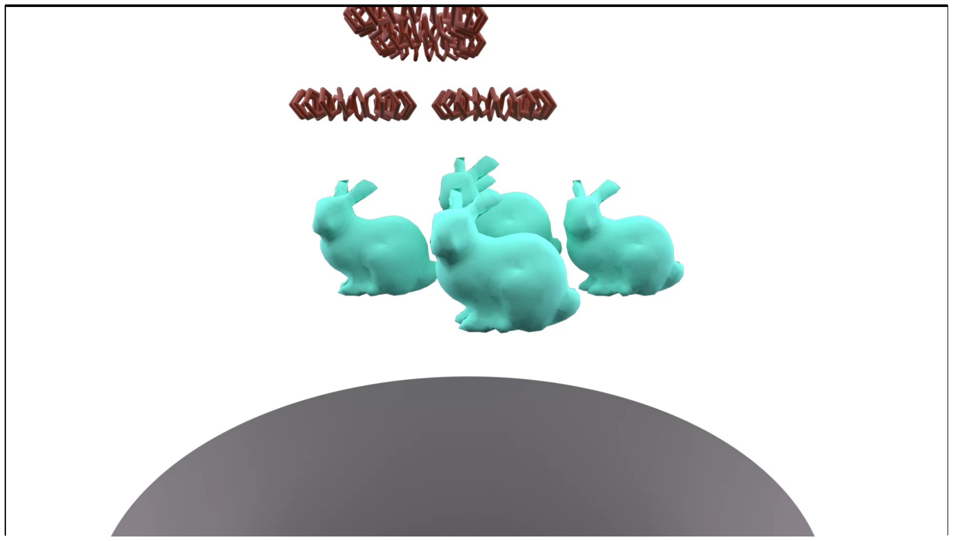
Another test. Let’s straighten a coil out entirely.



Again, ours bounces right back. The other ones fail. Again, you need to dial the timestep back by 100X to 10,000X before DER stabilizes.



Since the formulation is entirely vertex-based, you can throw it in with existing simulators. For example, here’s a bunch of volumetric bunnies, and we threw some hair ties on top of them.



Same shot, no bowl

[Kaufman et al. 2014]

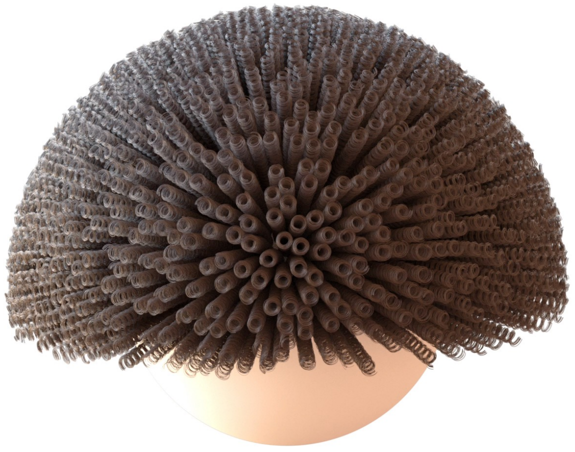
[Han et al. 2019]

[Daviet 2020]

[Daviet 2023]

The thing we care most about is hair. Everybody seems to use a hairball to test their algorithms, though it's always been straight hair.

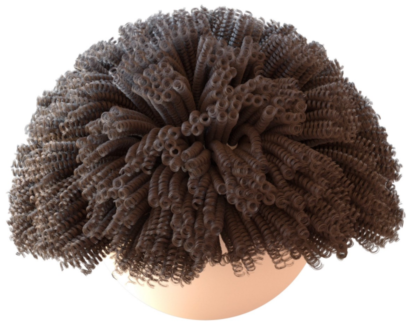
We're going to use one too, but for tightly coiled hiar.

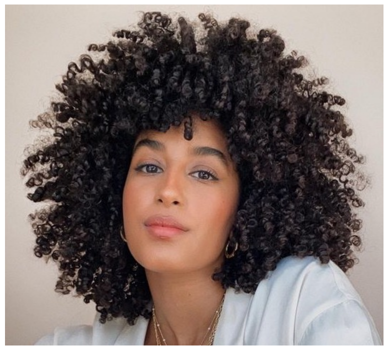


Here’s a hairball with 2000 wisps that we directly simulate. Each wisp is then rendered as 100 hairs.

This is all with a really large timestep, 1/30 of a second, and 3 Newton iterations. I’ll show you more Newton iterations in a second.

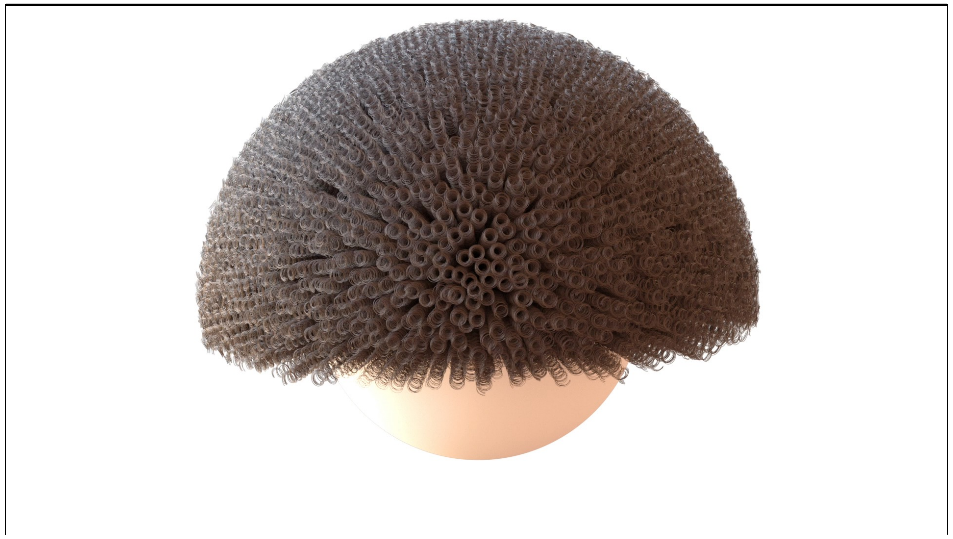
Collisions are enabled, so clumps automatically form between the wisps.





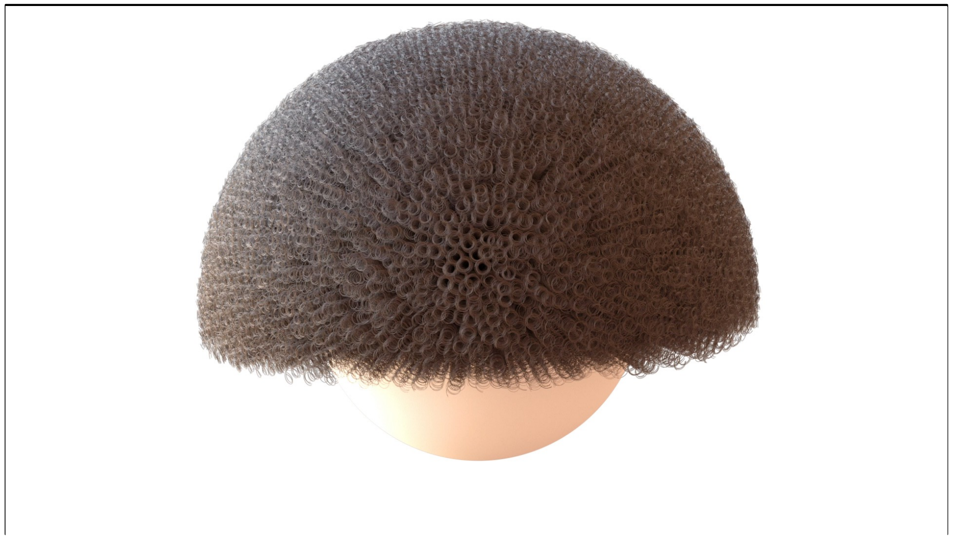
Leal Alexander
https://www.instagram.com/curlygallal/

This happens in real life. Here's the Instagram influencer Leal Alexander, and you can see the same clumps in her hair.

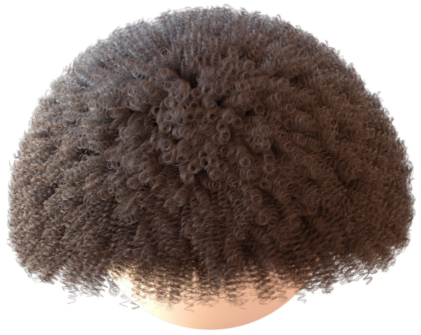


Here we simulated 4K wisps, and rendered each as 50 hairs. So, same number of total hairs, but twice the number of independent wisps.

At this level, clumps stop forming.



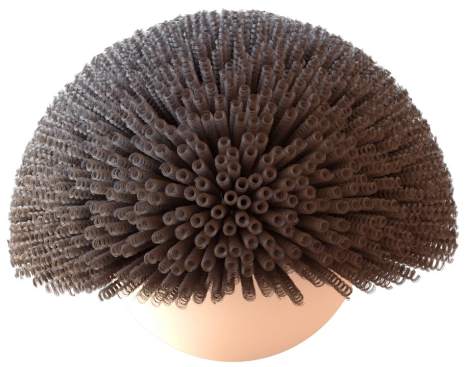
Finally, 8K wisps, rendered each with 25 hairs. A much more “picked out” look.



Leal Alexander
https://www.instagram.com/curlygallal/

This does match real-life. Again here is Leal Alexander with the same hair, but she picked it out. Clumps no longer form, and the look is fuzzier.

I just showed you 3 Newton iterations, to make the point is that it's stable.


3 iterations

12 iterations

I just showed you 3 Newton iterations, to make the point is that it's stable.

If you add more iterations, you get more dynamics. Here's the 2K example, but with 12 Newton iterations instead of 3

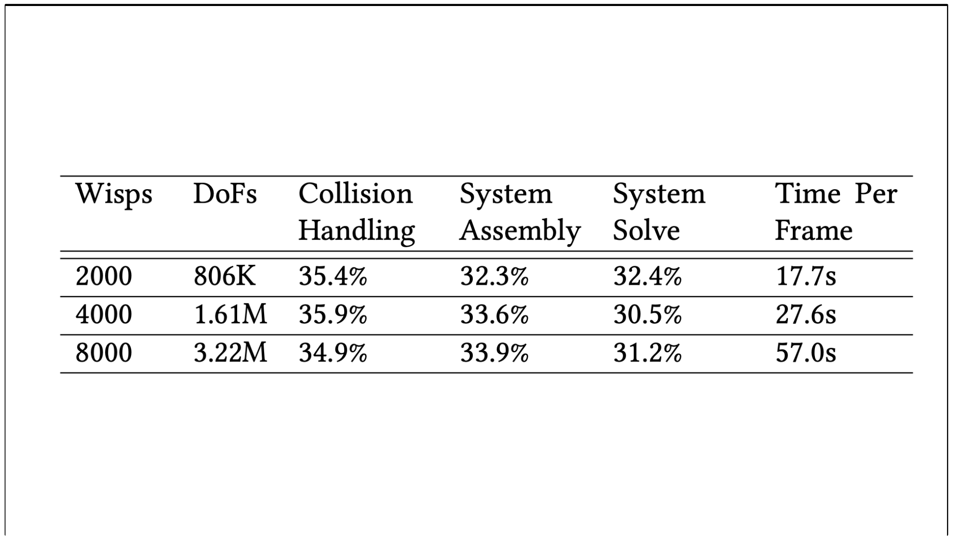
3 iterations
12 iterations

Here's the 4K example again

3 iterations

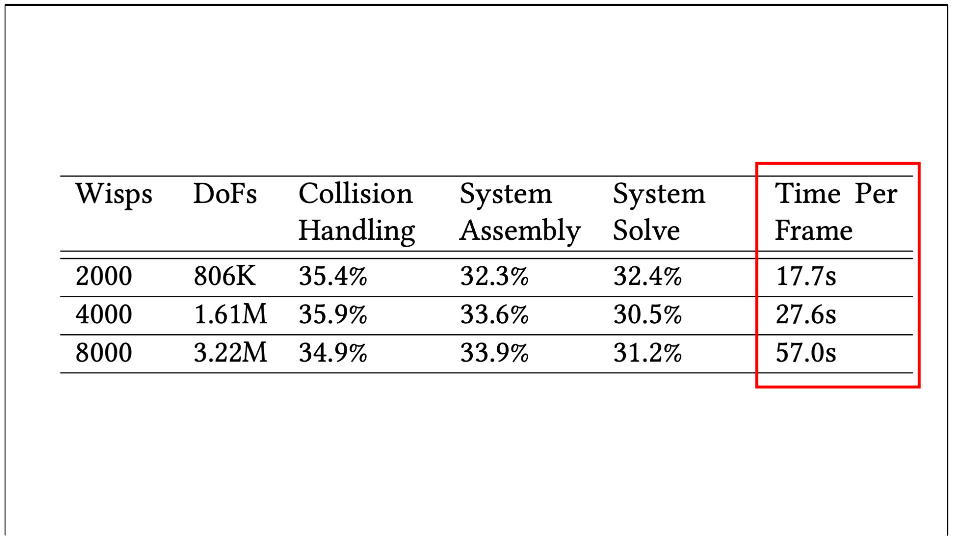
12 iterations

And here's the 8K example again.



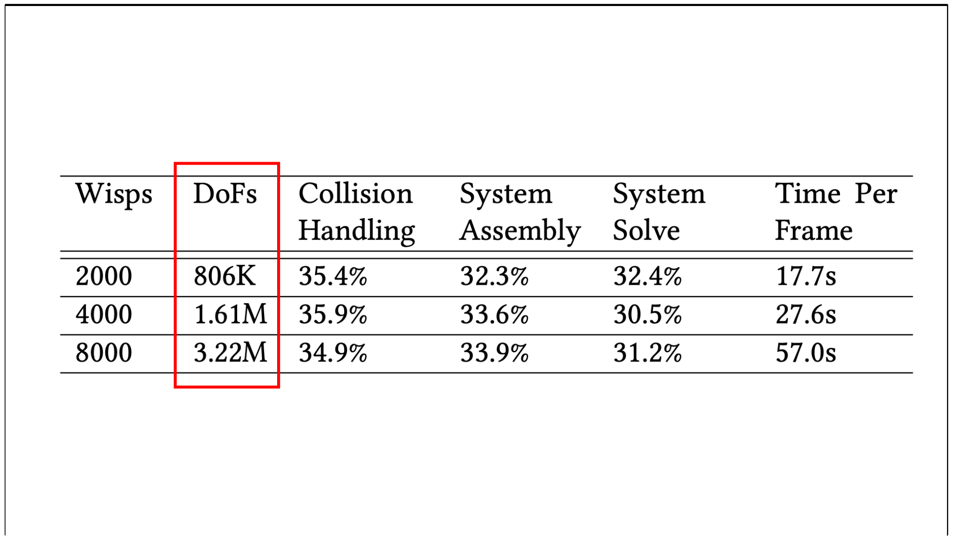
Wisps	DoFs	Collision Handling	System Assembly	System Solve	Time Per Frame
2000	806K	35.4%	32.3%	32.4%	17.7s
4000	1.61M	35.9%	33.6%	30.5%	27.6s
8000	3.22M	34.9%	33.9%	31.2%	57.0s

Here’s the running times we saw.



Wisps	DoFs	Collision Handling	System Assembly	System Solve	Time Per Frame
2000	806K	35.4%	32.3%	32.4%	17.7s
4000	1.61M	35.9%	33.6%	30.5%	27.6s
8000	3.22M	34.9%	33.9%	31.2%	57.0s

The total running times are here on the right, which might seem a little high



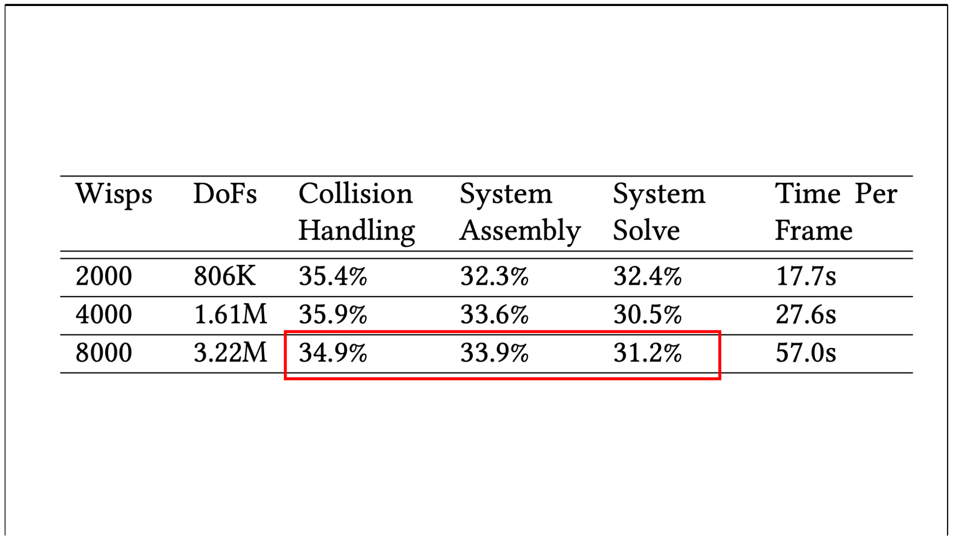
Wisps	DoFs	Collision Handling	System Assembly	System Solve	Time Per Frame
2000	806K	35.4%	32.3%	32.4%	17.7s
4000	1.61M	35.9%	33.6%	30.5%	27.6s
8000	3.22M	34.9%	33.9%	31.2%	57.0s

But we're looking at lots of degrees of freedom – each wisp has LOTS of vertices, way more than straight hair.

And, this is not some optimized GPU implementation.

You can take our energy and try to throw it onto the GPU or come up with a position-based dynamics formulation. Lots of opportunities there.

(next paper only has 81K DoFs)



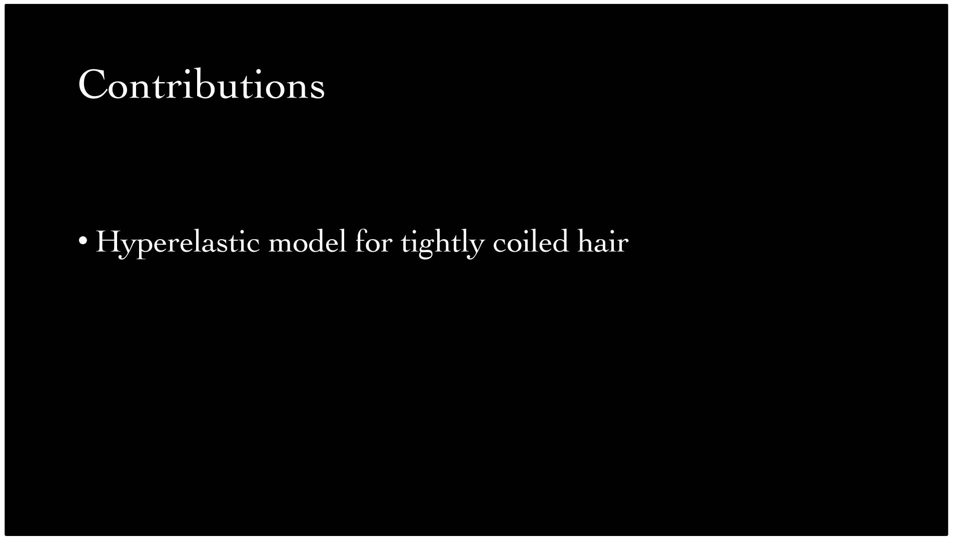
Wisps	DoFs	Collision Handling	System Assembly	System Solve	Time Per Frame
2000	806K	35.4%	32.3%	32.4%	17.7s
4000	1.61M	35.9%	33.6%	30.5%	27.6s
8000	3.22M	34.9%	33.9%	31.2%	57.0s

This is a relatively unoptimized CPU implementation, so you see the classic 1/3 1/3 1/3 split here between collisions, assembly, and solve.

If you want speed, you could take our energy and try to throw it onto the GPU or come up with a position-based dynamics formulation. Lots of opportunities there.

Contributions

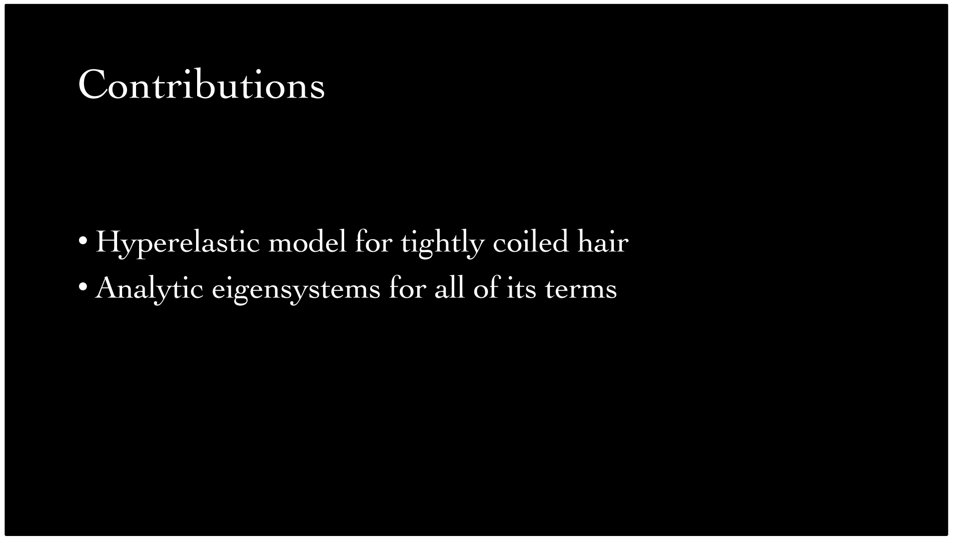
Alright, let’s wrap it up.



Contributions

• Hyperelastic model for tightly coiled hair

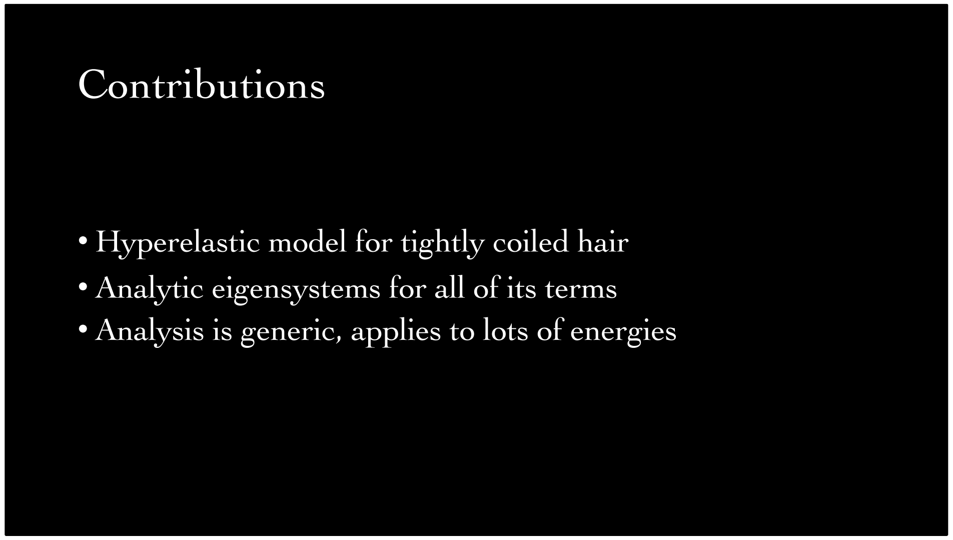
we've presented a new hyperelastic model for tightly coiled hair



Contributions

- Hyperelastic model for tightly coiled hair
- Analytic eigensystems for all of its terms

We've derived analytic eigensystems for all of its terms



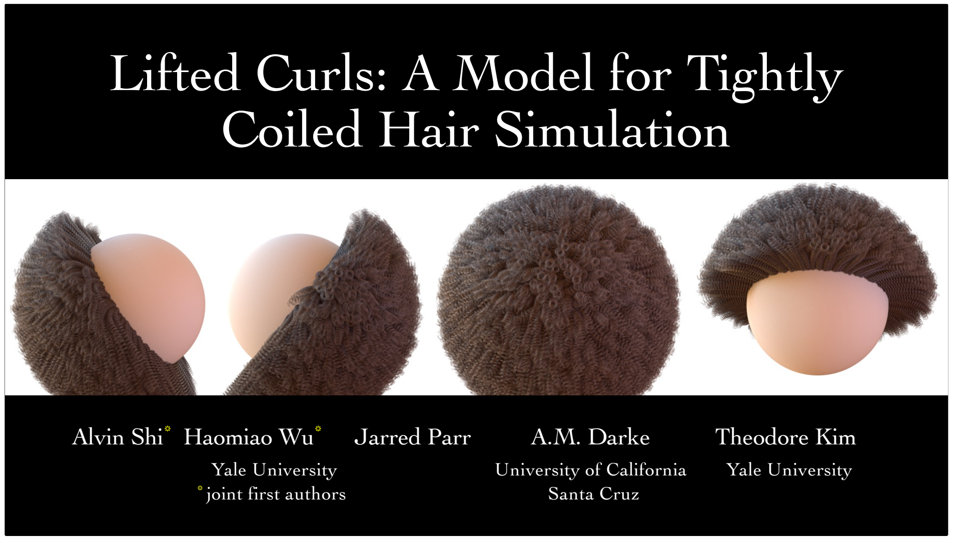
Contributions

- Hyperelastic model for tightly coiled hair
- Analytic eigensystems for all of its terms
- Analysis is generic, applies to lots of energies

And the analysis is generic, and applies to lots of other energies

We even saw it stabilize DVT quite a bit.

Still lots of room for speedup and optimization.



Lifted Curls: A Model for Tightly Coiled Hair Simulation

Alvin Shi[°] Haomiao Wu[°] Jarred Parr A.M. Darke Theodore Kim

Yale University University of California Santa Cruz Yale University

[°]joint first authors

So -- that’s Haomiao and Alvin’s paper.

Acknowledgements

- The Bungie Foundation
- The Teng and Han Family Fund
- Adobe Research
- NSF IIS-2132280

Thanks to all of our generous sponsors for funding this work.

Thank You

And thank you for listening. I'd be happy to take questions now.