Chapter 2

Deformation Fundamentals

2.1 Do I Need to Read This Chapter?

If you are brand-new to deformation simulation, or are coming back to it after some time away, the answer is: yes.

There are several existing introductions to deformation geared towards computer graphics, including the classic Witkin and Baraff (1997) as well as the more modern Sifakis and Barbic (2012) or Bargteil and Shinar (2018). If you are a practitioner that is already familiar with those materials, you can probably skip most of this chapter. The one caveat is that many of the fast, compact structures we show later on will depend on the notation for higher order tensors from Golub and Van Loan (2013). This does not show up that often in computer graphics, so you may still want to read Chapter 3.

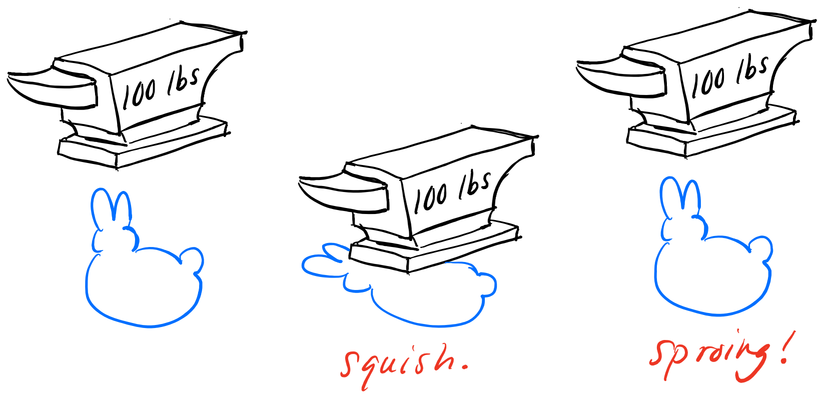

Figure 2.1.: The behavior of a hyperelastic bunny. Even when it is crushed by a 100 lb. weight, it recovers its original shape afterwards.

2.2 What Kind of Squashing Are We Talking About?

We will be looking at a specific form of solid deformation known as *hyperelastic* deformation. Say we take a bunny and squash it with a 100 lb. weight (Fig.2.1). When we remove the weight, it will sproing back to its *exact original shape*. This is the essence of hyperelasticity: when all external forces are removed, a hyperelastic object returns to its original shape. It does not matter if the bunny was stretched to a million times its original length, or crushed down into a pancake. Once the forces are removed, it bounces back to its original bunny shape.[1]

The original shape is thus considered quite special, as it is what the object "remembers" and always "tries" to return to, and is assigned a special name: the **rest shape**. In fact, when we squash a bunny with a 100 lb. weight, it is taking on the *best possible* or *closest possible* shape to the rest shape that it can muster under the current conditions.

But what do we mean by *best* or *closest*? Best compared to what, or closest with respect to what? This is really two questions.

1. How squashed am I? Alternately, how far am I from my original shape? Or: how badly deformed am I compared to my original shape? Most concretely: can I compute a quantitative *deformation score* that tells me exactly how stretched or squashed I am, relative to my original shape?
2. How much should I push back? Once I know how exactly how deformed I am, what should I do with this knowledge? Is being *really* deformed considered *extra*-bad, so I should push *extra*-hard to return to my original shape?

Let's look at these questions in turn.

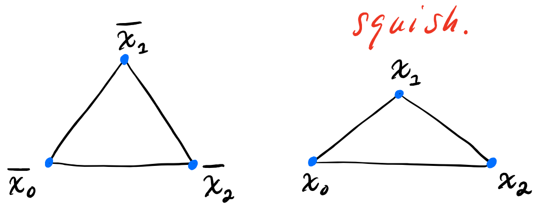

Figure 2.2.: On the left is the original, rest-shape triangle. On the right we (slightly) deform the triangle by moving $\mathbf{x}_1$ downwards.

2.3 How Squashed Am I?

First, let's look at an *obvious* way to measure deformation that will turn out to be *wrong*. From there, we can think about what went wrong and come up with something better.

[1]Clearly, this does not always happen in reality. If you stretch a rubber bunny too much, the fibers inside will slightly tear. When you let go, it will "remember" its rowdy treatment, and instead return to a slightly-stretched-out-looking-bunny shape. This is called *plastic* deformation, and can be layered on top of hyperelastic deformation, so we put it aside for the moment.

2.3.1 Measuring The Wrong Way

Let's look at a triangle (Fig. 2.2). We will denote its vertices in the **rest shape** using an overbar as $\bar{\mathbf{x}}_0$, $\bar{\mathbf{x}}_1$, $\bar{\mathbf{x}}_2$. After the triangle has been squashed away from it original shape, we denote its **deformed shape**[2] as $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$. One obvious way to *compute a score* for how deformed we are is then:

$$\Psi_{\text{wrong}} = \sum_{i=0}^{2} \|\bar{\mathbf{x}}_i - \mathbf{x}_i\|_2. \tag{2.1}$$

Here, we use $\|\cdot\|_2$ to denote the 2-norm. Don't fret if you haven't seen this notation before; we have a description of it in Appendix C.1.1. It is a generalized shorthand for denoting "distance". We will also interchangeably refer to scoring functions as *energy functions*. For our purposes, scoring and energy functions have the exact same meaning.

The scoring function Ψ_{wrong}, certainly looks reasonable. As you squash $\mathbf{x}_1$ downwards more, or stretch it upwards more, the score (energy) definitely increases. We definitely want the score function to equal zero when no deformation is occurring, and some big number when lots of deformation is occurring. If that's all that is required, Ψ_{wrong} certainly seems to get the job done.

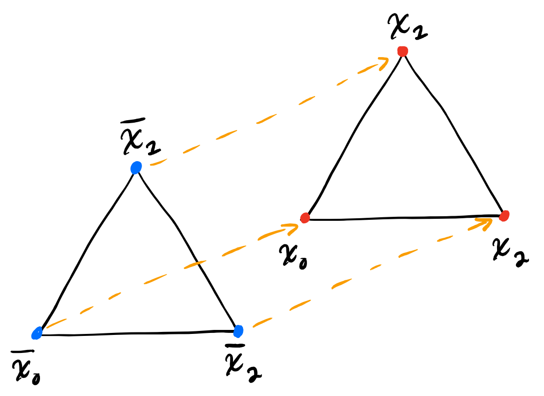

Figure 2.3.: We translate the original triangle (blue vertices) to the upper left (red vertices). Does this triangle count as "deformed"?

Or does it? The problem is that a bunch of things that we *would not* intuitively consider deformation also gets lumped in with the score. In Fig. 2.3, we have translated the triangle a small distance to the upper right. If we use Ψ_{wrong} on this new triangle, then it will show up as a non-zero score.

However, by any reasonable definition, this triangle is not *deformed*. Instead, it has been *moved*, or *translated*. Fig. 2.4 shows another problem case. If we *rotate* the triangle, then again, Ψ_{wrong} will return a non-zero score, mistakenly regarding this triangle as

[2]Other texts may use the terms *material* and *world* coordinates, so if you ever come across these, just remember that **rest = material** and **deformed = world**.

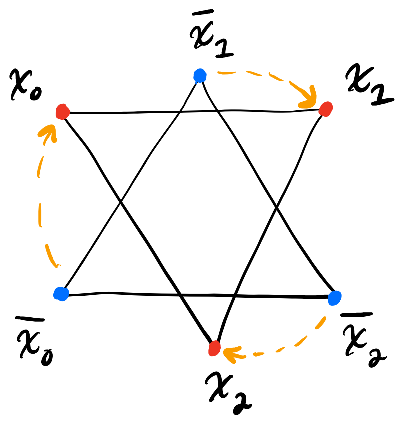

Figure 2.4.: We rotate the original triangle (blue vertices) clockwise (red vertices). Does this triangle count as "deformed"?

"deformed". However, by any reasonable definition, the triangle has been *spun* or *rotated*. It has not been deformed.

What we want is a scoring function Ψ that is *translation and rotation invariant*. If a triangle has been merely translated or rotated, its deformation score should show up as zero. Let's see if we can compute a score that will accomplish this.

2.3.2 Removing the Translation (Easy)

One way to encode the transformations that a triangle can undergo to use an affine map:

$$\begin{aligned} \phi(\bar{\mathbf{x}}) =& \mathbf{F}\bar{\mathbf{x}} + \mathbf{t} \\ =& \mathbf{x}. \end{aligned} \tag{2.2}$$

Here, $\mathbf{t} \in \Re^2$ and $\mathbf{F} \in \Re^{2\times 2}$ in 2D, and $\mathbf{t} \in \Re^3$ and $\mathbf{F} \in \Re^{3\times 3}$ in 3D (see the symbol table in Appendix A). There are only a few things that an affine transformation can encode:

- Rotation
- Scaling
- Reflection
- Translation.[3]

We saw that *rotation* and *translation* kept showing up and polluting our deformation score, even though we don't intuitively consider those as "deformations". However, *scaling* matches our notion of deformation quite nicely. Can we just isolate and measure that?

[3]You may ask: what about *shearing*? Wasn't that in linear algebra class too? That can be written in terms of rotations and scalings, so we can omit it here.

As a first step, let's see if we can carve away the translation, which is relatively easy. The translation lives entirely in $\mathbf{t}$, so if we perform some manipulation that cuts away that component, we are done. The canonical way of doing this is to take the derivative with respect to the rest shape:

$$\begin{aligned} \frac{\partial \phi(\bar{\mathbf{x}})}{\partial \bar{\mathbf{x}}} &= \frac{\partial}{\partial \bar{\mathbf{x}}} \left(\mathbf{F}\bar{\mathbf{x}} + \mathbf{t} \right) \\ &= \mathbf{F}. \end{aligned} \tag{2.3}$$

Since we are taking the derivative (gradient) of the affine map (deformation), the matrix $\mathbf{F}$ that pops out is called the *deformation gradient*. This matrix now contains rotation, scaling, and reflection, but *not translation*. If we base some sort of deformation score on $\mathbf{F}$, there is no way that translation can now pollute the result.

2.3.3 Deformation Gradient: Really Important

It is hard to overstate how important the deformation gradient is. Almost everything we do for the rest of this chapter will deal with $\mathbf{F}$. The variable has an unwieldy name, and everybody (Bonet and Wood (2008); Bower (2009); Marsden and Hughes (1994); Belytschko et al. (2013)) uses $\mathbf{F}$ to denote it, even though neither "deformation" nor "gradient" starts with an 'f'. Ideally the variable would have a snappy name and an intuitive symbol, but that is not the world we live in.[4] We are stuck with $\mathbf{F}$, so just remember that $\mathbf{F}$ is important.

You may be asking yourself: if $\mathbf{F}$ is so important, are you going to tell me how to compute it? Absolutely. It is a little involved, and takes a bit of space, so it has its own section in Appendix D. Taking for granted that we know *how* to compute this quantity, let's examine *what it means*.

2.3.4 Removing the Rotation (Not So Easy)

We now have $\mathbf{F}$, where translation has been mercifully subtracted off, but scaling, rotation and reflection remain entangled. How can we remove the rotation? This will turn out to be more difficult than removing the translation, and there will actually be more than one answer.

Let's try to build a simple score function out of $\mathbf{F}$. The simplest way to boil a matrix down to a scalar score is to take its *squared Frobenius norm*, denoted $\|\cdot\|_F$. As with the 2-norm in §2.3.1, don't worry if you're not intimately familiar with this norm. We document it in detail in Appendix C.1.2, but since this is the first time it is appearing, we will write it

[4] In geometry processing, it is sometimes denoted with J, presumably because it is the deformation *Jacobian*. Better, but still not great.

out explicitly,

$$\begin{aligned}\Psi_{\text{Dirichlet}} &= \|\mathbf{F}\|_F^2 \\ &= \sum_{i=0}^{2}\sum_{j=0}^{2} f_{ij}^2,\end{aligned} \tag{2.4}$$

where we have numbered the entries of $\mathbf{F}$ thusly:

$$\mathbf{F} = \begin{bmatrix} f_{00} & f_{01} & f_{02} \\ f_{10} & f_{11} & f_{12} \\ f_{20} & f_{21} & f_{22} \end{bmatrix}. \tag{2.5}$$

This score function is also known as the *Dirichlet energy*, and it *does not get the job done*. Remember, we want the deformation score to return zero when there has been zero deformation. However, the Dirichlet energy will return zero when all the entries of $\mathbf{F}$ are zero, i.e. $\mathbf{F} = \mathbf{0}$.

If we plug $\mathbf{F} = \mathbf{0}$ into our original deformation function $\phi(\bar{\mathbf{x}}) = \mathbf{F}\bar{\mathbf{x}} + \mathbf{t}$, it corresponds to the case where all the points $\bar{\mathbf{x}}$ are crushed into a zero-volume black hole centered at $\mathbf{t}$. As a scoring function, the Dirichlet energy might be a great black hole detector, but it's a crummy deformation measure.

2.3.4.1 *A First (Wrong) Attempt*

The Dirichlet energy returns a zero when $\mathbf{F} = \mathbf{0}$. What does an energy (score) that returns zero under zero deformation even look like? Well, if $\mathbf{F} = \mathbf{I}$, then certainly no deformation is occurring. In that case, $\phi(\bar{\mathbf{x}}) = \mathbf{F}\bar{\mathbf{x}} + \mathbf{t} = \mathbf{I}\bar{\mathbf{x}} + \mathbf{t} = \bar{\mathbf{x}} + \mathbf{t}$, which is purely a translation.

Can we design a function that returns zero when $\mathbf{F} = \mathbf{I}$? Well, if $\mathbf{F} = \mathbf{I}$, then $\|\mathbf{F}\|_F^2 = 3$. So, let's try the scoring function:

$$\Psi_{\text{Neo-Hookean}} = \|\mathbf{F}\|_F^2 - 3. \tag{2.6}$$

This is not a good idea. While it is true that when $\mathbf{F} = \mathbf{I}$, the score will equal zero, we also want zero to be the *smallest score possible*. Otherwise, the score becomes difficult to interpret. A zero score means zero deformation, a score greater than zero means some deformation, but what does a score less than zero mean? Less than zero deformation is happening?

In the case of $\Psi_{\text{Neo-Hookean}}$, if $\mathbf{F} = \mathbf{0}$, then $\Psi_{\text{Neo-Hookean}} = -3$. So, it's a pretty bad deformation measure. Things get worse if

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sqrt{\frac{3}{2}} & 0 \\ 0 & 0 & \sqrt{\frac{3}{2}} \end{bmatrix}. \tag{2.7}$$

This refers to a Wile-E-Coyote-style *pancake state*, which is definitely *not* a zero deformation state. And yet, the $\Psi_{\text{Neo-Hookean}}$ score returns a zero. Apparently there are many sneaky configurations that can fool this scoring function into thinking that no deformation has occurred. We could try to apply a squaring Band-Aid, e.g. $\Psi_{\text{Neo-Hookean,Band-Aid}} = \left(\|\mathbf{F}\|_F^2 - 3\right)^2$, but this would only address the negative scores problem. The scoring function would still return zero for the same sneaky deformed states.[5]

However, for some reason it has the fancy name "Neo-Hookean", so it must be important. It has been studied extensively in mechanical engineering and material science for the last 80 years (Mooney (1940); Rivlin (1948)), because we will see later that if you add a few important components, it becomes a *very interesting* deformation measure. But, it's not very good in its current form, so let's put it aside for now.

2.3.4.2 *A Second (Still Wrong) Attempt*

Next, let's try something capricious and stupid-looking (C&SL), which I promise will eventually lead somewhere. Let's just *subtract* **F** *and* **I** and take its norm. Then, we are at least guaranteed that when $\mathbf{F} = \mathbf{I}$, our scoring function will equal zero:

$$\Psi_{\text{C\&SL}} = \|\mathbf{F} - \mathbf{I}\|_F^2. \tag{2.8}$$

Additionally, since the **I** is inside the norm, everything gets squared, so there is no possibility of our function producing meaningless negative scores. We can use identity B.16 from Appendix B to expand this out to:

$$\begin{aligned}\Psi_{\text{C\&SL}} &= \|\mathbf{F} - \mathbf{I}\|_F^2 \\ &= \|\mathbf{F}\|_F^2 + \|\mathbf{I}\|_F^2 - 2\operatorname{tr}\mathbf{F} \\ &= \|\mathbf{F}\|_F^2 + 3 - 2\operatorname{tr}\mathbf{F}.\end{aligned}$$

I have assumed that we're looking at the 3D case, which is why $\|\mathbf{I}\|_F^2 = 3$. Looking at the expansion, the score still isn't great. The first term is the black-hole favoring Dirichlet energy, which doesn't do us any favors. Then, we have the constant of 3, which looks suspiciously similar to our not-great Neo-Hookean energy, followed by a new $\operatorname{tr}\mathbf{F}$ term. If you haven't seen a matrix trace in a while, there is a refresher in Appendix C.2. Since this is its first appearance here, we will write it out explicitly:

$$\operatorname{tr}\mathbf{F} = \sum_{i=0}^{2} f_{ii} = f_{00} + f_{11} + f_{22}. \tag{2.9}$$

This scoring function *will* equal zero when $\mathbf{F} = \mathbf{I}$, but when $\mathbf{F} \neq \mathbf{I}$, weird things happen. For example, if **F** happens to be some rotation matrix,

$$\mathbf{F}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \tag{2.10}$$

[5]Fun fact: This squaring is equivalent to the (yet-to-be-seen) volume preservation term from the St. Venant Kirchhoff model. It is still not terribly useful all on its own though.

then $\Psi_{\text{C\&SL}}$ will return a non-zero score whenever θ is not zero or some multiple of 2π. But, these would still clearly correspond to deformation-free states! It's better than a black-hole detector, but not by much.

Overall, it's weird that $\operatorname{tr}\mathbf{F}$ is shows up at all in this formula. Since $\mathbf{F}$ can be an arbitrary, non-symmetric matrix, it is not clear that its trace is terribly meaningful outside of cancelling off $\|\mathbf{F}\|_F^2 + 3$ in the overly-specific case of $\mathbf{F} = \mathbf{I}$.

2.3.4.3 *A Third Attempt: St. Venant Kirchhoff*

How can we make a version of $\Psi_{\text{C\&SL}}$ that instead equals zero whenever $\mathbf{F}$ is a pure rotation? One way to do this is to exploit a nice linear algebra identity. If $\mathbf{F}$ is a pure rotation, then we know that it must be the case that $\mathbf{F}^T\mathbf{F} = \mathbf{I}$. This follows from the fact that $\mathbf{F}^{-1} = \mathbf{F}^T$ when $\mathbf{F}$ is orthonormal.

Well, why don't we compute $\mathbf{F}^T\mathbf{F}$, subtract off the rotation part, i.e. the $\mathbf{I}$ part, and see what's left over? These leftovers probably characterize the deformation in some reasonable way. We can write this down as:

$$\Psi_{\text{StVK, stretch}} = \frac{1}{2}\|\mathbf{F}^T\mathbf{F} - \mathbf{I}\|_F^2. \tag{2.11}$$

This is a pretty good idea! The idea is good enough that people have given it a special name: The St. Venant-Kirchhoff stretching energy, or "StVK, stretch", for short.[6] The individual components of $\mathbf{F}^T\mathbf{F} - \mathbf{I}$ are popular enough that they have special names and symbols:

$\mathbf{C} = \mathbf{F}^T\mathbf{F}$	The right Cauchy-Green tensor
$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right)$	Green's strain

Using these, the energy can be written in a highly compact form:

$$\Psi_{\text{StVK, stretch}} = \|\mathbf{E}\|_F^2. \tag{2.12}$$

Again, this is a decent way to measure deformation. We're using $\mathbf{F}$, so the translation has been subtracted off, and we using an $\mathbf{F}^T\mathbf{F}$ identity to factor off rotation. If we were going to complain about one thing though, it would be that the scoring function is *overly non-linear*. Performing the same expansion as we did for $\Psi_{\text{C\&SL}}$, we can get:

$$\begin{aligned}\Psi_{\text{StVK, stretch}} &= \|\mathbf{F}^T\mathbf{F} - \mathbf{I}\|_F^2 \\ &= \|\mathbf{F}^T\mathbf{F}\|_F^2 + \operatorname{tr}\mathbf{I} - 2\operatorname{tr}(\mathbf{F}^T\mathbf{F}).\end{aligned} \tag{2.13}$$

The leading term, $\|\mathbf{F}^T\mathbf{F}\|_F^2$, is 4th-order, or *quartic* in $\mathbf{F}$. The $\mathbf{F}$ gets squared by $\mathbf{F}^T\mathbf{F}$, and then squared again by the norm, $\|\cdot\|_F^2$, which makes it 4th order overall. We will see later that this 4th-order-ness adds additional computational complexity, but in a way that is not actually desirable.

[6] Actually, *I* call it the StVK stretching energy, because it's the first term (the stretching term) in the StVK energy. So if you look up "St. Venant-Kirchhoff", you'll see this term, plus another one. We'll take a look at that other one later.

2.3.4.4 *A Fourth Attempt: As-Rigid-As-Possible*

We took advantage of the $\mathbf{F}^T\mathbf{F} = \mathbf{I}$ identity, but are there other identities that we could have used? If we're specifically concerned with teasing apart the rotation and non-rotation component of a matrix, the *polar decomposition* is helpful:

$$\mathbf{F} = \mathbf{R}\mathbf{S}. \tag{2.14}$$

Here, the matrix $\mathbf{R}$ is the rotational part of $\mathbf{F}$, and $\mathbf{S}$ is the non-rotational (scaling) component of $\mathbf{F}$. This decomposition seems custom-built to solve exactly the problem we are interested in.

However, you need to be a little careful about using the polar decomposition, because its classic definition is slightly different from the version we want. Usually $\mathbf{R}$ is only defined as a *unitary* matrix, not a rotation matrix, so it only needs to satisfy $\mathbf{R}^T\mathbf{R} = \mathbf{I}$. This means that there could be a *reflection* lurking somewhere in $\mathbf{R}$, whereas we want $\mathbf{R}$ to be a pure rotation. In our case, if a reflection has to lurk somewhere, we would prefer that it do so in $\mathbf{S}$. We will call this specific flavor the *rotation variant* of the polar decomposition. Details on how to compute this are in Appendix F.

In short: if you end up calling the polar decomposition code in some library, make sure it is computing the *rotation variant*, not the traditional version. If it is not, you will need to write a small wrapper for the library call in the style of Figs. F.1 and F.2.

With that out of the way, let's assume you can compute the rotation-variant polar decomposition, and now have $\mathbf{R}$ and $\mathbf{S}$ in hand. What can you do with them? In StVK, we took $\mathbf{F}^T\mathbf{F}$ and subtracted off its rotational component, $\mathbf{I}$. Well, now we have a different representation for the rotation, $\mathbf{R}$, but can we do the same thing? Subtract the rotation component off of $\mathbf{F}$, and assume that the leftovers must characterize the deformation somehow?

The most basic, gee-I-wonder-if-this-will-work way of writing this down is:

$$\Psi_{\text{ARAP}} = \|\mathbf{F} - \mathbf{R}\|_F^2. \tag{2.15}$$

There are a few worrying-looking things about this formulation. Usually when you have two matrices, multiplying them together is a first thing to try, because you can see what the multiplication is *really* doing by peeking at the SVD or eigendecomposition. But, instead of multiplying, we're subtracting. Can we really just subtract two arbitrary-looking matrices from each other and get something that isn't total garbage? They are related via the polar decomposition, but does that really mean we can just start subtracting these entries from each other?

Surprisingly, the answer is yes. In fact, this energy function is so indispensable that it has been given a special name in geometry processing: the As-Rigid-As-Possible (ARAP) energy (Sorkine and Alexa (2007)). We'll do a little more analysis of this energy a little later to see how it is actually more clever than it first appears.

This energy does not have the "overly non-linear" problem of StVK. If we apply identity B.16, we get"

$$\begin{aligned}\Psi_{\text{ARAP}} &= \|\mathbf{F} - \mathbf{R}\|_F^2 \\ &= \|\mathbf{F}\|_F^2 + \|\mathbf{R}\|_F^2 - 2\,\mathrm{tr}(\mathbf{F}^T\mathbf{R}) \\ &= \|\mathbf{F}\|_F^2 + 3 - 2\,\mathrm{tr}(\mathbf{S}).\end{aligned} \tag{2.16}$$

The energy is at most *quadratic* in $\mathbf{F}$. We will see later that the introduction of $\mathbf{R}$ and $\mathbf{S}$ creates some computational challenges, but for the time being, it is undeniable that Ψ_{ARAP} is appealingly less non-linear than $\Psi_{\text{StVK,stretch}}$.

In physical simulation, the ARAP energy is actually the first term in the popular *co-rotational* energy. This energy has been known in mechanical engineering since at least the 1980s (Rankin and Brogan (1986)), and the ARAP energy corresponds to the specific mechanical case where Poisson's ratio is set to zero. This energy was rediscovered by several graphics researchers in the early 2000s (Müller et al. (2002); Etzmuss et al. (2003); Irving et al. (2004)), which created some confusion because they each gave it a different name, such as "stiffness warping" or "rotated linear". However, these models are all equivalent, and nowadays everybody seems to have settled on "co-rotational" as the name for this energy.

2.3.4.5 *Summary of Energy (Scoring) Functions*

We have now seen a bunch of different energy functions. Let's summarize their pluses and minuses:

Energy	**Pros**	**Cons**
Dirichlet	Not many. A good black hole detector?	Doesn't measure deformation effectively.
Neo-Hookean	Returns zero when $\mathbf{F} = \mathbf{I}$	Returns zero for other configurations too, and can also return negative scores.
C&SL	Returns zero when $\mathbf{F} = \mathbf{I}$, and does not return negative scores.	Does not return zero under pure rotations.
StVK	Returns zero when $\mathbf{F}$ is a pure rotation, and reasonable scores otherwise.	Overly non-linear (quartic).
ARAP	Returns zero when $\mathbf{F}$ is a rotation, reasonable scores otherwise, and is quadratic	That $\mathbf{R}$ term is going to cause trouble.

As I said at the top, removing the rotation from the score is not easy, and there is more than one strategy. Now that we have a bunch of options, what should we do with them?

2.4 Force Computation: How Much Should I Push Back?

Now that we can measure how deformed we are, we can return to the second question from §2.2: *How much should I push back?*

The deformation measure directly generates forces. Surprise! When we were muddling over different ways to measure deformation, we were actually trying out different elastic energy potentials. The two are completely equivalent for our purposes. The deformation measure directly dictates the force response. If you want to design a new material model, come up with a different way to measure deformation.

To get the forces $\mathbf{f}$ on the nodes of a triangle in 2D, we stack the vertices of our triangle ($\mathbf{x}_0$, $\mathbf{x}_1$ and $\mathbf{x}_2$) into a big vector,

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix} \in \Re^6, \tag{2.17}$$

and then take the gradient:

$$\mathbf{f} = -a \frac{\partial \Psi}{\partial \mathbf{x}}. \tag{2.18}$$

Here, a is the area of the original triangle at rest. A really tiny deformed triangle the size of an ant is not going to exert the same force as a deformed triangle the size of the Empire State Building, so we should scale it.

That's all well and good, and Eqn. 2.18 looks straightforward, except that all the energies we looked at in §2.3.4.5 were written in terms of $\mathbf{F}$, not $\mathbf{x}$. In fact, we made a big deal out of the fact that $\mathbf{F}$ doesn't have all these junky measurement-polluting properties of $\mathbf{x}$, like translation, and that it was clearly the better primary variable. Having gone down that path, how are we supposed to take the gradient with respect to $\mathbf{x}$ now?

2.4.1 Computing $\frac{\partial \Psi}{\partial \mathbf{x}}$

We pushed the computation of $\mathbf{F}$ to Appendix D, but we're going to pull the final result of that discussion back up here. For a triangle, we pack its original $\bar{\mathbf{x}}_i$ and deformed $\mathbf{x}_i$ vertices into two matrices, $\mathbf{D}_m$ and $\mathbf{D}_s$, thusly:

$$\mathbf{D}_m = \left[\begin{array}{c|c} \bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_0 & \bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_0 \end{array} \right] \qquad \mathbf{D}_s = \left[\begin{array}{c|c} \mathbf{x}_1 - \mathbf{x}_0 & \mathbf{x}_2 - \mathbf{x}_0 \end{array} \right]. \tag{2.19}$$

The deformation gradient is then a composition of these two:

$$\mathbf{F} = \mathbf{D}_s \mathbf{D}_m^{-1}. \tag{2.20}$$

First the good news: we need the gradient with respect to $\mathbf{x}$, not $\bar{\mathbf{x}}$. If we instead needed $\frac{\partial \Psi}{\partial \bar{\mathbf{x}}}$ things would get ugly real fast, because $\bar{\mathbf{x}}_i$ appears in $\mathbf{D}_m$, which is inside an inverse, $\mathbf{D}_m^{-1}$, so we'd need to figure out the derivative of a matrix inverse.

Now the bad news: it's still pretty ugly. Let's take a really simple energy, $\Psi_{\text{Dirichlet}}$ as an example:

$$\Psi_{\text{Dirichlet}} = ||\mathbf{F}||_F^2. \tag{2.21}$$

Here's what we're going to have to do:

- Plug $\mathbf{F} = \mathbf{D}_s \mathbf{D}_m^{-1}$ in to $\Psi_{\text{Dirichlet}}$.
- Multiply everything through to get a massive one-line equation for $||\mathbf{F}||_F^2$.
- Take the derivative of this massive equation six times, once for each entry in $\mathbf{x}$, and stack the results into a force vector.
- Hope you didn't make a mistake in your derivation somewhere. Port the expressions into C++, and hope you don't make a mistake transferring these equations from paper to code.

2.4.2 Don't Do It This Way

Let's gaze at this ugliness for just a minute longer. Just a minute though. After that I have to look away, and tell you there's a better way. To start, we can tag each entry in $\mathbf{D}_m^{-1}$ thusly,

$$\mathbf{D}_m^{-1} = \begin{bmatrix} m_0 & m_2 \\ m_1 & m_3 \end{bmatrix} \tag{2.22}$$

then we'll try to multiply out the matrix $\mathbf{F}^T\mathbf{F}$ in anticipation of cramming everything together for $||\mathbf{F}||_F^2$.

$$\begin{aligned} \mathbf{F}^T\mathbf{F} &= \begin{bmatrix} m_0 & m_1 \\ m_2 & m_3 \end{bmatrix} \left[\begin{array}{c|c} & \\ \mathbf{x}_1 - \mathbf{x}_0 & \mathbf{x}_2 - \mathbf{x}_0 \\ & \end{array}\right]^T \left[\begin{array}{c|c} & \\ \mathbf{x}_1 - \mathbf{x}_0 & \mathbf{x}_2 - \mathbf{x}_0 \\ & \end{array}\right] \begin{bmatrix} m_0 & m_2 \\ m_1 & m_3 \end{bmatrix} \\ &= \begin{bmatrix} m_0 & m_1 \\ m_2 & m_3 \end{bmatrix} \begin{bmatrix} (\mathbf{x}_1 - \mathbf{x}_0)^T(\mathbf{x}_1 - \mathbf{x}_0) & (\mathbf{x}_2 - \mathbf{x}_0)^T(\mathbf{x}_1 - \mathbf{x}_0) \\ (\mathbf{x}_2 - \mathbf{x}_0)^T(\mathbf{x}_1 - \mathbf{x}_0) & (\mathbf{x}_2 - \mathbf{x}_0)^T(\mathbf{x}_2 - \mathbf{x}_0) \end{bmatrix} \begin{bmatrix} m_0 & m_2 \\ m_1 & m_3 \end{bmatrix} \end{aligned} \tag{2.23}$$

Just to clean things up, let's introduce more temporary variables:

$$\begin{bmatrix} (\mathbf{x}_1 - \mathbf{x}_0)^T(\mathbf{x}_1 - \mathbf{x}_0) & (\mathbf{x}_2 - \mathbf{x}_0)^T(\mathbf{x}_1 - \mathbf{x}_0) \\ (\mathbf{x}_2 - \mathbf{x}_0)^T(\mathbf{x}_1 - \mathbf{x}_0) & (\mathbf{x}_2 - \mathbf{x}_0)^T(\mathbf{x}_2 - \mathbf{x}_0) \end{bmatrix} = \begin{bmatrix} d_0 & d_1 \\ d_1 & d_2 \end{bmatrix}. \qquad \text{(Note it's symmetric)}$$

Now let's forge ahead further,

$$\mathbf{F}^T\mathbf{F} = \begin{bmatrix} m_0 & m_1 \\ m_2 & m_3 \end{bmatrix} \begin{bmatrix} d_0 & d_1 \\ d_1 & d_2 \end{bmatrix} \begin{bmatrix} m_0 & m_2 \\ m_1 & m_3 \end{bmatrix}$$
$$= \begin{bmatrix} m_0^2 d_0 + 2m_0 m_1 d_1 + m_1^2 d_2 & m_0 m_2 d_0 + (m_0 m_3 + m_1 m_2) d_1 + m_1 m_3 d_2 \\ m_0 m_2 d_0 + (m_0 m_3 + m_1 m_2) d_1 + m_1 m_3 d_2 & m_2^2 d_0 + 2m_2 m_3 d_1 + m_3^2 d_2 \end{bmatrix},$$

and then finally smash everything together with the Frobenius norm:

$$\|\mathbf{F}\|_F^2 = \left(m_0^2 d_0 + 2m_0 m_1 d_1 + m_1^2 d_2\right)^2 + \left(m_0 m_2 d_0 + (m_0 m_3 + m_1 m_2) d_1 + m_1 m_3 d_2\right)^2$$
$$\left(m_0 m_2 d_0 + (m_0 m_3 + m_1 m_2) d_1 + m_1 m_3 d_2\right)^2 + \left(m_2^2 d_0 + 2m_2 m_3 d_1 + m_3^2 d_2\right)^2.$$

Okay, I have to stop now. Cripes, it hurts to even look at this. The expression is big enough that it barely fits on two lines. I probably made a mistake in there somewhere; good luck finding it. Next we'll have to substitute back in $d_0 = (\mathbf{x}_1 - \mathbf{x}_0)^T(\mathbf{x}_1 - \mathbf{x}_0)$ and from there, $\mathbf{x}_0 = [x_0 \quad y_0]^T$, as well as $d_1, d_2, \mathbf{x}_1$ and $\mathbf{x}_2$, and then take multiple derivatives of the resulting hideousness. Also, things get worse in 3D, and the Dirichlet energy isn't even that complicated an energy! Things sink to even deeper depths of hideousness for more complex material models.

You could use a computer algebra system like Mathematica or Matlab's Symbolic Toolkit to handle all this. That would just offload the ugliness generation to the computer, and the final result would still be ugly. Surely there is a better way.

2.4.3 What Now?

There *is* a better way, which we will describe in the next chapter. When we do, you will see that the *forces* are now cleaner to compute, but the *force gradients* (a.k.a the *energy Hessians*) are still ugly.

You may be tempted to take the coward's way out here. All of this ugliness started because I insisted that we use $\mathbf{F}$ to compute our deformation measure instead of $\mathbf{x}$. If we had just stuck to $\mathbf{x}$, we wouldn't be in this mess, because computing $\frac{\partial \Psi}{\partial \mathbf{x}}$ when Ψ is written purely in terms of $\mathbf{x}$ is *much easier.* Just look at this gorgeous-looking spring-mass energy:

$$\Psi = \frac{\mu}{2}\|\mathbf{x} - \mathbf{x}_0\|_2^2 \tag{2.24}$$
$$\frac{\partial \Psi}{\partial \mathbf{x}} = \mu(\mathbf{x} - \mathbf{x}_0). \tag{2.25}$$

SO EASY. Don't be seduced though. If allow ourselves be sucked down into this Charybdis, we'll be limiting ourselves to the world of spring-mass models, and the richer universes of squishing and squashing volumetric responses will be inaccessible. Let's not do that; let's see how far we can take this $\mathbf{F}$ thing.