Functional Geometry and the "Traité de Lutherie"

HARRY MAIRSON

harry.mairson@verizon.net

Harry Mairson, a computer science professor at Brandeis University, devised a computer language to draw violins, following the methods of construction used by François Denis in his book "Traité de Lutherie." In this very engaging talk, Harry presents what he did and the reasoning behind it. Interested persons are encouraged to contact him, and to get and experiment with his code.

Harry Mairson: First, let me say that if you're interested in this and interested in trying it out, I'd love to talk with you. Please feel free to send me an email.

I'm an amateur luthier, not a professional. I'm a computer science professor. At the National Science Foundation they have a rubric that they call computational thinking. This is a way of saying that there are many different systems, and constructions, and ideas on the function of cellular and biological systems, and on the function of economic systems, and so on, that at base level can be explained by an algorithm—a method that can be written down. I think that my dual interest in lutherie and computer science is really expressed by that concept. The way a luthier would talk about computational thinking would be to say, there's a method for everything. There's a technique for everything.

For instance, take reaming the pegholes in a cello. I have a gizmo, a method, for doing this. My jig holds the cello and has a mirror, so I can stand behind the cello and look along two axes, down the cello and in the mirror, without moving my head, so I can easily ream straight pegholes. There's a method for everything.

David Hockney, a famous artist, wrote a book about ten years ago called "Secret Knowledge". He noticed that, all of a sudden, in the 15th century, artists started painting better. The art historians said, "people just got better at painting". But David Hockney, who's an artist himself and knows something about how hard it is to do things, said, "No. They did something. They had a method." He advocated a particular line of argument, that they were using a *camera lucida* and various optical techniques to do more accurate painting. This was taken to a greater extreme in a recent movie by Penn and Teller, called "Tim's Vermeer", about a software engineer in graphics who built a replica of Vermeer's room so he could paint "the Music Lesson" using a *camera lucida*. So there's a method for everything, and that's part of what I want to talk about today.

I stopped reading computer science books, and started reading books about violins and violinmaking. I stopped giving nerdy computer science talks, and I started giving talks like this, to people like you. Which is all just a disclaimer; there are some things that I know a lot about, and I feel I can speak about with some authority. There are some things I learned more recently, while climbing the lutherie learning curve, some things I sort of know about, and some things I'm just improvising. But I'm a college professor, so the one rule is, keep talking. So, bear with me.

What I'm going to talk about today, after some introductory background, is the software artifact that I built, which is meant to be a programming language embodiment of the ideas in François Denis's book, "Traité de Lutherie". I think this is the coolest book I have ever read. I'd like to talk about what you could use the software for, in a practical way, and finally, the more speculative part of the talk. I'd like to try to describe how I think you could use software

like this to facilitate a better understanding of how stringed instruments evolved. It's what I like to call, when I write grant proposals with a straight face, "computational art history".

Enough of that. How are string instrument outlines designed? It's a bit of art history, to be able to look at them and go "this is by such-and-such a maker". People who can identify paintings have a similar kind of sense. Modern-day makers copy, with modifications, instruments of famous makers, who couldn't copy instruments of famous makers and so on and so on, ad infinitum. Somebody had to have the original idea. Somebody had to have a method for drawing them. Otherwise you have what I describe to my discrete math students as a non-well-founded induction. You just keep referring to a previous case and you have a loop that doesn't terminate. So where do the ideas come from, the ideas that let you draw the outline of a string instrument?

Heron-Allen's 1885 book, "Violin-Making: as it was and is" contains one idea. It's a way I don't find particularly convincing. A lot of the ideas in it come from an earlier manuscript, "Règles" by Bagatella (1782), which also has its limitations. The one I want to talk about today, obviously, is the method in the book that François Denis wrote in 2006.

In a word, the idea of the book is that it's just straightedge and compass. No graph paper, no rulers, no Vernier calipers, no protractors; there's nothing Cartesian about it at all. All of our modern ideas about measurement simply don't matter. It's done a different way. It's done the Euclidean way. I liken it to the difference between 10th grade and 11th grade. In 10th grade you learn geometry. In 11th grade they say "a circle's not a circle. A circle's an equation." So we're going back to where a circle really is a circle.

I wanted to learn how to do this, and then I started thinking, really? With a pencil and paper? With the pencil creep? With a compass that didn't go into exactly the right place in the paper? With my misreading of the book, where it said divide this distance by three, not two, stupid? And starting over. And I thought, I need to have a way where I can make thousands of mistakes per second and recover from them instantly. To make mistakes at that speed, you need a computer.

When I first read "Traité de Lutherie", it made me realize a number of things. One of them is, technical writing is very hard. Reading the book reminded me of when I had to assemble a gas grill for my parents; the assembly directions that were translated, word for word, from some non-Indo-European language. I also realized that there is a lot of repetition and parameterization in the text; places where he says, for instance, "use the same construction as in the Amati on page 123...". You look up the method and you have to plug in the right parameters to make the thing go. I also noticed that there's a hierarchy of design. There are various steps in the design of the outline. And I had to ask, which of the steps affect the subsequent steps? Which ones are geometrically and computationally independent? When we start asking questions like that, and having concerns like that, it's natural—at least for me—to turn to a programming language, or at least to a programming solution.

One of the things I'd like to communicate to you today is that a programming language is not simply a way of telling a computer what to do. It's a way of expressing ideas. There are languages that are appropriate for love letters, there are vernaculars that are appropriate for firing people, and there is language that is appropriate for describing outlines. So, that's what I'm trying to do: come up with a programming language that would talk about these outlines. It's what I call a computational thinking about the past.

What I've done so far is to have a system of graphics primitives on top of this programming language that I know, a sort of base-level programming language called "Scheme," using some ideas from a 1980s paper by Peter Henderson, that I read a long time ago. I wanted it to be language-based, not WYSIWYG ("what you see is what you get"), because I wanted to preserve the luthier's vernacular, virtually verbatim, as described by Denis. I wanted to be able to look at a page in the book and then look at the code, and see that the code says exactly what the book says. It just says it in a different typeface. It's basically a programmable straightedge and compass machine. It's the basic construction Denis uses, where geometric objects are constructed by intersections, with some abstraction mechanisms I'll talk about once I go on. And I have

a beginning catalog of instruments and related analysis.

The canonical example of a WYSIWYG system is Microsoft Word, which deserves a place among the ten plagues in the Book of Exodus. I don't use Microsoft Word. I use this thing called LaTeX. LaTeX is a markup language. It's sort of a model for what I'm talking about today. If I want to typeset a math equation, such as in this paragraph:

> Now consider the rational approximation of the subharmonic section. Let $s+m=1$ where s and m are an approximation of this section: then a better approximation is $s'=m$ and $m'=s+2m$. Solving $\frac{s}{m}=\frac{s'}{m'}$ gives $s=1-\frac{1}{\sqrt{2}}$, the smaller part of the section.

what I do in LaTeX is to write a file that has the markup:

```
Now consider the rational approximation of the
subharmonic section. Let $s+m=1$ where $s$ and
$m$ are an approximation of this section: then a
better approximation is $s'=m$ and $m'=s+2m$.
Solving ${s\over m}={{s'}\over{m'}}$ gives $s=1
{1\over{\sqrt{2}}}$, the smaller part of the
section.
```

The markup text is translated and compiled into the typeset formula. If the typeset of the math equation is wrong, I can ask, "what's in my original file that produced the wrong typeset equation?", and I can fix that. Contrast this with using Microsoft Word. You're typing something in italic and you decide, I don't want the italic. You backspace, deleting the italic text, and you start typing again,. But it's still in italic. Buried in the bowels of the file somewhere is some switch-to-italic thing that you can't get to. Now you go into animal sacrifice mode, where you think, "how much of this paragraph do I have to delete before it finally realizes that I don't want italic?" The advantage of non-WYSIWYG programming language is that you get to see what you did to get what you got. That's what I like about it.

I think the WYSIWYG of string instrument outlines would be some kind of CAD/CAM system where you can drag points along the screen, but you can't really see what made the constructions. I want to be able to look at code and see what kind of image it made. So I'm thinking of a markup language for "Traité de Lutherie". I want to have instructions that describe an outline. I want to take the friendly prose, French translated into English, and make the picture that Denis has in the book. I'd like to take those instructions and make them into executable computer code.

I have a toy problem to try and describe what this language does, and it's just drawing a pentagon. Drawing a triangle or a square with a straightedge and compass is easy, but how do you draw a pentagon? I know a pentagon's not a musical instrument, but we'll get to that momentarily. The code has two sections. The first section, the point placement section, just says where to put some points on the plane, using straightedge and compass ideas. The second part, the parts list, is just a compendium of what to output.

I should just mention, briefly, why it is that you can draw a pentagon with a straightedge and compass. This, I never knew. If you look at the pentagon there are two red triangles, a big red triangle, and then a little one sitting on its side (figure 1). They're similar triangles. They're proportionately similar, so the ratio of their sides is the same. The long side φ is sort of the "diameter" of the pentagon. The ratio of φ (the long side of the large triangle) is to 1 (the short side or base of the large triangle) as 1 (the long side of the small triangle) is to 1/φ (the short side or base of the small triangle). They have the same multiplicity. If you multiply the base of the big triangle, 1, by φ, then you get φ, the long side of the big triangle. If you multiply the short side of the small triangle, 1/φ, by φ, then you get 1. And so, since φ is one way of writing the "diameter" of the pentagon, it has to equal 1 + 1/φ, which is another "diameter",

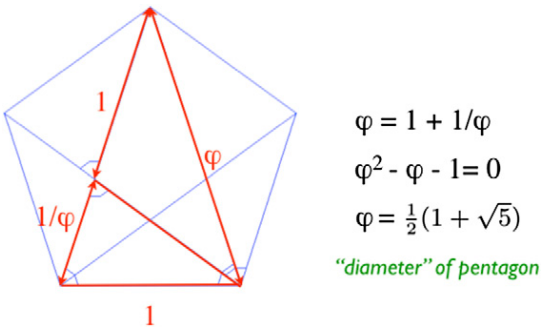

Figure 1.

the other long side of the large triangle. That's the first equation. From it, you get a quadratic equation (the second equation), which you learned how to solve in the 7th grade, using the Hippocratic Oath (laughter) and the solution is the Golden Mean. And so, the reason you can draw a pentagon with straightedge and compass is because, if you have a unit distance, you can draw that distance, the Golden Mean, and the rest follow.

So how do you do it? I will now take you step by step through this little baby piece of code.

First, there's a place on the plane that I just call the origin. From here we define a point, **p**:

```
(p (point (- (/ d 2)) (- (/ d 2))))
```

*Call **p** the point that goes down **d**/2 units, and left **d**/2 from the origin.*

We now define a second point, **q**:

```
(q (xshift p d))
```

*Let **q** be the point that is **p** shifted over by distance **d**.*

That's the one distance you see here. This is similar to the one parameter Denis uses, that generates everything else. So now we have **q** which is **p** shifted over by **d** units along the x-axis.

Now we want to define **m**, the midpoint of the line **p q**:

```
(m (midpoint p q))
```

*Let **m** be the midpoint of **p** and **q**.*

Now things get complicated. We define a point **r**:

```
(r (top (intersect
        (vertical m)
        (circle m d))))
```

***Vertical m** is the vertical line through point **m**. **Circle m d** is the circle whose center is **m** and radius is **d**.*

```
"to draw a pentagon with side d,"

(define (pent d)
(let* (
  (p (point (- (/ d 2)) (- (/ d 2))))
  (q (xshift p d))
  (m (midpoint p q))
  (r (top (intersect
       (vertical m)
       (circle m d))))
  (s (top (intersect
       (line q r)
       (circle r (distance p m)))))
  (t (top (intersect
       (vertical m)
       (circle q (distance q s)))))
  (u (top (intersect
       (circle p d)
       (circle t d))))
  (v (top (intersect
       (circle q d)
       (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
      (list (line t u)
            (line u p)
            (line p m)))
       )))
```

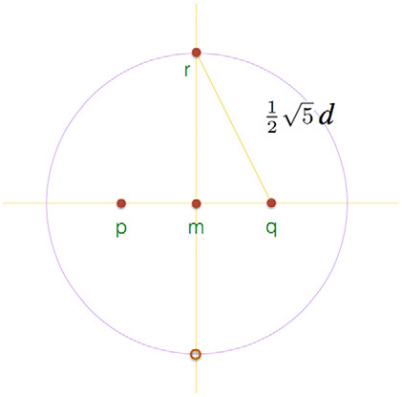

Figure 2.

This says to take the intersection of those two objects, the circle and the vertical line. There are two points, at the top and the bottom of the circle. *Take the top intersection and call it* ***r*** (figure 2).

Again, if you use the Pythagorean Theorem, and you know that the distance from **p** to **q** is **d**, then you know that the hypotenuse is ½ * √5 * **d**. That's almost the Golden Mean.

Now we want to define another point, **s**. This one will be on the circle centered on point **r**, but with a radius of distance **p m** - that is, half of **d**. It is located on line **q r**.

```
(s (top (intersect
        (line q r)
        (circle r (distance p m)))))
```

Make another little circle whose radius is the length ***p m*** *(i.e.,half of* ***d****), center it on* ***r****, extend the diagonal line though* ***q*** *and* ***r****, take the top intersection and call it* ***s****.* (figure 3).

The distance **q s** defines the radius of a new circle; the top intersection of that circle and the vertical line through **m** defines the point **t**, which is the top of the pentagram.

```
(t (top (intersect
        (vertical m)
        (circle q (distance q s)))))
```

Make a large circle whose radius is the length ***q s****; the top intersection of that and* ***vertical m*** *is the point* ***t****.*

Now you've found the top of the pentagon.

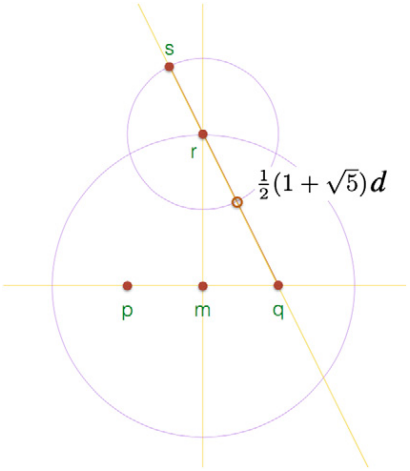

Figure 3.

Using a similar process, define the points **u** and **v**. This gives you all five points of the pentagon (figure 4).

```
(u (top (intersect
    (circle p d)
    (circle t d))))
(v (top (intersect
    (circle q d)
    (circle t d)))))
```

Once you've found the top of the pentagon, you can draw circles to find the left and right coordinates of the pentagon, and now you've got all five points - **p**, **q**, **t**, **u**, and **v**.

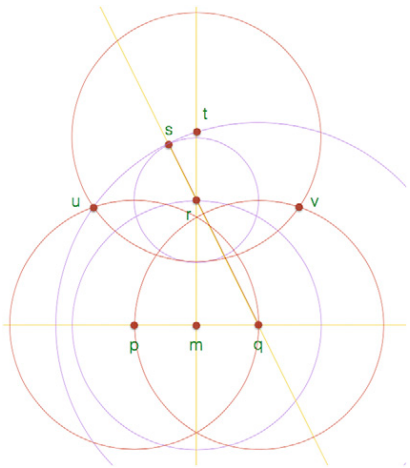

Figure 4.

All that is left is to connect the dots (figure 5), which is done with the parts list.

```
(make-curve t m
            (list (line t u)
                  (line u p)
                  (line p m)))
            )))
```

This defines a curve that begins at **t** and ends at **m**, and then mentions the various lines that you have to follow to get from **t** to **m**, which just gets mirrored.

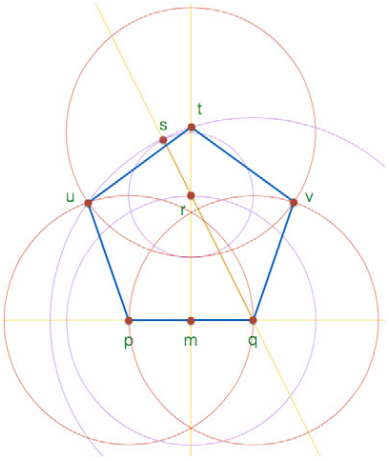

Figure 5.

Now, I know that's not a musical instrument. What's interesting, though, is if we get rid of the pentagon and some of the circles, there's a musical instrument hidden in there (figure 6): the lute of Arnault de Zwolle, which is from the 15th century manuscript in the Bibliothèque Nationale.

Just as the pentagon was my toy problem for how this language works, the lute is François' (and quite likely Arnault's) toy problem of what it means to draw a string instrument. It's a canonical toy problem for four reasons. First, it's written down; it's a written version of an oral tradition. Secondly, it's constructed using ruler and compass. Then, third and fourth, the way of laying them out is based on rational approximations, and on divisions of the square. Those

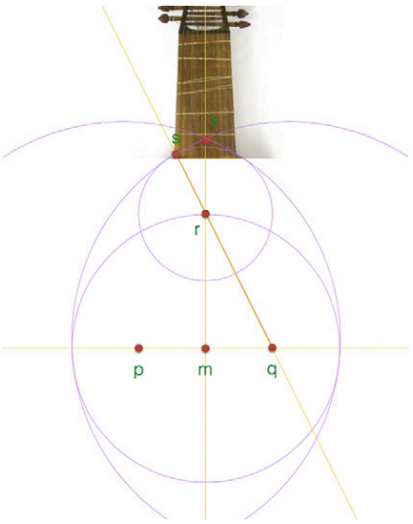

Figure 6.

are four things that are very important in the book.

Let me say something about the rational approximation. When you look at this lute (figure 7), you can see geometrically how it's drawn. There's a circle at the top, and if you mirror it down below, and you draw a circle in between, you will see there are two different radii, **m** and **s**. And it turns out if you divide **m** by **s**, you get the square root of 2. Which is

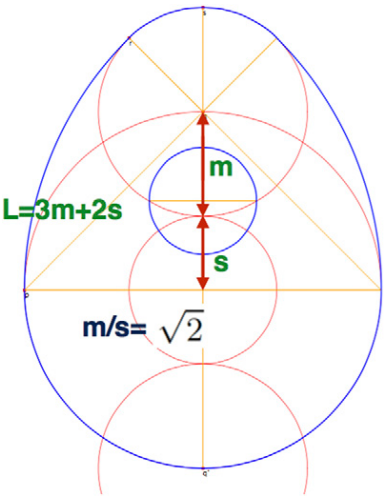

Figure 7.

not surprising, because that's the diagonal of a square. Part of the idea of instrument design is, if you take the square root of 2 and replace it with a rational approximation, such as the ratio 17/12, you can start drawing approximate lutes, so to speak. A lot of the constructions in François Denis' book come from this idea of rational approximation, using ratios of integers to approximate these kinds of ruler and compass constructions that are part of the toolkit that he uses in his book.

The fourth important thing is divisions of the square. Now, if you are bored silly by listening to me, you might be doodling. One thing you might do to doodle is draw a square, then draw a circle in the square, then draw a square in the circle and so on and so forth until your migraine takes over. But you might suddenly notice that "ooh, I'm doing the same sort of thing that he's talking about", because buried in this diagram (figure 8) is actually a lute trying to get out (blue outline). That's the division of the square, and it shows how the underlying structure of the lute comes out.

That's sort of preamble. In my programming archives, and you're free to look, just send me an email, what I did was code up this ruler and compass guide, and then I started repeating Denis's recipes out of his cookbook.

I've made code for the proportional construction of these instruments in Denis's book (see www.cs.brandeis.edu/~mairson/TDL):

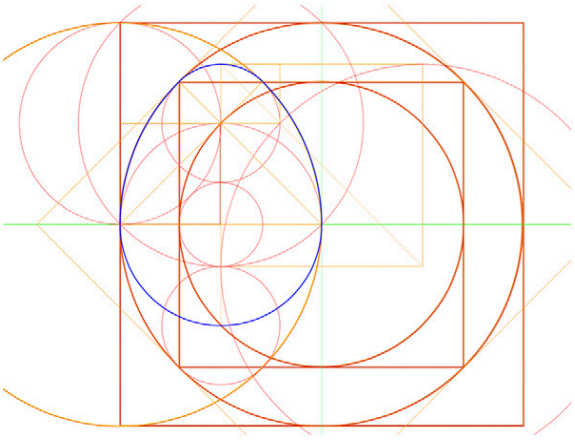

Figure 8.

- Violin by Andrea Amati [pp. 114-129]
- Violoncello by Josephus (filius Andrea) Guarneri [pp. 182-191]
- Violoncello by Domenico Montagnana *(Sleeping Beauty)* [pp. 192-206]

The construction of the code looks just like the constructional idiom of Denis's book. It's not a new method, just a different delivery method of the same thing.

The procedures that are in the code abstract over common constructional methods, in a way I'll describe. As I've said to François several times, "it's one thing to follow recipes, but I want to be able to cook. I want to be able to make stuff myself." So I've also tried doing that. I've also made code for the proportional construction of these instruments:

- Violoncello by Antonio Stradivari *(Mediceo)*, 1690
- Violoncello by Antonio Stradivari *(Cristiani)*, 1700
- Violoncello by Antonio Stradivari *(Countess of Stanlein, ex-Paganini)*, 1707

I'm sorry to continue a sort of Strad-fixation in this talk, because so much attention seems to be focused on Stradivari, but I picked these instruments for another reason, which I'll get to.

The code for any of these instruments comes in three parts. I won't go into all the boring detail here; contact me and I'll bore you then. First, there is the layout of the framework, which I call scaffolding, the horizontal and vertical lines. In this sort of system design, there's this Scheme/Racket language at the base, on top of that is my geometry engine, which is the ruler and compass machine. And then there's the code for the instrument itself. There is code for the lower bouts, the upper bouts, and the middle bouts; it's all fairly concise. It's basically what's in the book.

I talked about abstracting away from the ruler and compass construction. Think of straightedge and compass as a sort of machine language, an absolute base language. The idea is that you should be able to step away from the machine language and talk at a higher level. So, what could this possibly mean? One thing that has to be done in the Denis constructions is

something like inscribing circles inside of others. There's a whole chapter in the book on these geometric constructions. For instance, suppose you have two intersecting circles a and b, and you want to draw a circle c inside them, with a certain radius that in tangent to the interiors of those two circles. How do you place it? There are two solutions; the inscribed circle could be on the right or the left. Instead of going through the ruler and compass construction, I have code that says, call the "inscribe" procedure for circle a and circle b with the radius for circle c. There are two solutions, take the lower of the two.

I was talking to a luthier here yesterday who had done the Denis course, and he said, I can't remember now, a year later, how to do these various slights of hand, to get the points like this. I said well, if you can just abstract away from them, you've got code. You don't have to remember, you just plug in parameters and say, do it. And that's the value of programming languages, to structure complicated ideas, rather than having a lot of detail all at the same level.

Another thing that goes on in the constructions is reverse curves. This is used at the corners. You've got a big circle, which is like a hill. You say, roll a small circle of this radius up the hill until it's tangent at a certain point. You could do it with a series of straightedge and compass constructions. But again, we're better off just saying, do the reverse curve routine; here's the big circle, here's the radius of the small circle, and there's the point. Go do it.

There's a whole host of geometric problems, the ideas that are in this chapter of Denis' book, which can be solved once, at a higher level, by building them into the geometry engine. Then you don't have to do them anymore.

Another example is tangent lines between two circles. You can eyeball that one, but how do you do it analytically? It's interesting.

So what's any of this good for? Well, it's kept me out of trouble, let me write a few grant proposals, and my Dean doesn't bug me anymore. But also, I think it can be used to teach instrument design, math, programming—without numbers. It's all Euclidean, it's not Cartesian.

I'd like to build an online repository of famous designs, and I'm not simply talking about the PDF; I'm talking about the program, so people can look at the design that produced the plan, and modify it, so you can look at particular designs and change them. And maybe, if you're not consumed by proprietary angst, you will even put them back in the repository for other people to look at. I'd like to be able to use it as a construction prosthesis, so you can build molds more easily. And finally, the more speculative idea, my idea for computational art history. Or, as my teenage son likes to say, my plan for total world domination.

First, its use in fabricating a mold. How do you make a cello mold the normal way? You get a plan from The Strad magazine or somewhere, you trace out the outline, you move the outline in 5mm all the way around, with a pencil, you somehow transfer it onto a piece of plywood, you cut it out, you make a duplicate... every time you chain together these procedures, you're introducing error. If you could just do things once, and simply, and right, you could reduce your error. The next mold I make, I'm going to design it with this software, I'm going to print it out on a large piece of paper, I'm going to glue it onto the plywood, and cut it out. It'll just be right. So, I think this could help with mold fabrication. Before you get any sort of instrument going, you have to put some work into the infrastructure; this could minimize the infrastructure.

The beauty of PDF is that you can print it at any size. PDF is not a bitmap. PDF is a programming language that tells your printer what to do. It uses scalable vector graphics; it will print out accurate curves of all sizes.

You can make design modifications, from the silly to the sublime. Take the parameters and tweak them, and you can get different outlines.

You can do composite designs. This is something I've thought about because when I looked at the B-form cellos I thought, I don't really like the middle and upper bouts, I just don't like the curves. But I can use the upper two bouts of the *Mediceo* and the lower bout of the *Stanlein*. They're two completely different-sized instruments, but this is all proportional construction, so you can take the code for the upper two bouts and plug it in with the code for the lower bout, and have everything come out fitting together. And it almost does. There's a tiny bit of tweaking that I probably have to do, but this is what you can do. You can mix and match pieces.

And you can do parameterized design. I was asked, could you draw me a B-form pattern, but I'd like a smaller body stop, a smaller instrument size? Yes, this is not hard to do. So I think this is a more flexible tool for being able to generate outlines.

So now for the speculative part of the talk. Could we use a tool like this to search for something? I'd like to ask two questions. One of them is, where did the B-form cello come from? And the other is, what is the relevance of the proportional, geometric methods? What role did they play in the design of the B-form cello?

Bruce Carlson, in "The Evolution of the Stradivari Violoncello" (2004) has an illustration of a series of Strad cellos, a sequence of smaller cellos being made over time, all by Stradivari. And this is part of why I chose the Strad cellos to draw. The sequence of cellos reminds me of an evolutionary diagram. We know they got smaller. If you read Bruce Carlson's essay, he talks, in a sort of art-historical way, of how they got smaller. But I'd like to be a little more analytical. In what way, exactly, did they get smaller? What was really changing in the design? How did the curves change? How did the radii change? How did the placement of the circles change?

Secondly, the question of what's the relevance of geometric methods. One of the things that François says in his book, in chapter 3, is *"Close scrutiny of Stradivari's work shows a break with tradition alongside the master's classicism"*. What he's saying here is, the old proportional ways are beginning to die out, and the more modern ideas of measurement are starting: the kind of rulers that we know, the Cremonese points, the graph paper, the calipers, our more modern ideas of measurement. So, I'm looking at this chapter, and I thought, can I draw a B-form cello proportionally? Just completely proportionally, exactly the way Denis drew an Amati violin. Why do I need the points? I also wanted to know, if you wanted to draw a B-form, like the *Bass of Spain*, what did the earlier models, the *Mediceo* and the *Cristiani* have to do with that? How do you get from those to the *Bass*? From Cremonese point A to Cremonese point B? Part of my answer was, just try and do it, see how hard it is.

You can draw anything with a straightedge and compass. Anything, even the Mona Lisa. The question is, how hard you have to work. So when my code, my construction, began to look tons more complicated than anything I'd seen in the book, I knew I'd screwed up. But I thought, if I could keep my code at sort of the same level of descriptive complexity, then maybe proportional methods are okay. I also thought that it would be easier to proportionally draw the earlier cello outlines. They were more likely to be Amatisé.

I also thought, if you—Strad—are going to start experimenting with a new technology, such as a new idea of measurement, you'd be better off experimenting with violins. Violins are more likely to incorporate modern ideas because you can make a violin faster. You can experiment with the new technology and try things. When you have a huge overhead for building an instrument, as you do with a cello, you're more likely to stick with the older way of doing things. That's not a historical argument, that's not a Phil Kass argument, that's an *a priori* argument. It's based on logical principles, not historical evidence.

So, I wanted to try drawing one of the earlier models, and see where I screwed up. I actually tried drawing the *Countess of Stanlein* about a year and a half ago, and completely screwed it up, because I didn't know what I was doing. Now I know a little more about what I am doing. So the next I tried the *Cristiani*. I used pictures from the Cremona Strings Collection monograph, *"Violoncello Stauffer ex Cristani 1700",* and made tracings of the purfling on the front and the back, reflected. So I had four images, and put them all together, because the ribs are likely to be a little wiggly, and that way you're sort of averaging over all of the changes. So I did that, and the code reads like all the other code I wrote for the other instruments in the Denis book, with proportional things like regulating the length and the width, 7 parts to 5, all the sorts of things Denis talks about. I should add that I did this this summer, when I spent a week with François, and about every twenty minutes he had to box my ears. The French word that he used for me was "têtu"—stubborn. He helped me recover from a variety of mistakes. But what I came up with is a perfectly proportional way of printing out the *Cristiani* outline. And it has the same level of description we were seeing for the Amati violin, or the *Sleeping Beauty*, or any of that.

I laid my outline on a photo of the back of the *Cristiani* (figure 9). It's not perfect, because the drawing is completely symmetric, and the back isn't. There's wood shrinkage, there's parallax in the photo, and I may have screwed up some of the dimensions a little bit, but you get the idea. Now, as soon as you have an outline, you want to start plugging it in to every photo of every other instrument you can to see how they fit. I superimposed the idealized drawing of the *Cristiani* on a photo of the back of the *Castelbarco* Stradivari in the Library of Congress, and it's a nice fit. It starts providing evidence that those were made off the same mold.

The *Mediceo* is a much bigger instrument. I took the tracings of the *Mediceo* purfling, front and back, left and right reflected; same game, same everything. With one little exception. Unlike the *Cristiani*, there is a little tangent in the lower bout like you see in the upper bout for the Montagnana. In the upper bout for the Montagnana, this tangent gives you that sort of "manly-manly" look; you get something different here.

So with this tool we can start doing comparative cello designs. Now we have two outlines that are generated fairly accurately, and you can look at the two of them and say, what changed? What changed in the outline? We can try to relate comparative geometry, and also comparative method. And we have three ways of talking about it: by the curve of the instrument itself, by the underlying circles and arcs, and by the code that gives the proportional dimensions. Three ways, three perspectives on thinking about the same thing.

Finally, I got to the B-form cello, the *Countess of Stanlein*. I traced the purfling, front, back, superimposed, all as before. I took the code for the *Mediceo*, which is a much bigger instrument, and shrunk it and tried to fit it to the upper and middle bouts (figure 10). The outline of the Stanlein is in gray. If you look at the framework, the horizontal and vertical lines from the *Mediceo*, it fits the B-form cello. The horizontal lines, you can see, are in just the right places. There's just one thing that's wrong. If you shrink it like that, you have a difference in the lower bout. But if you take the drawing and shift it all the way to the lower bouts, you can see the curves and the dimensions are nice. So this starts

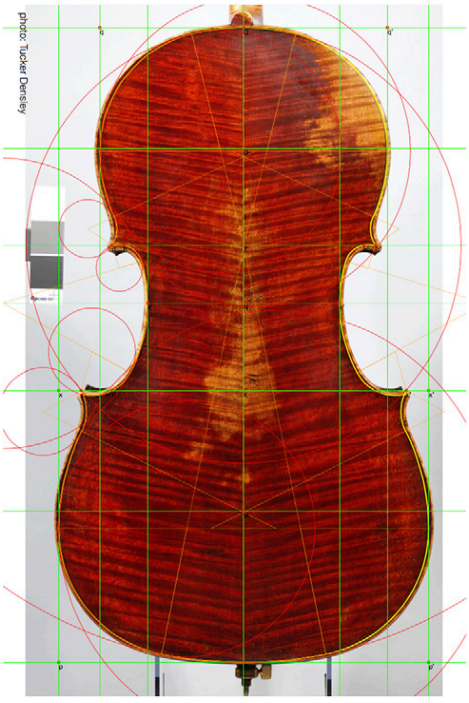

Figure 9.

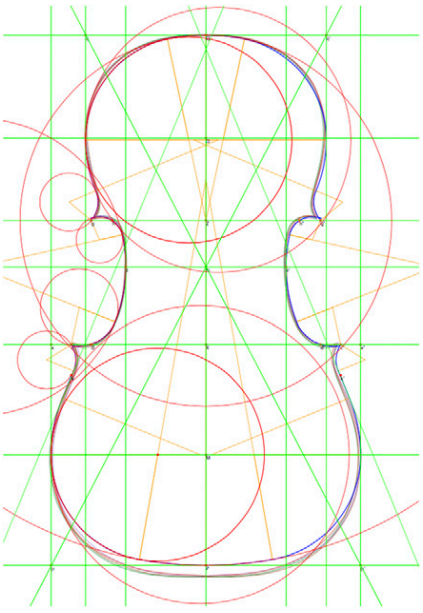

Figure 10.

making you think, in an *a priori* way, where did the B-form cello come from?

Here's where it could have come from. Do the math:

- The *Mediceo* is 792 mm long, with a huge body stop of 432 mm.
- A B-form cello is about 755 mm long, and the body stop is 400 mm and some change.

If you reduce the body stop by about 27 mm, from 427mm to 400 mm (mold dimensions) you're reducing the whole dimension about 6.3%. That 6.3% reduction on the mold length, which is 782 mm (take the length of the *Mediceo* plate, 792 mm, and subtract 10 mm for rib height plus overhang), all of a sudden you've got a mold that is 733 mm. That's too short. There's a 12 mm gap. It has to be put back somewhere. How was it done? How were the top two bouts reduced proportionally, the dimensions, so it looked right? How was this done, if not by proportional methods? If you move the lower bout down about 12 mm, you can then add the 12 mm back by patching in at the corners of the lower bouts.

So, the code that I wrote for the *Stanlein*, the B-form cello, is basically, "take the *Mediceo*, scale it down to the appropriate overall dimensions in the top two bouts, so it fits there. Move the lower bout down about 12mm, and then look at the *Mediceo's* tangents, which are already there, and just extend them a tiny bit". And there you have, pretty much, the B-form cello.

Anytime you make something, you're probably making it from something you made before, that you changed. Here are two artifacts, one that came earlier, and one that came later, and we have a reasonable explanation of how you might have gotten from one to the other. And one of the key things in doing it was understanding proportions.

Now as to why the gap was 12 mm, why it was that distance, is something else. I don't know. That's another dimension. I have no idea. So here's the drawing of the *Stanlein* superimposed on a photo of the *Stanlein* (figure 11). What's very nice is, I said, draw me a circle that is 400 mm from the top of the mold, a nice big smile right through the middle, and it goes right

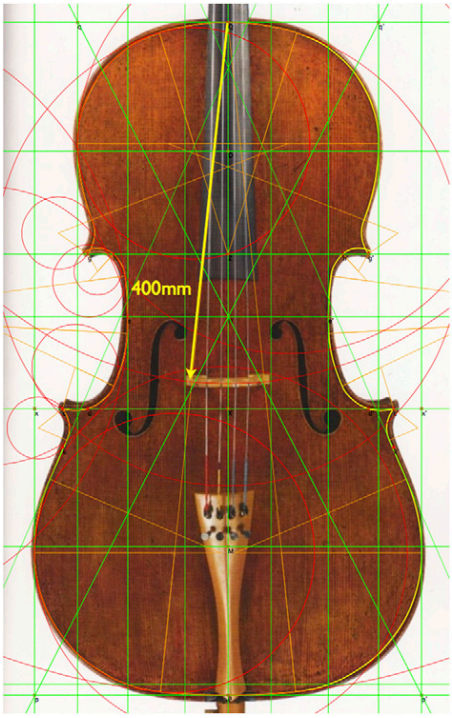

Figure 11.

over the top of the bridge. That made me happy, to think I knew something about the body stop. Here's pictures of the relative size of the two instruments (figure 12), the big blue *Mediceo*, and the little red *Stanlein*. On the right, I've scaled the drawings to match the upper and middle bouts, and we can see where the differences are.

When I started superimposing these pictures, I said to myself, I've seen pictures like this before. I've seen them in Stewart Pollens's book on Stradivari. He superimposed a lot of the molds and made drawings to show how were they different. The beauty of the internet is that you can send anybody email, even Stewart Pollens. So I wrote to him and said, how did you make these drawings? He answered, "*the line drawings were made by placing a sheet of translucent plastic film over life-size photographic enlargements of the forms and tracing them. The images of the forms were not adjusted to different proportions.*" So they are "as-is". The difference between the images that you see in his Strad book and the ones that I'm generating is that my superpositions are proportionally reduced.

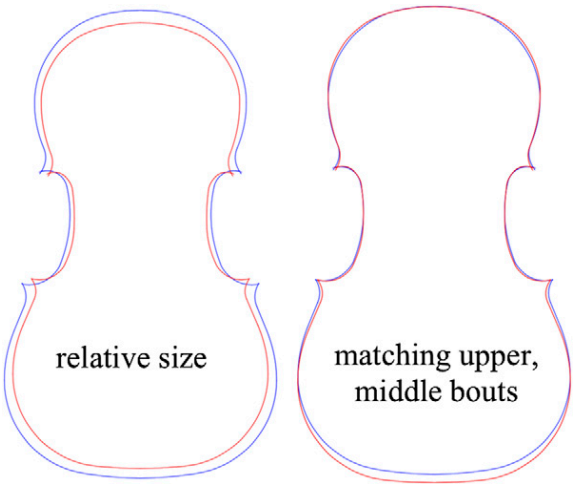

Figure 12.

I'm talking about a proportional translation, which I think is evidence that the proportional technology known earlier was still understood.

Even if you think that I'm totally blind, and you don't think that the *Mediceo* is the archetype for the B-form—and it might not be—it still makes sense that an earlier, proportionally-generated form could have been used to do this, according to the cut-and-paste analysis that I've done. Maybe someone else could do this better than I can, but you get the idea of the sort of thing I'd like to do. And, more importantly, you get the idea of the software tool that I think makes this possible, because you just can't do this with pencil and paper.

That's pretty much it. If you're interested in plugging in at any level, email me. Anything from, "would you send me 15 copies, from your large-scale printer, of the following five instruments..." to "could you send me the code so I can tweak it", or "I'd like to do some more profound design". Send me an email, come talk to me, because I'm really looking for people to try this out. I would like to try to make it better. I wrote ten grant proposals last year for this project, and one of them totally annoyed me by saying, "no one is ever going to use that except you". And I thought, "now I need to go to VSA and convince someone else to use it, and then I can write back to these people and say, 'You're wrong! They're breaking down the doors!'"

Another thing. Anytime you learn something new, the people selling the new thing say this is going to make everything easier. And it never does, because what it really means is now you have to learn something else, and okay, somewhere down the road it might get easier. Computers are a perfect example of that. One thing I want to say is, if you look at this code, and think, "he's a computer science professor, I'd never be able to do this myself", I want you to pretend that you know what you're doing. You'll really get far. My standard example of this is that I used to make up web pages for courses, and I didn't use any web page designer. What I would do is schmooze around the internet and find a page that some colleague had, that I liked. Then I would download the HTML and say, I'm going to replace his picture with my picture, I'm going to replace his phone number with my phone number, and pretty soon I had this really cool-looking page. Do I know how HTML works, really? No. I don't know any of the niceties of it, all the ins and outs; I have no idea. But you don't need to know them to be able to do this. You might be able to look at the code that I've written and say, oh I don't understand a lot of this, but I think I know how to change this number, and it won't blow up in my face. You can still experiment with pieces of it and make it do new things.

That's all I have to say, but I have a lot of people to thank. First, let me thank François Denis. Many of you don't know this, but François Denis has two full-time jobs. One of them is doing all the stuff you know about. The other is responding to my email. So, I'm eternally grateful to him, and if he hadn't written his book, I wouldn't be having all this fun. Secondly, let me thank Curtis Bryant, who is a cello maker and restorer in Boston who said to me, "you have got to read François' book. This is the way it is". He's given me tons of advice over the years. I need to thank Marilyn Wallin, who through a series of courses taught me just about anything I know about putting instruments together, and finally, my grad student, David Van Horn, now a professor at the University of Maryland, because I showed him a preliminary version of this and said to him "I just want to quit my job and do this." And he said, "No, Harry, this IS your job." That was some of the greatest advice I've ever gotten. So, thanks.

I'm happy to answer any questions. Email me or talk to me after. If you want to look at any of these drawings up here, I have the *Sleeping Beauty* here, the B-form, the *Countess of Stanlein*, maybe others.

Links

• Mairson, Harry, *Functional Geometry and the "Traité de Lutherie"*, 2013 ACM International Conference on Functional Programming, pp. 123-132. This is an academic version of this talk, and is available online: http://www.cs.brandeis.edu/~mairson/Papers/ICFP062-mairson.pdf

• You can download and install Racket from: http://racket-lang.org/download/

• To get the code, write Harry Mairson at harry.mairson@verizon.net or see www.cs.brandeis.edi/~mairson/TDL.