Using systems techniques to make CNC fabrication more fluent

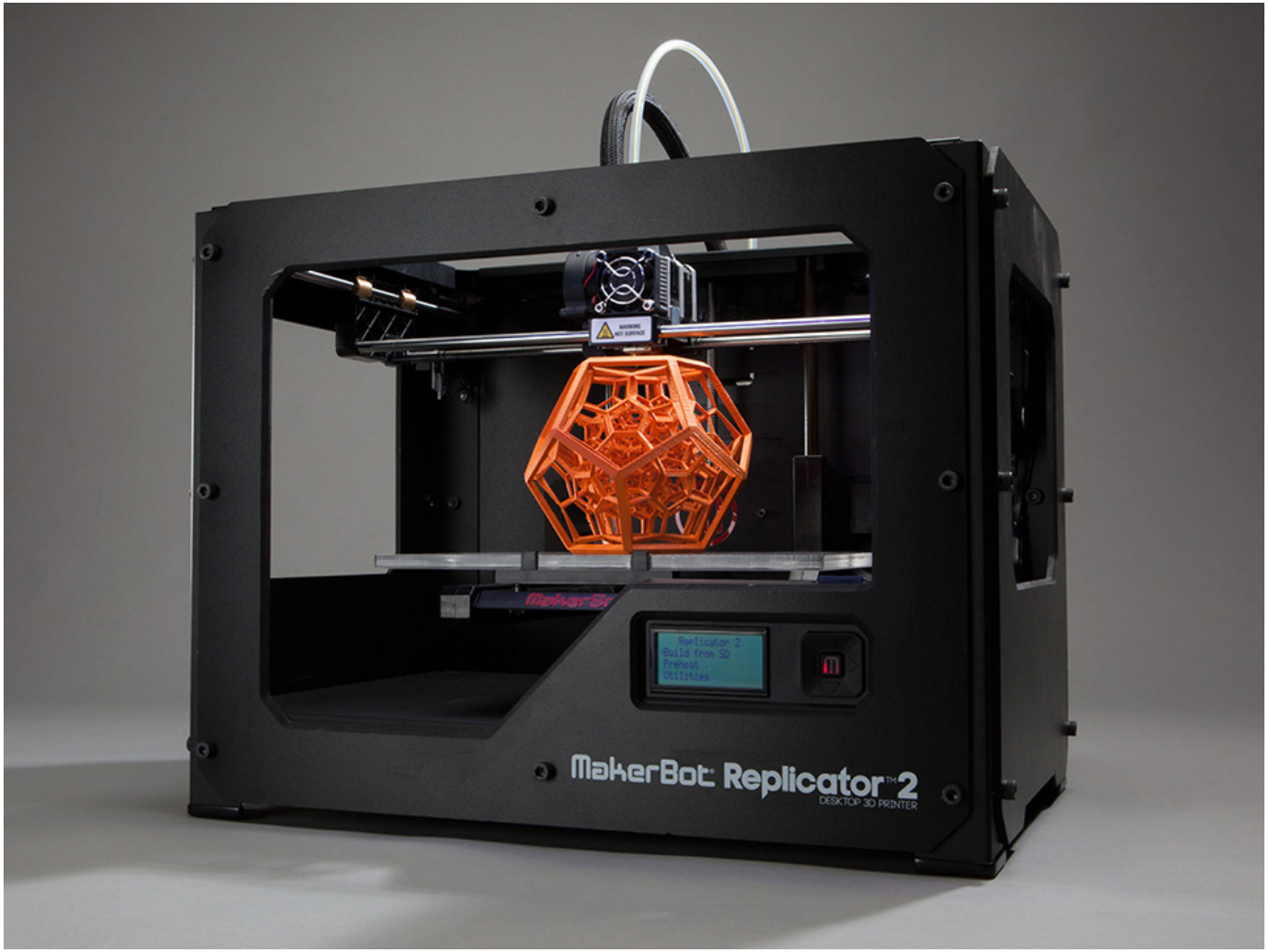
Dawson Engler
Stanford

Secure Internet of Things Project Retreat
Santa Cruz
June 7, 2018

Computer-controlled fabrication

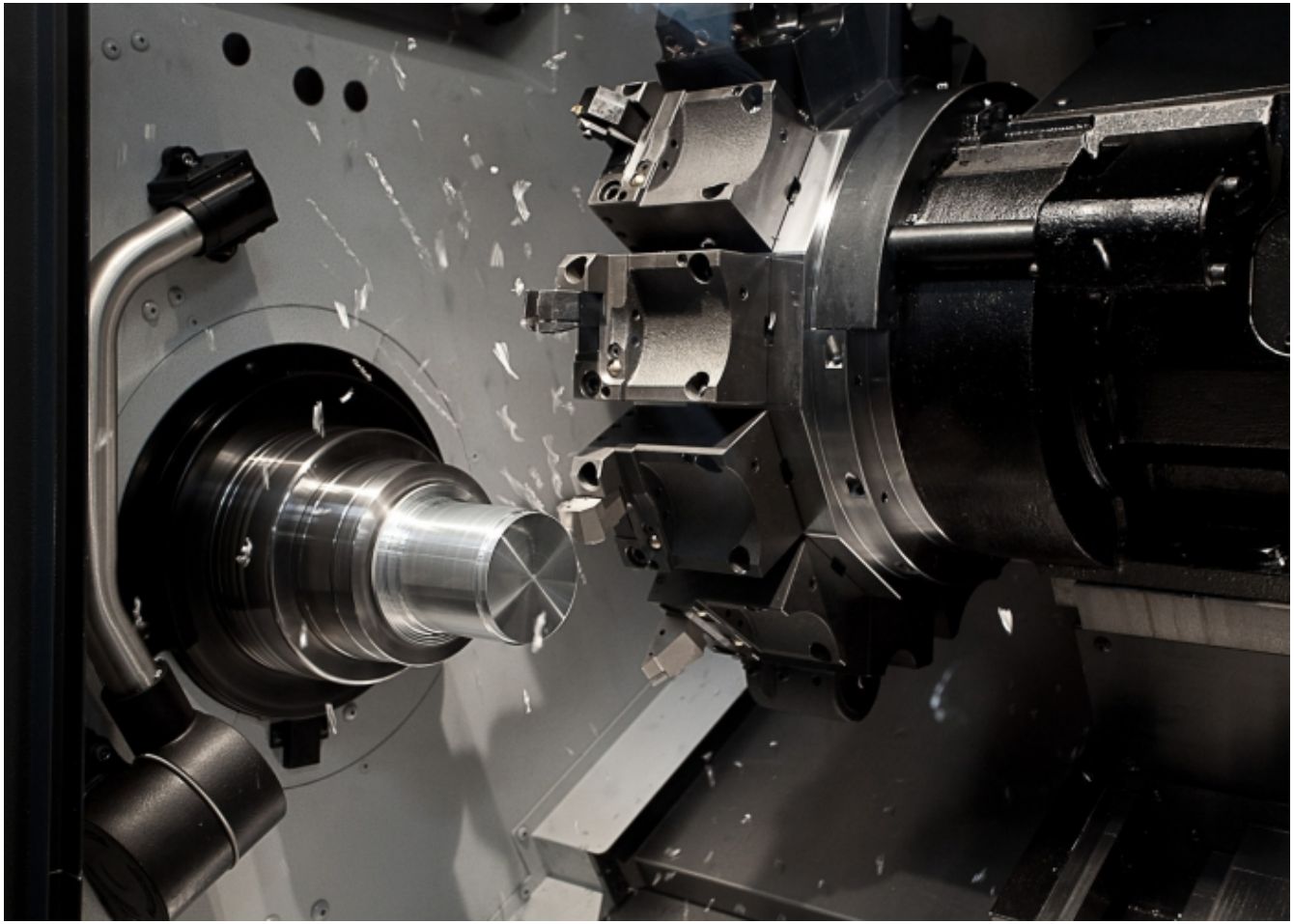
- ``CNC''
  - Started in 1960s.
  - Now is everywhere.
  - Generally: use computer control to turn a ballscrew by a specific amount, thus moving tool by a precise, deterministic distance along axis.
- Almost any object you touch contains some-to-all pieces:
  - Made by a CNC machine directly.
  - Or made by machines that were made by a CNC machine.

3d printing.



CNC mill.

CNC lathe



CNC router

CNC water-jet

Rules of thumb

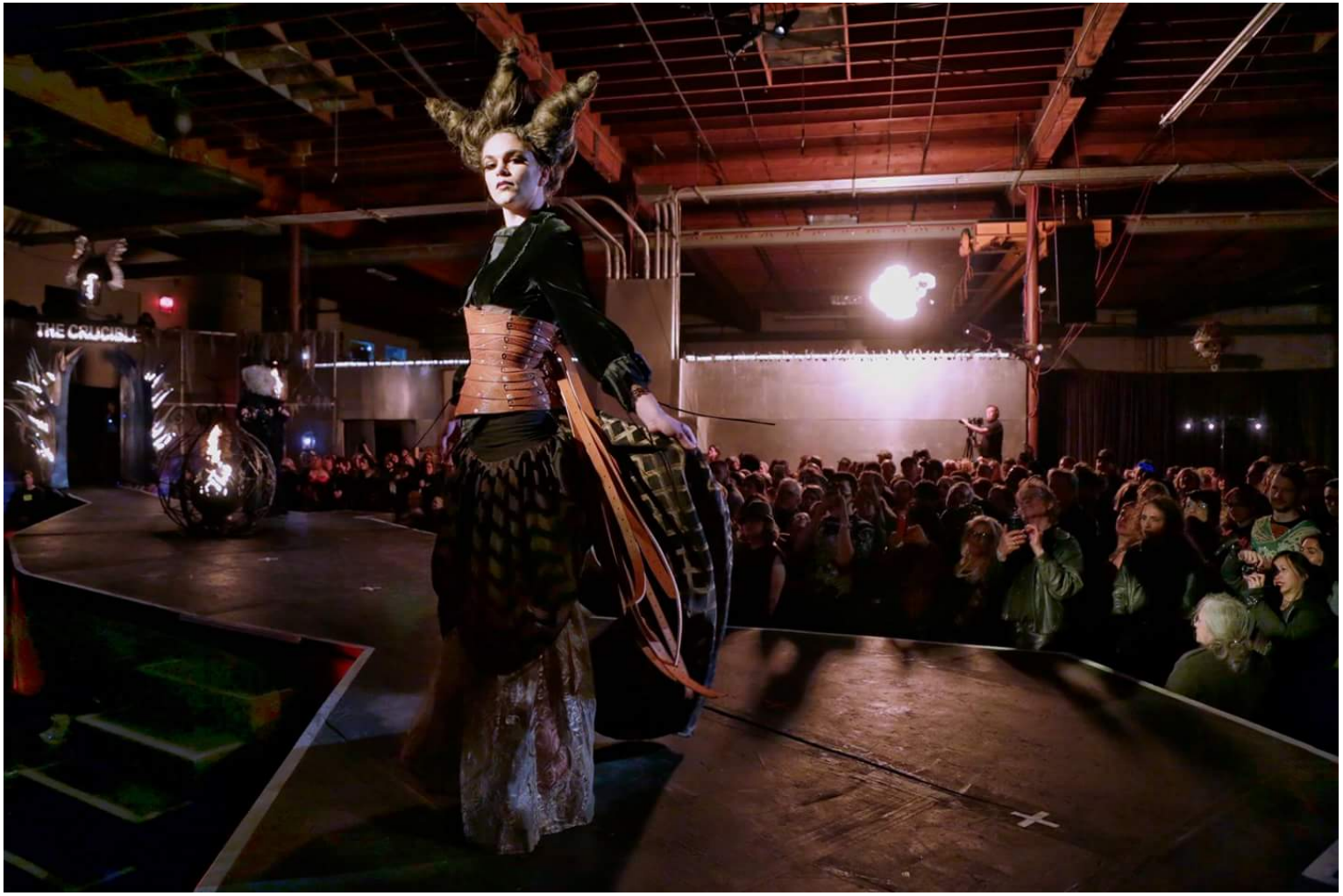
- Machines:
  - heavy (thousands of lbs)
  - precise (thousandths of inch),
  - very very dangerous,
  - easy to wreck. (designed for this!)
  - Expensive.
- Their software, generally:
  - Black box. Expensive. Complex. Problematic.
  - Designed for people who don't know how to program.
  - Labor intensive
    - Usually many mouse clicks to get something done.
    - Ephemeral program must re-type in each time.

Relevance to IoT

- CNC machines = meta-Things that make Things.
  - Embedded
  - Real time
  - Heavy users of sensors.
- IoT = significantly higher probability of fabricating a thing.
  - Will use CNC.
  - Faster prototyping and more accurate, safer fabrication is important.

Motivation: leather+metal for fashion shows.

Motivation: Non-embarrassment. Secret weapons



Reasonable test

- Failure is super embarrassing. Keeps honest.
- Need expressive:
  - Free form designs
  - Different materials
  - Lots of regularity, lots of difference.
- The rule:
  - Assembly can be manual (hammer, sewing)
  - All cutting is by machine. No exceptions.

- More than 10 shows
  - Last four can type “make” and get all programs emitted for 14 models.

This talk

- Improve all CAM systems through PL techniques
  - Low level analysis + rewriting = lift all boats.
  - .
- .

Low hanging fruit

- Almost all controlled by primitive language, Gcode
  - Origin, 1960s.
  - Ignored by programming language community.

```
T3 [ GET TOOL NUMBER 3 ]
 S16000 [ SET SPINDLE SPEED RPM ]
 M3 [SPINDLE ON]
 G53 Z0 [LIFT Z TO TOP]

 [TOOLPATH NAME: Drill 1]
 F5 XY [SET FEEDRATE FOR X AND Y]
 F50 Z [SET FEEDRATE FOR Z]

 G0 X0.000000 Y0.000000
 G0 Z0.800
 G1 X0.000000 Y0.000000 Z0.000000
 G0 X0.000000 Y0.000000 Z0.315000
```

- Many problems have easy PL solutions.
  - Retargetting.
  - Optimization.
  - Error detection.

Tenths: gcode rewriting system

- simple, extensible g rewriting system.
  - Reads in gcode, user-written extensions transform/check, re-emit.

  - Rewriting code = can work with output from anything. Lift all boats.
  - (~ Similar to link-level optimization for machine code.)

Optimization (speed, accuracy)

- CAM software has to produce results quickly.
  - Toolpaths, optimization deal with NP complete problems
  - Also deal with variables the CAM software may not know about.
  - Gcode optimization lets us do extreme and/or more insightful opts.
- Is written by a software company.
  - You will often understand how to machine your materials with your tools on your machines better than a software (CAM) house.
  - They can write the optimization passes you use.
- Often you figure out how to fabricate better over time.
  - Gcode rewriting lets you slip these tricks back into old code en masse.

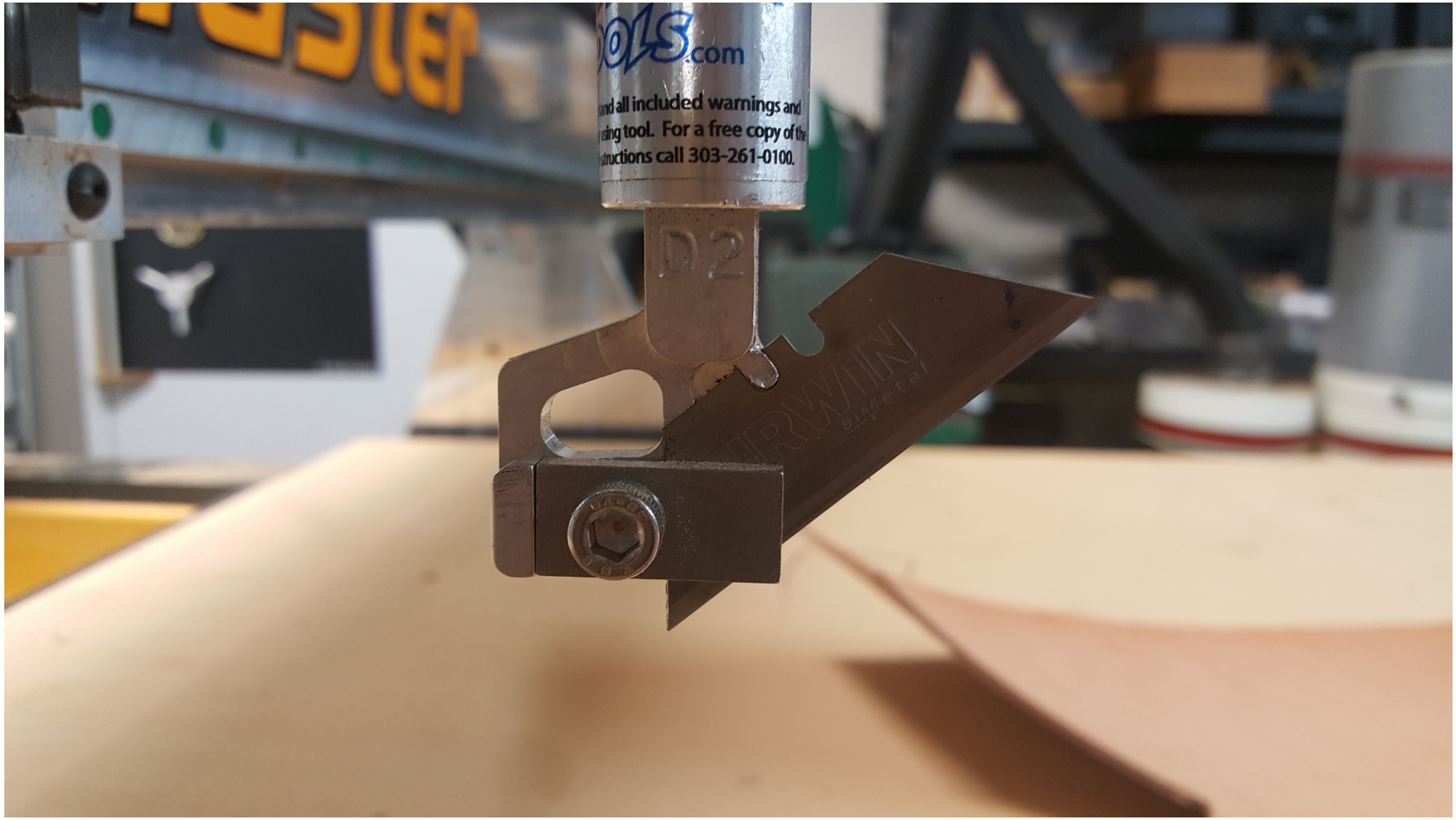
Router

Fabrication pipeline

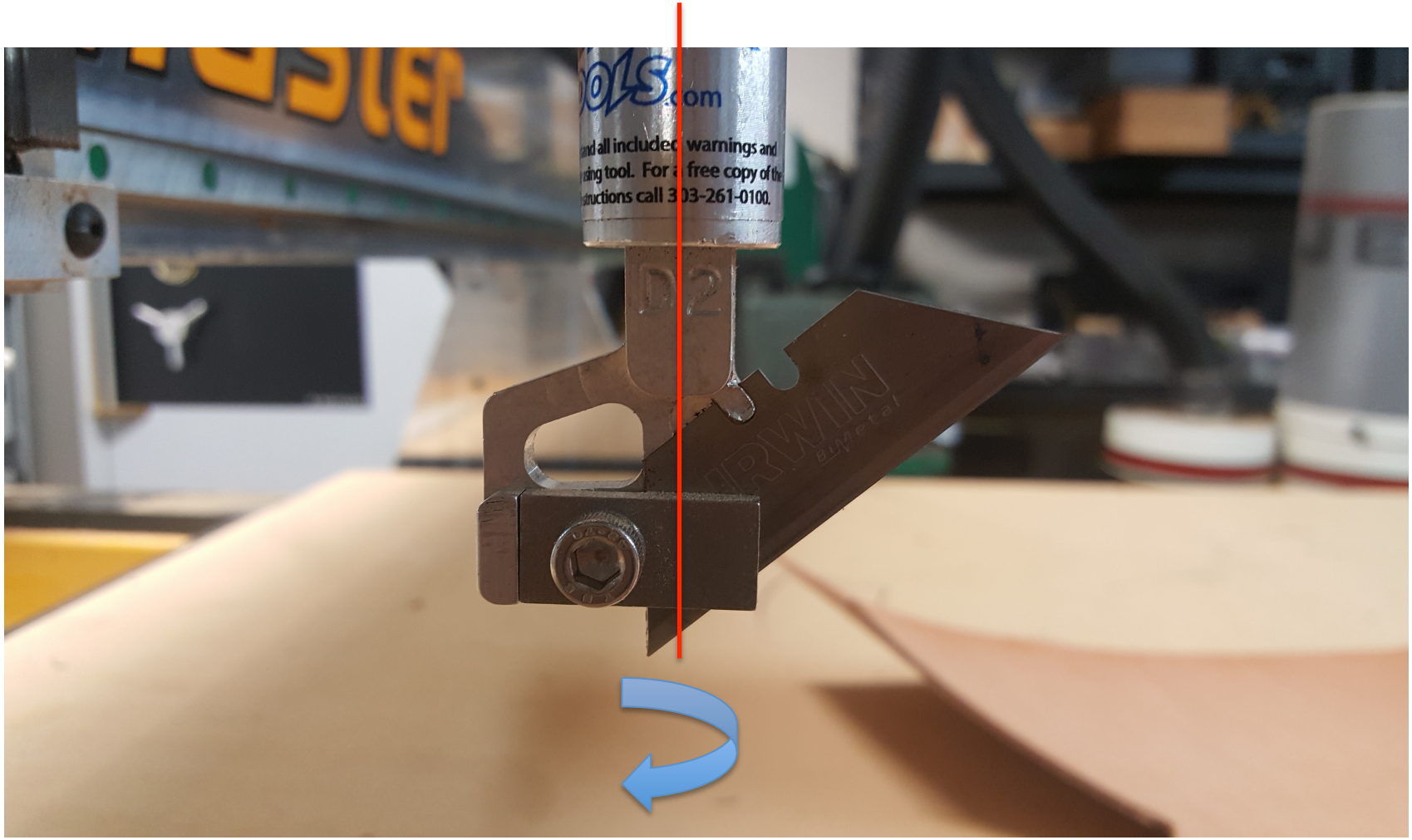
- Fabricate using a common pipeline:

  - The software it ships with (vcarvepro) has many problems.
  - Closed system. Can't do anything to fix.
  - From bitter experience: Working around manually = difficult, dangerous.
  - Using gcode rewriting makes it easy.
    - Problems that can be solved similarly show up everywhere.
    - Picked these b/c easy to explain and make video.

Cutting tool: dragknife



3rd party cutting tool: dragknife

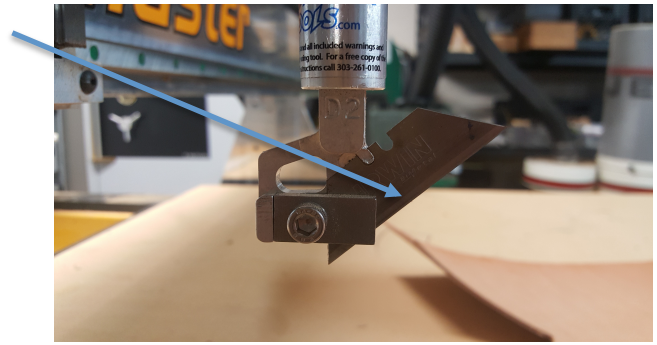


CISC vs RISC

Fixed bugs. Now fix quality

Post-processing to optimize cutting

- The commercial system isn't very accurate.
  - Overcuts b/c razor has a 45 degree angle.
  - Doesn't have option for accurate pivots
  - Seems to have some math issues.



- Solution:
  - Implement our own accurate dragknife tool path generation.
  - Use rewriter to rip out old toolpath and insert our more accurate version in.
  - Communicate to the rewriter by using a fake tool with special name
- Common pattern:
  - optimize drilling, or cutting speeds, etc by post processing
  - Similar to object-code optimization. (OM, etc)

Real use

Real use

Real use

Final product

Rewrite old code with new tricks

- Common: Work with material / tool / machine more?
  - The more tricks you come up with.
  - But old code stays in the old way.
  - Use gcode rewriting to rewrite it with the brave new way.
- Examples:
  - Stress relief
  - Strap cutting tricks

Stress relief cuts

- Obvious in retrospect, not so much at the time.

Works much better (accordion)

Strap cutting is tricky.

- Multiple errors in vid
  - Cutting strategy waste
  - Holes = sieve.
  - …

Can this program destroy the machine?

- Have done bug finding for almost 20 years.
- Can import these techniques, with interesting twists.
- Makes a bunch of problems much easier.

- Currently: have to completely trust the code you run.
  - Did you bump the keyboard? Stray mouse click? Machine can destroy itself.
  - Many properties can verified right before
  - Mechanically reduce trust dramatically.

- Obvious: just do static checks.
  - E.g., make sure don't cut outside window.
  - Would have prevented a near destruction from solidworks!
  - Many of these. They work as you expect.

- Less obvious: cross checking (simple, but serious win).

Cross-checking in one slide

- From regular checking: is this code correct?
  - Typical: no spec.
  - Also typical: other code exists that implements the same interface.
  - So: Cross check (symbolically, statically, dynamically)
  - Any difference = an error. If one is correct, you've shown (sometimes proven) the other is correct too.

  - Works here too!
  - Main game: to see if program is safe, try to link it to others.

Can program crash machine?

- Old program:
  - no idea if it uses the same tools on the machine
  - If it fits within the current work envelop.
  - Look at gcode? (Ha)
  - Re-cam? (easy to make even worse mistakes).
- Simple finesse:

Can program crash machine?

- Old program:
  - no idea if it uses the same tools on the machine
  - If it fits within the current work envelop.
  - Look at gcode? (Ha)
  - Re-cam? (easy to make even worse mistakes).

- Simple finesse:
  - You know the last few programs you have run.
  - They have (apparently) not crashed the machine.
  - So simply cross check the old code against these.
  - If it uses the same tools, and fits within the same envelop = safe. (Fairly typical) Otherwise extract the delta.

Can program crash machine?

- Old program:
  - no idea if it uses the same tools on the machine
  - If it fits within the current work envelop.
  - Look at gcode? (Ha)
  - Re-cam? (easy to make even worse mistakes).

- Simple finesse:
  - You know the last few programs you have run.
  - They have (apparently) not crashed the machine.
  - So simply cross check the old code against these.
  - If it uses the same tools, and fits within the same envelop = safe. (Fairly typical) Otherwise extract the delta.
  - Very easy (50 lines) to do on 3-axis machine.
  - Will scale to 5-axis very soon.
- Has been a serious nagging problem in shop.
  - Once you look at the right way is pretty easy.

Other patterns: previous version versus this one

- After crude geometry cut, then start tweaking:
  - Tool depth of cut and feedrate
  - Tool type
  - Order of cuts
  - Adding a single feature.
- A stray mouse click / keyboard can destroy everything.
  - Have to run at 1%, get ready to hit the stop.
- However, is easy to “delta verify” that only that single part changed.
  - --only-doc tool=1
  - --only-feed tool = 3
  - --only-tool-swap tool = 3
  - ….
- Can verify the rest was untouched. Much much improved.

Summary

- Gcode:
  - From the 60s.
  - Controls almost all CNC machines.
  - Entirely ignored by the programming language community.
  - Machinists have many problems b/c have a passive relationship with code.

- Automatically post-process it to:
  - optimize, retarget, correct.
  - Even trivial passes make big different for real people.
  - We use it all the time in our studio.

Fluent: flexible, freeform fabrication

- Programmatic, parametric modeling and fabrication:
  - Model geometry in 3d or input from files (DXF or even gcode)
  - Robust, simple flatten to 2D.
  - Programmatically add features (rivet holes, seam lines, grooves.)
  - Automatically emit gcode to cut.
- Nice-to-killer features:
  - “One touch”
  - Turing completeness = have to anticipate much less
  - Robust.
  - Easy to do novel CNC tricks: have control, have all semantics.
  - “Delta design”: import geometry from different sources, quickly easy modify / combine / remix.
  - “Constraint-based naming”: robust feature naming even with parametric changes

Example: corset

Make corset from thick leather from four annuli

Did this kind of pattern for ~ 1.5 years with solidworks.

Arguably best 3d->2d CAD.

Very

Very

Very

Painful. (google: “solidworks” “sucks”)

Constantly broke.

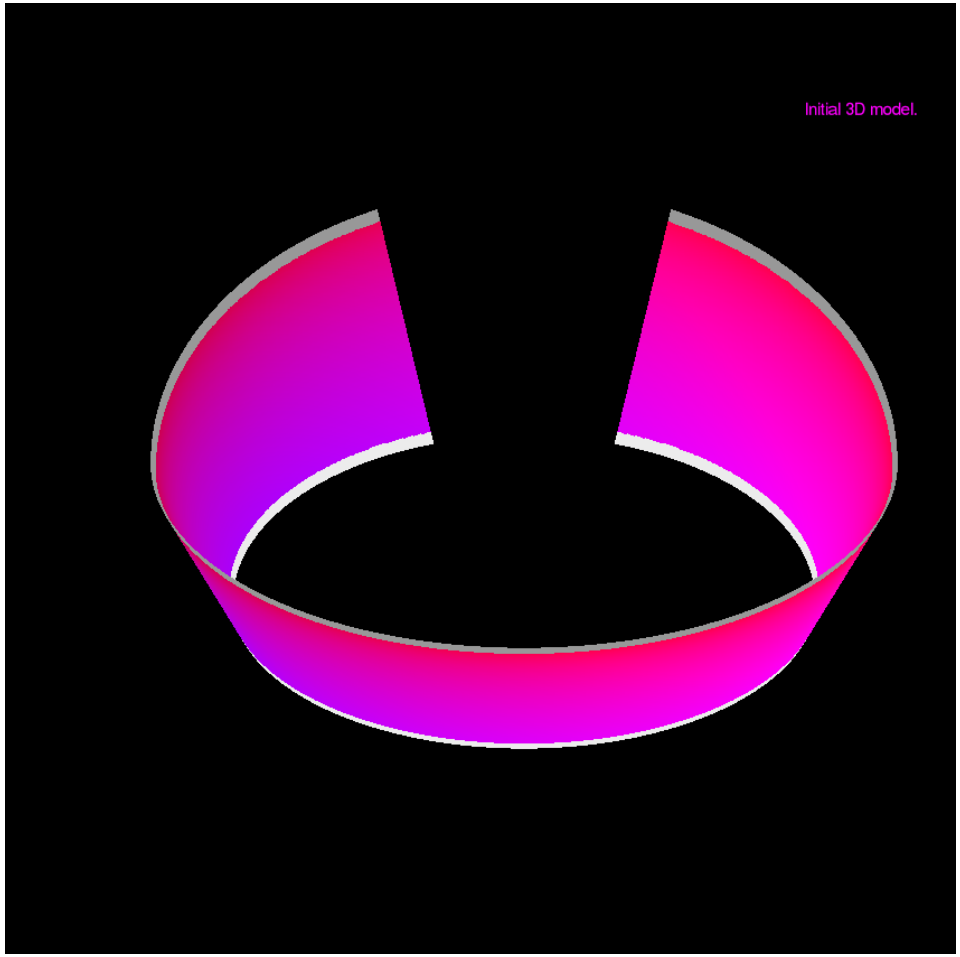
Wearisome amount of manual effort to convert to gcode.

At best: 20-30 minutes / iteration

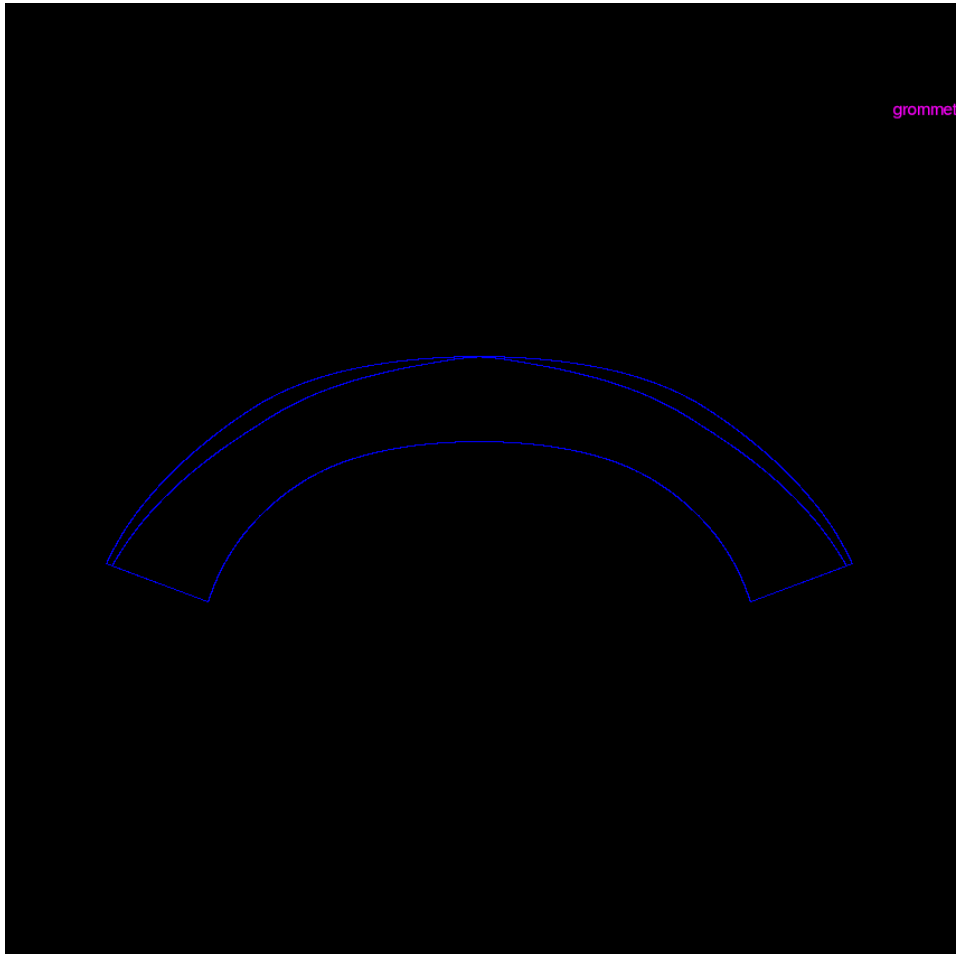
Worst: Forget steps, make mistake

Fluent makes much easier.

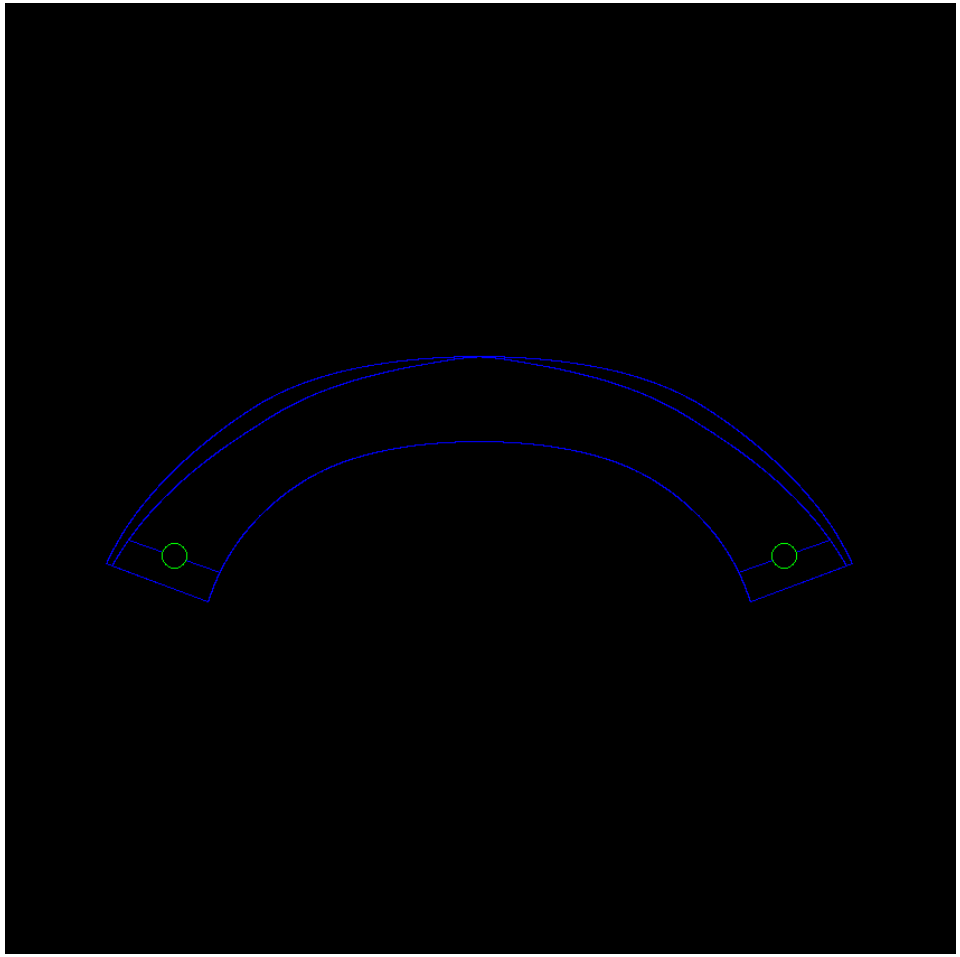
3D bottom corset strip



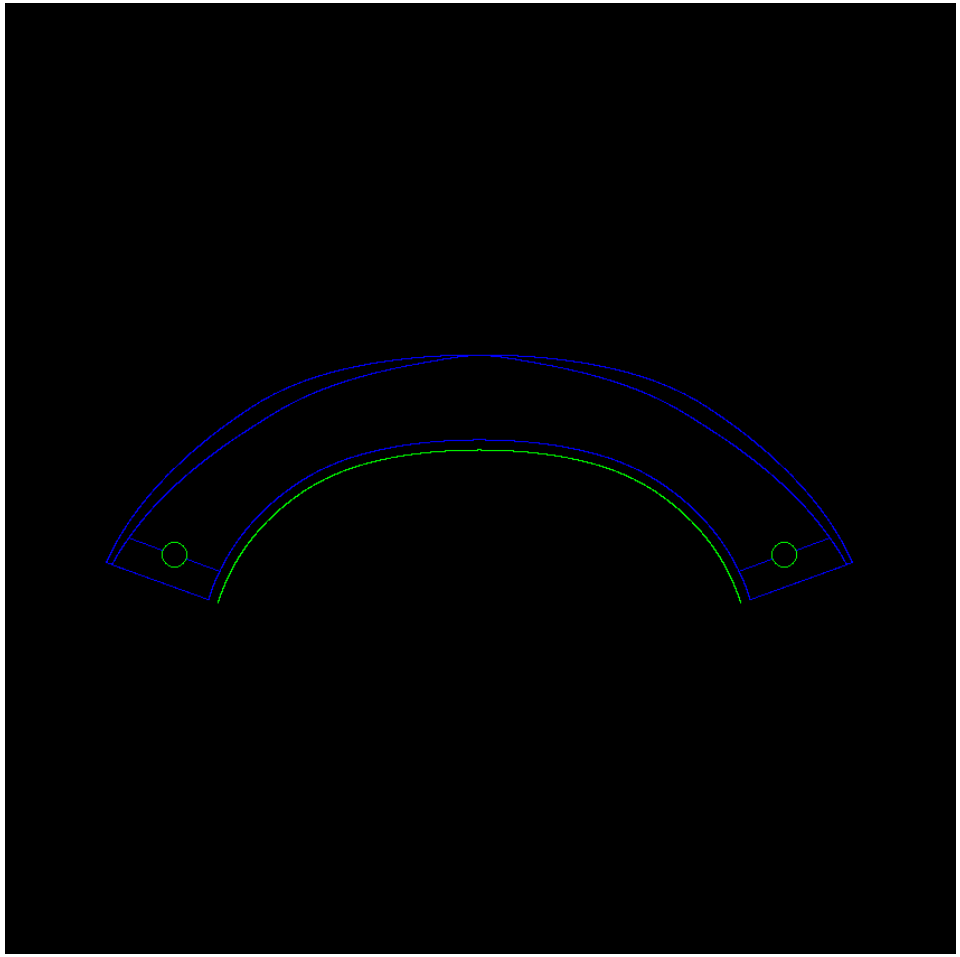
Shape hemline, flatten to 2d



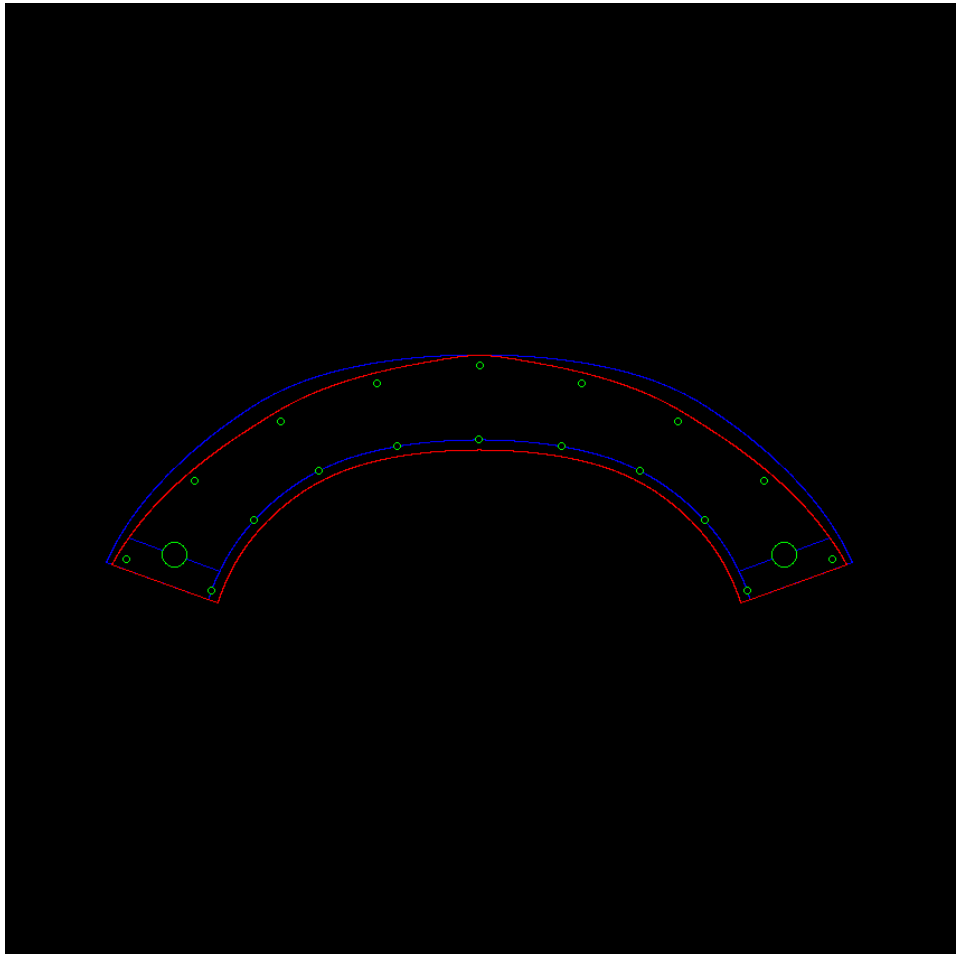
Add grommet holes



Add seam allowance

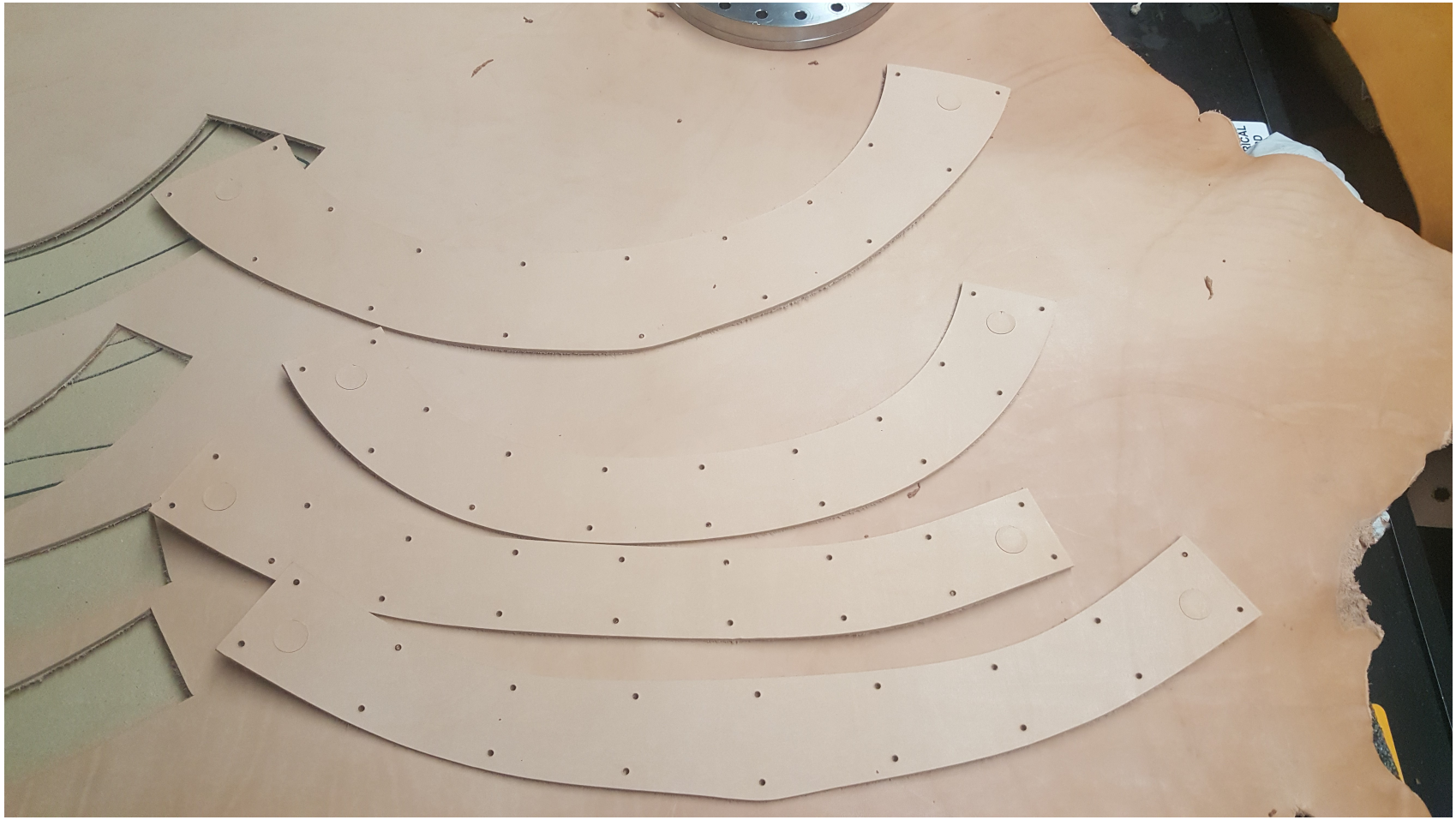


Add rivet holes

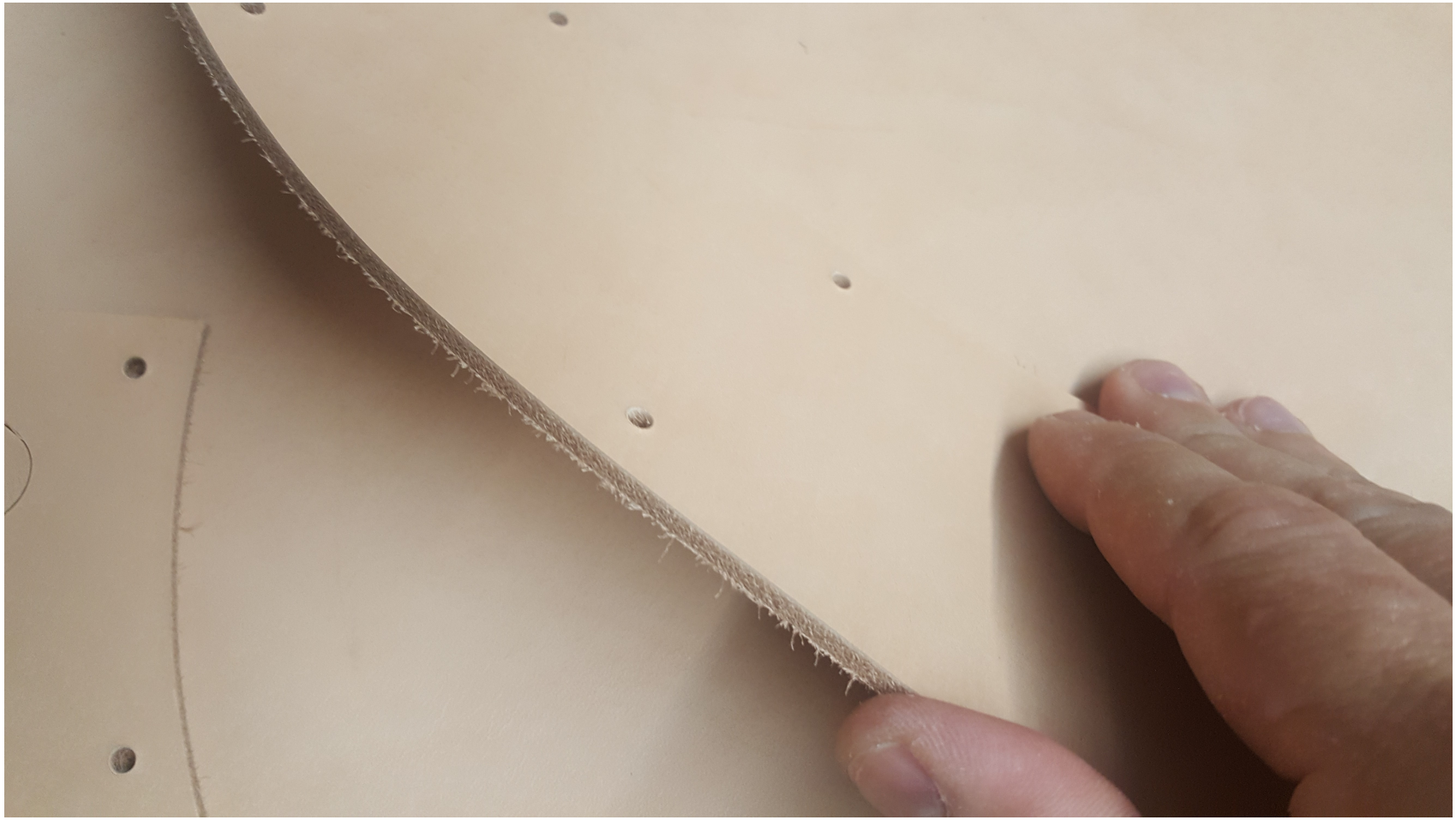


Auto-gen and cut from leather

Auto-gen and cut from leather



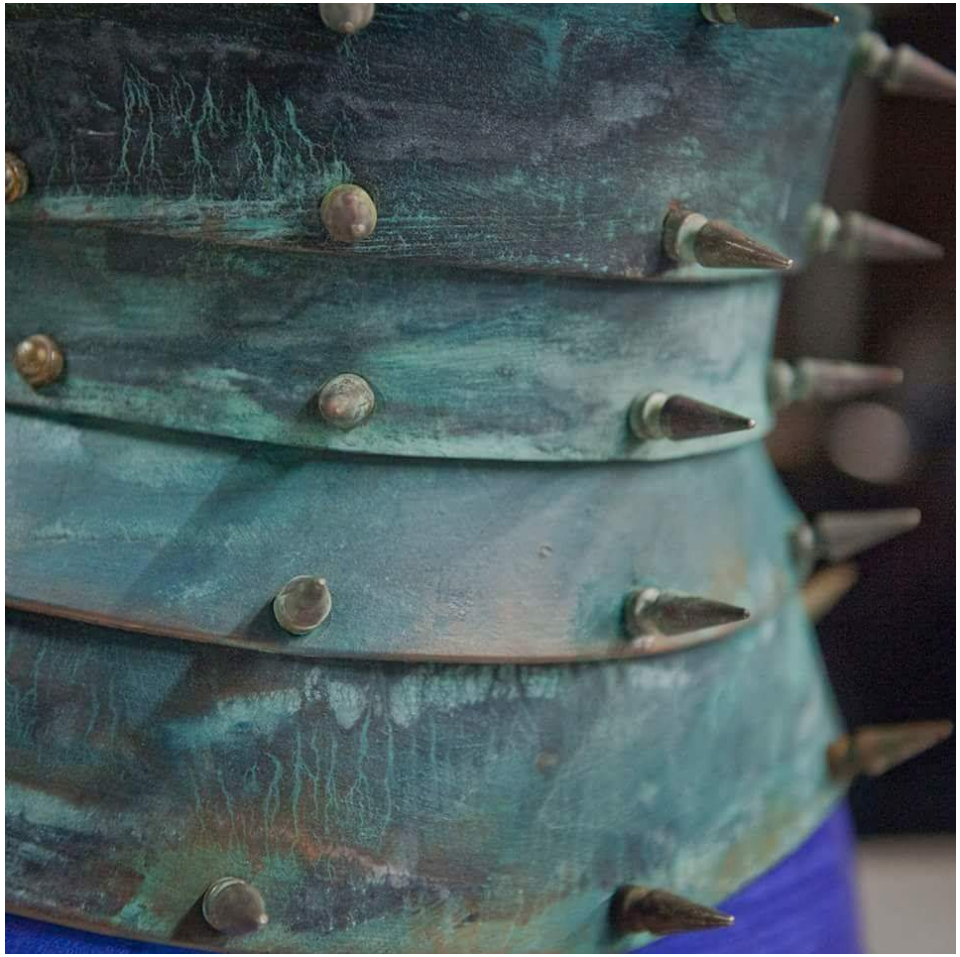
Auto-gen and cut from leather



Assembled

Assembled

CNC = quick, easy variations



Show

Show

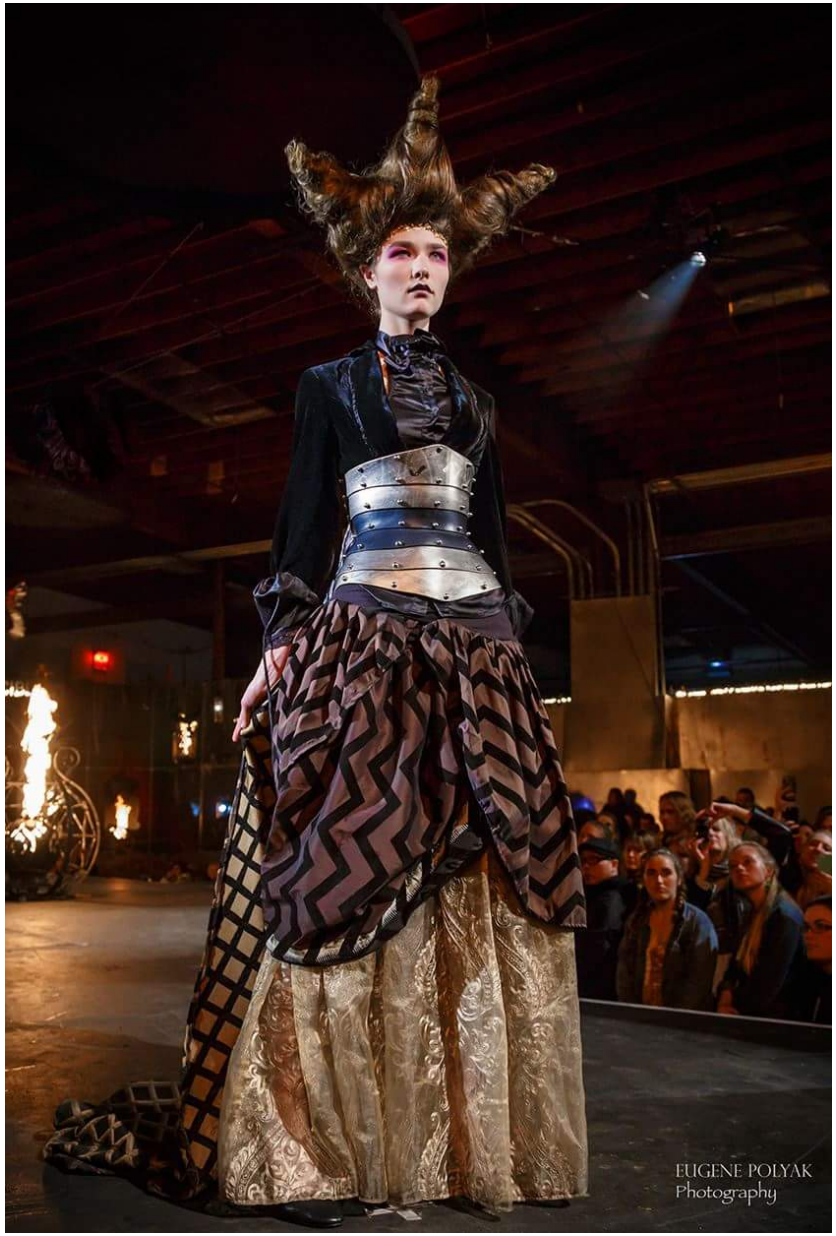
Show

Programmatic = power

Easy to add a new strip with a few lines of code change to interpolate.

Re-compute holes, etc seamlessly

Was tedious enough with CAD that never did.



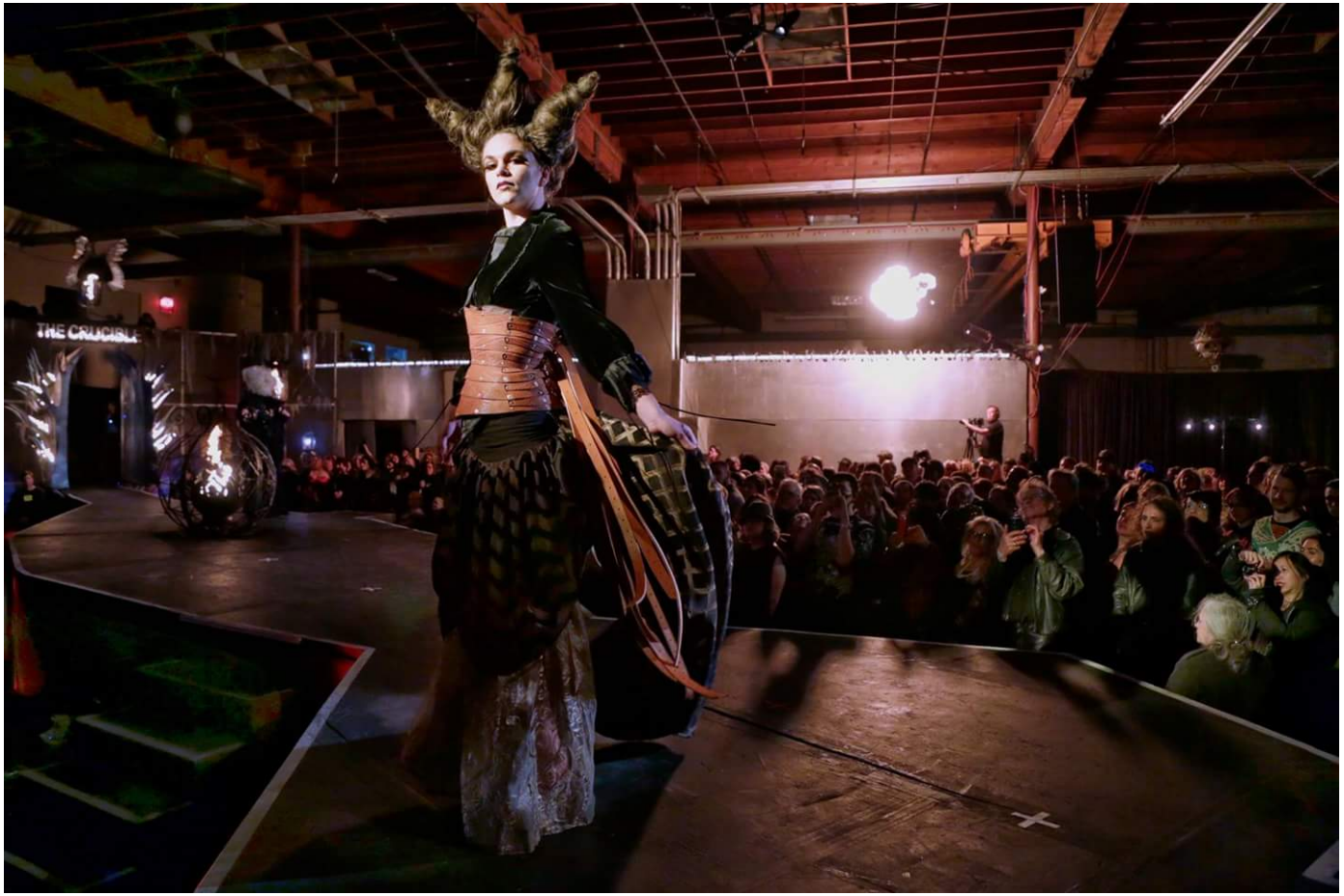
Few lines = many variants

Programmatic = power

Add a new closure method with a few lines.

Programmatic = power

Add a new closure method with a few lines.



Programmatic = power

Compute holes for ¼ inch thick strips.

Non-trivial. Hole spacing of top strip is a function of curvature in 3d.

AFAIK no CAD system can do this (and would be tricky with a GUI even if they could).

Fluent makes this simple: can match holes up on each strip with a small amount of code.

Programmatic = power

Spatial pun: take lower strip of corset, recut as a scaled down neckpiece from brass.

Because we can write code to manipulate, trivial to rescale this way.

Switch from cutting leather to cutting brass with a single line change.

Programmatic = power

Spatial pun: take lower strip of corset, recut as a scaled down neckpiece from brass.

Because we can write code to manipulate, trivial to rescale this way.

Switch from cutting leather to cutting brass with a single line change.

Computational to visual

Encode new cutting techniques: makes it easy to have complete control over cutting and still keep gcode genereation automatic.

one person figures out, adds library, everyone can use.

Computational to visual

Encode new cutting techniques: makes it easy to have complete control over cutting and still keep gcode genereation automatic.

one person figures out, adds library, everyone can use.

Co-fabrication: metal and leather

Co-fabrication

Fluent

- 3d modeling with integrated code generation
- Makes simple things simple, fancy things possible.
  - Easily add novel cutting techniques
  - Easily add novel geometric calculations.
  - A lot of control for optimizing cutting.
  - One touch fabrication.
- Last show:
  - one key press ~ all cutting programs for all outfits.
- Changing measurements?
  - few numbers, then press key again.
  - Major win.

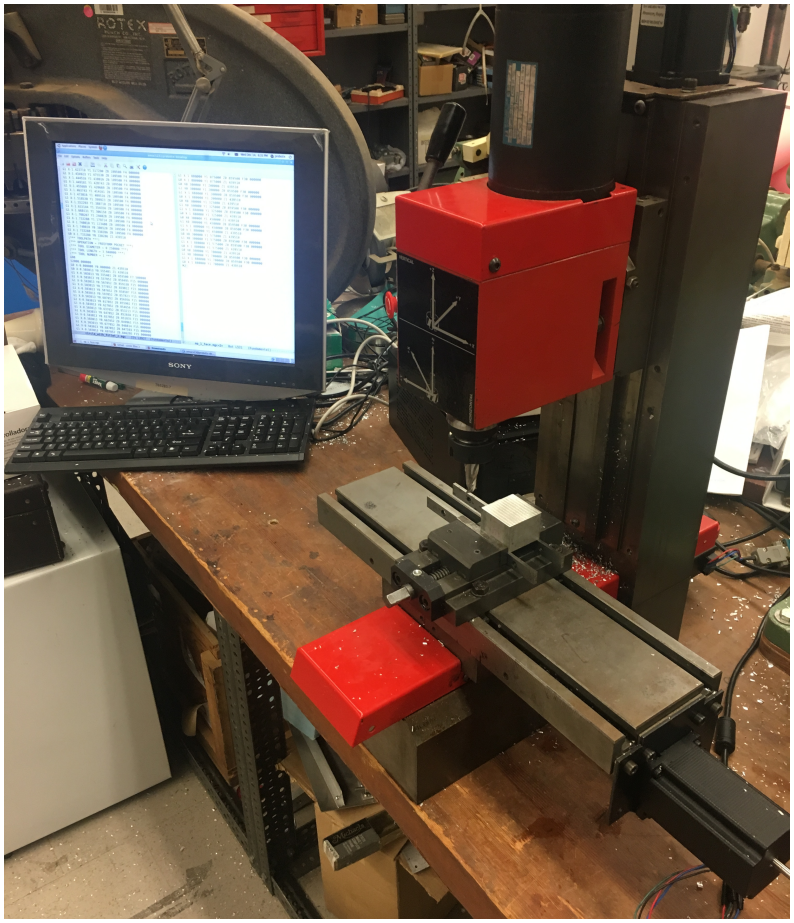
Thanks!

Thanks!

Synthesizing CNC Mill Programs

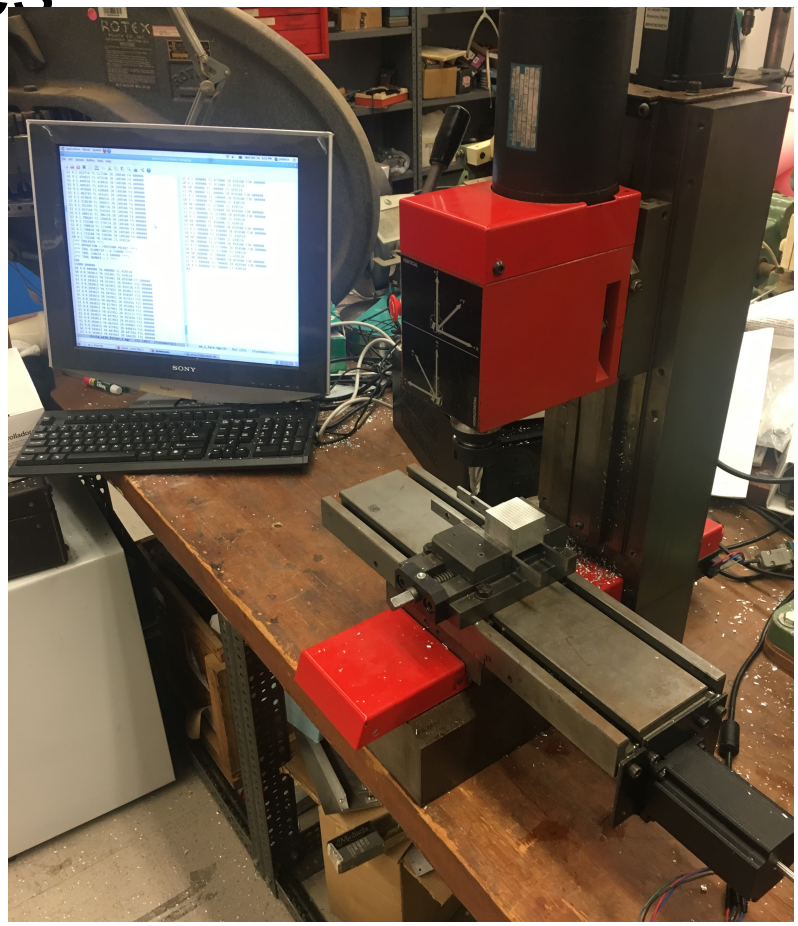
What is a CNC Mill?

- Computer controlled metal cutting machine
- Ubiquitous: Almost every modern machine shop has at least one
- Expensive: Thousands to hundreds of thousands of dollars
- Programmed in G-code, simple control language invented in the 1950s



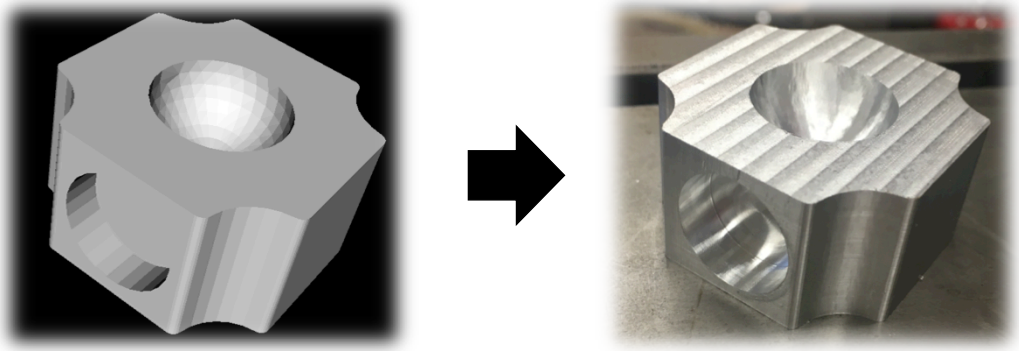
Programming Challenges

- Expensive -> Want high utilization
- Made out of metal -> Any programming bug is expensive, maybe tragic
- Requires a machinist to make tricky geometry / physics decisions

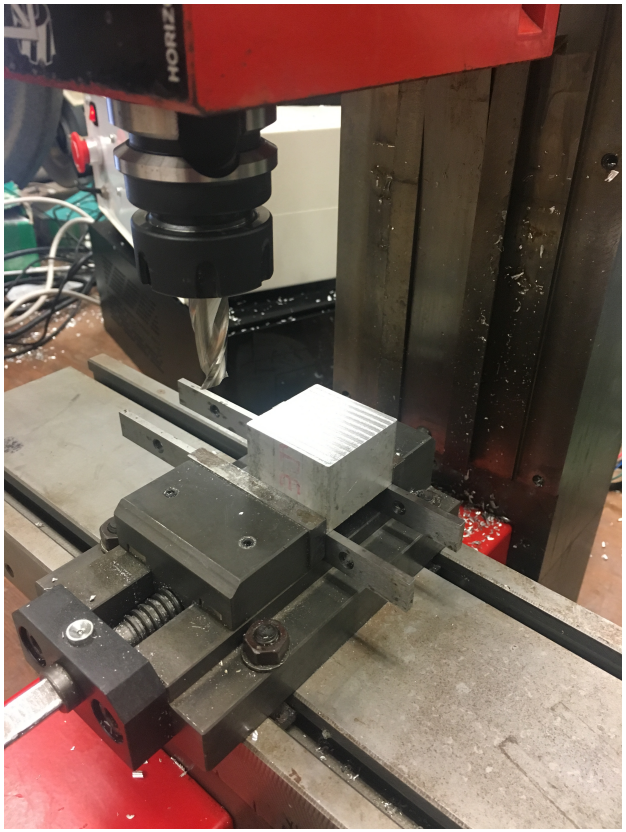


Milling Program Synthesis

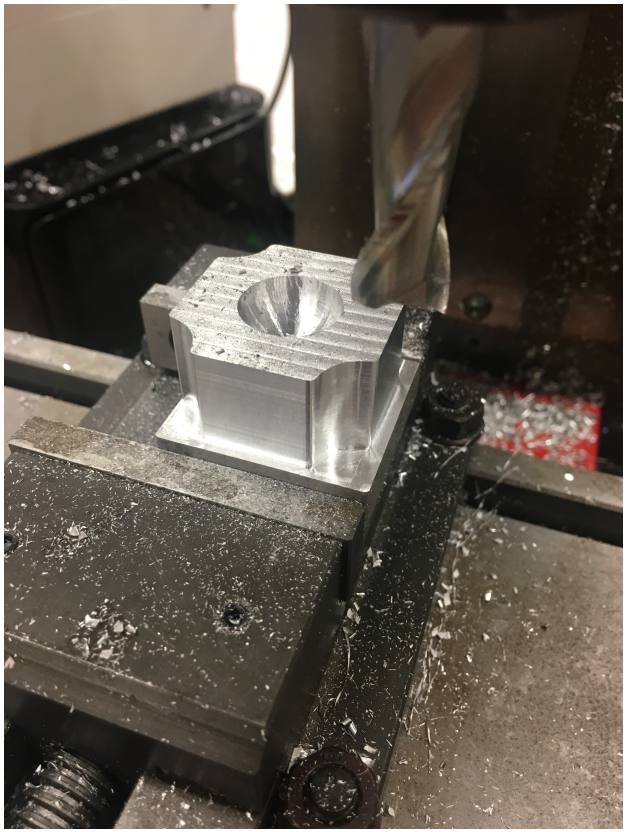
- Inputs:
  - CAD model of the part
  - List of cutting tools
  - Size of the initial stock
  - Dimensions of the vice
- Output: A crash free plan to cut the part



Setup 1

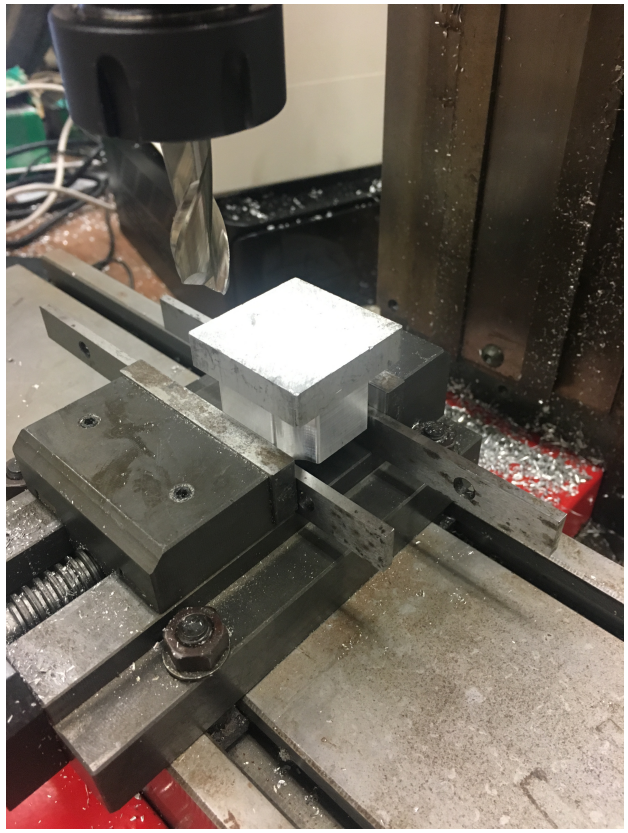


Start

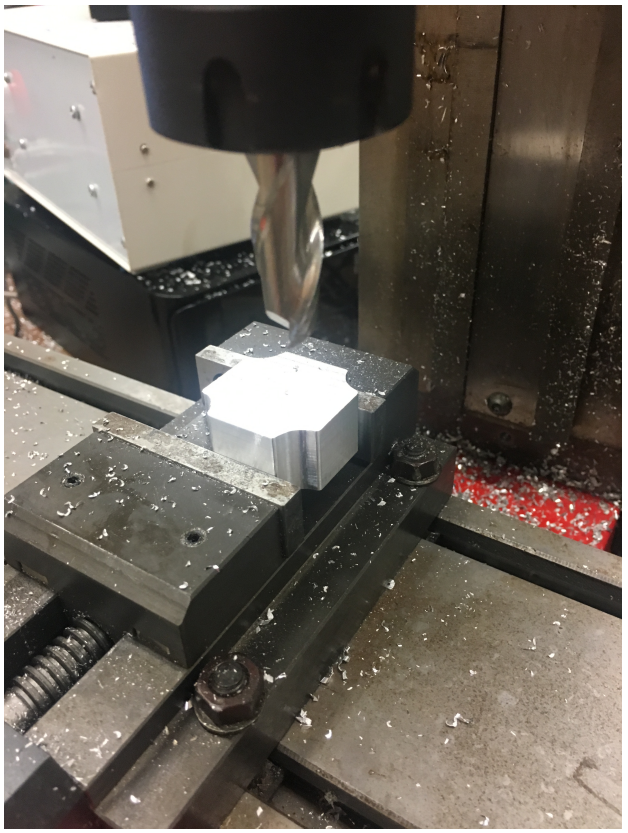


End

Setup 2

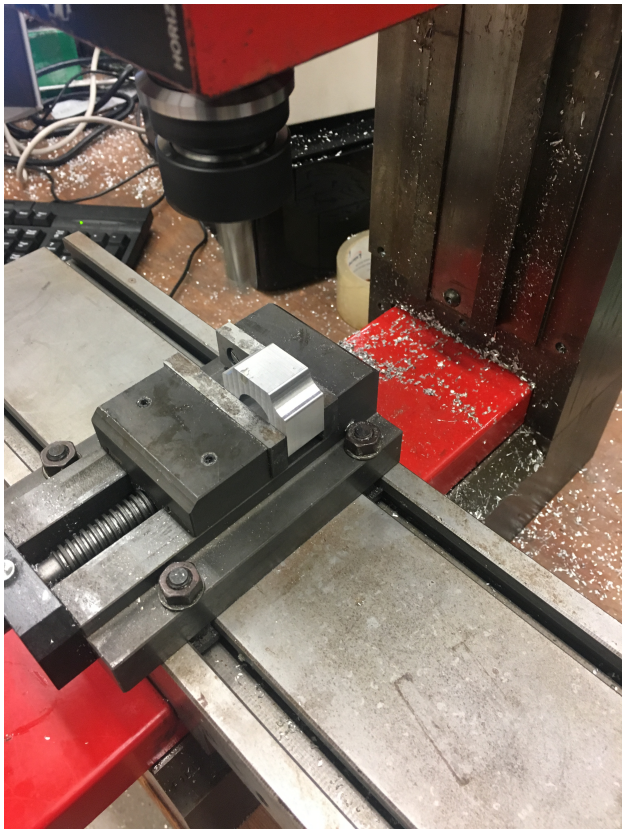


Start

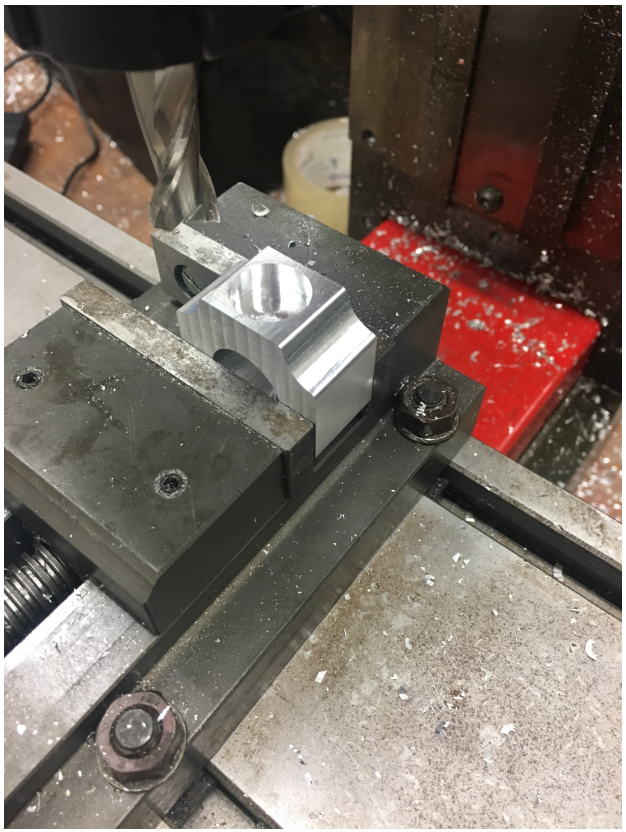


End

Setup 3

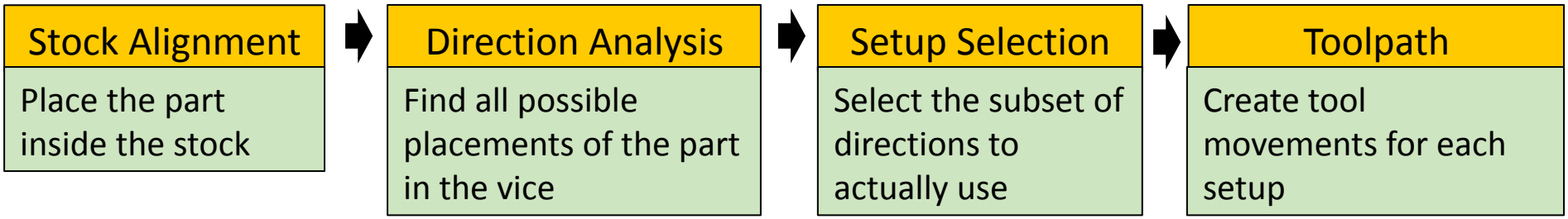


Start



End

Planning Pipeline



Conclusion

- To our knowledge this is the first fully automatic, end to end process planner for 3 axis mills that has been tested on real parts
- Plans are crash free by construction
- Plans are toolpath overlap free by construction

Future Challenges

- Extend to more advanced machine tools, 4, 5 axis mills, mill turns
- Handle more tool types
- Support automatic design of custom fixtures for more irregular shapes
- Manufacturability feedback

Dawson Engler

- Associate Professor, Stanford
  - MIT PhD (exokernel operating system)
  - Co-founded Coverity
    - Commercialized static checking work
    - 1000s of customers, bought by Synopsis
- Research: find real bugs in real code
  - Agnostic on method, main religion = results.
  - Static, Model checking, Symbolic execution
- These days:
  - Design and CNC fabrication (this talk).