TPMS:

1

- [ ] 3D DIC (Ø3 or Ø7) → Probably going to need some guidance here to make it even be usable.
- [x] Comparison between the two

FEA:

- [x] Actual stress/strain RPU response

XCT:

Get from SimpleWare post-Otsu

- [x] Reg. primitive sample scan
- [x] Reg. Otsu output vs. attenuation histogram
- [x] Def. primitive sample scan
- [x] Def. Otsu output vs. attenuation histogram

Comparisons:

- [ ] Binary comparison
- [ ] Signed comparison
- [ ] Residual comparison
- [ ] Variance/distance comparison

Other/Extra:

- [ ] Standard vs. Realistic Primitive FEA → Actually... it seems as though the standard is fairly realistic. → Might be a nice quick grab → Thinking top, front, ortho

it's not quick, but I'll let it keep nice & grab

- Look @ Ø5pØØ3 & Ø5pØØ7 for good 3D DIC outputs
- In the meantime, edit realistic FEA for smooth outputs RPU
- While those are going, find good XCT outputs to evaluate
  → May have to re-reconstruct 63ØØ lb. sample w/ new techniques

FEA:

- Change EPX to RPU ✓ (elastic, plastic lines 1,345,601 - 1,345,656)
- Output more than 12 steps (12Ø, perhaps) ✓ (line 1,345,~~677~~ 711-ish)
- Output IVOL (see other INPs) ✓ (line 1,345,715)
- Field Output get rid of variable = PRESELECT ✓ (line 1,345,711 also)
- Changed TPMS material declaration to RPU
  ↳ Actually came out quite nicely!

FEA plot:

Time to output something... Looking @ 2 outputs, namely Rxn @ top & S22/LE22 over full part. Prioritizing first bit.

- RF is node-based. Do I have a nodeset for the top of this model?
  ↳ I have it for the bottom of the model, which should suffice.
  Called: nset = NS_TILTED_SAMPLE_VØ_WITH_YMIN
  instance = PART-1-1

→ Request appropriate node set
→ Output 'RF' from appropriate subset

Note: looks like we sum these total nodal forces, not average.
& it's our third node set, so we index at [2]

Of course it couldn't be that easy. All Øs...

↳ I think I just need to talk to Dr. P on this one...

Upon return: → My YMIN node set has Ø reaction forces, but the top platen has them. Should I just take all of the top plat.?

→ Mohsin has made a Set-4 that is the underside of the ~~po~~ bottom plat. which has the most positive RF. I guess we try it & see what happens...

↳ That at least gives me values. Let's also output the displacement of Set-3 (top of top plat.) to get true F/D data
→ & we'll just grab the average.

Value for force peaks at 3Ø kN, which sounds a little high...
↳ 6744 lbs → Actually not bad!

Okay, I have my theoretical S/LE plotted amongst the lab data.

↳ Next it is time to do some reconstructing. I will redo it based on new methods. No need to worry about getting the correct SOD, SDD because we have a standard glued on top!

- Window normalization
- Spot filter aim 0.13% → 0.125%
- Ring filter has absolutely NO correlation
- Don't forget 6.5535 prompt

From recon. PPT:

SDD = ~~411~~ mm	924
SOD = ~~194~~ mm	217
CCW	
RA = ~~1062.6~~	1060.2
Tilt = ~~0.2°~~	Ø
Skew = ~~0.55~~	0.29
VC = ~~1024~~	1069
HC = ~~1024~~	1021.8

Alright, reconstruction underway. Now what?

↳ Meanwhile I'll go grab a good prim 30% RPU ODB from the compute node & look at the output w/ a similar strain.

↳ Seems all of my samples only went to ~ 6% strain, not quite enough to see instability from the "correct" samples

↳ Oughta just rerun it and edit the INP file. Which I'll still go and find on the compute node either in bubbleGum or lumpySpace

↓

Turns out ~~a~~ basically ~~the~~ AI study was on EPX. Found a good one, ~~an~~ now just change to RPU & 25 mm compression & see what happens (hopefully it is perfect)

↳ Geez... -3.5 mm. What are we supposed to learn from that?

Now I'm waiting on a slow reconstruction and a long simulation. What can I do in the meantime?

↳ I think the Otsu & cumulative vs. threshold value will be a really cool plot. First I'll have to figure out which model to run it all on. → Seem to remember running on my own computer.

Now... where were we. The FEA is getting stuck at around 18½ mm, which is okay because samples only reached ~ 16 mm. Seems like a good sign even.

I have a completed deformed recon. Looks spectacular btw. Let's histo & otsu a reg one, then we'll crop this guy.

↳ I need to recon the normal one, too. Going w/ 60 keV as the likely one from deformed. Going to have to do some acrylic scaling here...

Need to also determine an alignment anchor for my deformed part.

- Can probably go ahead & kill the FEA. Progressed 0.007 in 106 steps.

↳ Well... we used an EXC model instead of INC. Should've done a better job paying attention there...

LOOSE THREADS

Current loose threads:

- Need an FEA of the correct, inclusive model → DONE
- Crop my deformed reconstruction → DONE!
- Reconstruct the undeformed model → In progress, out of my hands now → DONE!
- Analyze reconstructed models' acrylic rod for data comparability

On def. crop. Anchor top right of bottom platen to top right of image space. -1.78° on first side, crop some easy space

-1.7813° bicubic → Resliced from left, avoid interpolation

Cropped x = 1.9869, y = 2.1404 → Cropped top-bot., left-1224,
→ Reslice from left again
→ Cropped left = 214, rt. = 1358, top = 0, bot. = 1378
→ Reslice left again

Meanwhile searching for a 05p00something to simulate. Found one. Meshing now in SimpleWare.

Currently aiming to work up a nice FEA. I have the geometry & NSETs from SimpleWare. Now all there is to do is paste it into the previous attempt, make sure NSET names line up, & I'm all good!!

- Mat'l.:
- NSET BCs

Upon return:
- Why is Abaqus giving me an XML parsing error?
- Otsu the deformed cropped sample
- Crop the regular primitive sample

Abaqus. First an XML error. Changed nothing & now "rpu has already been used". I know the other one ran. I'll just try it by carefully pasting everything.

Yeah, fresh start grabbing 05p009 that worked (but was wrong form factor) and replacing nodes, elements, & node sets with information from the correct form factor. And now that's running!

Next will be & cropping my undeformed primitive sample. Cropping and rotating.

Projection 1796 → rotate CW 25.997°, cropped x = 388 y = 267 w = 1449 h = 1617
→ Resliced from top

Slice 148 → rotated 90° right
→ rotate CCW 0.3197°

Slice 1438 → cropped 0.519 top, h = 5.05. Otherwise full width.
→ Resliced from top

Slice 841 → rotate CCW 1.112°

Slice 1053 → crop top @ y = ~~43~~ 94 bot. @ 1406 slice 1054 → rt. @ 1303, left @ ~~139~~ 137

Pres.: 2

- Find architecture pics
- FMD video
-

Actually I only really need the architectures & I'm done w/ content. Then just need to do the storytelling.

Got it all. Have 9-10 slides total, which will be more than enough to last 15 minutes.

Okay, it is Otsu & histogram time. I suppose I should use the acrylic for comparison.

Use z-scaling. ~~Dal~~ Deformed dataset → A
~~N~~ Regular model → B

1. Get ~~σ~~ mean μ_s, std dev. σ_s of standards

$$z_A = \frac{x - \mu_{s,A}}{\sigma_{s,A}}$$ where x is values in A standard

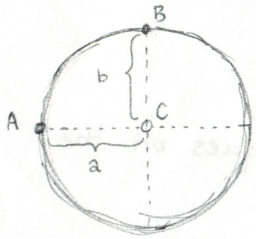
$$z_b = \frac{y - \mu_{s,B}}{\sigma_{s,B}}$$ where y is values in B standard

Then let's see how those compare to each other in z-space. Then we can Otsu on both, virgin and z-space. Perhaps histogram both as well?

Acrylic B 82-126

Shit was pissing me off. Revisit in the morning. → No longer pissed.

Yeah we'll just go with Python. Visit the circle in imagej & calc. its position



A = (55, 173)
B = (187, 41)
∴ C = (187, 173)

a = 187 - 55 = 132 ✓
b = 173 - 41 = 132 ✓

Looks excellent! Made for dataset B. Now A. Back to ImageJ briefly.

Acrylic A 54-74 A = (41, 218) B = (171, 88)
∴ C = (171, 218)
a = 171 - 41 = 130 ✓
b = 218 - 88 = 130 ✓

~~σ~~

μ_A = 3,303.79	μ_B = 4,161.30
σ_A = 124.44	σ_B = 150.15

Okay. I'm glad I know how to do all of this. But also. Irrelevant. Just make the damn full dataset histogram & Otsu. Get this shit over with.

Upon return:
- Histogram both datasets
- Otsu both datasets
- Plot.

Both datasets stored in NPY files. → Saved numpy isn't flat. Have to do that.
Histogram looks kinda bad. Try way increasing bins. Helped. xlim 0→10k 5k. Not improving bin resolution. Cut off data above 5k. Only lose 7600 points out of 1.9 billion. Closer. Black too high. Try log scale. Log scale looks weird. But better than not... ~~But better~~ But meso region comes out so high... Try discluding 0. → Clipped data 10 ≤ data ≤ 5,000. We'll see... Lost 310,000,000 from 0-10 range.
Yep. Data 10→5k. Reg. scale. xlim 0, 4250. Looks good.

Otsu: $\sigma^2 = W_b W_f (\mu_b - \mu_f)^2$

- Load dataset
- Only look @ data 0 → 4250 (keep whole set)

```
values=[]
for i in range(0, 4250):
    ~~Wb = Wb~~
    ~~Wb = fun~~
    if i % 35 == 0:
        print(i)
    Wb = len(data <= i) / len(data)
    Wf = ~~Wf~~ 1 - Wb
    μb = np.mean(data[data < i])
    μf = np.mean(data[data ≥ i])
    temp = [i, np.sqrt(Wb * Wf * (μb - μf)**a)]
    values.append(temp)
```

values = np.array(values)
- np.save("standardOtsu.npy")
  → Would work, but just too cumbersome. Found a way w/ cGPT using histogram.

Full dataset = 1662.25
Cut dataset = 1630.86
More bins → = 1635.74

Several of the high values for the data are probably machine artifacts anyways. So not a big deal if they don't get included.

Otsu changes:
- Save Otsu value
  → As a np array
- Save plot data

Now we take the histogram code & plot in the Otsu values.

- Load Otsu value & plot data
- Plot!!

xlim (0, 4500)
ylim1 (0, 5.5e7)
ylim2 (0, 1.2e25)

↳ Looks fair. Room for improvement, but good enough to keep moving on.

Now get deformed dataset worked out.

- Create histogram → See max reasonable value ✓ ✓ → Trial 10k, looking like 5k also
- Run Otsu ✓ → 1213.38
- Plot & make it look pretty. ✓
  ylim1 → (0, 3.5e7)
  ylim2 → (0, 8e24)
  xlim → (0, ~~5000~~ 4500)

~~Upon return~~: It's time to revisit my previous comparison work. I'd really like to find the old stuff because I was VERY careful to keep dimensions in order.

Yep, today is comparison day. Let's get it. I remember doing comparison work on my laptop & dealing w/ memory issues which may not have occurred in workstation. Checking Sublime. → Actually didn't immediately see it. Perhaps was in VS Code by then?

→ I have a .coding file named comparisons, but it is all about the deviatoric stuff. Which is definitely what I'm remembering doing on my laptop. Looking in the wrong place! >:(

→ The presentation which contains it was created ~~in~~ May 14. Gives me a date range at least.

→ Found a bunch of code from back then. May be getting close.

→ Seems I've moved the XCT stuff off my drive & onto dpxct.
Original path was G:/20240328-joey-05p006-90deg/Projections/
final model

→ Hm. Does not seem to be on dpxct.

→ Checking scratch directory. → YES!

This will take forever I'm sure of it. But it will be highly, highly valuable to have all data available.

Well... only 5 hours to go. Or 22, I just looked back. What to do in such an amount of time... → See the total file size of course!

Only 180 GB... yeah, might as well find a way to kill some time. 3.5 GB done.

I want to stay focused on this, so I'm going to try & look @ a little FEA in the meantime. The realistic Mohsin simulation could come in handy after all since it was platens moving downward.

Here's what I'll do for that:

- ~~Pa~~ Make a new directory
- Place in it the INP, ODB, RPU information
- ✓ Search Mohsin folder for *.cae → No luck. BUT! I could maybe import original INP & go from there?

Trying to make this simulation myself. Duplicated the steps. I want a quick turnaround as a test so I'll try just 3mm disp. down & up w/ just 10 outputs each.

↳ Seems to have worked perhaps... We'll really know when we get to the end of the loading step. At least not getting any major warnings & no errors.

→ Looking at 05p009, looks perhaps like a total strain near 27%. Going for more exact: -16.015074 mm - (-0.001764 mm) = 16.013306 mm.

Yeah, once this sim. comes out I'll try 16.013 boundary conditions with 300 outputs on each step for that nice granularity.

Already 0.5/2.0 time steps. And 22% (41.5 GB) done on the transfer.

Next I think will be to play with some deviation code. Maybe some toy problems.

Sim. seems to be struggling on the unload, but nothing too catastrophic yet. And 29% done on comparison transfer. Only 4½ hours remain.

How exciting! Even at just 3 mm compression I witnessed some relaxation or whatever. I got plastic deformation! :o

→ Now time to output the real deal! Which I will probably need to free up some space for since I'm putting 180 GB on. Goodbye 75keV & 90keV studies! You're off to the scratch directory!

Returning. Looking towards Abaqus & comparison methods. Will be really cool to get to the variance method today.

Okayy... Neither my transfer nor the simulation finished. ~~Going t~~ Transfer really close, but oughta clear up some more space for development.

↳ Also, weird on the simulation. Didn't seem to have trouble, just ran to 98% on step 1 & stopped. My suspicion was a space issue. I'll clear up a good bit of room here and go again.

My 75keV & 90 keV are each 108 GB. Check that they're on ISAAC & delete. Will be plenty of space.

Yep! Goodbye! Allocated B)

→ Got the simulation re-attempt submitted again.

Now to get back into some comparison techniques on the

next page!

3 Alright, getting into comparisons. FEA done thru step 2 (3:49 pm).
↳ Started @ 3:38 pm
↳ Step 8 (4:02 pm)
↳ Step 20 (4:31 pm)

Paper1Usable → ComparisonWork

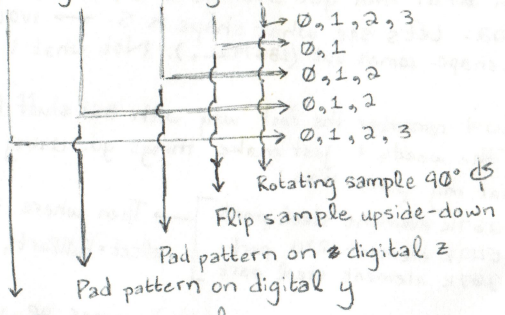
Relevant-ish ~~files~~ scripts:
- fullModelBinarizing.py
- fullModelHistogram.py
- fullModelImporterDataAnalysis.py
- fullModelMatrixSaver.py
- fullModelMaxValues.py
- fullModelPositionAnalysis.py
- otsuThresholding.py
- quickThresholder.py
- stlConversionVoxel.py
- XCT_Projections_Renaming.py

Got:
- stlConversionVoxel: Turning STL to voxels
- quickThresholder: Used to convert grayscale to colors. We can do this in the future using SimpleWare instead
- otsuThresholding: Just outputting the histogram bin center & corresponding Otsu value like we've seen recently.
- fullModelPositionAnalysis: Found the best alignment for digital & real sample padding
  ↳ Output to orientations.txt → Check out.
- fullModelMaxValues: Just output max values from each frame. Doesn't sound especially useful.
- fullModelMatrixSaver: Converted image directories to 3D *.npy files
- fullModelImporterDataAnalysis: Load digital & real *.npys, find difference, print sum & shape
- fullModelHistogram: Made histogram of ~~bin set of~~ each cross section, summed to make full model
- fullModelBinarizing: Converted all images to 0 or 1 given a threshold value

---

Okay. Let's check & see what the best orientation was:
288 orientations evaluated. Wow. And I have 6 identical outcomes. Probably using some symmetry.
Now to figure out the meaning of y-r, y-d, z-d, y, z.

~~y-real in~~
yreal in pad_y_real
ydigi in pad_y_digi
zdigi in pad_z_digi
flipy in ~~flipy~~ flip_y
flipz in flipz



One of best was 0, 2, 0, 0, 3

yreal = 0 → real, pady, pady, pady
ydigi = 2 → pady, pady, digi
zdigi = 0 → digi, padz, padz
flipy = 0 → DON'T flip digi on y
flipz = 3 → spin digi about z-axis 270°

So it sounds like I always import the XCT reconstruction (real) and only pad it. And I import the STL-generated sample (digi) and pad it then flip it.

Now I'd like to replicate this to see if I've applied the best already or not.

Upon return:
- Compare the two datasets given above parameters & check out its comparison error.

Yep. Looking at the previous day. Gonna load up my real & digital arrays in *.npy form, then do nothing special. Just run up a [0,0,0,0,0] and then do my [0,2,~~2~~,0,3]. For values confirmation.
↳ 1.895977% error
↳ 1.892125% → 0.2% improvement over

On controls, just ensuring both *.npys are coming in the same shape.
Basically yes.
real.shape = (1408, 1171, 1174)
digi.shape = (1408, 1172, 1172)
→ Good. Now comparing.

Running. Time to load both datasets: 23.5s Time control: 58s Time optimum: error

Now it'll go well. Control time: 58.12s Error: 1.895977% ✓
Optimum time: 70.15s Error: 2.117892% ✗ :(Hmm...

Upon confirmation I'd like to save these optimum arrays for my comparisons upcoming.

Did we make an error? Set it up wrong? Yep! I got the correct value for [0,2,2,0,3], but needed to have done [0,2,2,0,2]

Correct optimum error: 1.892125% → Heck yeah.

Now we'll save these as optimum_digi, optimum_real.

Honestly... I'm not sure that the binary comparison will be any good visually. Still may be interesting for a simple value.

↳ I've been doing differences, but ~~that~~ maybe sums could be interesting as well. We'll see! Do some brainstorming around.

Wait. My optimumReal... is it binary? Ugh. Complicates things.

↳ Yes, it is. Which is fine for at first, but it would be better to hold on to a full data version.

Okay... well. So I'm using this "old" dataset because I did a really good job at lining things up. Only problem is it's a shit reconstruction. Sooo full of ringing artifacts. I'm going to have to get into Octopus & do another recon. or something.

↳ I'll rebuild the 60 keV guy. But will involve some reconstructing.

Here I am returning to the methods. Still reconstructing. And since reconstruction isn't exactly a fast process, we can do some playing around with other things in the meantime. I can look into comparison methods with my worse dataset as a standing. Also I should take some time to output the force/disp. data from my more realistic RPU FEA. See how that comes out, get started thinking about the deformed model comparison.

Alright, meantime digging in to RPU → testLong.odb. Now we have two steps, so we'll have to adjust the Python script accordingly.

Printing a few things to ensure good form: • NSET names, • Steps, • That's all, • Instances

- Instances: Bottom platen, part, top platen
- Node sets: All nodes, part self contact, part ymin, set3 (top plat. top), set4 (bot. plat. bot)
- Steps: Step-1, Step-2.

I really should be outputting the part ymax node set so I can get the most accurate displacement data. So I'm starting to really accumulate a bunch of open threads here... Let's list:

- Still reconstructing 60keV model
- May need to reconstruct 75keV model
- Play around with comparison on bad recon.
- Create new FEA which includes ymax & improved boundary condition
- Play around with old FEA ODB for F/D two steps.

Main focus will be FEA, 60 recon. in background meanwhile

Many more node sets exist than are sent to the ODB. Looks like I can make separate field output requests.

First testing modified BC. → Just displace ~~1 mm~~ → 16.013/150 = 0.1068 mm/step
→ Output like 4x/step. → 0.5 mm @ 5 out/step

Meanwhile I want to see what the binary difference between optimumDigi and optimumReal looks like in image form. First: What is the data like in each? Both 1s & 0s?

↳ Yes, both binary.

↳ Looks good. Next we figure out what to do for those outputs. Let's think about what we'd like:

- S/LE/IVOL full sample
- RF bottom platen bottom
- Position top sample

→ So can I output this all specifically from within the F-Output section of Abq.?

1. Create all desired sets
2. ~~incl~~ Create corresponding foutputs PER step
3. Run & check

1. → Created BottomPlatenBottom, FullPart, & PartTop
2. → BPB1, RF; → FP1, S, MISES, E, LE, EVOL; → TP1, U

While the FEA continues, I am finishing up reconstructing my 60keV. The binary comparison looks fine, but may be too difficult to understand to make it in the paper. We march on. → Vol. recon 1825 to ~~510~~.500

~~In the~~ Now I will be croprotting the 60keV and generating a signed differences 3D plot, → Just do binary + 2 & we'll worry about thresholding in SimpleWare.

↳ Yes binary + 2. I have done abs() + 2. Oops.

Again with the signed difference method, I think it looks pretty cool. But it might be difficult for readers to interpret. But it's saved anyhow.

Next to generate something with a bit more granularity. But then I'll need to make a new array since I've been working with the binary version. And I'll have to → let's bullet it:

- ✓ Create/find raw real array
- ✓ Save it • Pad it.
- Find Otsu
- Find mean values > Otsu

Searching... → This is such a dogshit recon. btw...

↳ Found it, directory called ~~final~~Model. finalmodel

Sounds like I'll want fullModelMatrixSaver.py

→ Some changes req'd to ensure full data richness.

→ Saved shape = (1174, 1171, 1408)

→ binarized Optimum.shape = (1408, 1174, 1174)

Not really interested in figuring this out rn. Loose threading it.

~~Upon return:~~

- Check FEA output for foutputs
- Croprot 60keV
- Set up shit set for optimum

★ Artificially destroy single

★ Try to make 10-50% error in necking region → Substantiate w/ simulations

Good morning. Getting going on FEA today so I can run that long realistic simulation! On the hunt for field outputs. Not in odb. Try odb → steps. Now try → keys. More diggings does not seem to be in steps. Going odb → rootAssembly. I saw nodeSets & elementSets. Quick check to see if mine made it. Yes, I see my created node & element sets. Dope. No data seems to be in there. I think I need to visit each frame to get... I see. We go odb → step → frame → fieldOutput → getSubset → nodeset/elemset name

So we'll save those sets as names & visit some frames & get some data.

- Get node/element set indices from odb.rootAssembly.nodeSets.keys()
  → Name given indices for easy keeping
- Iterate through steps:
  - Iterate through frames:
    → Grab data from frame with data = odb.steps[step1.name].frames[i].fieldOutputs['u'].getSubset(region=setName).values

Grabbing names, I've got BottomPlatenBottom at [1], and PartTop at [4] on the node sets. Then on element sets we have FullPart also at [1].

Cool & we'll want the RF on bottomplatenbottom, the U on parttop, and the S & IVOL on fullpart.

My type(data) = 'FieldValueArray'. → It actually plays nice if I just wrap the full statement with a Numpy array call! :)

So now what? → I'd want S * IVOL/sum(IVOL) as one. Sum magnitudes RF on BPB. And get average U on PT. → But with S we do need to grab just S22. Let's see what shape is S: → IVOL was something like (183000L,). S.shape comes out (183792L,). Not what I was expecting to be honest.

Now I remember the fast way with this stuff (I think). → I did not. Time to get out of the weeds & just make things go. Using the "old" method of data retrieval. Note that my part has:

- 267k element steel part
- 549k element RPU part
- 267k element steel part

→ Then where is 183k coming from?
elset=FullPart, start line 1,333,386, end line 1,351,855 → @ 16/line & 18,469 lines, 295,000 elements.

Let's see what np.sum(IVOL) comes up with. For IVOL over FP, Weird. & get: Shape: (183792L,) and sum looks like a directory... I probably need to run .data on every point.

↳ Let me see len(IVOL on FP) briefly. → Also 183792. Okay. And a dir of a random value in data?

→ My IVOL set sum comes out to 8674 mm^3. Even just the skins should come out at 5×50×50×2 = 25,000 mm^3. Maybe I only managed to grab the surface elements on the part...

↳ I'll try to see if I got any data out on other element sets despite not explicitly requesting it. → I do not. → I'll edit my FP Foutput accordingly using PART-1-1.PT_TILTED-SAMPLE_V0.

↳ Letting that run now.

4

- Waiting on new ABQ test since I had bad starting point. While I wait on that I can still look @ previous & check BPB & PT over time.
- I would also like to do a ['u'] on ~~plot~~ top platen top so I can watch unloading occur. → Actually much better than even part top.
- Probably ship these out on the compute node later. Get them all shoved out so I can save the compute node some space and trauma.

→ And we've actually already worked RF. Cool. Let's make our two new damaged samples over here in Flattpack. See what kind of trouble we can get into.

↳ Going to land these in G:\Paper1Usable\Mohsin\STLs so we can produce the node & element data with SimpleWare.

↳ No skins allowed in custom uploads FP. No prob., add later. → Just have to make the "blank" 50 mm³ instead of 60 mm tall.

On my skinnyBar sample, 25% neck = 1.564 mm, 30% neck = 2.521 mm. Pretty major difference. Going to try 29% & see what happens. 29% neck = 2.332 mm

↳ We'll give 28.5% Vf → 2.268 mm → 10.04% red. ↳ A 7.5% reduction.

Upon return: • Check up on fOutput2 ✓. Begin long ~~ex~~ sims on reg., miss., skinny
• Croprot • ~~R~~Miss., skinny comparison

Alright, think I'm ready to make my INP files. Get some long simulations running! Going to start with the standard, make it all nice w/ elsets, nsets, boundary conditions. Then write to INP, edit with other model nodes and elements, & open back up in Abaqus, delete node & element sets, then write to a new INP, ensuring 150 output steps for each foutput. → Alright, go.

↳ Note, I may need to code up a ~~nod~~ element set for new models. We'll see. → Sorta, but easy & no code! :)

→ Don't get rid of old node & element sets! → But fOutputs are fine. → to delete.

Original part lines nodes 320,452 → ~~1,012,312~~ & 1162,600
elems → 460,745 → 1,010,534 (+549,789)
462,507 → 1,012,296

Imported new part, some things a little wonky. But we'll get there.

↳ Try translating part -x 25mm → Also -5 in ~~z~~ direction.

Alright I think I have my inputs put together nicely. Now we go to ISAAC & see if we can't get these bitches to run. Had to dos2unix, but no immediate errors are apparent. All made it to > 1 min. in the run category at least.

↳ All about to cross the 20 min. mark. All STAs indicate good progress as well.

Feeling like getting into croprot for a bit. Let's get it. → (324, 264, 1332, 1596)

Quick FEA check-in: ↳ Running @ 1:18:00 total time.
skinny & missing having a little trouble. Checking out their ODBs in case they're going crazy due to my translations.
(61, 57, 1485, 1244)

Upon return: • Finish croprot • Bad ~~exam~~ samples comparisons • Look into FEAs
• Write code for data extraction • Images!!!

Alright, we probably cutting some corners this morning. Just focus on spitting out these images for TIA.

Gonna make comparisons using first stlConversionVoxel. Just output all 3 samples @ same output resolution.

Okay FEA is a bust. No problem, we can still do the bad ~~im~~ samples comparison. And the good sample FEA. Two goals. Okay my standard FEA also didn't work out. We'll just run comparisons I guess.

For a good visual, I want to SEE the missing parts. So I would do...

If I do (good-bad) I get just the imperfections. Then multiply that by 2, then add bad. Let's see.

0 477 954 1431 1908 2385 2862 4.69 x 7.04

My last session was super rushed. I tested BCs & outputs on a short term, but failed to do the same on a replaced model. Simple mistake that I need to remediate. That's my 1st order of operations today. I'd like to not have to translate my models. Start there. Figure out the "center" of tilted sample, then make that happen for both bad samples.

tiltedsample0-2	min (-25, -5, 0)	max (25, 55, 50). Those y-values are weird.
standardSample	min (-25, -5, 0)	max (25, 55, 50). Probably came from tilted.
missingSample	min (-25, 0, -25)	max (25, 60, 25). Checks out.
skinnyBar	min (0 , 0 , 0)	max (50, 60, 50). Hmm...

Well. Expected missing & skinny to be the same. OH! But I had to add platens to missing & not skinny. All checks out then.

So now we need to fix them via code. That means loading, modifying, and print it. Ideal: (-25, -5, 0)

✓ Missing: (-25, 0, -25) → Add (0, -5, 25)
✓ Skinny: (0 , 0 , 0) → Add (-25, -5, 0)

Perfect. We're centered. Now we back out & write these as the nodes to their SimpleWare outputs. Done. And we'll open them in Abaqus to check they worked. → They did not work because the node indices must be written as integers. We'll fix that. → NotePad++ and CTRL+H.

Made 3 NSETs → FullPart, PartTop, PartBottom. 1 ELSET → Auto named.

Wrote INPs.

Now what? → Need to integrate into tiltedSample. Find an ODB that worked fine and work from there.

We'll start from the beginning where Mohsin left us with tiltedsample0-2. Pulled in that INP file.

Tiltedsample0-2:

3 parts → bottom-platen, part-1, top-platen

2 mats → STEEL, EPX ← steel

3 sections → section1set1, 2.2, 3.1

Assembly → 3 instances → part-1-1, top-platen-1, bottom-platen-1

8 sets → part self contact, part ymin, set-3 (top plat. top), set-4 (bot. plat. bot.), full bot. plat., full sample, full sample again, full top platen

4 surfaces → bottom-lattice, platen-bottom, platen-top, top-lattice
↳ All contacting surfaces

2 steps → Just initial & loading

2 interactions → intprop-1-1, intprop-1-2
M: plat_top, S: top-lattice
M: plat-bot, S: bot-lattice

1 interaction property → acting on intprop-1~~-~~ w/ friction penalty 0.3

2 contact controls → cntctil-1, ""-2. Only seems to need 1 tbh.

Now what? Let's start by duplicating the INP & replacing the part with our own.
We'll start with "missing". Node start (320,459 - 469,600) El (462,602 - 1,012,391)
Missing: nodes (10 - 142,063) El (142,065 - got it)
Updated element set count for full part & set-2. (6th & 7th sets from earlier)
→ Save & open in Abq. → Sample seems to be centered :)
→ Let's start by matching sets. instances seem happy.
Sets → self contact is whatever. doesn't use full sample. → part ymin ✓. → set-3 ✓. → set-4 ✓.
→ full_bot_plat ✓. → full part ✓. → full part again ✓. → top_plat_full ✓.
Surfaces: Likely where we went wrong previously. → yes, lattice top & bottom both need to be redone. → Both looking good.
Following through, the steps, interactions, interaction properties, and contact controls seem to be in good shape. I'll decrease that compression to 1mm & we'll see if everything agrees. → Submitted. We'll see how this goes!
Hey! After 0.083 time (1 output) we seem to be working! Woohoo! I'll follow right behind here with skinnyBar → nodes (10 - 140,301) El (140,303 - 685,698)
Updated element set count! ✓
Also seems to be correctly centered. → Sets look good. → Surfaces fixed.
Went ahead & submitted a short skinny.
Meanwhile missing short is successful. → Going to make a missing up_down short.
↳ Added up-step, fixed its boundary condition, & submitted.
→ And we're looking good on the down-only short skinny (first 4 time steps).
→ I'm feeling confident enough to ~~su~~ make & submit the long versions of these to ISAAC. Let's go! Just changing fOutput to: • Include EVOL • Be 150 outputs and make the BC go down to -16.013 & back up to 0.

OH!! → And change the material EPX plastic values to RPU!!! And elastic.

	miss	skin	tilt
EPX	✓	✓	✓
EVOL	✓	✓	✓
150	✓	✓	✓
Step	✓	✓	✓
BC1	✓	✓	✓
BC 2	✓	✓	✓

→ Cool. Writing INPs

Job file	miss	skin	tilt
jobname	✓	✓	✓
16 cpus	✓	✓	✓
job =	✓	✓	✓
input =	✓	✓	✓

Got those jobs submitted. The missing short UpDown looks like it'll be successful. All ISAAC jobs have steps. Amazing.

Meanwhile: Time to look at croprot again:

Upon return: • Finish croprot • Compare samples • Extract Abaqus values

Good morning. Only tilted completed. Perhaps others failed catastrophically, which I wouldn't mind. Pulling those files down now. Doing some croprot this morning. (30, 72, 1204, 1206) → (30, 74, 1196, 1200)
~~Worth~~ Likely worth creating a small script to output Abaqus data. Getting close on time, so I should consider most relevant things.

- Bad sample comparison
- Bad sample Abaqus F/D

↳ Honestly that's about it I'd say.

Top: y=35 bot. y=1409
Right: x=1189 Left: x=13
Top: y=14 Bot.: 1185 → h=1171

It's a nice crop. It'll have to sit as a loose thread for a bit, but that'll be fine.

Loose thread: Otsu & best comparison on fully rotated & cropped 60 keV

Okay. F/D time. We extract RF on 'SET-4' and U on 'SET-3'. That should literally be it. All needed.
All samples have the same steps & sets. So far so good.
We'll want the average U on 'SET-3' and sum RF on 'SET-4'.
Okay. I think I made that happen. Although I'm just averaging all of U and summing all of RF. I need to refine both of these to just the [1] spot on U and just the magnitude of RF.
↳ And the whole file only took around 3 minutes thanks to my good coding!
Made 2 simple changes to my script. Will see success shortly! B)
↳ Should've made those 2 simple changes twice. Otherwise this is working.

While I run these, I can work on my sample comparisons.
I'm loaded on all. All have same shape, nice. Right. These samples are already binarized because they came from STLs. Next.
They hold 255 max, but are dtype('float64'). Going to overwrite those now to ~~bt~~. ~~Hmm... Go~~ No. Why 255? They can be ±1. That first. I can store them as bool.

Quick values: 1,941,990,884 voxels
3,103,422 ~~skinnyDiff~~ diffSkinny → 0.1598% error
4,132,535 diffMissing → 0.2128% error

whereas max stress was:
12.34 MPa standard
10.11 MPa missing → 18.1% decrease
10.99 MPa skinny → 10.9% decrease

5. It's time to wrap this sucker up. So what's left? The few items I can think of include:

- ✓ binary skinny difference
- ✓ binary missing difference
- ✓ cleaned plot of skinny, missing, standard
- ✓ image matrix of skinny/missing/standard versus strain
- development of residual statistics/visualization → considering ignoring this one tbh
- development of variance statistics/visualization

These will be easy. I will first begin with the binary differences. Want to just save that binary as a *.npy and visualize later. Made skinnyMissingBinary.py. Done. For sending to images, we can just try changing the bool to int8 & saving as a photo. Hm. My PNGs all saved as 16-bit. No wonder I couldn't see my 8-bit values. Had to convert my images to type "L". Getting what I was expecting now. Uh oh! Skinny & missing came out the same. Think there may be an error in my *.npy file. Skinny looks as it should. Okay and skinnySample.npy and missingSample.npy have different sums by 3.6 million. They should be fine. Found it. Got lazy on the copy/paste for saving the difference arrays. Running each back now & we should be in good shape. Yep ✓. For visualization I'm hoping to load both standard & diff image stacks & overlay them, but I may have to combine them all. Okay, I can load in additional images, but they must be same dimensions. My standard has a transposition error. Using ImageJ to reslice from top.

→ Yeah, that ended up working out nicely. I'll bring in skinnyDiff and we can pose for some photos.

All photos taken :) → The composite photo actually came out quite nicely.

Think I'll plot my stuff in Python. Not have to deal with Excel's wackiness. I wanted to programmatically remove the plateau tails in the data, but I think it's a ~~better~~ more efficient use of my time to hard-code it. Plot looking nice!

Do we really do the deformation matrix? It could look cool, it could look shit. Only one way to find out!!

A few guidelines for these photos:

- They should all have the same scale → do a zoom-to-fit on start & go from there
- Remove words >:(→ Finally found it. That's what I'll do upon my return.

Alright. Got some big ideas this morning. Keep multi-plot, but also split it up to be annotated by the image matrix. Eh? Nah. Let's just do the full plot & a 6-image matrix aiming at: • Ø-strain all • 1.6% strain all • 8% strain all • 17% strain all • peak strain standard • 20% strain unloading

How do we align these? 16.013 mm/150 steps → 16.013/60 = 26.689% strain

26.689%/150 steps = Ø.1779% strain/step.

1.6% strain = step 9 + 1 = 10 8% strain = step 45 + 1 = 46 • 17% strain = step 96 + 1 = 97

peak = 151, 20% unloaded = 6.689% from start = step 38 + 1 = 39

So we grab: All: ~~Ø~~, ~~9~~, ~~45~~, ~~96~~, ~~150~~; standard ~~151~~, ~~38~~ → from unloading stage

→ And one final thing. Let's figure out how to freeze the contour plot at whomever's maximum von Mises stress. → First, who is it? Well, likely standard due to max overall stress. → I'll keep von Mises for each in their respective rows. Also seems platens must be removed for every revisit of a model, so we'll just collect all photos of a single sample all at once. Crop settings: 5.62", 5.44", 3.86", 1.03"

Boom! A nice, clean, annotated plot of the stress/strain data accompanied by the models' deformation matrix. Looks cool!

Next? → I suppose I oughta put together the last bit of sample comparison data. Before I dig into all of that, I believe I owe one more "best" fit for the XCT sample I looked at last. I know I've done some recent(ish) coloring & comparison, but I don't think that model was optimized.

→ I'm honestly not even sure where its croprot images are. Need to find those. Good thing I detailed everything I did! Also check SimpleWare for a potential lead w/ recent projects.

So I'm coming back after a bit of a break here. I want to look for my most recent 60 keV croprot. OH! They're all in Projections. That was easy. Okay I have my binarized model. Now to save it as a Numpy array. And I'll save the STL → images as its npy as well. AH! Already appears to be done as standardSample.npy. standard.shape = (1409, 1174, 1174). binaryfinalmodel = (1408, 1171, 1174)

So we can pad on top/bottom just ~~one way~~ two ways, & 4 ways front & back. Then flipped or not(2) & 4 spin orientations for a total 2*4*2*4 = 64 orientations. We'll find the optimum using last optimum code & report back soon.

But first we save binaryfinalmodel to a npy. Alright, ran a slightly updated "fullModelMatrixSaver.py" and received binary_final_real.npy → No, killed it. Would've saved as 16-bit. Actually would've been 64-bit, ugh. Yeah boolean was way faster.

Now we can rig up our old optimal position code and figure out our best orientation!

Made a new script since some change will be more significant.

Here goes nothing on that...

Hmm... something is up because a lot of my numbers are the same and some of them are zero... Weird.

→ I fixed a boolean error, got a trial error of 35%, then ran the script & got the same weird values including Øs.

It is late and I am struggling to find the brainpower for debugging like this.

Something real wacky is going on here. Just got that 35% error again...

.3*5% + 1*1% = 0.15% + 1% = 1.15% = ?? 30%*50³ + 100%*50²*10 = 37500 + 25000 = 62500

62500/50³*60 = 0.416̄

np.sum(real)/(real.shape[0]*real.shape[1]*real.shape[2]) = Ø.4138 → works.

= Ø.4172 → makes sense!

Then where is this darn 35% coming from?? digi: → We'll find her tomorrow.

I did some noodling & think the best thing to do here is to turn my ~~binar~~ boolean arrays back into images & evaluate them in ImageJ. 35% error is ridiculous for parts that take up 41% space... Just saved all as png's. Now to see the errors of my ways.

WHAT THA FUCK!!! → All of my "real" images are out of order. I must've called a listdir() and not a sorted(listdir()) when building out my matrix. → Yep. Resaving that numpy array & I'll output it again just for validation sake.

Glad I've figured this out so I can start back into the real stuff.

Okay great. I've just produced a 1.965% error. Now for the optimal.

But I'm getting the same ~~errors~~ values as yesterday. Maybe my flips are changing the dtype?

NO! My rotations are ON digi... oops. That's what using old code gets you.

And now somehow I've given real.shape = (2816, 1174, 1174) lol.

Found it. NOW we spitting out realistic values.

All values hovering right around the 2% mark. Some of that likely because when I looked at my platens in the "real" visualization, there isn't a single "full" square. Maybe should've lowered my Otsu threshold value. But I'd rather just push this out the door & we go with what we've got.

Alright, my best outcomes were [1,1,1,Ø] and [1,1,1,2] with a 1.915% error.

That's (pad-y, real, pad-y, pad-y), (pad-z, real), real=np.rot90(real, k=1, axes=(1,2))

and real=np.rot90(real, k=Ø or k=4, axes=(Ø,1))

→ So yeah, those are the same thing.

I'll save these as optimum npys. Also curious what the percent becomes as I shave slices off of these guys.

Dang it that gave me a 2.252% error! Now what... Also saving is taking forever...

→ I don't even have anything in my orientations table that returns this value...

wtf is going on here o.o

Okay. Reproduce [1,1,1,0] = 1.915% error. Orientation = [pad_y_real, pad_z_real, flip_z, flip_y]. Need to run a little test on my rot90s briefly. I think axes(0,1) is what flips us on our head & therefore should be the flip_z value. Should only take [0,1] and have k=flipz*2. → Yes. So far so good. And then axes(1,2) should be [0,1,2,3] and no multiplication.

Running my optimum position again for sanity check. Got my 1.915% on [1,1,0,1]. Now let's reproduce that for saving. Changes incorrectly applied. Got them correct, got my 1.915%! Woo!

Now I'd like to see how that number improves if I shave off some edges.

↳ Actually that may make it into the paper now I find it so interesting!

↳ Yeah I think it makes an interesting point about ~~the~~ some of the difficulty with capturing the edge of a part with XCT. But that we're largely focused on poor internal geometry & that external errors on the order of 0-0.5mm should be discoverable via alternative metrological methods.

Well... I think it's k-d tree time... Their example: ideal = $\begin{bmatrix}1&0&1\\0&1&0\\1&1&0\end{bmatrix}$ actual = $\begin{bmatrix}1&1&0\\0&1&1\\1&0&0\end{bmatrix}$

I bet building out such a huge tree will take forever. Only 810,000,000 values to build it across...

↳ Took about 12 minutes. Actually not too bad.

I hate to say it but... It may be about time to start writing while we let this kdTree stuff work its magic.

↳ I wish I put something in the code to track post-tree progress. But oh well.

HEY! → Here's a way to plot my comparisons! → I'm smelling an image matrix again. We dig into the false-positive/false-negative stuff Chat-GPT was pushing earlier. Yeah. And I can use the accuracy formula it gave me earlier as a nice metric for the binary comparison. How exciting this all is!

↳ So accuracy will be the best to show, but may also be worth including precision:

```
accuracy = (TP.sum() + TN.sum()) / total_voxels
precision = TP.sum() / (TP.sum() + FP.sum())
```

I got my colored photos of distance. One issue is that it seems to be too good! Colors are hardly noticeable honestly. Got a nice distribution of distances to better understand what I'm dealing with here. I'm happy with the information, but now I must figure out how to display it all. → SimpleWare will only accept data as a grayscale, so that's a potential avenue albeit inconvenient.

↳ I also need to increase the sensitivity of my color mapping. → By, like, a ton.

I'm really happy, it has been a great morning session. Next we'll finalize our visualizations and tease out some residuals stuff. Otherwise there's nothing left to do but get to writing!!!

I'm back. Ready to look into plotting things. Let's target VTK to start. Trying my best at visually toolkitting here. Had to better filter my distances I think. That's a start, but not the whole story. Now plotting values 0 to 1, but that means ~~air and a~~ agreed-upon solid & also air are being fully transparent & I only want that for my air. We should be plotting agreed-upon solid. Going fast like a herd of turtles. But I think I've cleverly implemented a method to fix opacity. Now fixing it to match Numpy and VTK matrix dimensions. → Now VTK is starting to overstay its welcome. I'm giving it a last chance before moving on to something else.

Seems like PyVista could be worth trying. C-GPT gives me deprecated commands >:(→ But I think some Googling may have fixed that issue because now it's taking forever to run.

Losing hope on PyVista as well, but I've found some people talking about it with matplotlib. We'll see.

Using ax.voxels may show some promise. → Actually not that either. But I *might* be able to make this poly3DCollection method work with another full course of 🅱 Pomos tonight.

We *will* accomplish voxel plotting today, mark my words!!!

↳ I just know Matplotlib will shit itself if it has to turn out 800 million voxel cubes...

↳ Starting with a random 100^3 cube inside model. shape (1404, 1174, 1174)

choose (800:900, 200:300, 400:500)

Well... I managed to plot some cubes, but they're shaped unlike my model. → Because I asked for values > 11.60 instead of those < 11.60. Now we're plotting 1 million cuboids rather than 500 and the plot figure isn't responding...

↳ If it will have this much trouble on 0.05% of overall model it likely isn't worth exploring.

I'm trying to load it up in Fiji, who does not like these transparent voxels. I may be able to get what I want by making these transparent voxels black. Then I can implement a threshold on the Fiji viewer & stop seeing the black.

Updated code seems to produce what I'm interested in. Now we'll see if Fiji agrees with my hypothesis. → Seems my ringing artifacts are coming through in the colormap. I'll fix that up in the future & after smoothing out actual visualization.

One other potential visualization → yeah, let's try it later. We're going to code up SimpleWare to make a cividis for us. It'll let me make the background transparent. Wish I had thought of this one earlier. Better late than never.

1536	4608
793	2379

→ LFG on SimpleWare!!! → I do have some real hope on this one finally. → I THINK IT'S WORKING!!! This means I can likely finish visualizations up tonight and move on to writing through the weekend. This is actually super exciting!!!

↳ I do have a small concern as I watch this set of masks go by that cividis will have not been dynamic enough a range of cmap. So much dark. But at least I'll have all the methods developed and can change it later if necessary.

It's also really kind of SimpleWare to allow me to make, at this point, 43 individual masks.

↳ 49 total!

→ Next will be to make some false positive/true negative/whatever models & plot those out. It'll be quick, I'm on it.

Okay I've got the ability to output & save boolean TP, FP, TN, FN. I think what would be most valuable would be to plot the model itself, with the voxels on it that are wrong. i.e., the FPs. Then I would show the values that were "missed" from the digi model. So yeah. ~~So~~ That would be FNs. So we'll save FP & FN.

The FP & FNs look gorgeous. May need some color work, but morphology came out clean.

Cropped ~~6.6", 6.5", 3.42", 0.5"~~ ~~5.78", 6.67", 3.76", 0.41"~~ 6.28", 6.67", 3.56", 0.41"

Yeah, I hate to say it, but I think it's writing time. I'm just waiting for this distance visualization script to wrap up. I'll ~~be~~ pack it away & really won't have much left to do except write. I'm actually quite looking forward to it.

→ All these distances are too good.

There's not much to hand-write when you're in writing mode. I'll just make notes of my word counts.

Great day!!

1/5/25 → 4089 → 4097 → 4251 → 4375 → 4422 (+333)

Probably → 4424 → 4424 → 4424 → 4696 (+272) → (+605)

→ ? → 4912 → ? → 5164 (+468) → (+1073)

↳ need a table of sorts to discuss the dog bone data. And ~~it~~ make RPU04 ~~dott~~ dashed line.

From this paper in biomimetics, [19] found primitive buckling ~~&~~ in primitive sheet structures

[19] ✓, Zhang "Energy absorption char." → Also (and especially) (maybe only) Maskery's "Insights" show the buckling effect quite well.

Maybe also [27] from Insights? → We'll just go with Maskery.

Okay, we've talked about all the physical, mechanical testing that has been performed. We have remaining XCT, FEA, FEA comparison, XCT comparison. It's obviously between XCT & FEA... XCT ~~is~~ relates as we've introduced full models & isn't derivative. FEA *is* derivative, but we just finished talking about ~~it~~ mechanical stuff! → I think we go: FEA → XCT → XCT comparison (TP/FN/etc.), → FEA comparison (skinny, missing, etc.)

* Probably worth rerunning the basic FEA using better material parameters.*

Today (1/6/25) words: 5164 → 5165 → 5294 → 5445 → 5487 (+323)

Crop 6.48 5.85 3.96 ~~3.24~~ 0.41

Moving on to the next sheet. It's been good here.