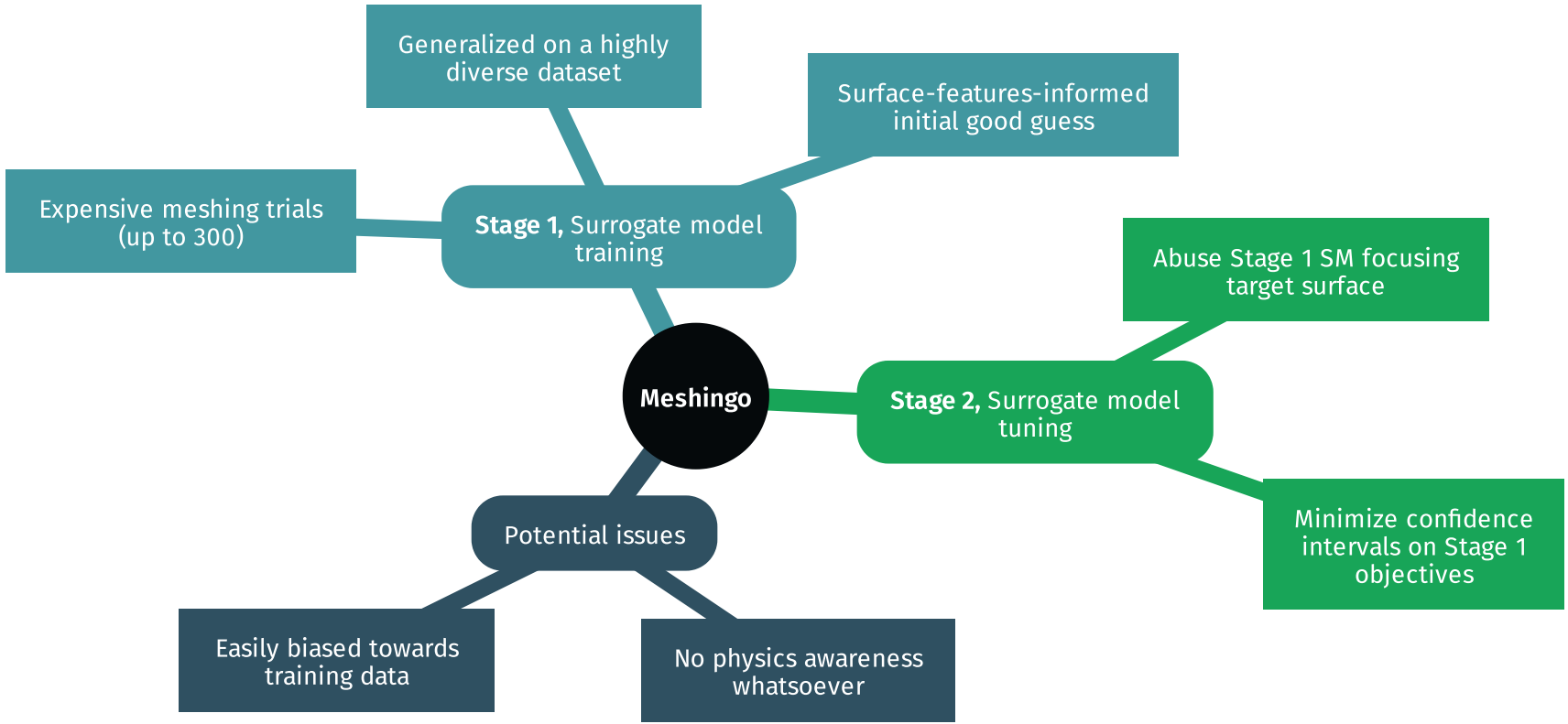
Meshingo: A cell size recommender system

Fully automatizing cfMesh mesh generation...

Mohammed Elwardi Fadeli, Apr. 2025

Table of Content

Motivation and Objectives



Disclaimers

Focus of the presentation

Because of the sheer amount of unpredictability associated with this kind of ML model, what is presented here is a general idea of the strengths and weaknesses during the prediction of a "good-enough" cell size starting from an STL file - that would be meshable on a mid-range laptop machine - using Bayesian Optimization.

Swappable-components

All components are easily swappable, including meshing tools, BO algorithms, objective functions for the optimization and so on...

What to expect?

Models trained using this approach are not supposed to be general-purpose:

- There is a "range" of predictable cell size a model is valid for (dictated by objective functions, parameter ranges, training data). Eg. A model would only function within (1e-6, 1e-2) range of cell sizes
- This approach is in very early stages of development, so it is best used to infer insights instead of actual

More Disclaimers

Sample surrogate models

The sample surrogate shown here was trained for ~130 trials (~100 of which were successful) on unprocessed STL files (~60 models downloaded from all over the internet). The testing STLs were of course never seen by the surrogate.

Unprocessed STLs?

In production, the training dataset must be made invariant to rotation, scale, and number of its triangles. `cartesianMesh` doesn't care about number of triangles but objective functions must.

Debating decision records

I keep a public log of ADRs which you can debate at any point, by PR'ing against the respective markdown file. Feel free to create new markdown files there if you feel new decisions need to be discussed.

Instant Cell Size predictions!

Try it out now! It's really just one command (at least for the 1st stage)!

```
# Get a sample surrogate model; shipped with an Apptainer container
apptainer pull meshingo.sif oras://ghcr.io/foamscience/meshingo:0.0.1
# Get oriented, see where things are, and what is available
apptainer run meshingo.sif info
# Get meshingo out of the container (can also just clone)
apptainer run meshingo "cp /opt/meshingo meshingo" && cd meshingo

# Get SLT model in place
cp '<your-stl-file>' testing_dataset
# Use the surrogate to get a few (at least 2) case configurations
apptainer run meshingo.sif 'meshingo validate
          --model /opt/surrogates/Meshingo
          --training-set /opt/surrogates/geometric_features.csv
          testing_dataset/<your-stl-file>'

# Inspect the respective meshDict files to minimal/maximal cell sizes
foamDictionary -expand <case_path>/system/meshDict

apptainer run meshingo.sif 'meshingo --help' # for more stuff to do
```

Instant Cell Size predictions!

You get "objective-oriented" suggestions


```
{
    "125": {
        "job_id": 2885729,
        "case_path": "/tmp/meshingo/trials/Meshingo_trial_bd1bd3261cad5dae7e66f3e3f65aa203",
        "case_name": "Meshingo_trial_bd1bd3261cad5dae7e66f3e3f65aa203",
        "best_for_objectives": [
            "SurfaceDifference"
        ],
        "predictions": {
            "CellSizeDecayed": 0.228648880741723,
            "SurfaceDifference": -0.0279852980695916,
            "MeshIssues": 1.3640759324078426,
            "CellCount": 8.473390924663022
        }
    },
    ...
}
```

Bayesian Optimization setup

Tech Stack used:

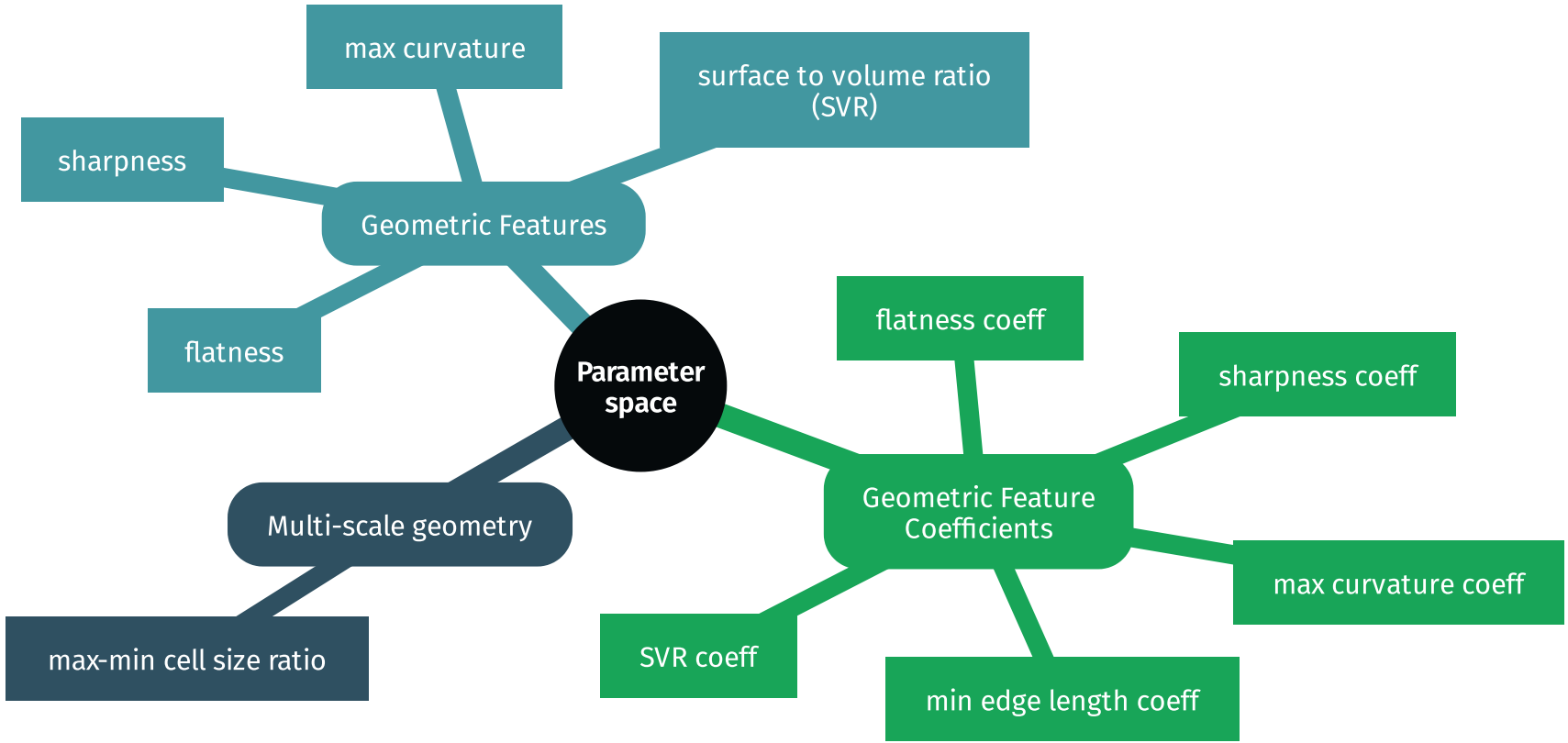
1. cfMesh's `cartesianMesh` as a meshing tool.
2. foamBO as the optimization driver for both stages.
   - Unattended choice of the BO algorithms hyper-parameters
   - Based on Parameter range/type + Objectives
3. meshingo Python package to handle
   - Geometric feature evaluation for surface STLs
   - Objective function evaluation

A convenient CLI for the whole workflow

Get a good bunch of STL models, and

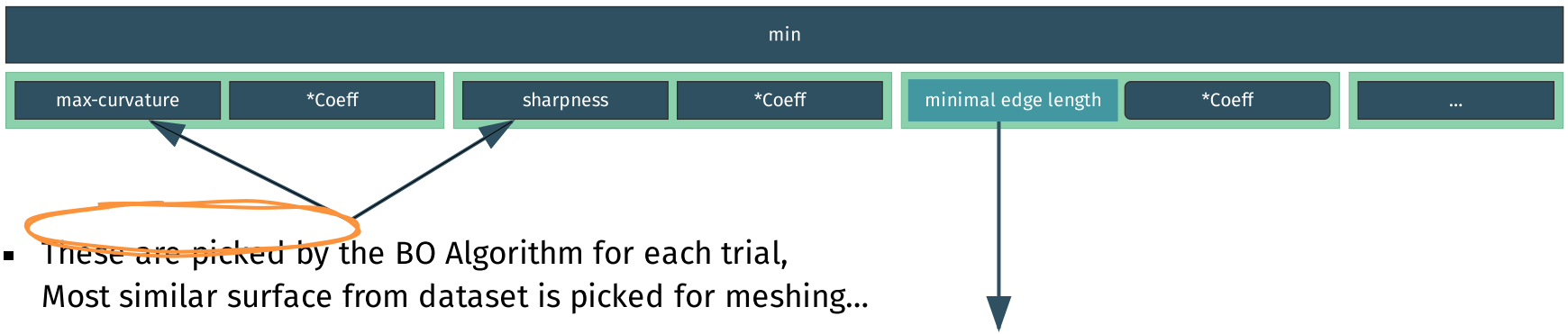
```
# Install needed dependencies
./app/meshingo install

# Prepare your STLs, only needed for training stage
cp <my_STLs*> training_dataset/

# Training stage, run once to produce a surrogate model
export STAGE1_MAX_OCTREES=9 # Optional, default is 9, more means more RAM usage
./app/meshingo train --stage1-name Meshingo training_dataset

# Try it on a target STL (takes only few seconds to setup meshing cases)
./app/meshing validate --model stage1/Meshingo <path_to_stl>

# See if SM is biased towards some geometric features [optional]
./app/meshingo bias-scan --stage1-name Meshingo --threshold 0.05

# Fine-tuning stage, uses the SM instead of the meshing tool [optional]
./app/meshingo predict --stage1-name Meshingo --stage2-name MY_TARGET <path_to_STL>
```

Stage 1 - Surrogate model training



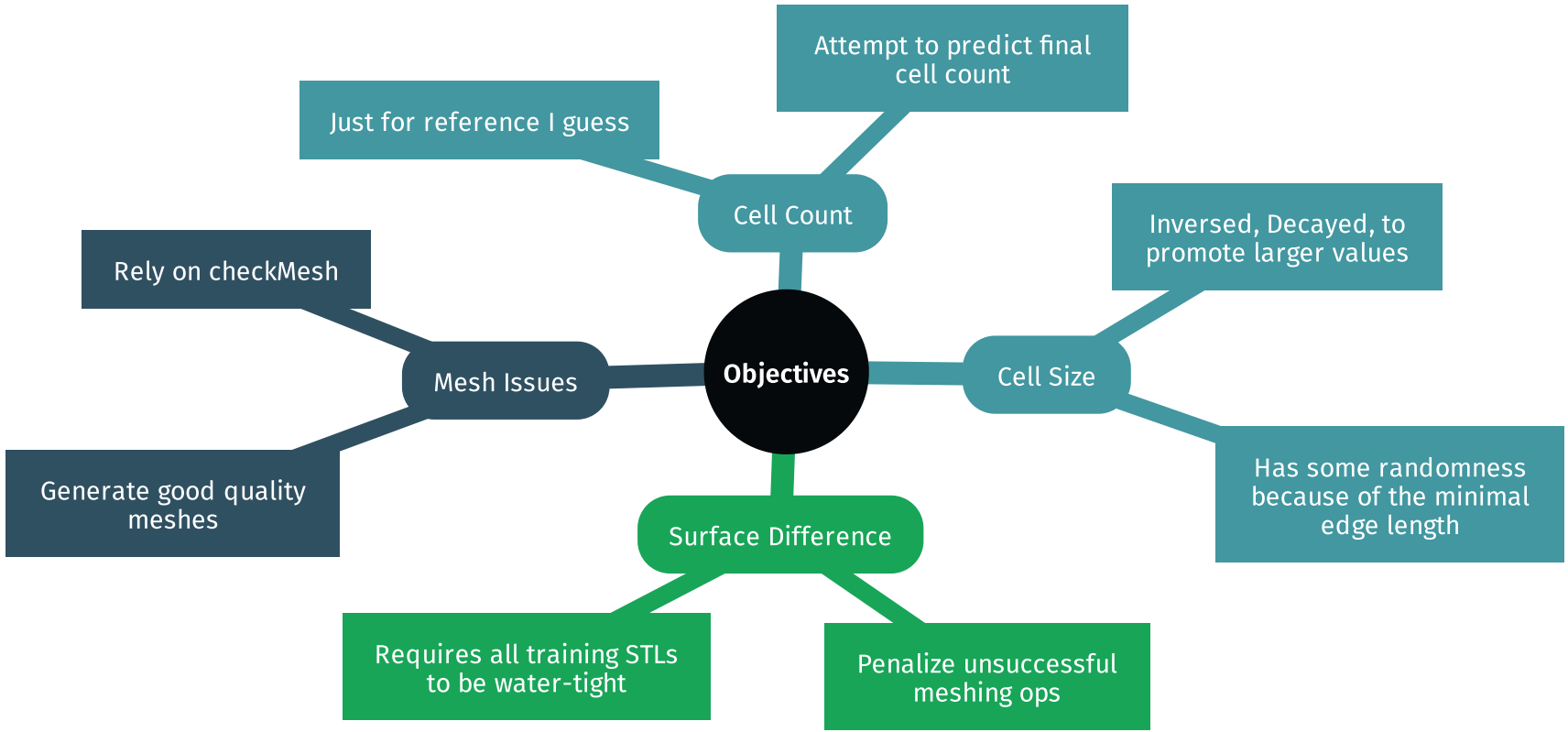
Stage 1 - Surrogate model training

The `Cell-Size` objective function is calculated as follows, then an inverse-decay is applied:



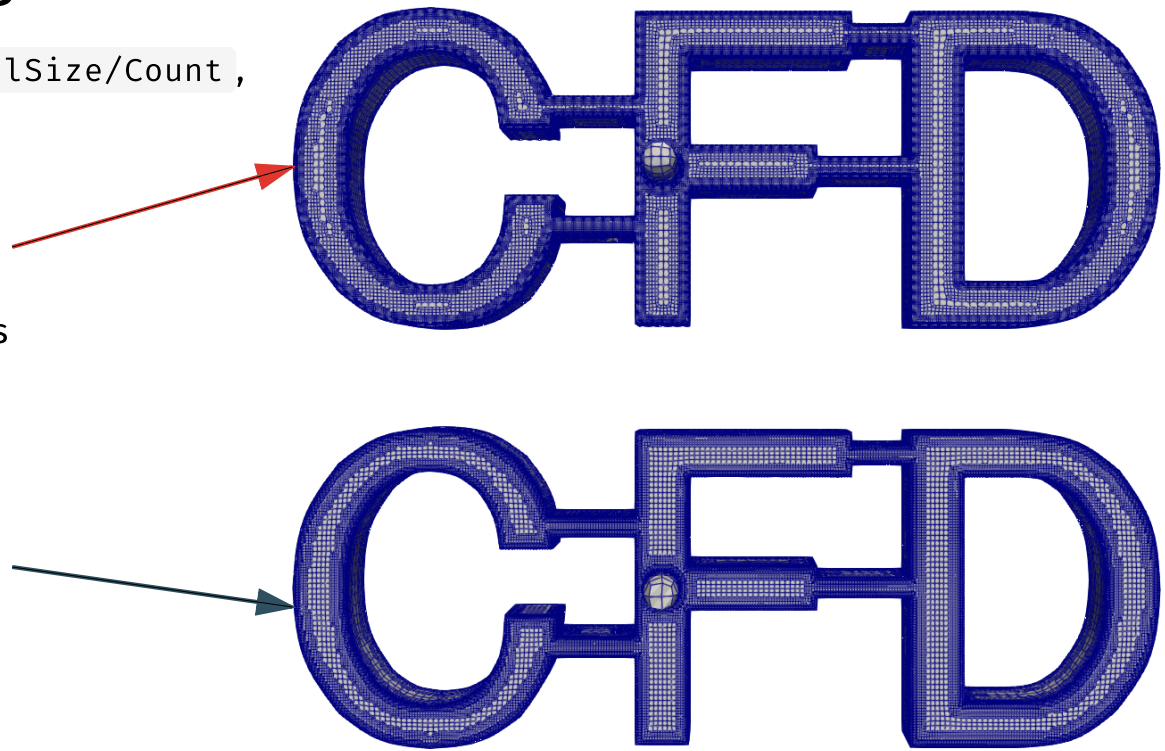
- These are picked by the BO Algorithm for each trial,
  Most similar surface from dataset is picked for meshing...
- Minimal edge length, is the smallest length that is supposed to capture all surface features.
  Prematurely computed by flooding the STL with rays and recording shortest ray-model intersection distances

Stage 1 - Surrogate model training



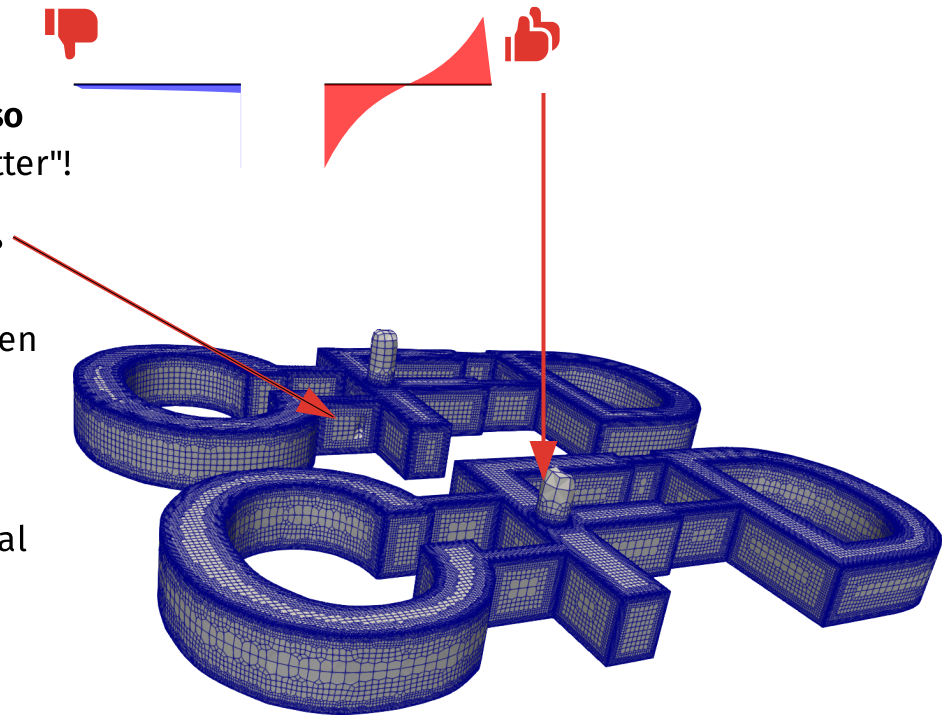
Investigating Objective Functions for Stage 1

- Best settings for focusing on `CellSize/Count`, and `Mesh Issues`
  - Min Cell Size: **5.48968e-05**
  - Max/Min aspect ratio: **10.1755**
  - 1 050 183 cells, no mesh issues
- Best settings for focusing on `SurfaceDifference`
  - Min Cell Size: **4.35166e-05**
  - Max/Min aspect ratio: **69.793**
  - 713 235 cells, no mesh issues

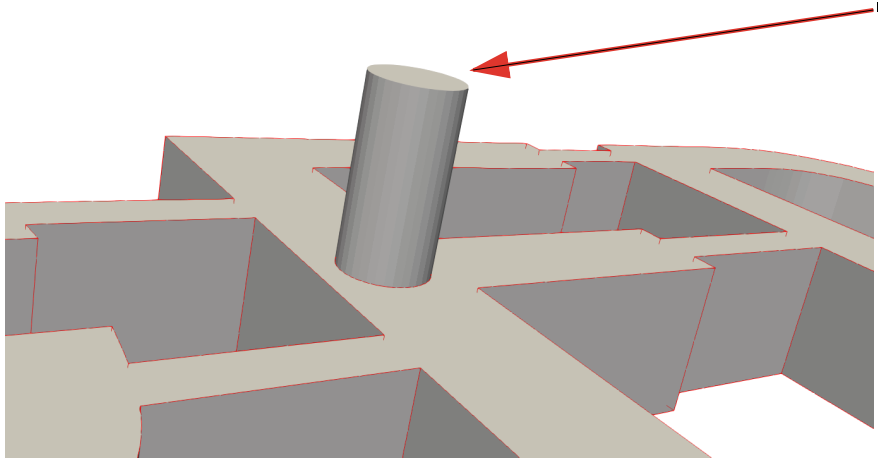


Investigating Objective Functions for Stage 1

- `SurfaceDifference` metric is **currently not so good**; Doesn't translate to "smaller value is better"!
- Can discover **random** `cartensianMesh` **bugs**?
- Overall, the Pareto Frontier data, used on unseen STL files
  1. provides a decent cell size out of stage 1
  2. needs some polishing around curvy/spherical regions



Investigating Objective Functions for Stage 1



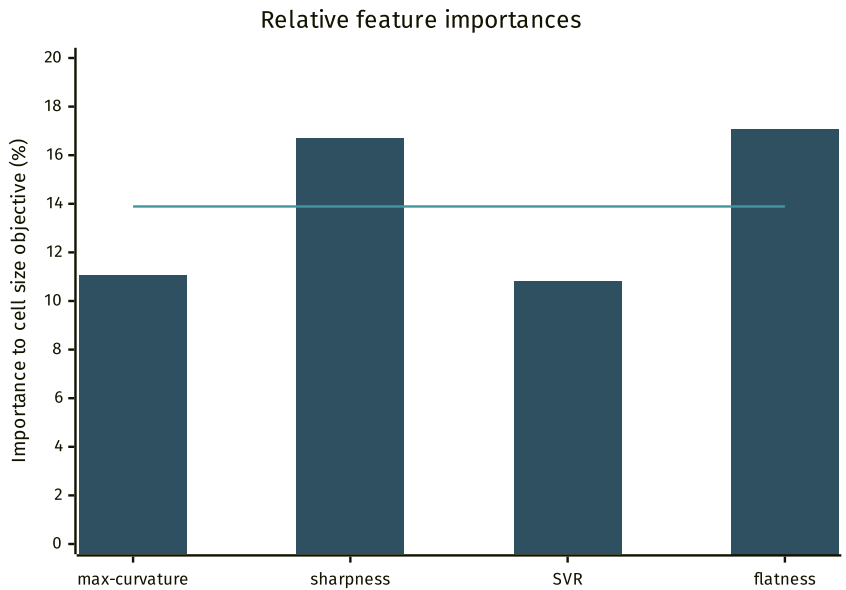
- BTW, This is how the STL file looks like.
- Red lines denote (auto) detected feature edges.

  This one was missed on purpose
  - Delegating cell size to "feature-edges angle"
  - But predictions are way off (`1e-3` training improvement bar):

Model inference	Cell count	mesh issues
SD	25.3m	1
SCI	21.3m	0

Checking for model Bias

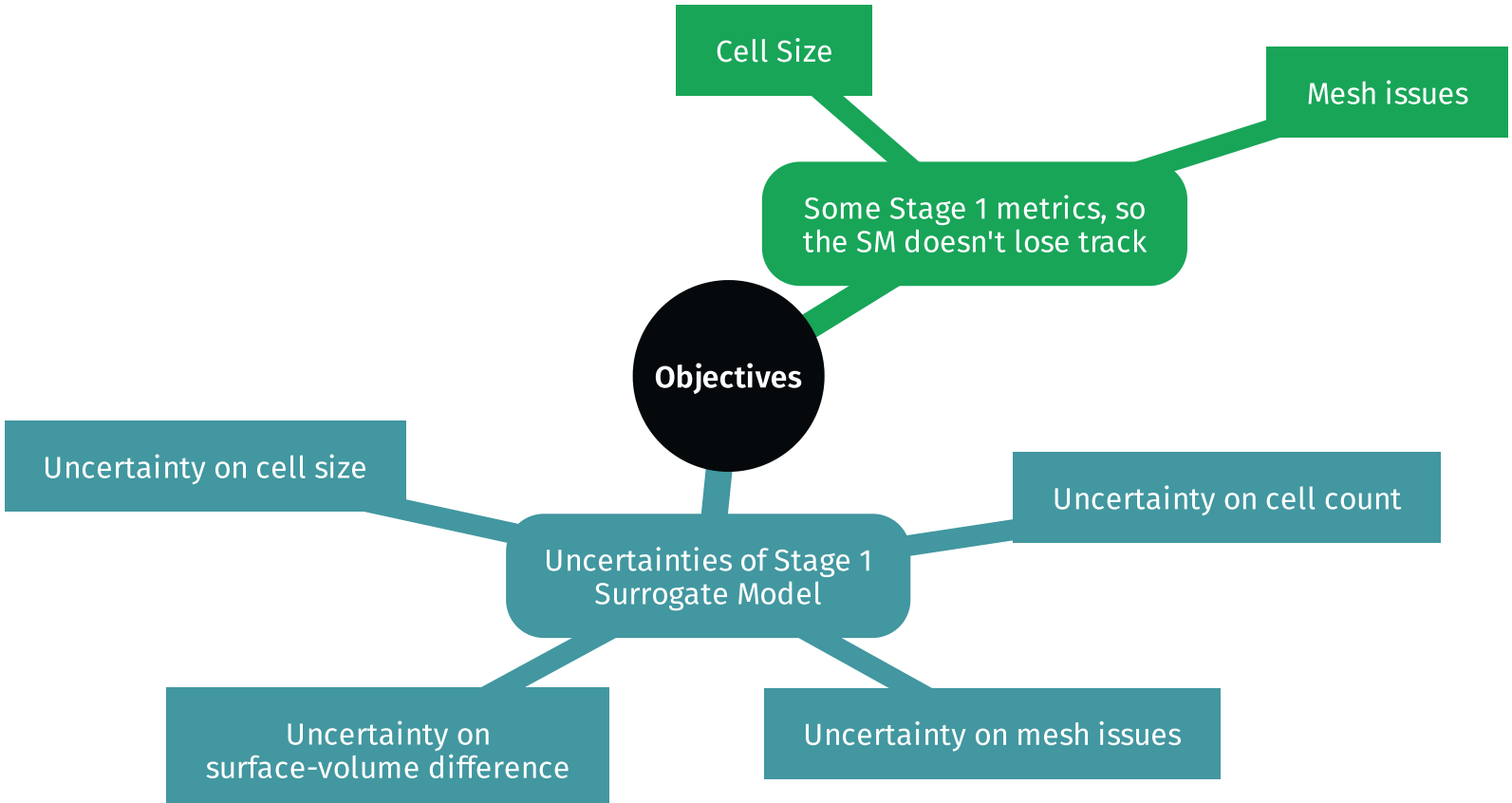
We got a surrogate model from Stage 1, hopefully a well trained GP.



- This model is **a little** biased towards sharp and flat surfaces.
- But this might be OK if the model is supposed to work on a certain "type" of surface models.

Stage 2 - Surrogate model fine-tuning

Stage 2 - Surrogate model fine-tuning

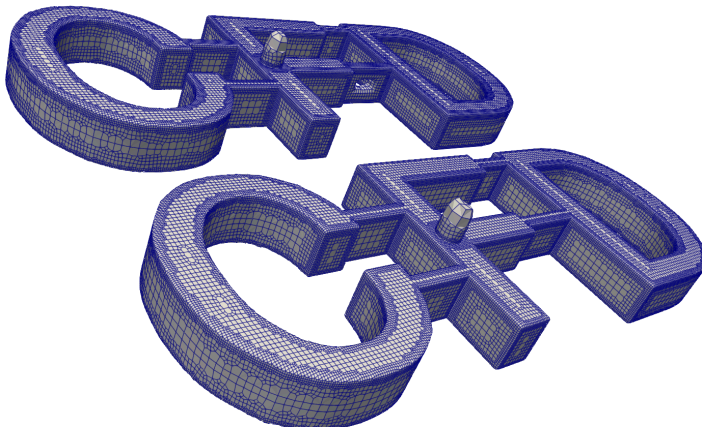


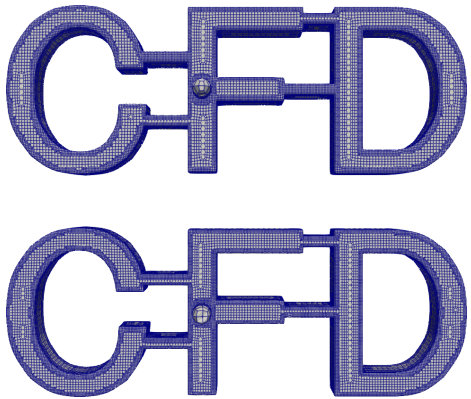
Stage 2 - Surrogate model fine-tuning

- Now, we take the surrogate model from Stage 1, and use it instead of the meshing tool
  - This is much faster
  - But bias from Stage 1 is inherited
- Instead of picking the most similar surface from a training set, use the target STL
  - This helps the model discover around the target STL
  - But takes a bit of time (additional 30-40 trials)

- By the end, a pareto-frontier is generated for the Stage 2 objectives
  - A few cell sizes are suggested, together with their predicted performance
  - The user can then choose the cell size to use based on their impact on the objectives

Stage 2 - Surrogate model fine-tuning





Top, best for `CellSize` and uncertainty on `SurfaceDifference`
Bottom, best for uncertainty on `cellSize`, `Cell Count` and `Mesh issues`, and the `Mesh Issues` objective itself

	Cell Size	maxMin ratio	Cell Count (CI)
SZ	4.34e-05	350	854.594k (13%)
U(CCI)	2.96e-04	10	92.424k (9%)

Stage 2 - Surrogate model fine-tuning - Remarks

- Current implementation is inefficient
  - Re-fits the surrogate model on each trial
  - Depending on the target STL, might need 200+ trials to converge
    - to a `1e-3` improvement bar

- Reducing confidence intervals still doesn't make predictions on `CellCount` fall within the confidence intervals
  - Probably needs way more trials
  - Cell count is normalized by STL surface area. Any wrong doings here?

Stage 2 - Surrogate model fine-tuning - Remarks

- Does the improvement in finding lower-cell-count configurations justify the extra effort for stage 2?
- Maybe a BO in 2nd stage isn't the best use of Bayesian Optimization?
  - Active learning? Refining/Validating untrusted regions from stage 1
  - Multi-fidelity BO for stage 2? Picking whether to use the surrogate or performing an actual meshing operation... BUT this will mesh the target STL multiple times
- Explore models other than Gussian Processes?

For now, stage 2 is there just for deriving insights and experimenting with the surrogate model

The full story (ADRs)

	Invalid Date
	Invalid Date
	Invalid Date
	Invalid Date
	Invalid Date
	Invalid Date
	Invalid Date
	Invalid Date
	Invalid Date
	Invalid Date

Critique and Discussions

- Focusing on **Automation & Unattended Meshing**
- Data-Efficient Surrogate Model Training, not needing 1000s of STLs as a GNN/CNN would
- BO is a good match since meshing operations can be so expensive

- Mediocre handling of geometric features importances
  - `min(flatness*Cf, sharpness*Cs, ...)`
  - Maybe a CNN on voxalized STLs is better?

- Graph Neural Networks (GNNs) as replacement for Stage 1 as GNNs are better suited for learning spatial relationships in geometric data
- Running a simple potential flow through the mesh can reveal where refinement is needed...
- Discontinuities not well modeled , assuming smooth meshing responses is not wise...
- Multi-scale geometries may require more than a single max-to-min cell size ratio
- Heavy bias towards the `cartesianMesh` configuration (eg. no boundary layers... etc)

Thank you for your attention

GitHub · More stuff like this