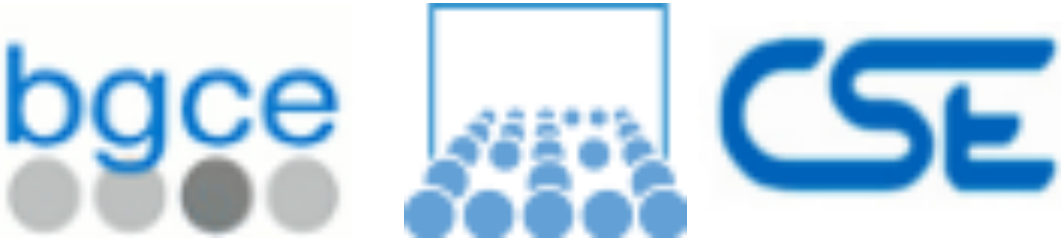
BGCE First Milestone Meeting

BGCE Project: CAD – Integrated Topology Optimization

S. Joshi, J.C. Medina, F. Menhorn, S. Reiz, *B. Rüth*, *E. Wannerberg*, A. Yurova

Technische Universität München

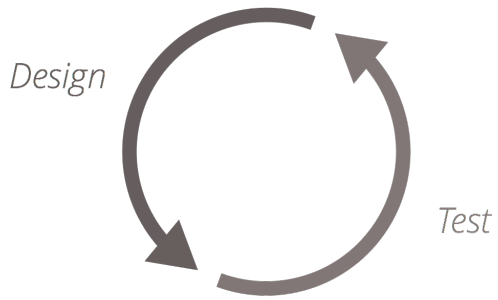
August 5, 2015



Contents

Motivation

Current Design Process:



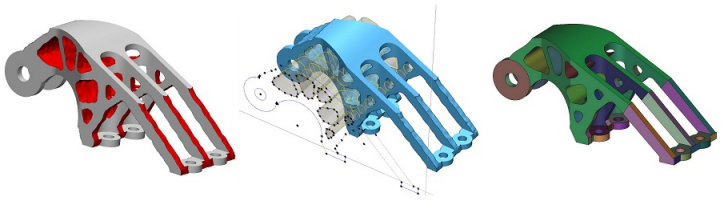
- Iterative and redundant
- Time consuming

Motivation

Current Design Process:

- Iterative and redundant
- Time consuming

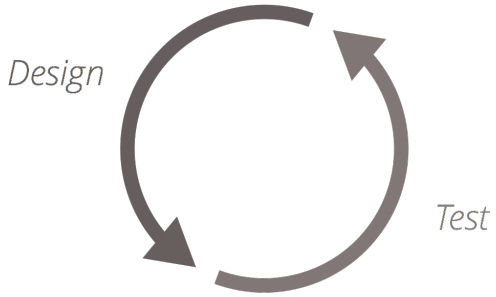
Topology optimization



- Promoted by additive manufacturing

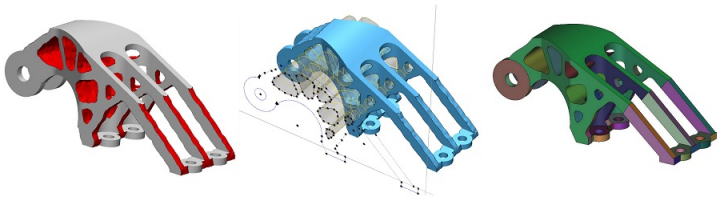
Motivation

Current Design Process:



- Iterative and redundant
- Time consuming

Topology optimization



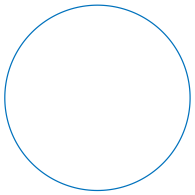
- Promoted by additive manufacturing

Focus:

Convert optimized geometry to **lightweight** and **scalable** CAD formats

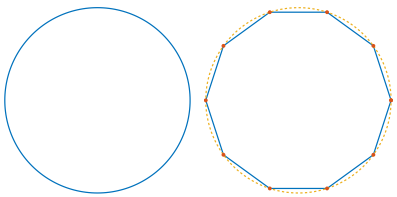
Workflow Overview

CAD design



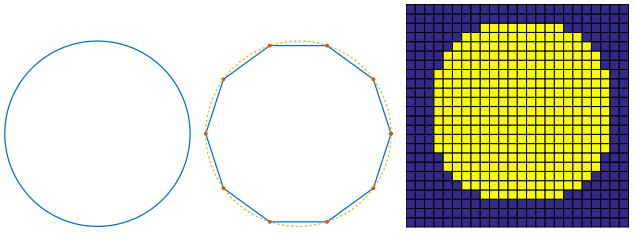
Workflow Overview

STL Interface



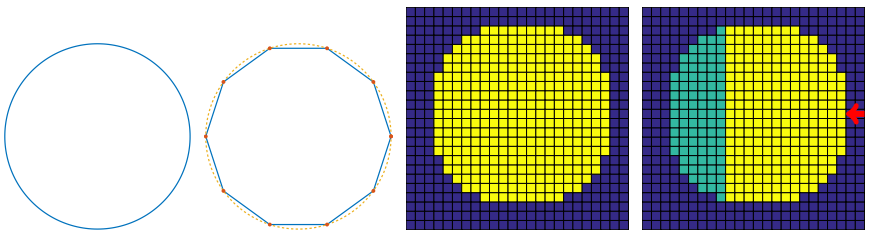
Workflow Overview

Voxelization



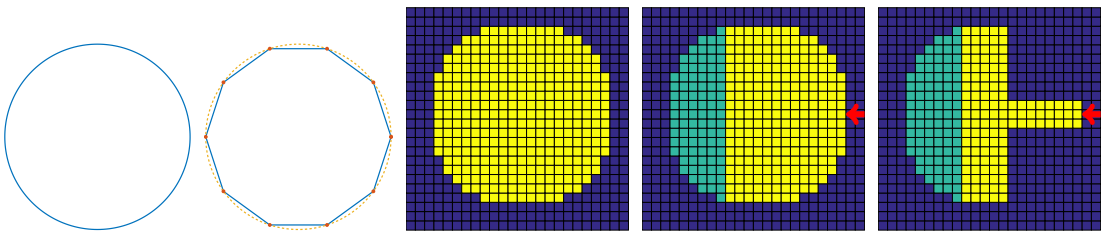
Workflow Overview

TPD input file - Specification of loads and fixtures



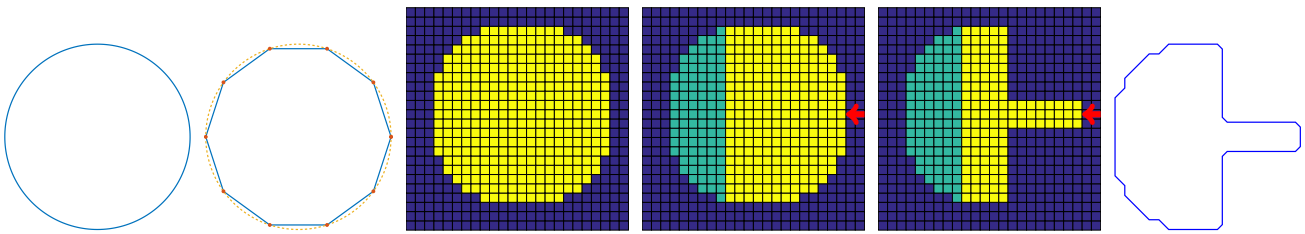
Workflow Overview

Topology optimization



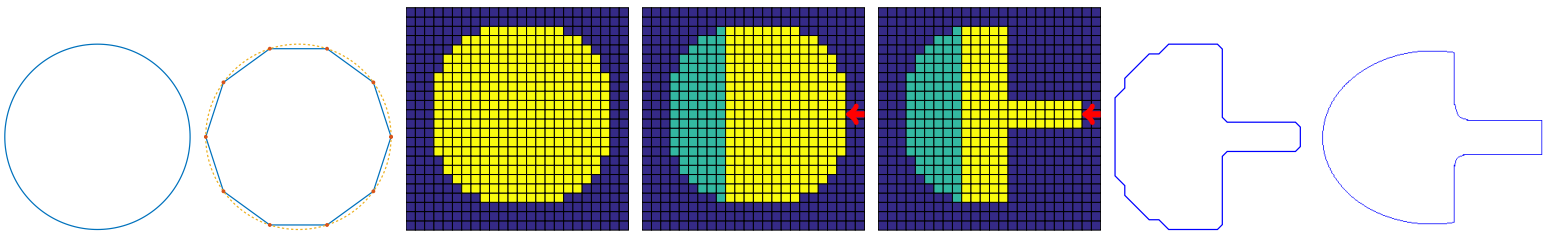
Workflow Overview

Voxelised geometry



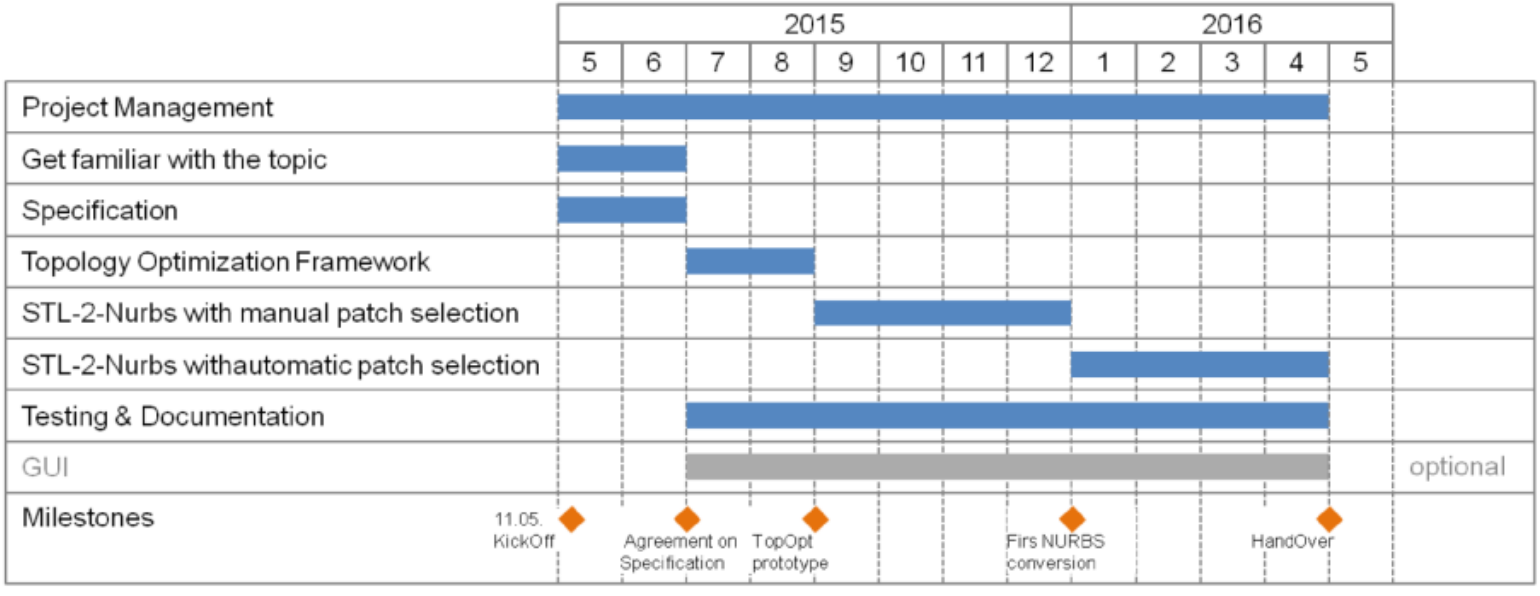
Workflow Overview

Post-processing: Parametrization, Feature recognition



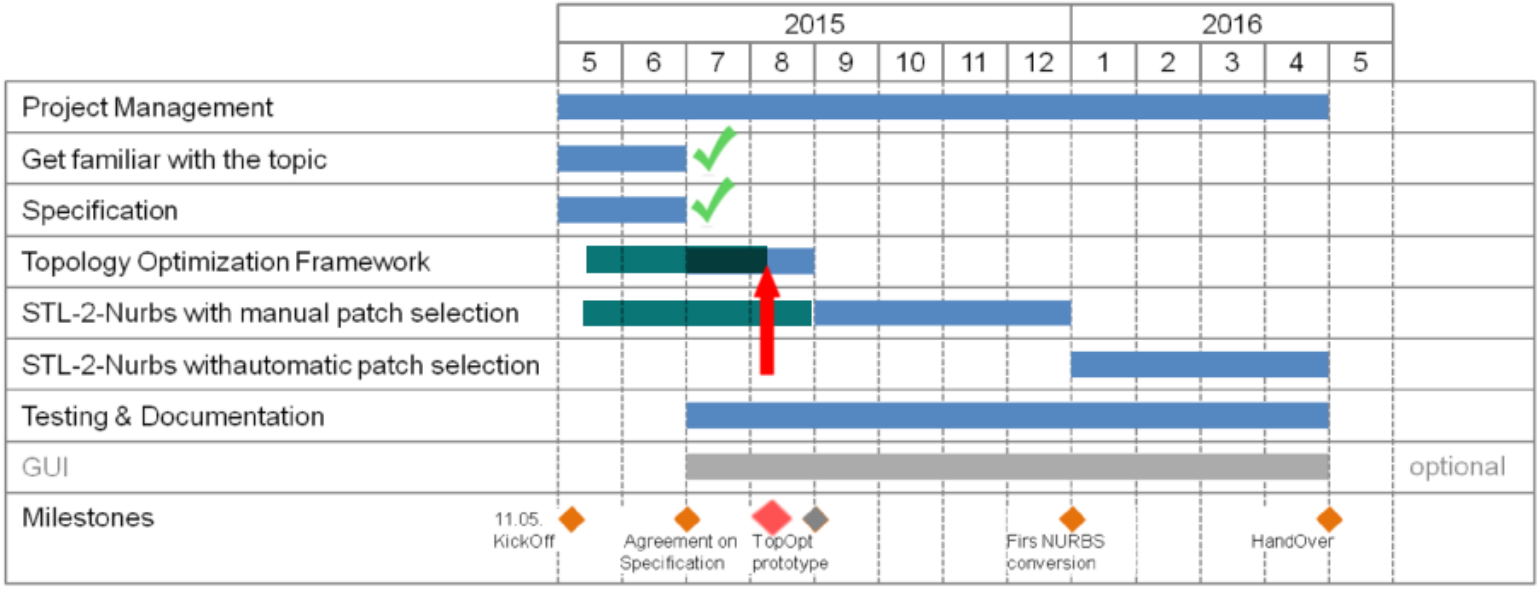
Schedule & Milestones

Schedule:



Schedule & Milestones

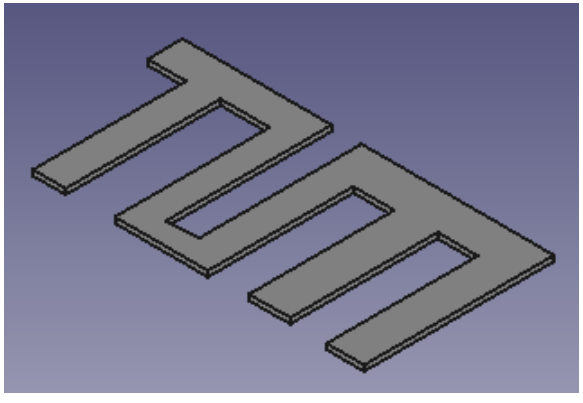
Schedule: (current)

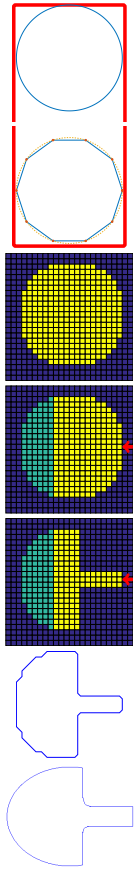


CAD to STL

Tools:

- Create original CAD geometry in CAD program





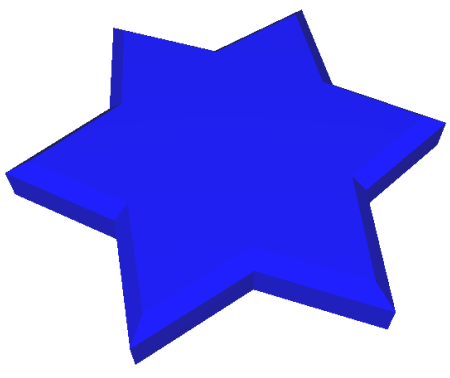
Interface:

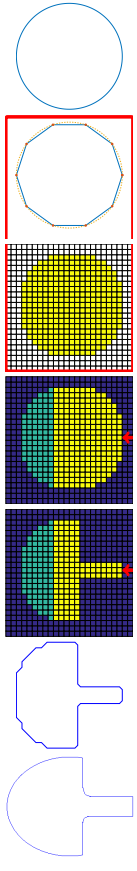
- Most CAD programs offer: Export to STL...

From STL To Voxels

Tools:

- Common Versatile Multi-purpose Library for C++ (CVMLCPP)
  - Takes .stl file and returns a binary file with the given voxel size
- Custom script to read binary file and output it as ascii.vtk

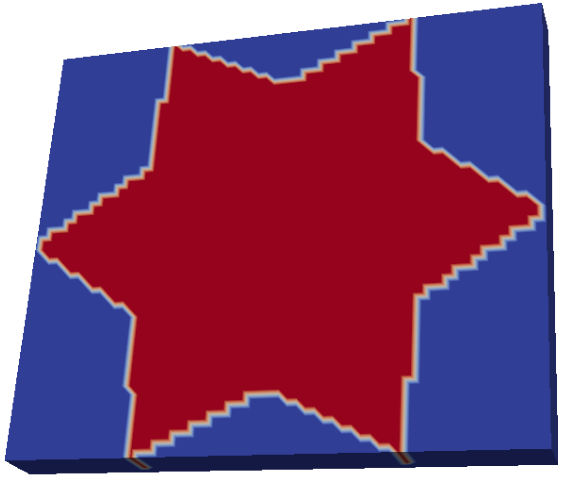


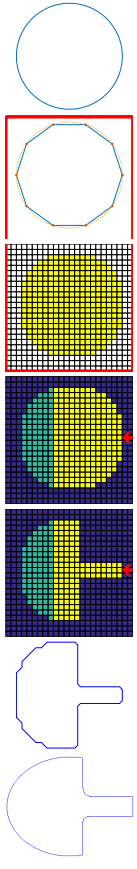


From STL To Voxels

Tools:

- Common Versatile Multi-purpose Library for C++ (CVMLCPP)
  - Takes .stl file and returns a binary file with the given voxel size
- Custom script to read binary file and output it as ascii.vtk

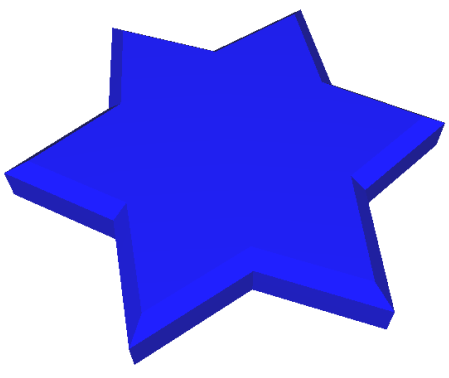


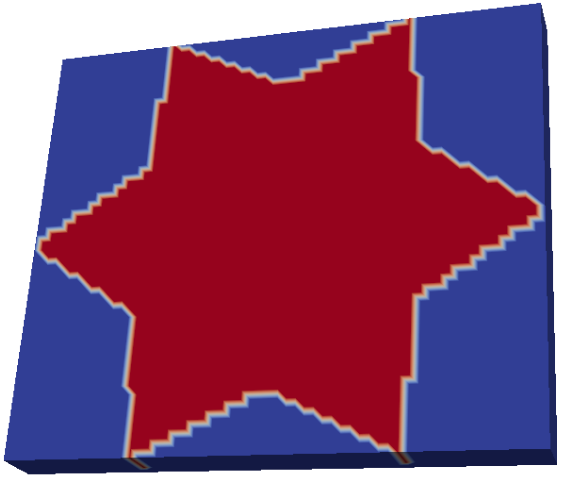


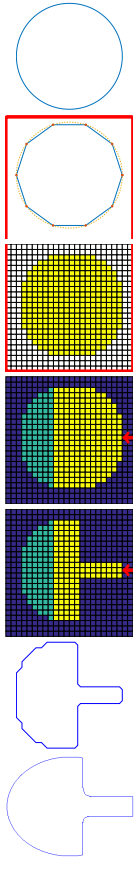
From STL To Voxels

Tools:

- Common Versatile Multi-purpose Library for C++ (CVMLCPP)
  - Takes .stl file and returns a binary file with the given voxel size
- Custom script to read binary file and output it as ascii.vtk



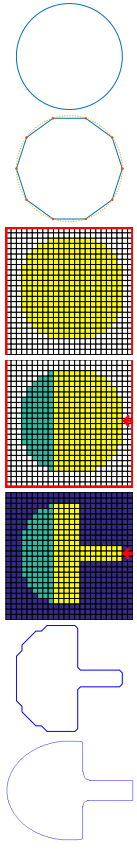




Load and fixture specification

Boundary conditions required - how to specify?

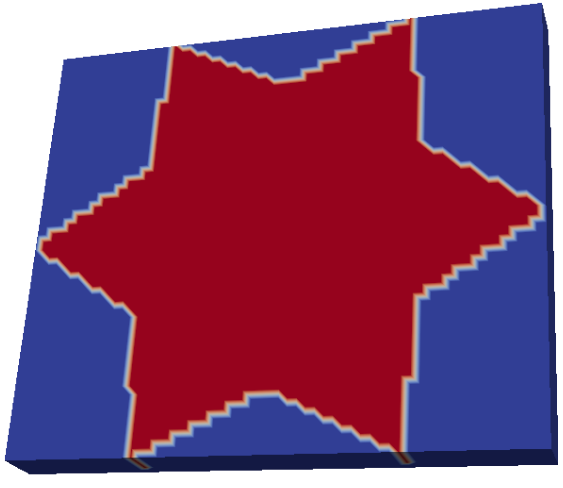
- Current state: Manual specification
- Idea 1: Metafile before Voxelization step

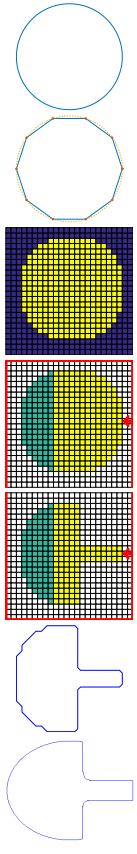


Topology Optimization

Tools:

- ToPy for topology optimization
- Custom script for generating .tpd file
  - Takes binary output from CVMLCPP and generates ToPy input
  - Sets non-voxel cells to passive elements
  - Adds boundary conditions manually

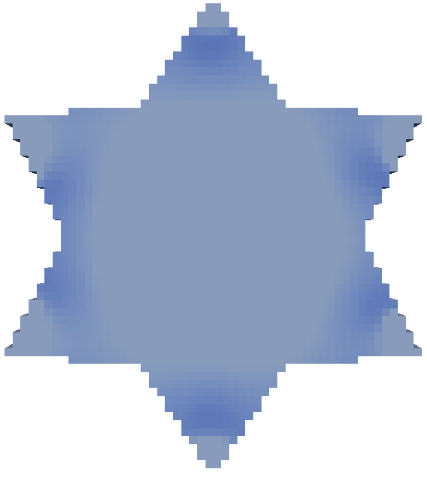


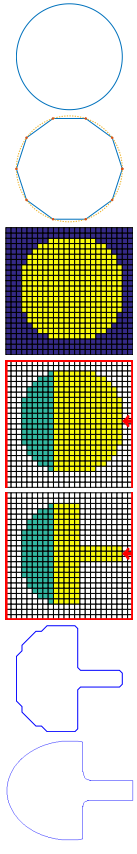


Topology Optimization

Tools:

- ToPy for topology optimization
- Custom script for generating .tpd file
  - Takes binary output from CVMLCPP and generates ToPy input
  - Sets non-voxel cells to passive elements
  - Adds boundary conditions manually

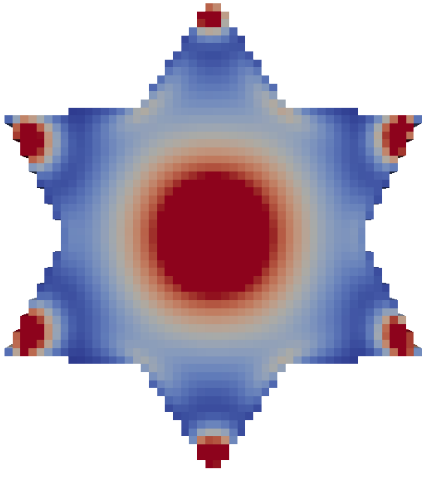


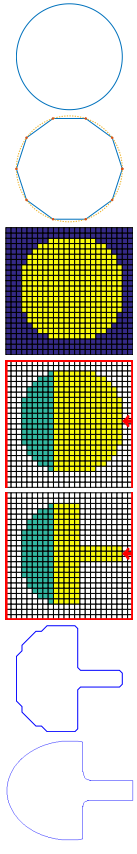


Topology Optimization

Tools:

- ToPy for topology optimization
- Custom script for generating .tpd file
  - Takes binary output from CVMLCPP and generates ToPy input
  - Sets non-voxel cells to passive elements
  - Adds boundary conditions manually

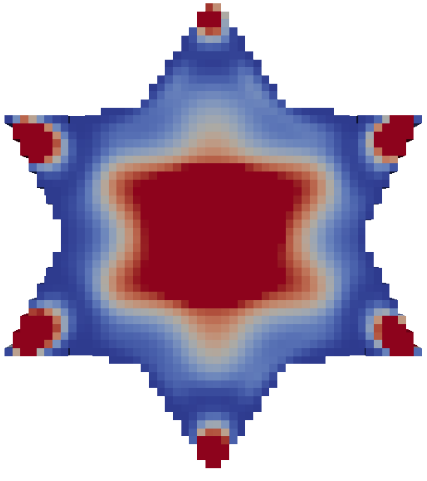


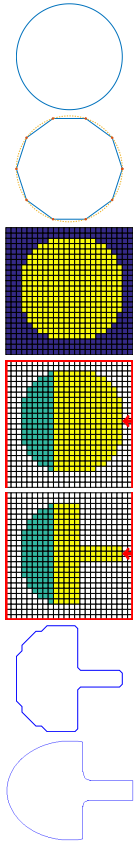


Topology Optimization

Tools:

- ToPy for topology optimization
- Custom script for generating .tpd file
  - Takes binary output from CVMLCPP and generates ToPy input
  - Sets non-voxel cells to passive elements
  - Adds boundary conditions manually

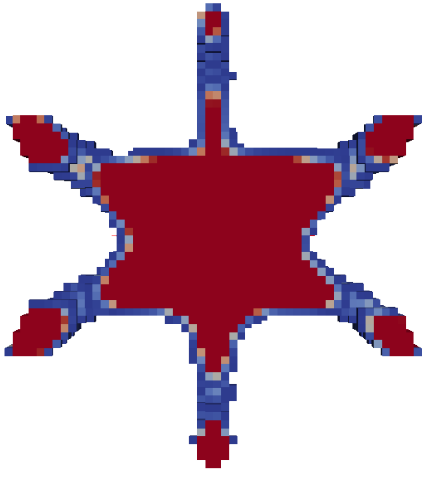


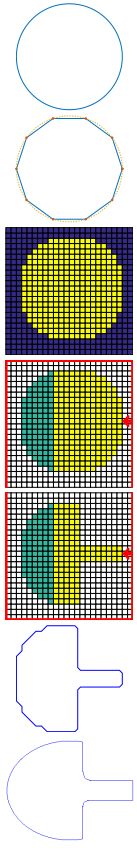


Topology Optimization

Tools:

- ToPy for topology optimization
- Custom script for generating .tpd file
  - Takes binary output from CVMLCPP and generates ToPy input
  - Sets non-voxel cells to passive elements
  - Adds boundary conditions manually

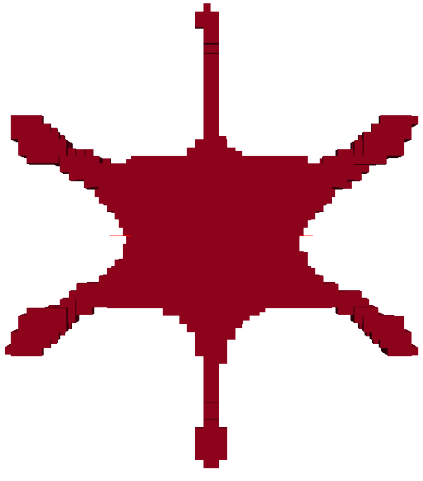


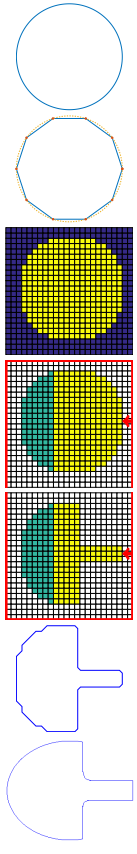


Topology Optimization

Tools:

- ToPy for topology optimization
- Custom script for generating .tpd file
  - Takes binary output from CVMLCPP and generates ToPy input
  - Sets non-voxel cells to passive elements
  - Adds boundary conditions manually

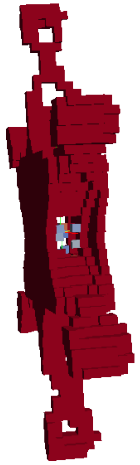


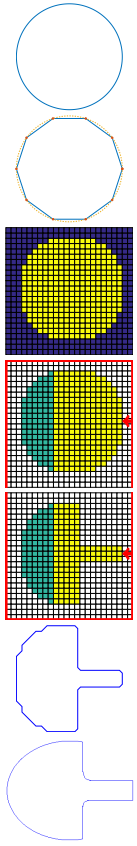


Topology Optimization

Tools:

- ToPy for topology optimization
- Custom script for generating .tpd file
  - Takes binary output from CVMLCPP and generates ToPy input
  - Sets non-voxel cells to passive elements
  - Adds boundary conditions manually

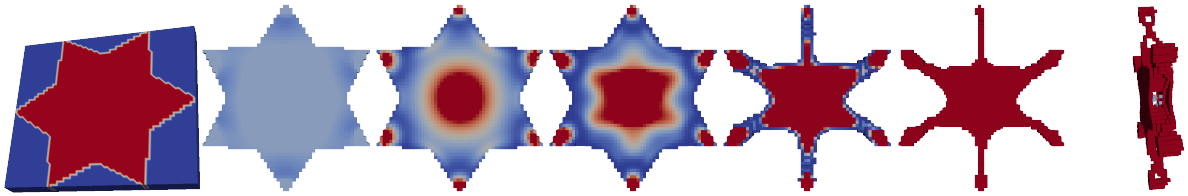


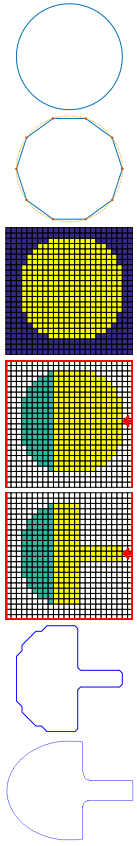


Topology Optimization

Tools:

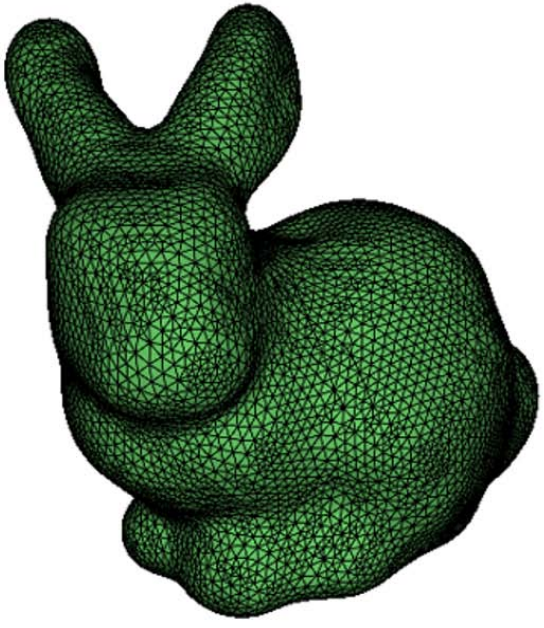
- ToPy for topology optimization
- Custom script for generating .tpd file
  - Takes binary output from CVMLCPP and generates ToPy input
  - Sets non-voxel cells to passive elements
  - Adds boundary conditions manually

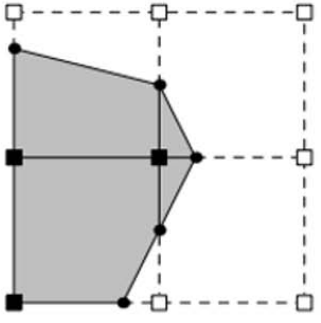




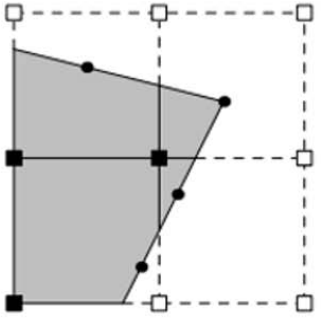
From Voxel to Mesh Geometry

- Extract isosurface from voxel information
- Algorithms: Marching Cubes, Dual Contouring, Extended Models

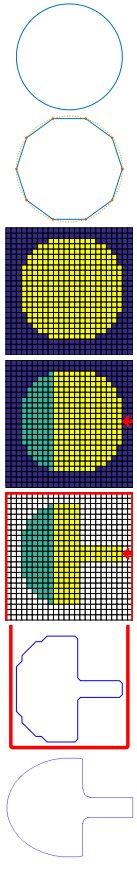




MC

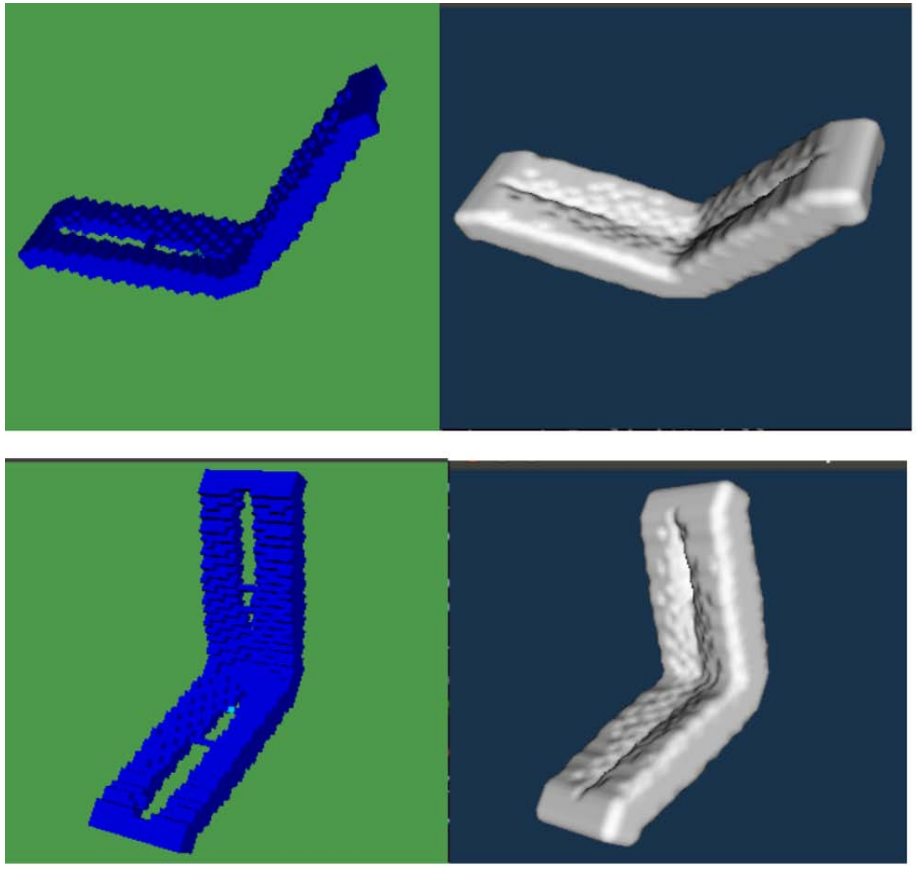


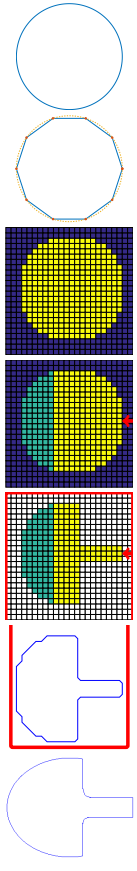
Dual method



Surface Extraction

Contour Filtering using Implicit Modelling

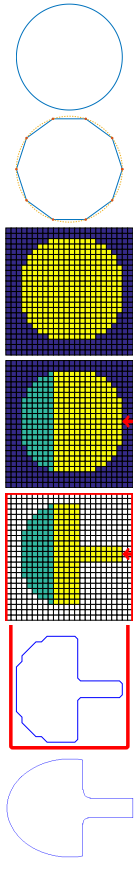




Problem: Holes are not taken into account

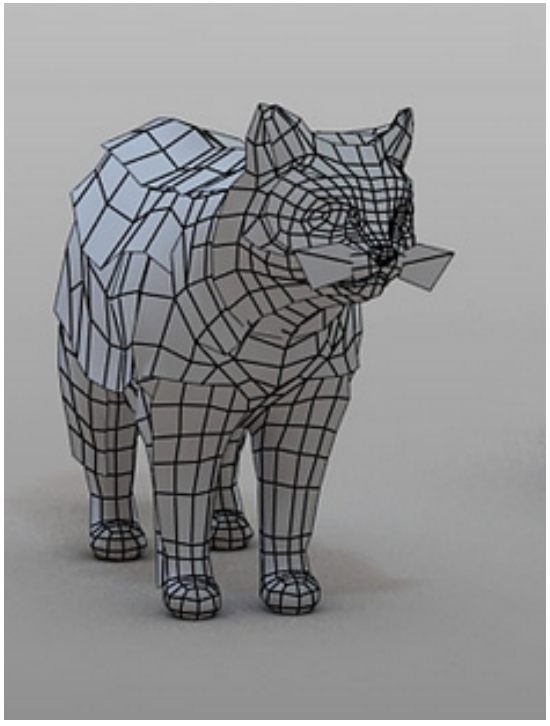
Decimation

- Fine mesh to a coarser mesh through Decimation- Reduction of number of triangles. (Upper: 50% Lower: 90%)
- Smoothing step is needed in between



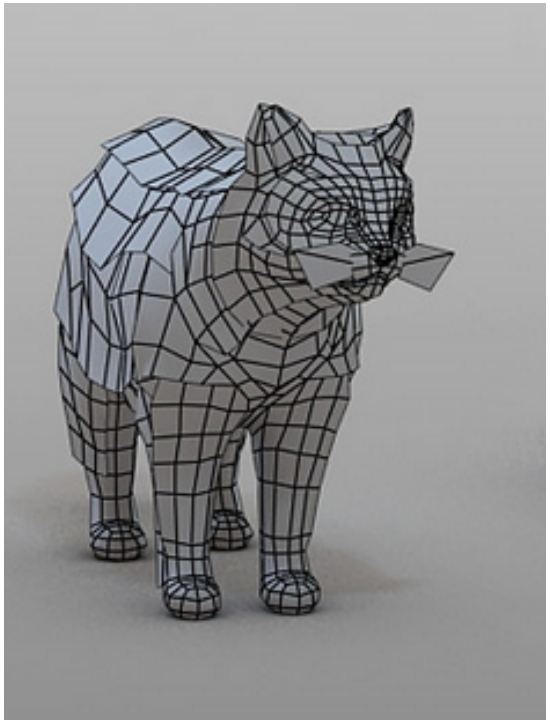
Current status

- What do we have so far?



Current status

- What do we have so far?
- What if we try to pass it to an engineer?



How to make CAD understand our data?

B–Spline

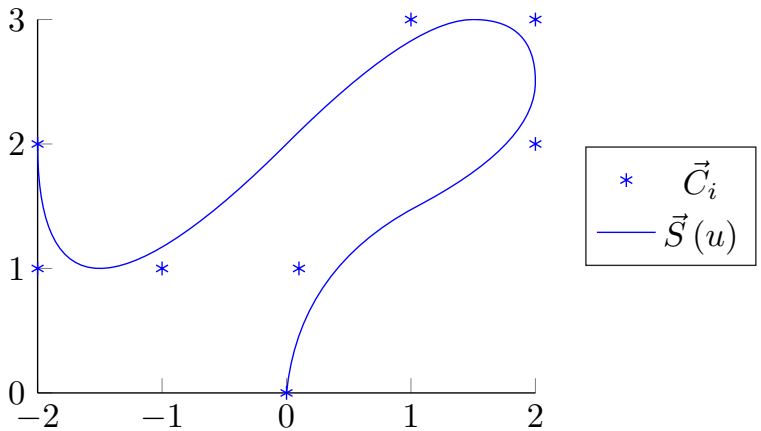
$$\vec{S}(u,v) = \sum_{i,j=1}^{n,m} \vec{C}_{i,j} N_i^p(u) N_j^p(v),$$

where p – degree of the B–Spline surface and n, m – number of control points in each direction.

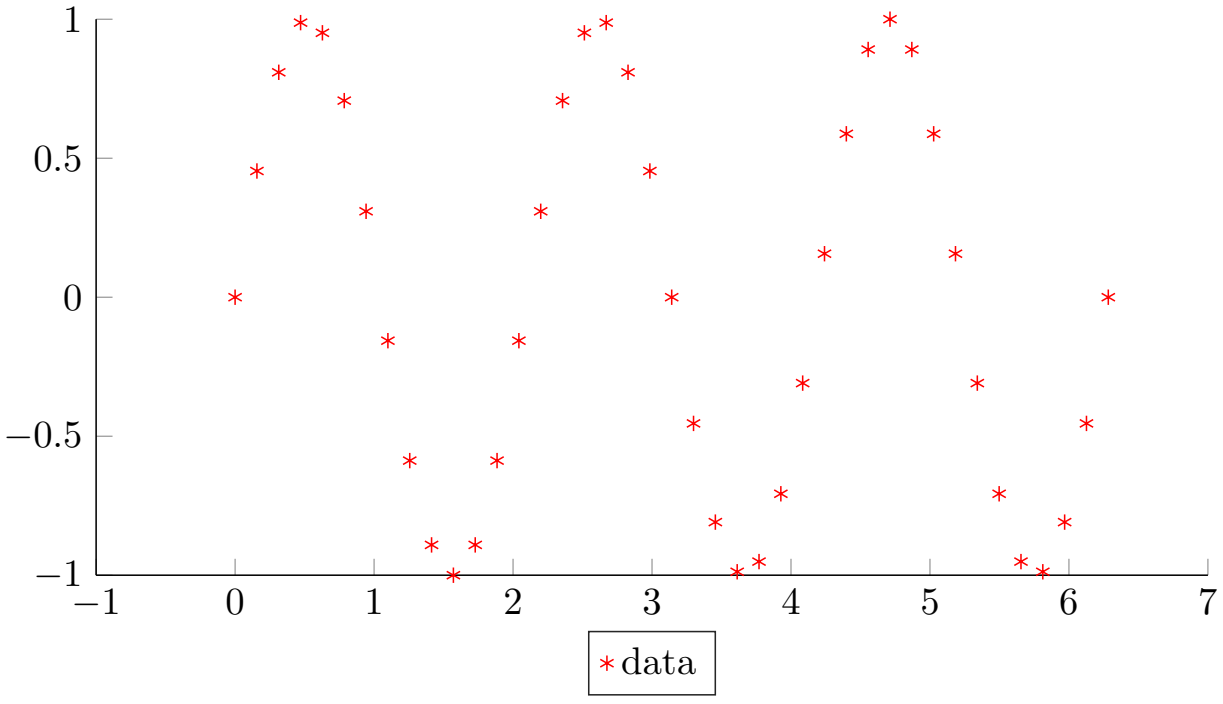
B–Splines

- offer great flexibility for handling arbitrary shapes
- are CAD–standard

Engineers are working with CAD



B–Spline Fitting

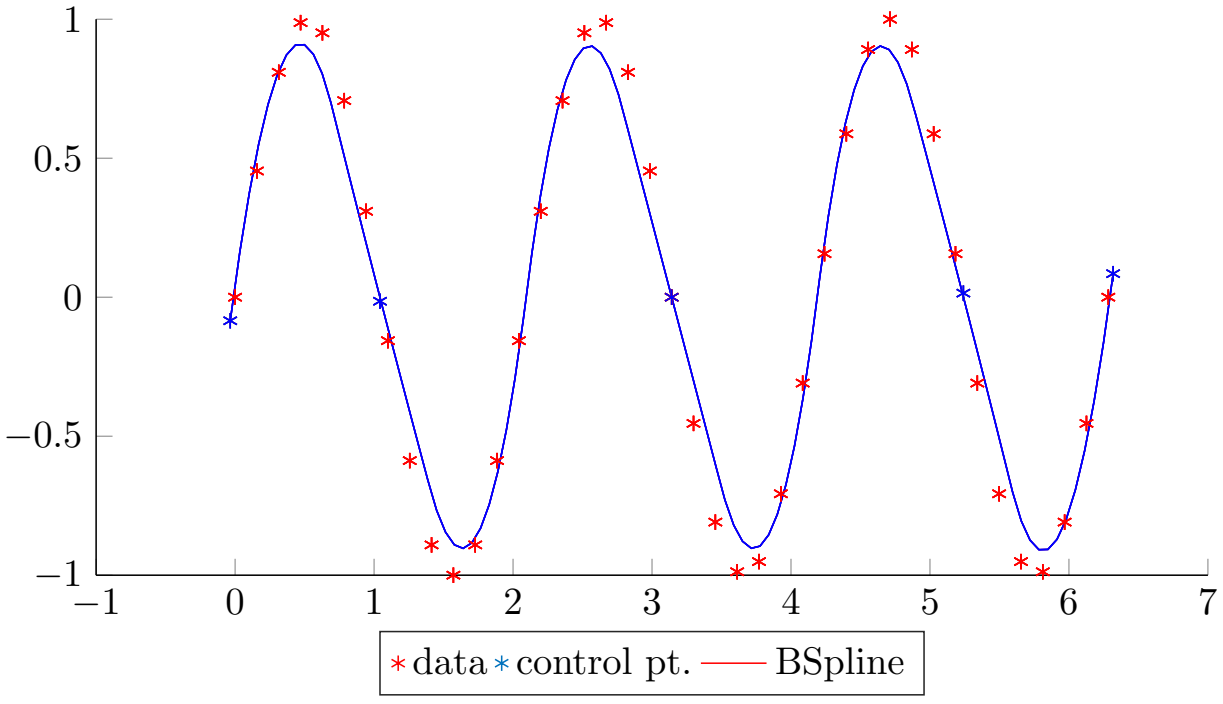


Goal:

Find B–Spline representation of our data!

$$\vec{S}(u_\alpha, v_\alpha) \approx \vec{P}_\alpha$$

B–Spline Fitting



Goal:

Find B–Spline representation of our data!

$$\vec{S}(u_\alpha, v_\alpha) \approx \vec{P}_\alpha$$

B–spline fitting: Least squares

The task:

Find control points $C_{i,j}$, such that the B–Spline surface

$$\vec{S}(u,v) = \sum_{i,j=1}^{n,m} \vec{C}_{i,j} N_i^p(u) N_j^p(v)$$

approximates our dataset of points $\left\{\vec{P}_\alpha\right\}$.

This leads to *minimization problem*:

$$\vec{S}(u_\alpha, v_\alpha) \approx \vec{P}_\alpha \forall \alpha \leftrightarrow \min_{\vec{C}_{i,j} \in \mathbb{R}^3} \sum_\alpha \parallel \vec{P}_\alpha - \vec{S}(u_\alpha, v_\alpha) \parallel_2$$

B–spline fitting: Least squares (cont.)

Resulting system looks like:

$$\sum_{i,j=1}^{n,m} \vec{C}_{i,j} N_i^p(u_\alpha) N_j^p(v_\alpha) \approx \vec{P}_\alpha \quad \forall \alpha$$

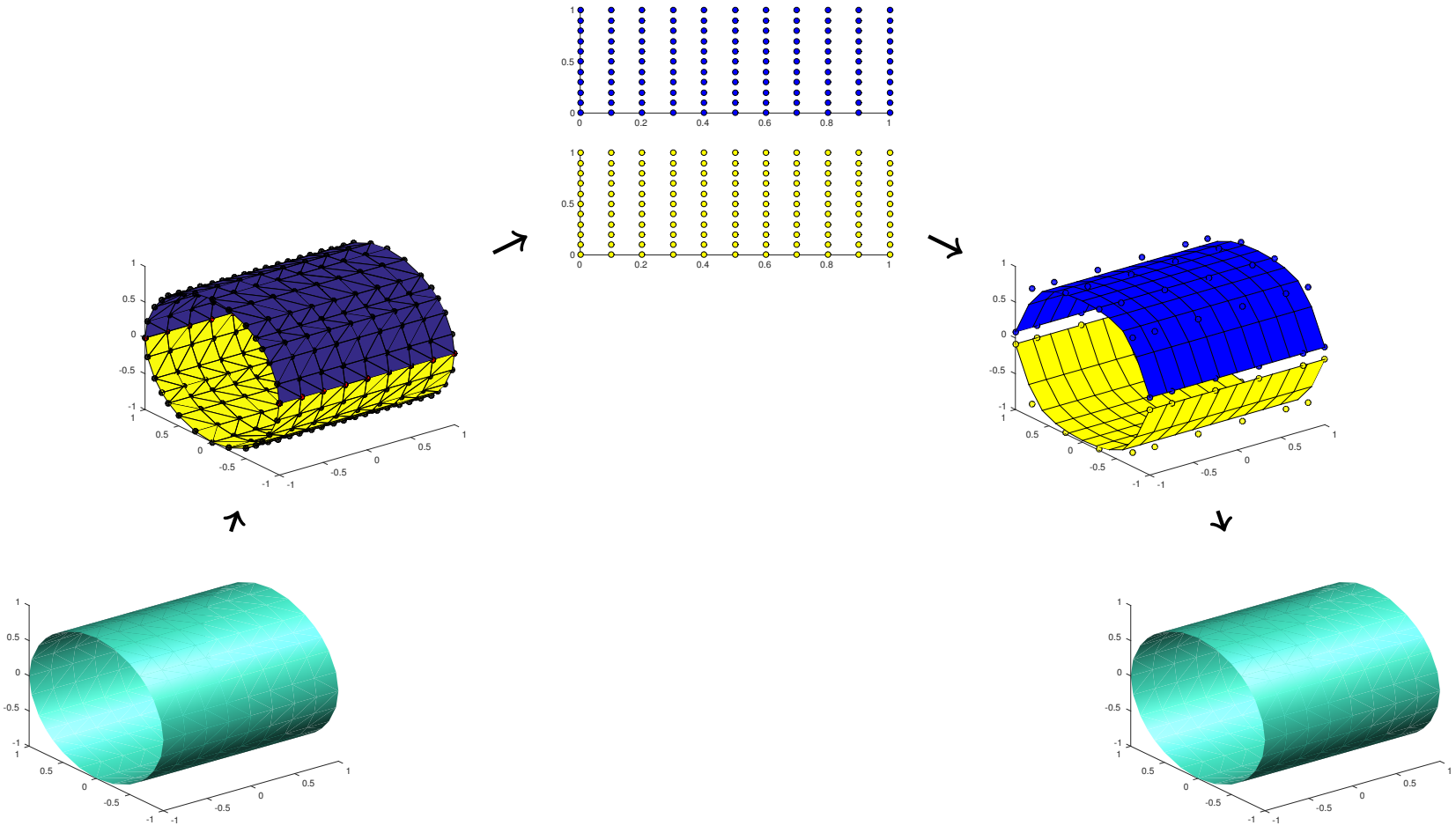
Or, in matrix–vector form:

$$AC \approx P$$

Our system matrix A depends on $\{u_\alpha, v_\alpha\}$

B–Spline Fitting pipeline according to Becker, Schäfer, Jameson

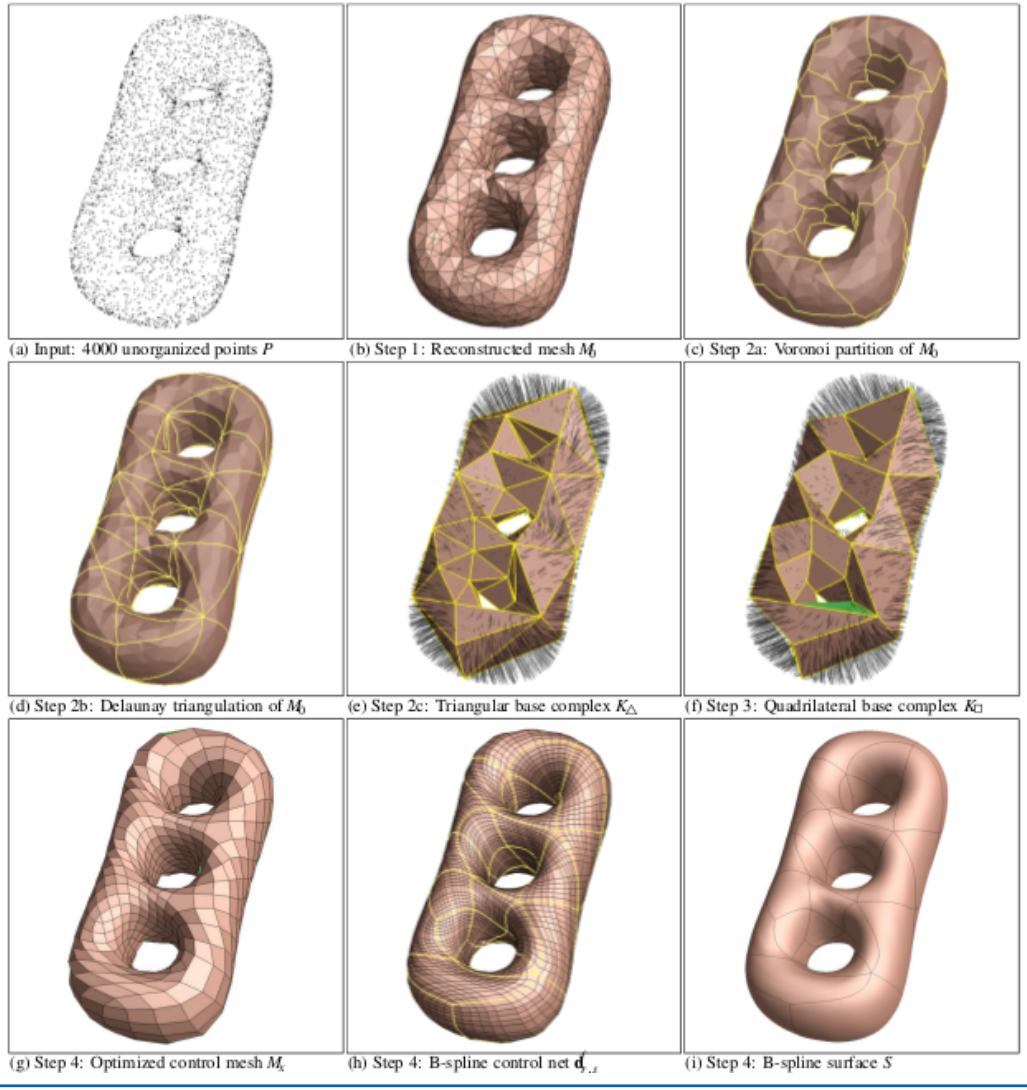
How to deal with a complex shape?



B–Spline Fitting: Open questions

- How to distribute our data into patches?
- How to parameterize obtained patches?
- How to connect several patches after fitting?

B–Spline Fitting pipeline according to M. Eck & H. Hoppe

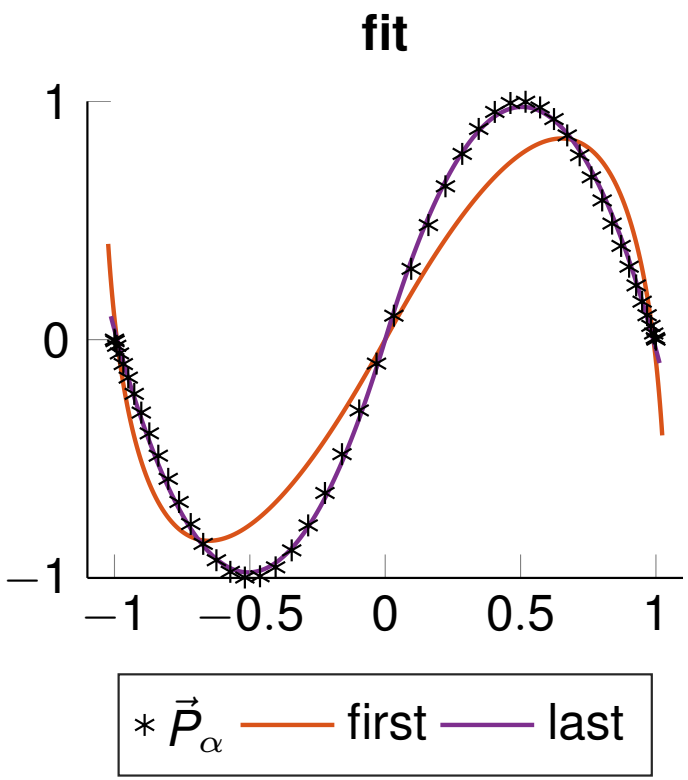


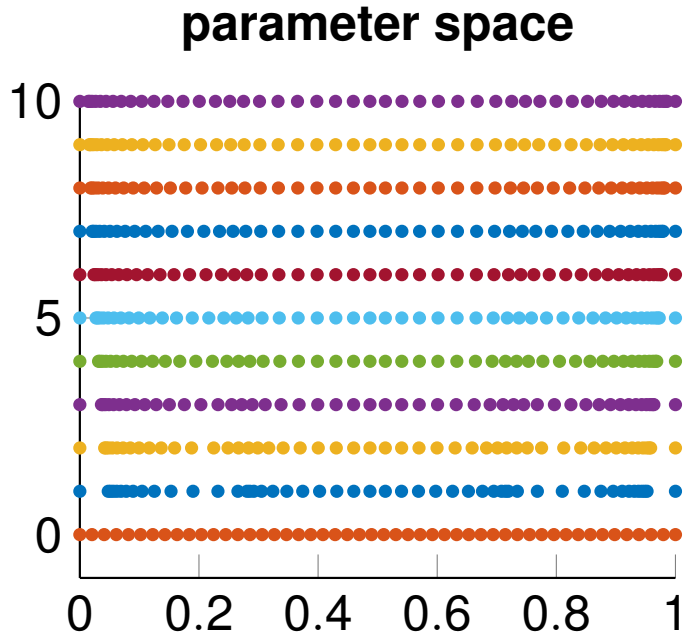
(a) Input: 4000 unorganized points P
(b) Step 1: Reconstructed mesh M_0
(c) Step 2a: Voronoi partition of M_0
(d) Step 2b: Delaunay triangulation of M_0
(e) Step 2c: Triangular base complex $K_\triangle$
(f) Step 3: Quadrilateral base complex $K_\square$
(g) Step 4: Optimized control mesh M_s
(h) Step 4: B-spline control net $\mathbf{d}_{r,s}$
(i) Step 4: B-spline surface S

B–Spline Fitting: Parameter correction

The task:

For *fixed* control points $C_{i,j}$, find an optimal parametrization $\{u_\alpha, v_\alpha\}$.





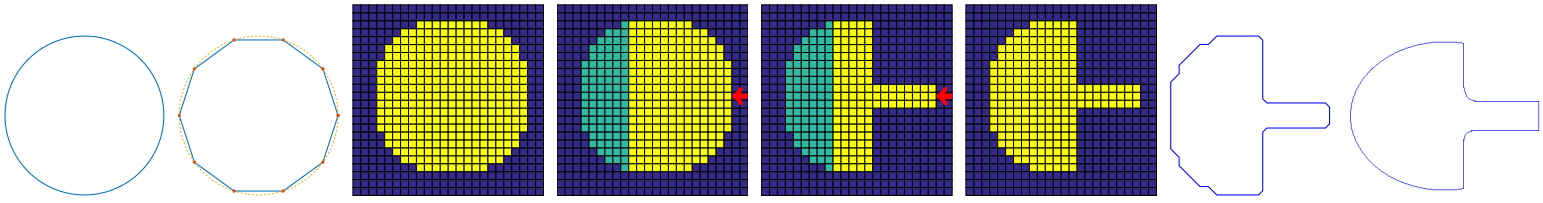
What is done?

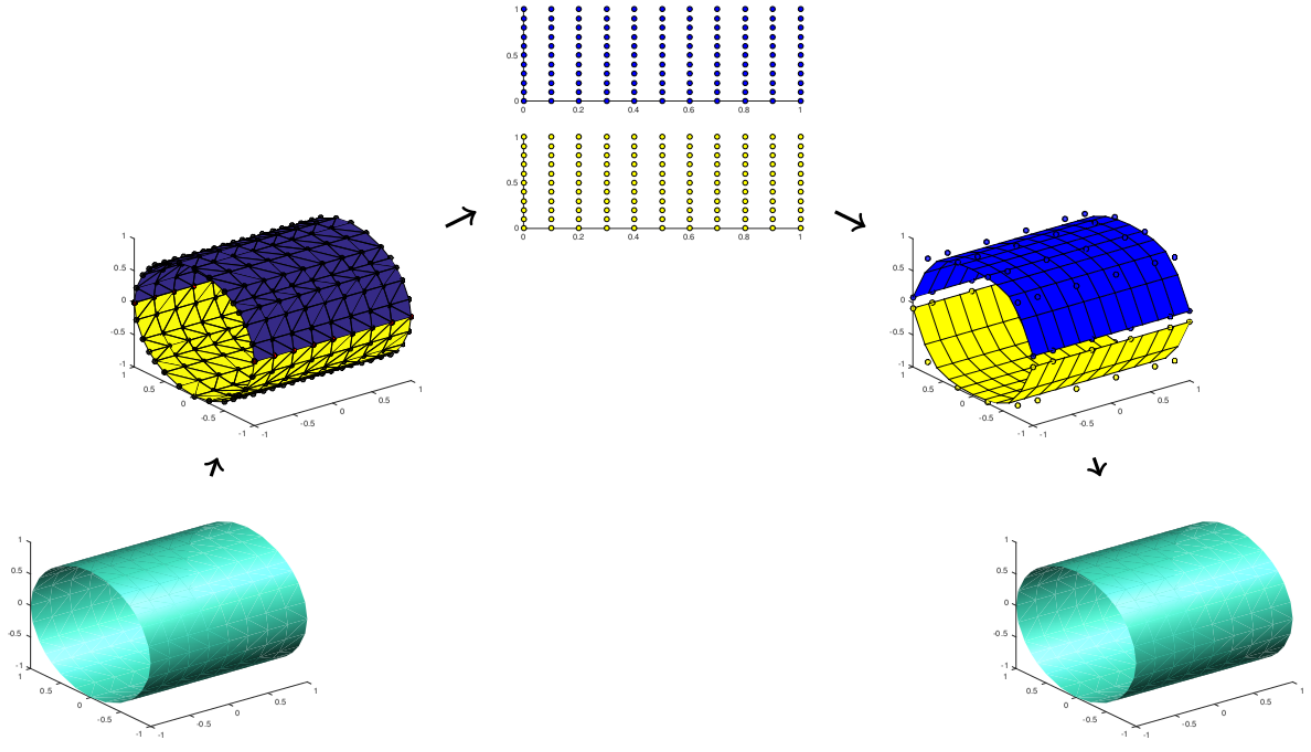
- First part of the pipeline from CAD model to optimized voxel model:
  - ✓ CAD to STL with e.g. FreeCAD
  - ✓ STL to Voxels with CVMLCPP
  - ✓ Voxels to ToPy input with custom script
  - ✓ Topology optimized geometry with ToPy
  - 🕒 Surface reconstruction with VTKToolbox
- B–spline fitting
  - ✗ Automatic patch selection
  - ✗ Parametrization of obtained patches
  - ✓ B–spline fitting using least squares
  - 🕒 Smooth connection of patches
  - ✗ Conversion back to CAD

What is next?

- Automation of the first part of the pipeline
- Integration of boundary conditions handling
- Implementation of remaining B–spline fitting steps (based on work of M.Eck & H.Hoppe)
- Further research on algorithms considering voxel geometry

Thank you for your attention!





Literature

- **William Hunter** Predominantly solid-void three-dimensional topology optimisation using open source software
- **Gerrit Becker, Michael Schäfer, Antony Jameson.** ”An advanced NURBS fitting procedure for post-processing of grid-based shape optimizations”
- **Matthias Eck, Hugues Hoppe.** ”Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type”