Technische Universität München

BGCE Project: CAD – Integrated Topology Optimization

BGCE First Milestone Meeting

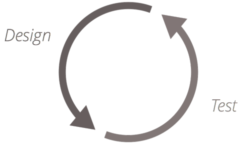
S. Joshi, *J.C. Medina*, *F. Menhorn*, S. Reiz, B. Rüth, E. Wannerberg, *A. Yurova*

November 1, 2015

Contents

Motivation

Current Design Process:



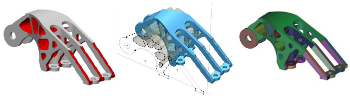
- Iterative and redundant
- Time consuming

Motivation

Current Design Process:

- Iterative and redundant
- Time consuming

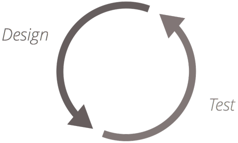
Topology optimization



- Promoted by additive manufacturing

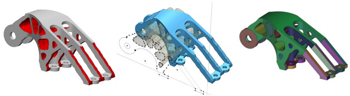
Motivation

Current Design Process:



- Iterative and redundant
- Time consuming

Topology optimization



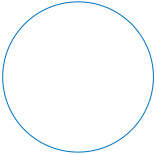
- Promoted by additive manufacturing

Focus:

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

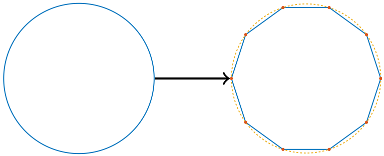
Workflow Overview

CAD design



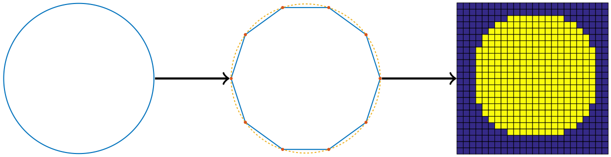
Workflow Overview

STL interface



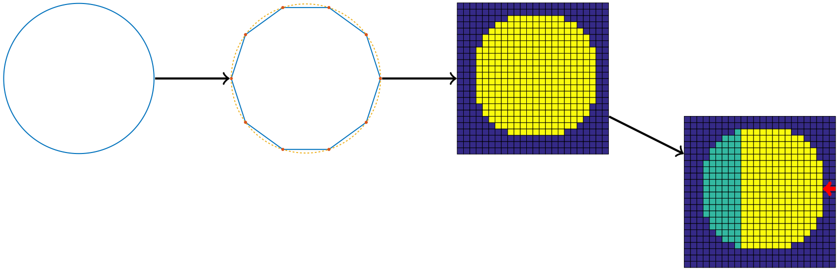
Workflow Overview

Voxelized topology



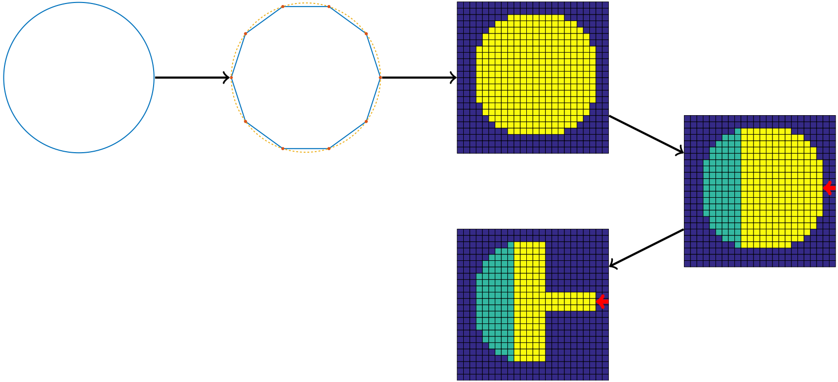
Workflow Overview

Specification of loads and fixtures



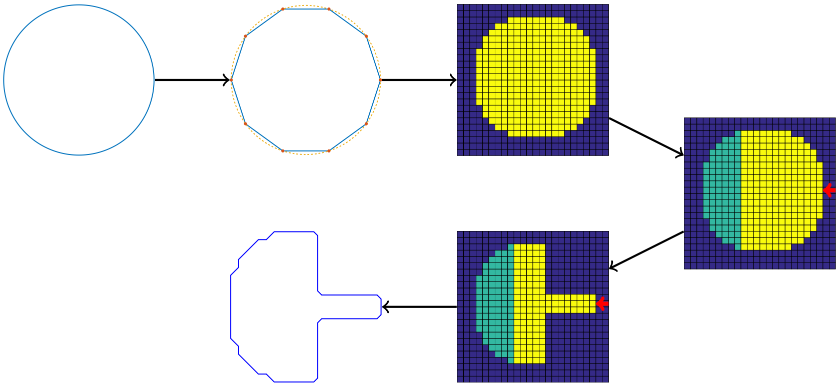
Workflow Overview

Optimized topology



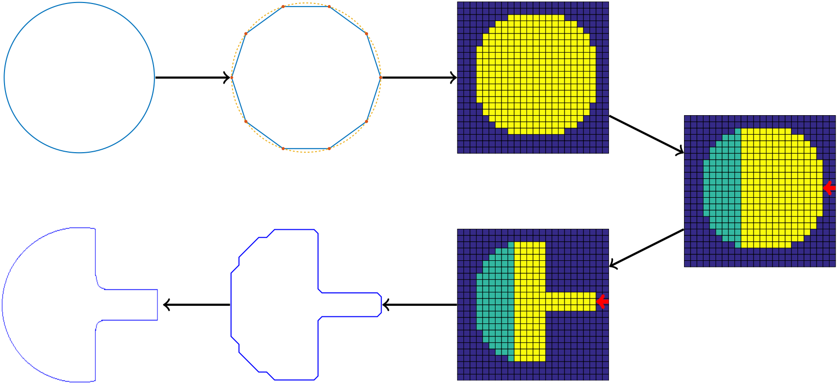
Workflow Overview

Surface extraction

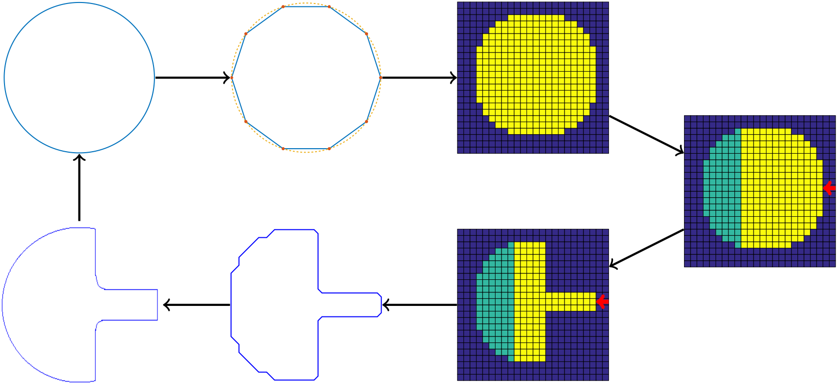


Workflow Overview

Parametrized CAD-geometries

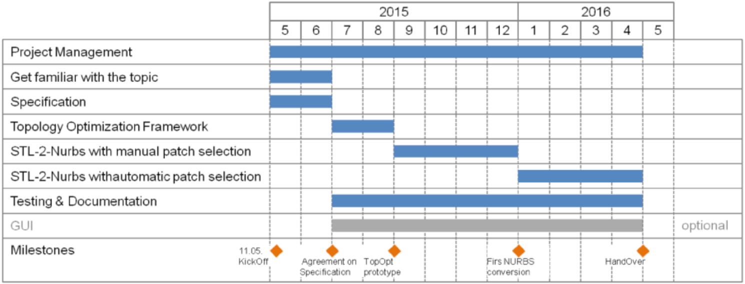


Workflow Overview



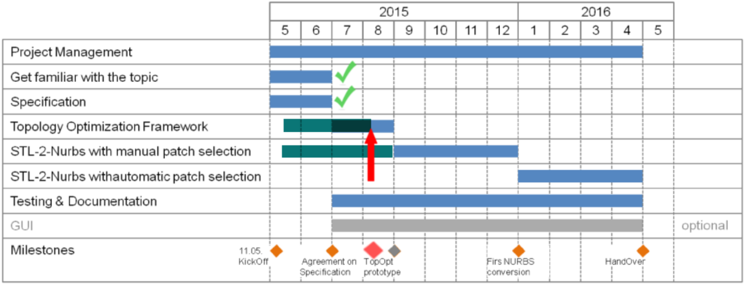
Schedule & Milestones

Schedule:

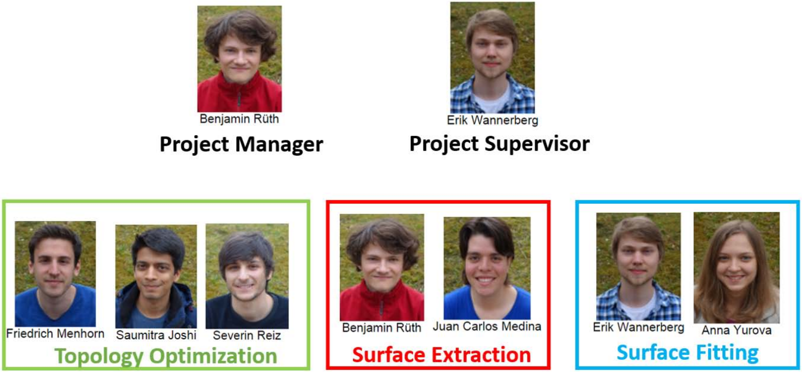


Schedule & Milestones

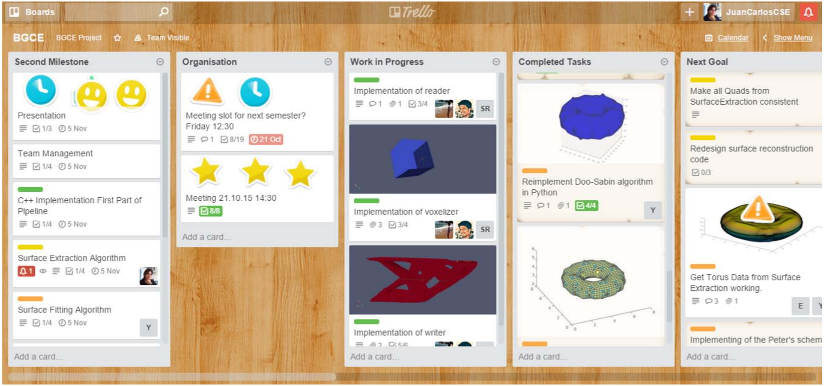
Schedule: (current)



Divide and Conquer



Project management



Contents

Status DRAFT

Last milestone

- ✓ Manual voxelization using CVMLCPP
- ✓ ”Hard coded” script for ToPy input
- ✓ Topology optimized geometry using ToPy
- ✗ Recognition of boundary conditions

Today

- ✓ Voxelization with OpenCascade
- ✓ Extraction of loads, fixtures and active elements through colouring
- ✓ Automatic ”one click” pipeline to surface reconstruction

The user's view DRAFT

- Model geometry in favorite CAD tool (FreeCAD, OpenSCAD)
- Colour faces where boundary conditions are applied

  Red Fixture

  Green Active

  RGB RGB value in $[0 \leq R < 255, 0 \leq G < 255, 0 \leq B < 255]$ for load vector
- Save model as *STEP with Colours* and *IGES with Colours*
- Run ***NAME*** *filename force_scaling*

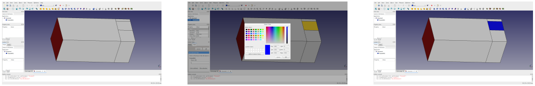
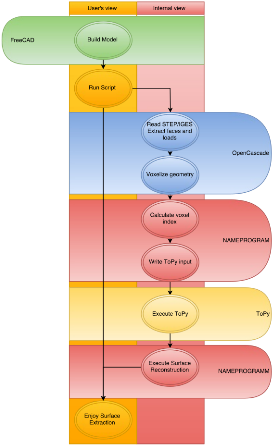
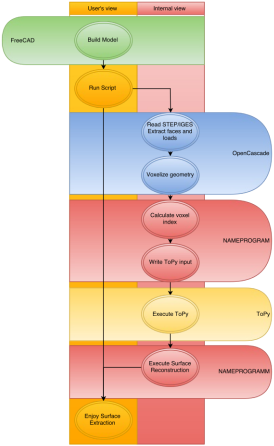


Figure : Color faces in FreeCAD



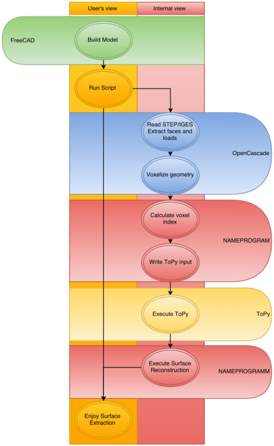
The internal view DRAFT

- The pipeline:
  1. Read STEP and IGES file, extract colours and faces
  2. Voxelize faces using OpenCascade
  3. Calculate index for each voxel for ToPy
  4. Write ToPy input file
  5. Execute ToPy on the input file
  6. Execute Surface Reconstruction on ToPy vtk output



The internal view DRAFT

- The pipeline:
  1. Read STEP and IGES file, extract colours and faces
     - STEP file holding the colours
     - IGES holding the structure



The internal view DRAFT

- The pipeline:
  1. Read STEP and IGES file, extract colours and faces
  2. Voxelize faces using OpenCascade
     - Included open cascade voxelizer

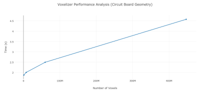
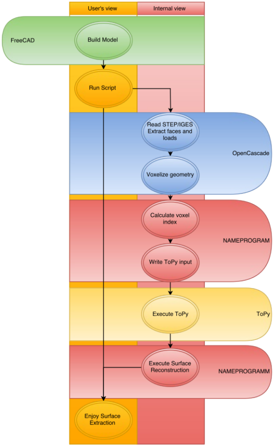


Figure : Scaling of voxelizer

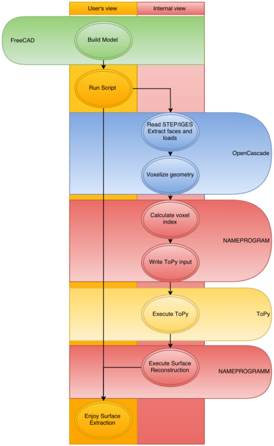


The internal view DRAFT

- The pipeline:
  1. Read STEP and IGES file, extract colours and faces
  2. Voxelize faces using OpenCascade
  3. Calculate index for each voxel for ToPy
     - Different indexing for elements and nodes in ToPy

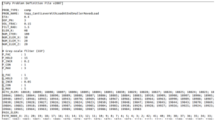
```
# ==========================================
# === Discretisation of the design domain ===
# ==========================================
# 2D: Y              3D: Y
#     |                  |
#     +---X              +---X
#                       /
#                      Z
#
# 1---5---9
# | 1 | 5 |
# 2---6---10
# | 2 | 6 |
# 3---7---11
# | 3 | 7 |
# 4---8---12
#
```

Figure : Indexing in ToPy



The internal view DRAFT

- The pipeline:
  1. Read STEP and IGES file, extract colours and faces
  2. Voxelize faces using OpenCascade
  3. Calculate index for each voxel for ToPy
  4. Write ToPy input file
     - Each voxelindex is specifically written



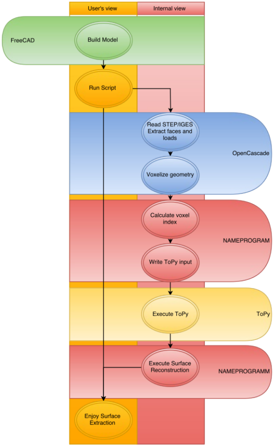
```
[ToPy Problem Definition File v2007]

PROB_TYPE:   comp
PROB_NAME:   topy_CantileverWithLoadAtEndSmallerMovedLoad
ETA:         0.4
DOF_PN:      3
VOL_FRAC:    0.15
FILT_RAD:    1.5
ELEM_K:      H8
NUM_ITER:    100
NUM_ELEM_X:  50
NUM_ELEM_Y:  20
NUM_ELEM_Z:  20

# Grey-scale filter (GSF)
P_FAC      : 1
P_HOLD     : 15
P_INCR     : 0.2
P_CON      : 1
P_MAX      : 3

Q_FAC      : 1
Q_HOLD     : 15
Q_INCR     : 0.05
Q_CON      : 1
Q_MAX      : 5
ACTV_ELEM: 18810; 18809; 18808; 18807; 18806; 18805; 18804; 18803; 18830; 18829; 18828; 18827; 18826; 18825; 18824; 18823; 18
18866; 18865; 18864; 18863; 18890; 18889; 18888; 18887; 18886; 18885; 18884; 18883; 18910; 18909; 18908; 18907; 18906; 18905;
18948; 18947; 18946; 18945; 18944; 18943; 18970; 18969; 18968; 18967; 18966; 18965; 18964; 18963; 18990; 18989; 18988; 18987;
19830; 19829; 19828; 19827; 19826; 19825; 19824; 19823; 19850; 19849; 19848; 19847; 19846; 19845; 19844; 19843; 19870; 19869;
19884; 19883; 19910; 19909; 19908; 19907; 19906; 19905; 19904; 19903; 19930; 19929; 19928; 19927; 19926; 19925; 19924; 19923;
19966; 19965; 19964; 19963; 19990; 19989; 19988; 19987; 19986; 19985; 19984; 19983
PASV_ELEM:
FXTR_NODE_X: 21; 20; 19; 18; 17; 16; 15; 14; 13; 12; 11; 10; 9; 8; 7; 6; 5; 4; 3; 2; 42; 41; 40; 39; 38; 37; 36; 35; 34; 33;
```

Figure : Script for ToPy



The internal view DRAFT

- The pipeline:
  1. Read STEP and IGES file, extract colours and faces
  2. Voxelize faces using OpenCascade
  3. Calculate index for each voxel for ToPy
  4. Write ToPy input file
  5. Execute ToPy on the input file
     - Topy runs....

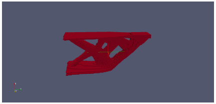
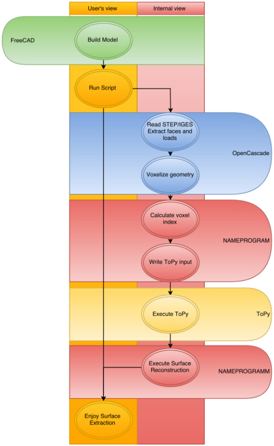


Figure : ToPy Output



The internal view DRAFT

- The pipeline:
  1. Read STEP and IGES file, extract colours and faces
  2. Voxelize faces using OpenCascade
  3. Calculate index for each voxel for ToPy
  4. Write ToPy input file
  5. Execute ToPy on the input file
  6. Execute Surface Reconstruction on ToPy vtk output
     - Running dual contouring algorithm

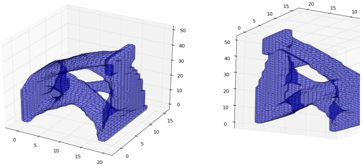
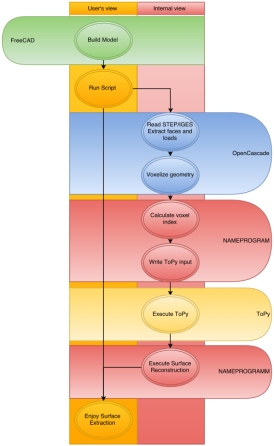


Figure : Surface extraction for Cantilever



The next steps MOVE TO LATER

- GUI for input
- Speed up ToPY
- Usage of different optimizers

Contents

Status

Last milestone

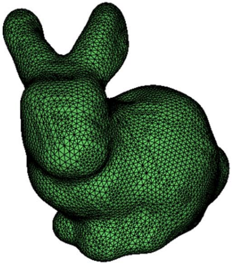
- Surface reconstruction with the VTK Toolbox

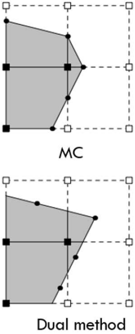
Today

- ✓ Extraction of voxel data from Topy
- ✓ 3D Dual Contouring program
- ✓ Coarsening and non-manifold edge treatment
- ✓ Projection to quads and respective parametrization
- Interface to NURBs

From Voxel to Mesh Geometry

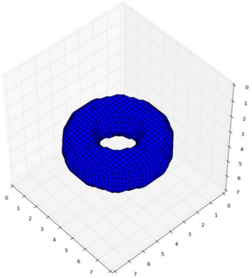
- Extract isosurface from voxel information
- Algorithms: Marching Cubes, Dual Contouring, Extended Models
- Problems with VTK's Marching Cube implementation

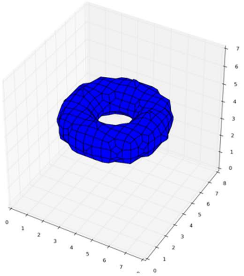




Dual Contouring

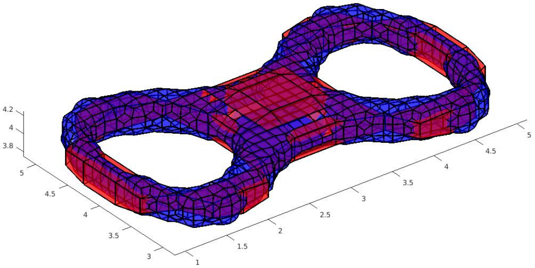
- Python implementation- Use of powerful libraries, including VTK
- Output: Closed surface made out of *quads*
- Coarsening is needed for surface fitting's algorithms





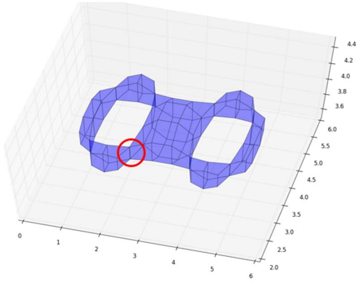
Dual Contouring

- Python implementation- Use of powerful libraries, including VTK
- Output: Closed surface made out of *quads*
- Coarsening is needed for surface fitting's algorithms



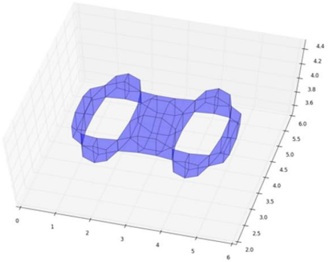
Dual Contouring- Problems

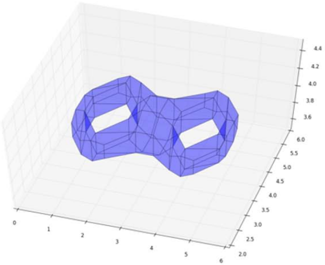
- **Non-manifold edges** appear
- One edge can only belong to two quads for the surface to be closed
- Special treatments in the implementation to avoid them



Dual Contouring- Problems

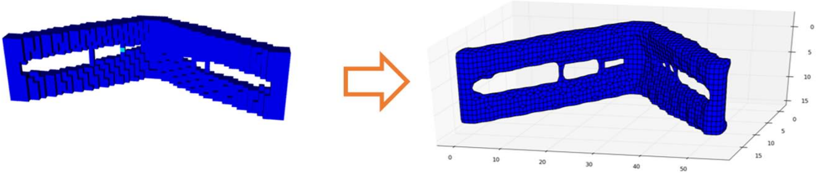
- **Non-manifold edges** appear
- One edge can only belong to two quads for the surface to be closed
- Special treatments in the implementation to avoid them





Dual Contouring- Input

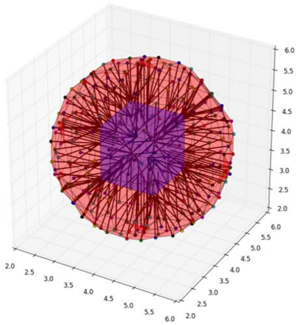
- Sixth step of the DRAFT pipeline- Interface between Topology Optimization and Surface Extraction
- Special implementation to use voxel data from Topy as input

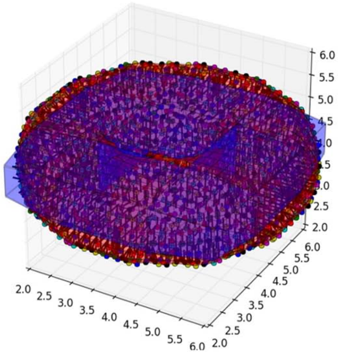


Demo

Projection and Parametrization

- Points from finer grid are projected to quads of the coarser grid
- Parameters u and v are found for each quad
- This information is needed for the algorithms in the last part of the pipeline





Contents

Status

Last milestone

- ✗ Automatic patch selection
- ✗ Parametrization of obtained patches
- ✓ B–spline fitting using least squares
- ◷ Smooth connection of patches
- ✗ Conversion back to CAD

Today

- ✓ Automatic patch selection – moved to the surface extraction part
- ✓ Parametrization of obtained patches – moved to the surface extraction part
- ✓ B–spline fitting using least squares
- ✓ Smooth connection of patches
- ✗ Conversion back to CAD

Long way to smoothness

Peters' scheme:

Given the control mesh M_x

1. Refine the *control mesh* 2 times using Doo-Sabin refinement
2. Construct a tensor product Bezier patches (biquadratic or bicubic) centred on the each vertex of the refined *control mesh*

According to Peters obtained surface is G^1 smooth

Add pictures for Doo-Sabin and for fitting

Long way to smoothness

Main ideas

- Use the mesh obtained from Dual Contouring as a *control mesh*
- Modify the fitting step to take advantage of the **Peters' scheme**

$$\downarrow$$

$$E_{dist}(V_x) = \sum_{i=1}^{N} \| p_i - y_i V_x \|_2^2 \rightarrow min, \tag{1}$$

y_i - coefficients obtained from the Peters' scheme theory.

What is achieved?

- Smoothness of the fitted surface is now guaranteed by construction
- Fitting is possible for more complex shapes achieved by using an information from the Dual Contouring algorithm

Improved pipeline

Insert a nice new pipeline in tikz: orig points $->$ dual contouring $->$ clouds $->$ Doo-Sabin $->$ fitted surface together with clouds

Before and after

May be some pictures with a really easy shape, which we were able to fit last time and it was not smooth and the ones with new fancy smooth shapes

What is next?

- Finishing of the implementation of the improved pipeline in Python
- Full integration with Surface Extraction part
- Introducing of the *fairness functional* in order to deal with more complex shapes
- Implementation of the *adaptive refinement* in order to control a maximum error tolerance
- Implementation of the *parameter correction* for the improved pipeline
- Exporting the results back to CAD

Contents

What is done?

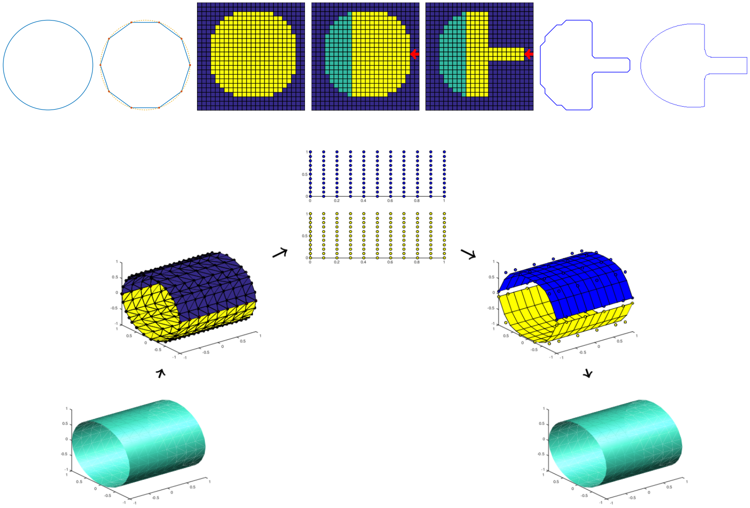
- First part of the pipeline from CAD model to optimized voxel model:
  - ✓ CAD to STL with e.g. FreeCAD
  - ✓ STL to Voxels with CVMLCPP
  - ✓ Voxels to ToPy input with custom script
  - ✓ Topology optimized geometry with ToPy
  - 🕒 Surface reconstruction with VTKToolbox
- B–spline fitting
  - ✗ Automatic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

Contents

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”