Technische Universität München

BGCE Project: CAD – Integrated Topology Optimization

BGCE Final Milestone Meeting

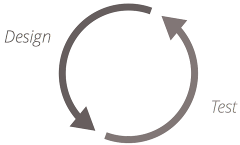
S. Joshi, J.C. Medina, F. Menhorn,
S. Reiz, B. Rüth, E. Wannerberg, A. Yurova

February 26, 2016

Contents

Motivation

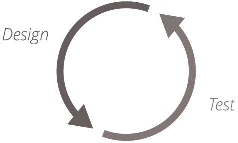
Current Design Process:



- Iterative and redundant
- Time consuming

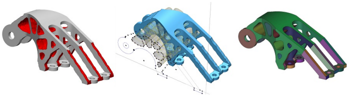
Motivation

Current Design Process:



- Iterative and redundant
- Time consuming

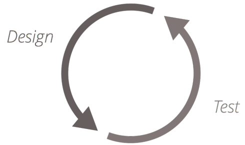
Topology optimization



- Promoted by additive manufacturing

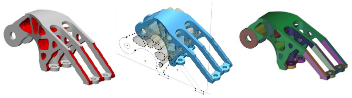
Motivation

Current Design Process:



- Iterative and redundant
- Time consuming

Topology optimization



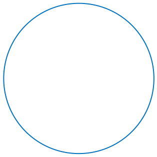
- Promoted by additive manufacturing

Focus:

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

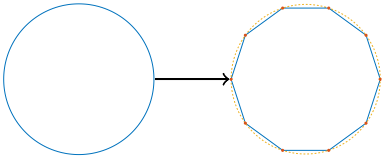
Workflow Overview

CAD design



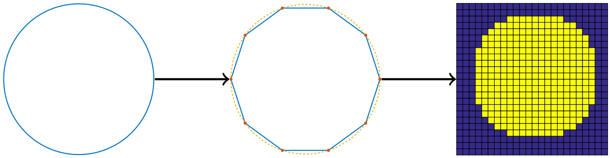
Workflow Overview

STL interface



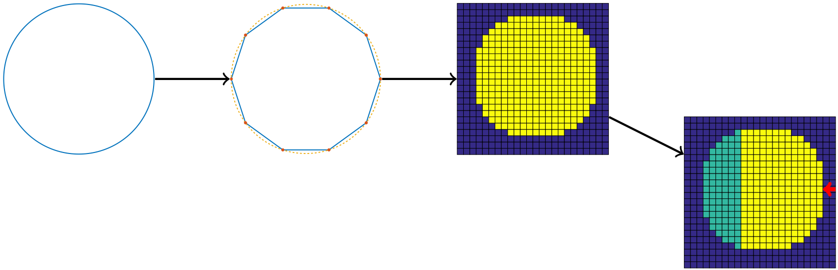
Workflow Overview

Voxelized topology



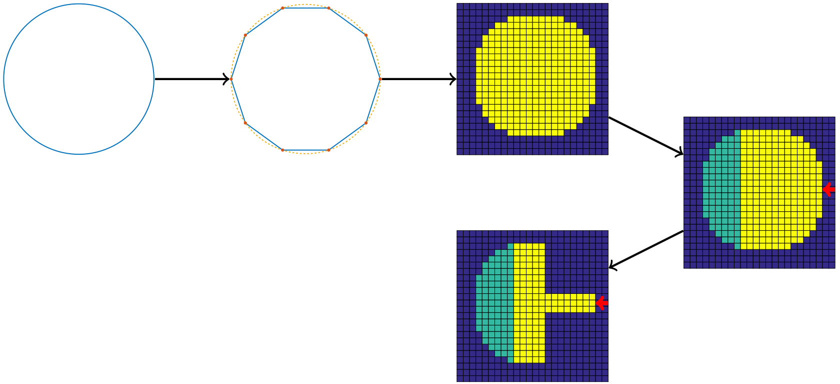
Workflow Overview

Specification of loads and fixtures



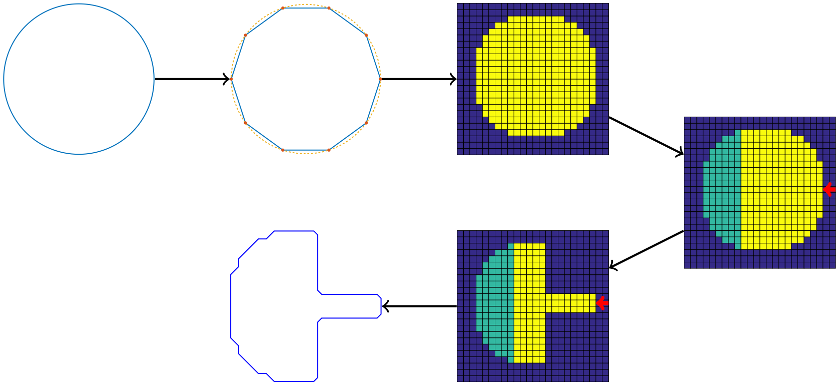
Workflow Overview

Optimized topology



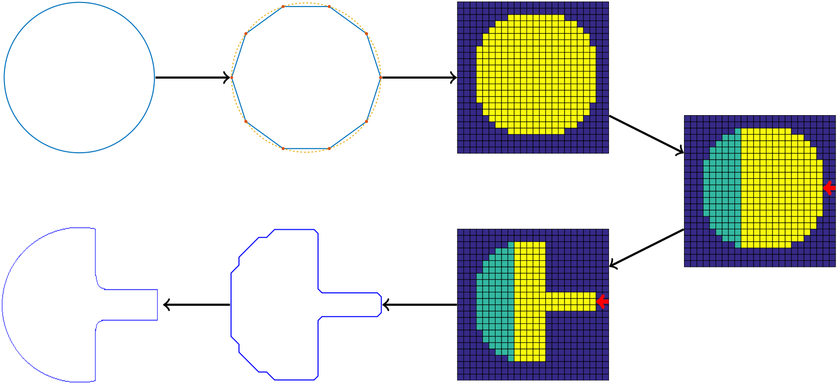
Workflow Overview

Surface extraction



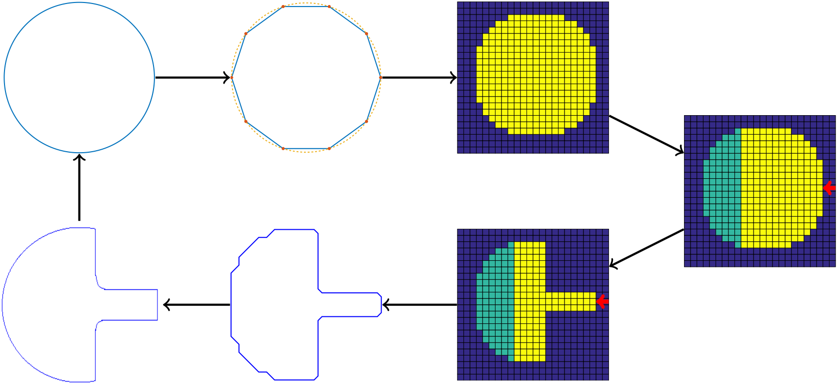
Workflow Overview

Parametrized CAD-geometries



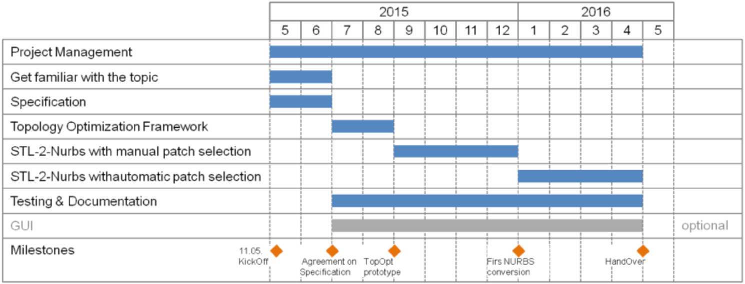
Workflow Overview

Iterative design process



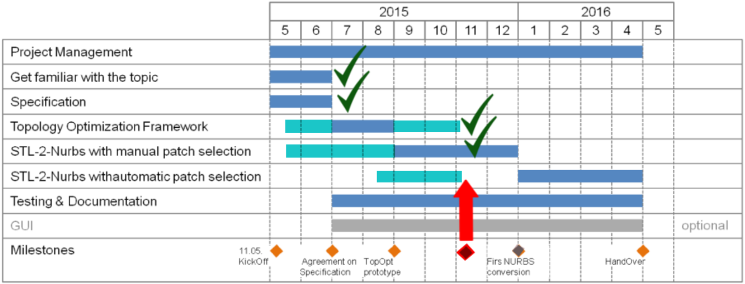
Schedule & Milestones

Schedule:

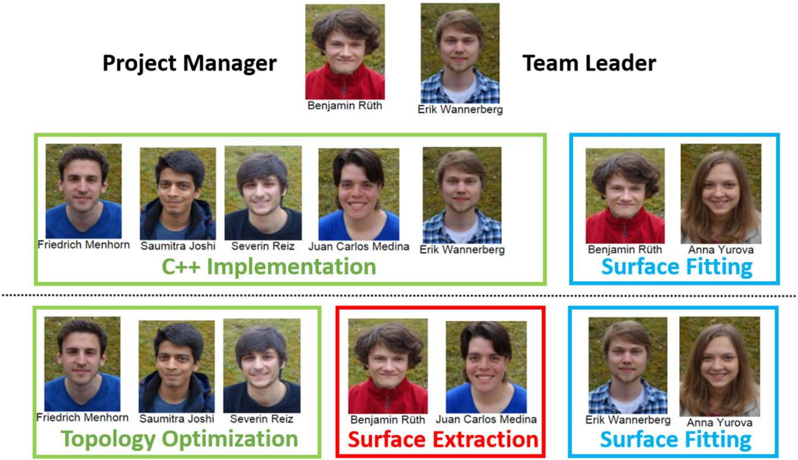


Schedule & Milestones

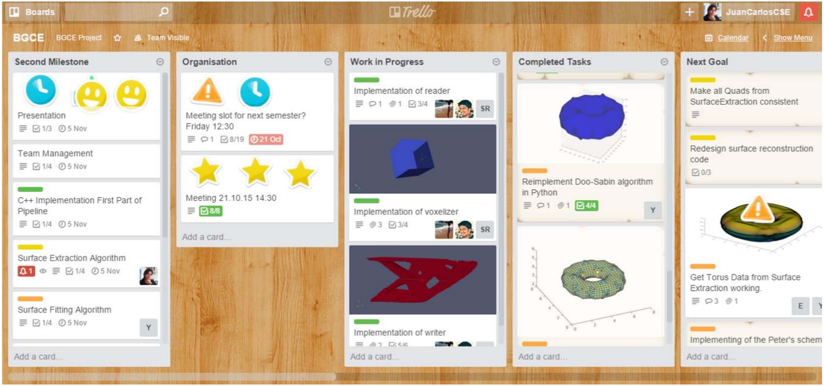
Schedule: (current)



Divide and Conquer



Project management



Contents

Status

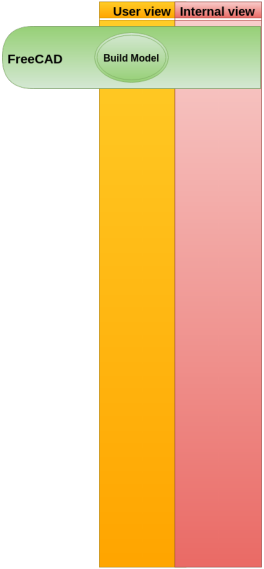
Last milestone

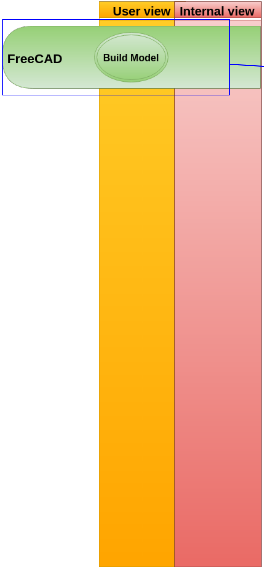
- ✓ Manual voxelization using CVMLCPP
- ✓ ”Hard coded” script for ToPy input
- ✓ Topology optimized geometry using ToPy
- ✗ Recognition of boundary conditions

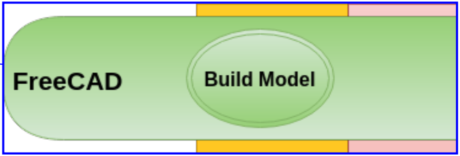
Today

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

User view	Internal view


User view
Internal view
FreeCAD
Build Model

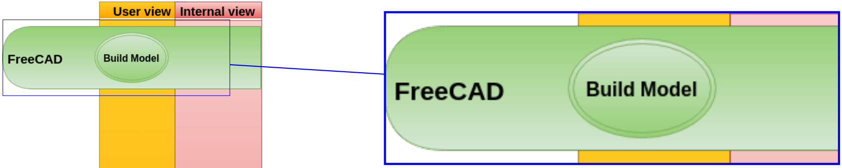




- Model geometry in your favorite CAD tool
- Colour faces for boundary conditions

  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`



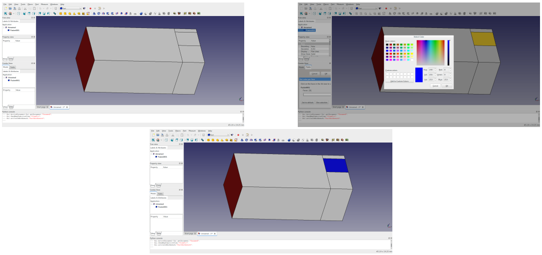
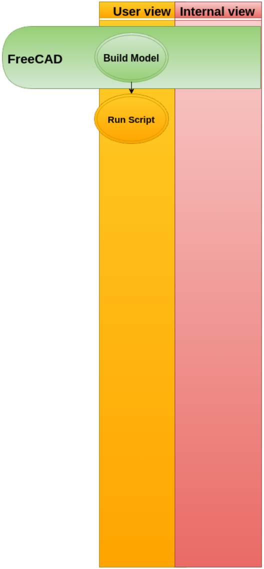
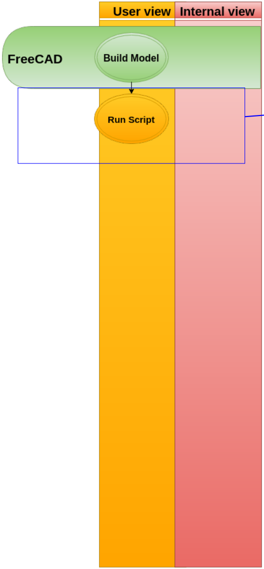
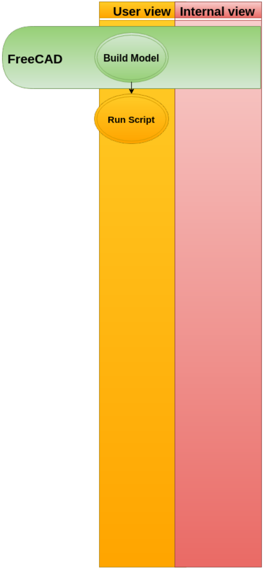


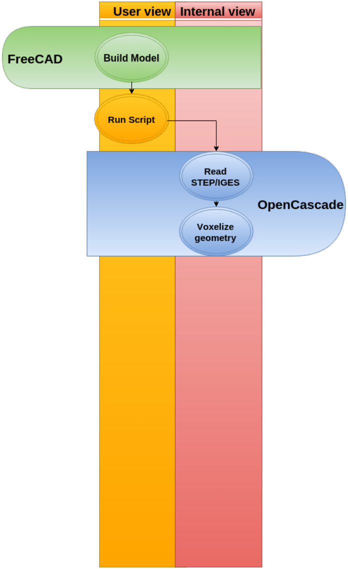
Figure : Color faces in FreeCAD

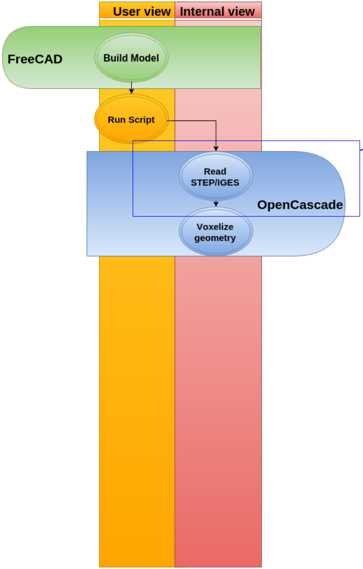

User view
Internal view
FreeCAD
Build Model
Run Script

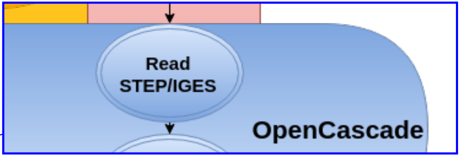


→ Run
`CADTopOpt filedir filename force_scaling refinement`


User view
Internal view
FreeCAD
Build Model
Run Script

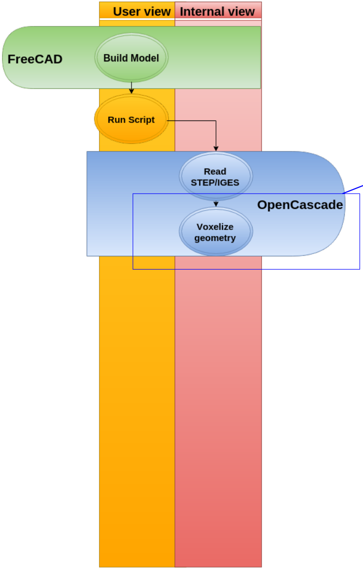

User view
Internal view
FreeCAD
Build Model
Run Script
Read STEP/IGES
Voxelize geometry
OpenCascade

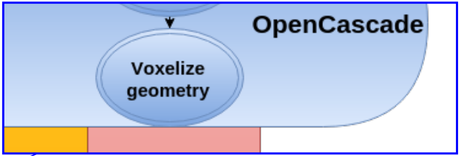




Read STEP and IGES file, extract colours and faces

- STEP holds the colours
- IGES holds the structure





Voxelize faces using OpenCascade

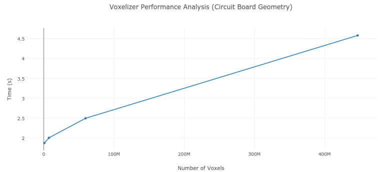
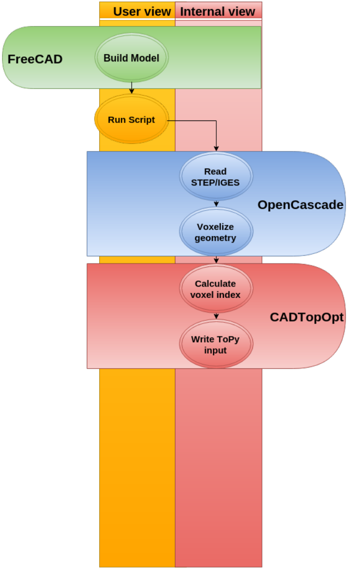
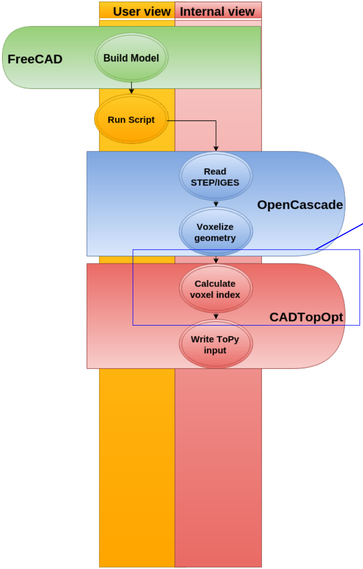
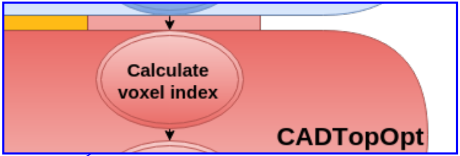


Figure : Scaling of voxelizer


User view
Internal view
FreeCAD
Build Model
Run Script
Read STEP/IGES
OpenCascade
Voxelize geometry
Calculate voxel index
CADTopOpt
Write ToPy input

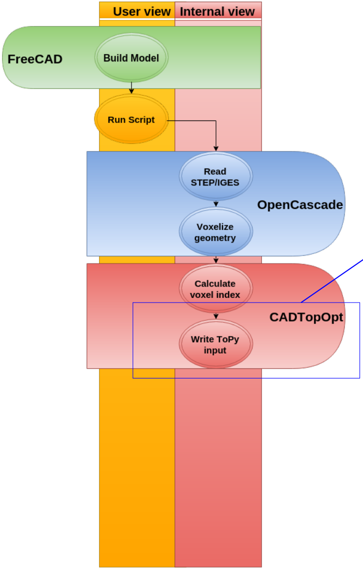


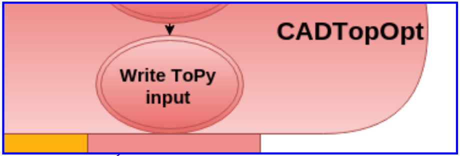


Different indexing for elements and nodes in ToPy

```
# ===========================================
# === 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 [1]

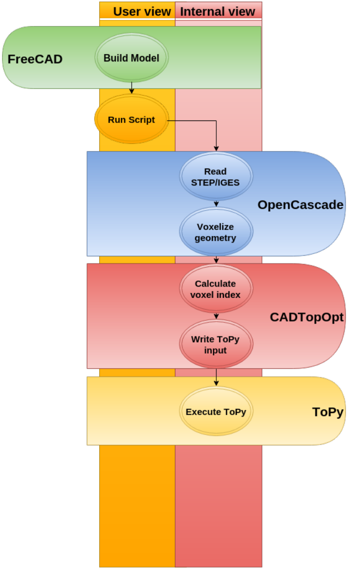


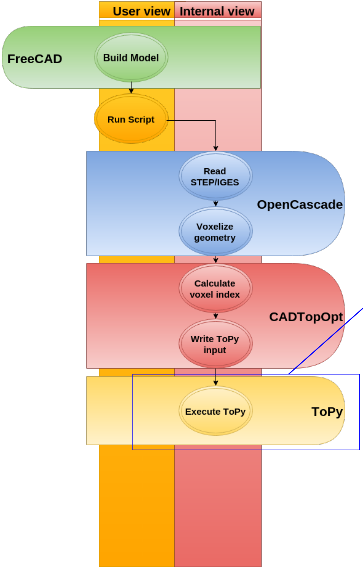


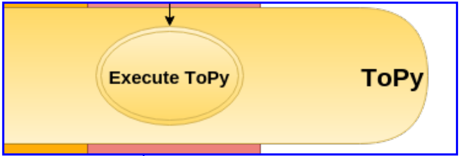
Each voxel index is specifically written

```
[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


User view
Internal view
FreeCAD
Build Model
Run Script
Read STEP/IGES
Voxelize geometry
OpenCascade
Calculate voxel index
Write ToPy input
CADTopOpt
Execute ToPy
ToPy





Execute ToPy on the input file

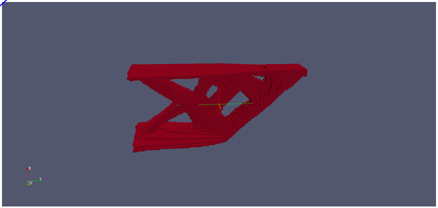
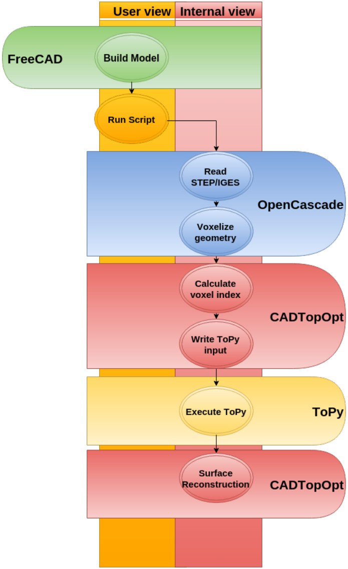
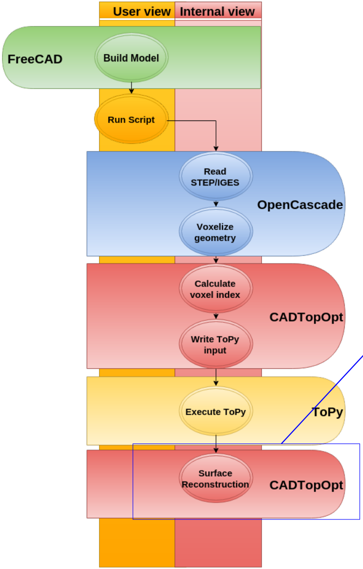
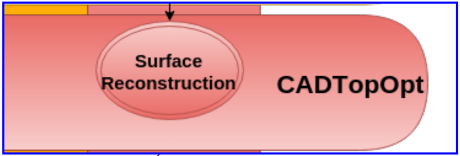


Figure : ToPy Output


User view
Internal view
FreeCAD
Build Model
Run Script
Read STEP/IGES
Voxelize geometry
OpenCascade
Calculate voxel index
Write ToPy input
CADTopOpt
Execute ToPy
ToPy
Surface Reconstruction
CADTopOpt





Run dual contouring algorithm

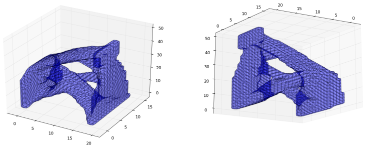
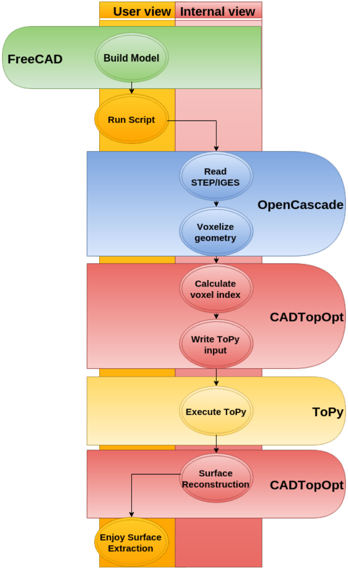
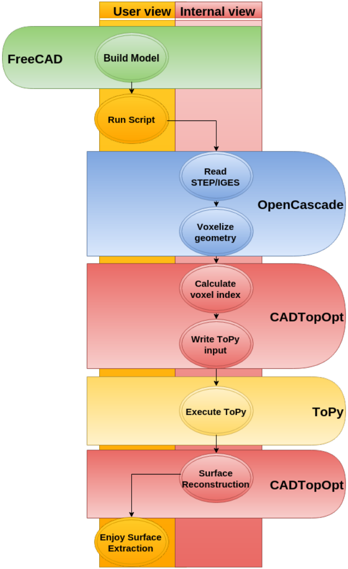
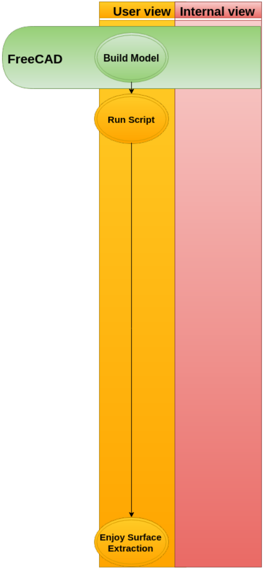


Figure : Surface extraction for Cantilever


User view
Internal view
FreeCAD
Build Model
Run Script
Read STEP/IGES
Voxelize geometry
OpenCascade
Calculate voxel index
Write ToPy input
CADTopOpt
Execute ToPy
ToPy
Surface Reconstruction
CADTopOpt
Enjoy Surface Extraction



But what does the user see?



But what does the user see?

Contents

Status

Last milestone

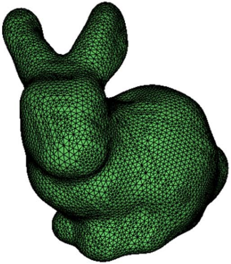
- ⏱ Surface reconstruction with the VTK Toolbox

Today

- ✓ Extraction of voxel data from Topy
- ✓ 3D Dual Contouring implementation
- ✓ Coarsening and non-manifold edge treatment
- ✓ Projection of datapoints onto 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



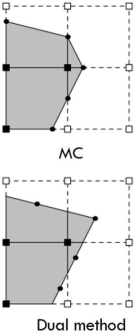
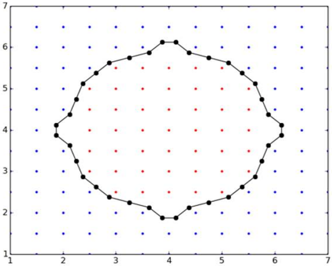
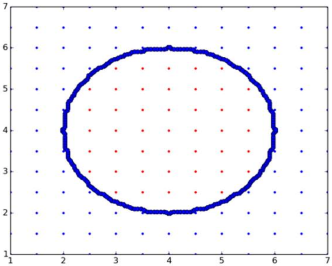


Figure : From [4],[5]

Dual Contouring

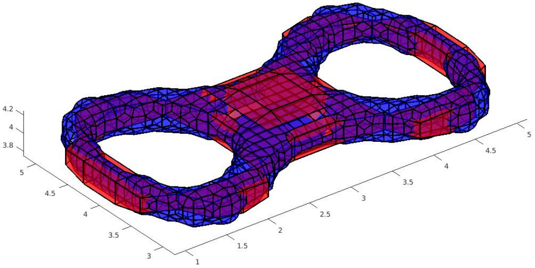
- Python implementation — Use of powerful libraries, including VTK
- Output: Closed surface made out of *quads*
- Coarsening is needed for surface fitting algorithms





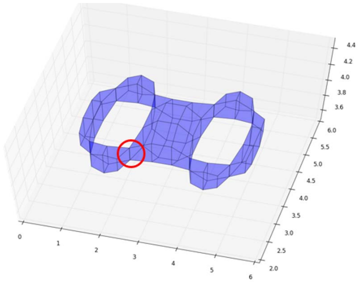
Dual Contouring

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



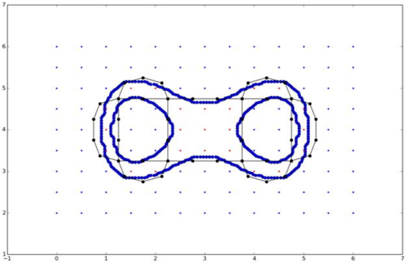
Dual Contouring — Problems

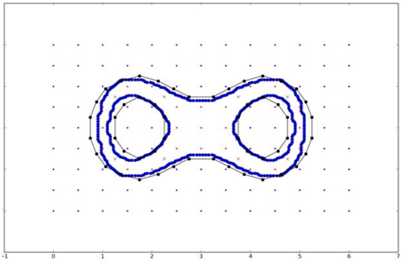
- **Non–manifold edges** appear
- One edge can only belong to two quads for the surface to be closed
- Special treatments in the implementation to avoid them



Dual Contouring — Problems

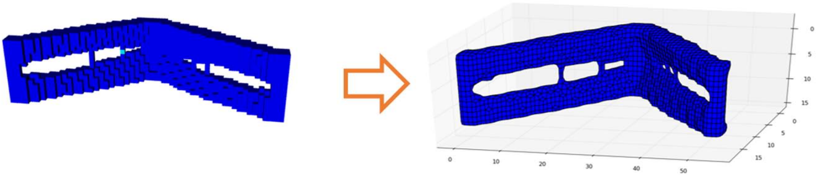
- **Non–manifold edges** appear
- One edge can only belong to two quads for the surface to be closed
- Special treatments in the implementation to avoid them





Dual Contouring — Input

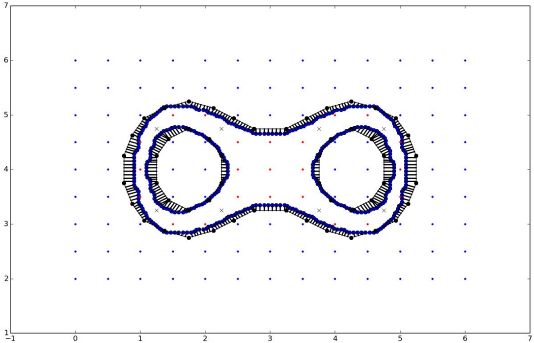
- Interface between Topology Optimization and Surface Extraction
- Special implementation to use voxel data from ToPy as input



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

B–Spline

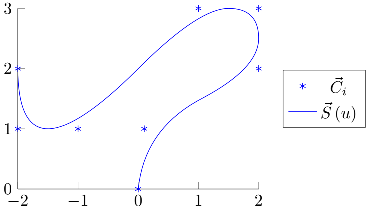
$$\vec{S}\,(u,v) = \sum_{i,j=1}^{n,m} \vec{C}_{i,j} N_i^p\,(u)\,N_j^p\,(v)\,,$$

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

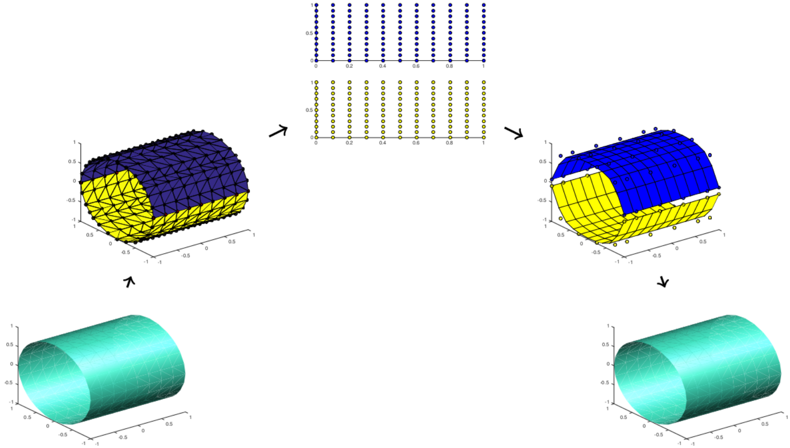
B–Splines

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

Engineers are working with CAD



B–Spline Fitting Pipeline [2]



Status

Last milestone

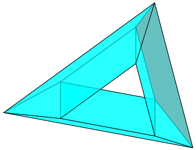
- ✗ Automatic patch selection
- ✗ Parametrization of obtained patches
- ✓ B–spline fitting using least squares
- 🕐 Smooth connection of patches
- ✗ Conversion back to CAD

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 – modified
- ✓ Smooth connection of patches
- ✗ Conversion back to CAD

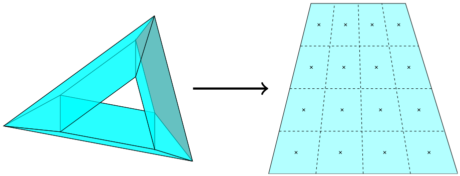
Long way to smoothness – Peter's scheme

Control mesh



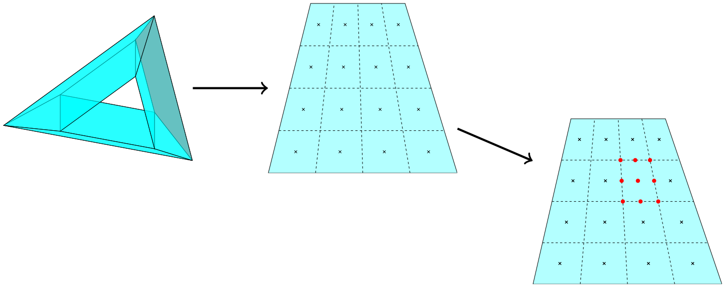
Long way to smoothness – Peter's scheme

Refined control mesh



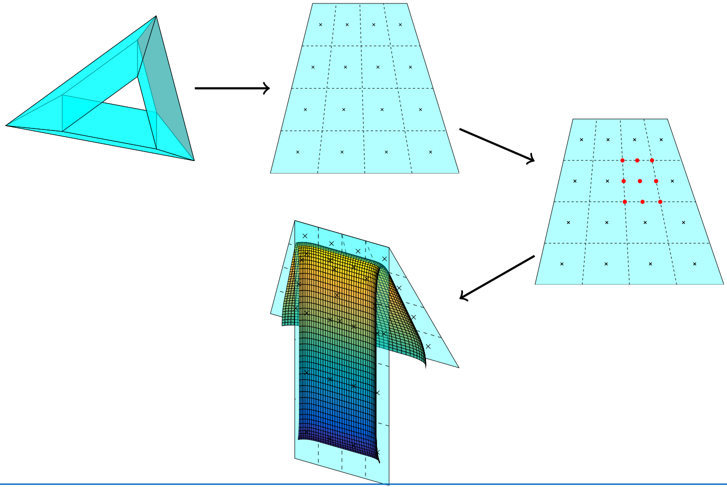
Long way to smoothness – Peter's scheme

Bezier control points



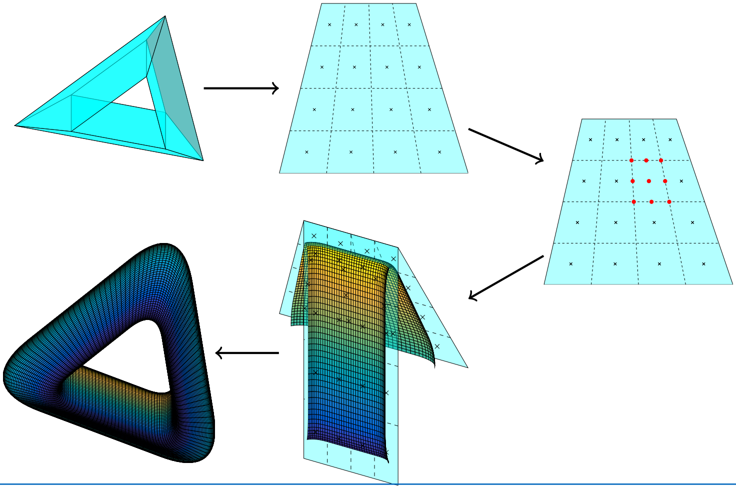
Long way to smoothness – Peter's scheme

B-Spline patch



Long way to smoothness – Peter's scheme

Peters' surface



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**

$$\downarrow$$

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

y_i - coefficients obtained from the Peters' scheme theory.

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**

$$\downarrow$$

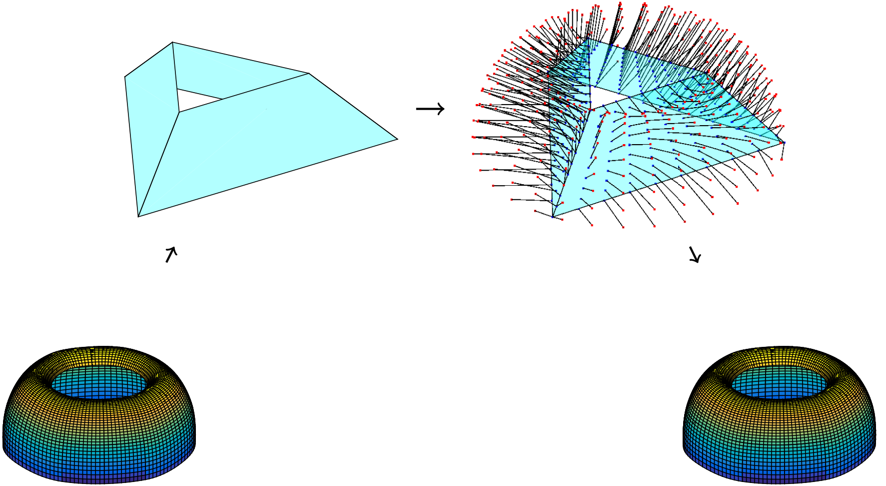
$$E_{dist}(V_x) = \sum_{i=1}^{N} \| P_i - y_i V_x \|_2^2 \rightarrow min,$$

y_i - coefficients obtained from the Peters' scheme theory.

What is achieved?

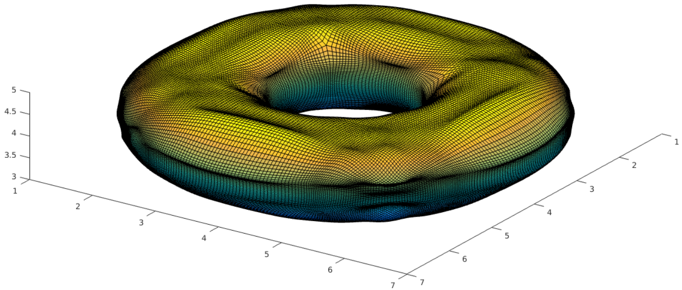
- Smoothness of the fitted surface is now guaranteed by construction
- Fitting of more complex shapes achieved

Improved pipeline[3]



Possible optimizations

- Introduction 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



Contents

What is done? What is next?

- Topology Optimization
  - ✓ Pipeline from CAD model to optimized voxel model
  - ✓ User input of boundary conditions
  - 🕐 Support for complex geometries
  - ✗ GUI for user interaction

What is done? What is next?

- Topology Optimization
  - ✓ Pipeline from CAD model to optimized voxel model
  - ✓ User input of boundary conditions
  - ◷ Support for complex geometries
  - ✗ GUI for user interaction
- Surface Extraction
  - ✓ Dual Contouring for simple geometries
  - ✓ Provide necessary data for Surface Fitting
  - ◷ Interfaces
  - ✗ Adaptive and topology safe Dual Contouring

What is done? What is next?

- Topology Optimization
  - ✓ Pipeline from CAD model to optimized voxel model
  - ✓ User input of boundary conditions
  - 🕒 Support for complex geometries
  - ✗ GUI for user interaction
- Surface Extraction
  - ✓ Dual Contouring for simple geometries
  - ✓ Provide necessary data for Surface Fitting
  - 🕒 Interfaces
  - ✗ Adaptive and topology safe Dual Contouring
- Surface Fitting
  - ✓ B–spline fitting using least squares
  - ✓ Smooth connection of patches using Peters' scheme
  - ✗ Conversion back to CAD

Remaining questions

Python

- ⊖ First part of the pipeline is in C++
- ⊕ Second part of the pipeline is now in Python
- ⊕ Easy to port from the original MATLAB prototypes

C++

- ⊕ First part of the pipeline is in C++
- ⊖ Second part of the pipeline is now in Python
- ⊖ Cumbersome to implement

Remaining questions

Python

- ⊖ First part of the pipeline is in C++
- ⊕ Second part of the pipeline is now in Python
- ⊕ Easy to port from the original MATLAB prototypes

C++

- ⊕ First part of the pipeline is in C++
- ⊖ Second part of the pipeline is now in Python
- ⊖ Cumbersome to implement

ToPy Problem

- ⊕ Current implementation is using ToPy

Remaining questions

Python

- ⊖ First part of the pipeline is in C++
- ⊕ Second part of the pipeline is now in Python
- ⊕ Easy to port from the original MATLAB prototypes

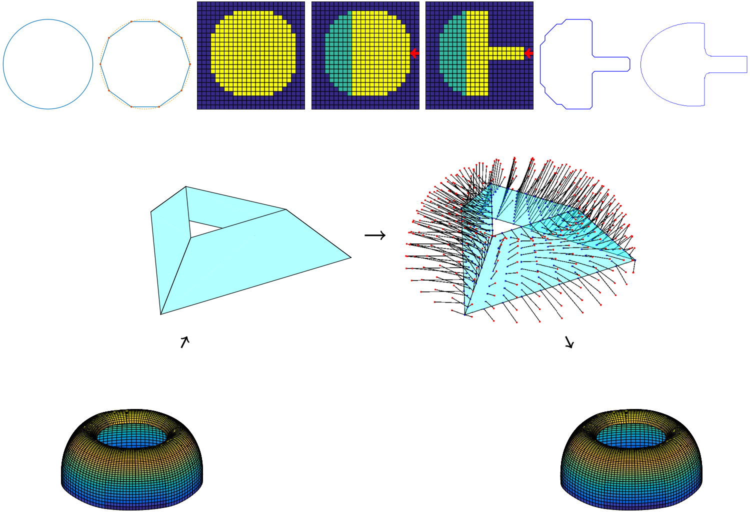
C++

- ⊕ First part of the pipeline is in C++
- ⊖ Second part of the pipeline is now in Python
- ⊖ Cumbersome to implement

ToPy Problem

- ⊕ Current implementation is using ToPy
- ⊖ ToPy is not available any more!

Thank you for your attention!



Literature

1. **William Hunter.** ”Predominantly solid-void three-dimensional topology optimisation using open source software”
2. **Gerrit Becker, Michael Schäfer, Antony Jameson.** ”An advanced NURBS fitting procedure for post-processing of grid-based shape optimizations”
3. **Matthias Eck, Hugues Hoppe.** ”Automatic Reconstruction of B-Spline Surfaces of Arbitrary Topological Type”
4. **Greg Turk, Marc Levoy** ”Stanford Bunny”
5. **Tao Ju, Frank Losasso, Scott Schaefer, Joe Warren.** ”Dual contouring of hermite data”

Projection and Parametrization on arbitrary quads

1. find least squares plane approximating quad

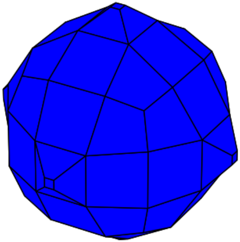


Figure : DC sphere

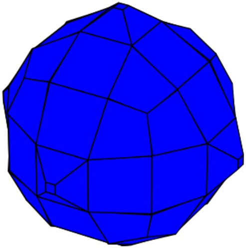
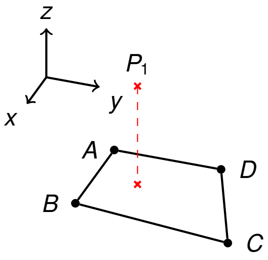


Figure : with plane quads

Projection and Parametrization on arbitrary quads

1. find least squares plane approximating quad
2. projection of datapoint onto plane



Coordinate transformation

system with basis

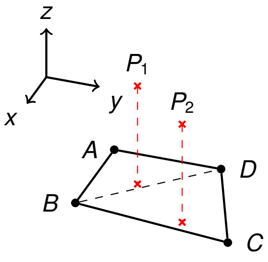
$$B_{BAD} = \begin{pmatrix} \vec{n} & \vec{AB} & \vec{AD} \end{pmatrix}$$

yields

$$(B_{BAD})^{-1} P_1 = \begin{pmatrix} d & u & v \end{pmatrix}^T$$

Projection and Parametrization on arbitrary quads

1. find least squares plane approximating quad
2. projection of datapoint onto plane
3. find corresponding parameters $[u, v] \in [0, 1]^2$



Problem:

✓ for P_1: $(u, v) = (0.5, 0.4)$

✗ for P_2: $(u, v) = (1, 1)$

Solution:

1. if we get $u + v > 1$
2. use B_{BCD} instead of B_{BAD}
3. set $u = 1 - u$, $v = 1 - v$