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

March 1, 2016

# Contents

# CAD issues

## Problems:

- Engineering-design processes are a pendulum!
- Topology-Optimization algorithms are a one-way street!

## Desired:

$\Rightarrow$ One-click optimization

$\Rightarrow$ Full-circle optimization process

# Features

## Fully integrated design process

- CAD to CAD
- Turnkey
- Standardized I/O

## Control to the user

- Resolution
- Smoothness
- Localized Optimization

## 100% open source

# Power to the user

## Geometry

- Original geometry
  - Load(s)
  - Fixture(s)
- Modifiable region(s)
- Non-modifiable region(s)

# Power to the user

## Geometry

- Original geometry
  - Load(s)
  - Fixture(s)
- Modifiable region(s)
- Non-modifiable region(s)

## Solution Parameters

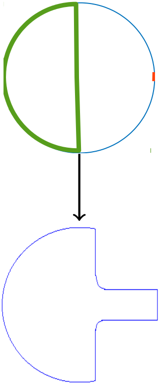
- Optimization accuracy
- Level of surface approximation
- Surface smoothness

# DEMO

# Scalability and Performance

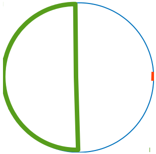
# Overview: Workflow

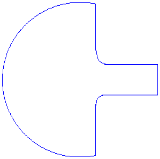
What the user sees:



# Overview: Workflow

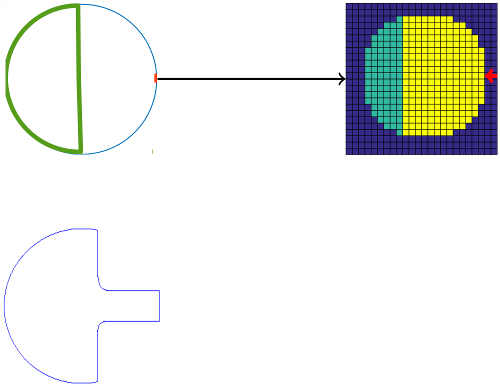
CAD design including specification of loads and fixtures





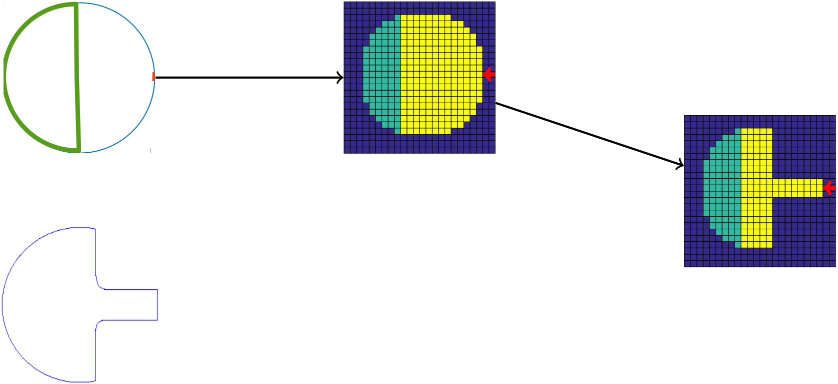
# Overview: Workflow

Voxelized topology



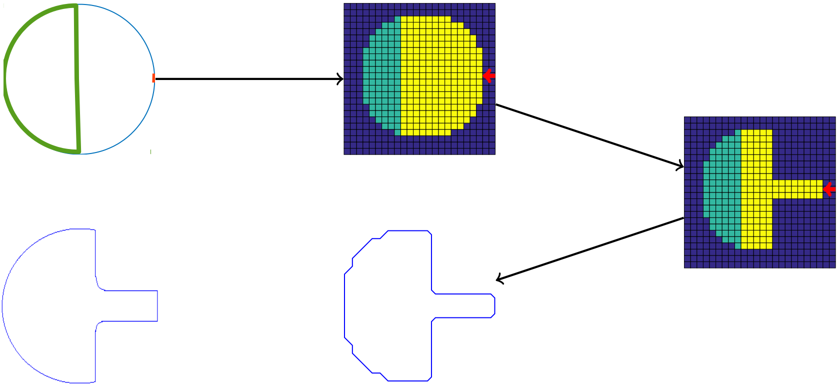
# Overview: Workflow

Optimized topology



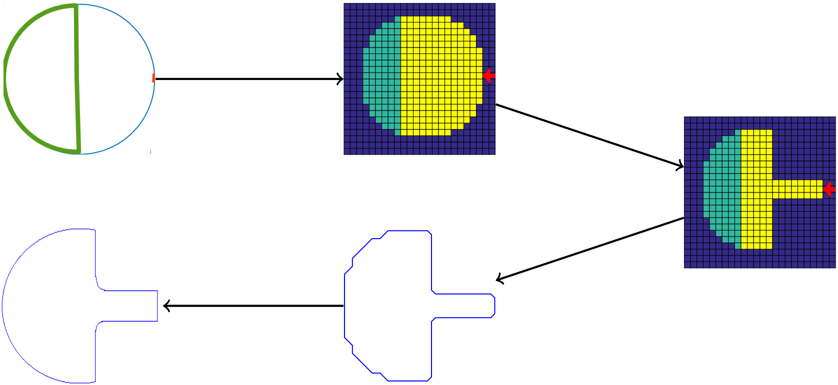
# Overview: Workflow

Surface extraction



# Overview: Workflow

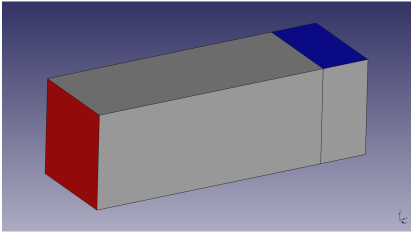
Fit B-Spline surface



# Contents

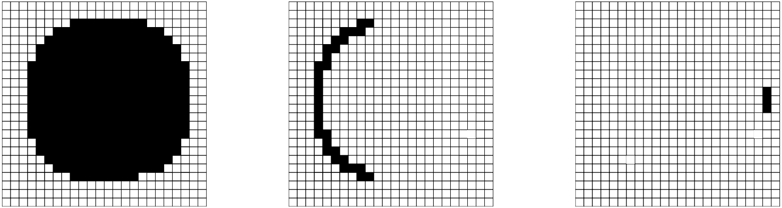
# CAD files: Color code

- Red faces (RGB=[255,0,0]): Fixture
- Green faces (RGB=[0,255,0]): Non-changing region
- Colored (RGB=[0-255,0-255,0-255]): 3D loading vector
- Linear force scaling (according to user-specified parameter)
  One Byte: 0-126 negative, 127 zero, 128-255 positive direction



# Voxelized geometry

- OpenCascade STEP/IGES CAF reader
- Voxelize faces/geometry seperately: Boolean (0/1) grid for
  1. Active voxels (geometry)
  2. Fixture voxels
  3. Non-changing voxels
  4. Load voxels



# Topology Optimization

# Contents

# Status

## Last milestone

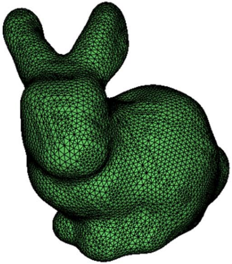
- ⏱ Surface reconstruction with the VTK Toolbox

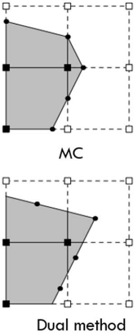
## Today

- ✓ 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

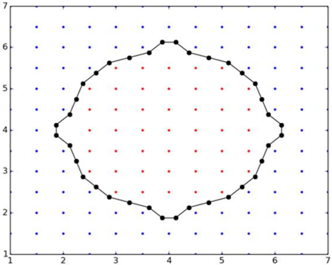


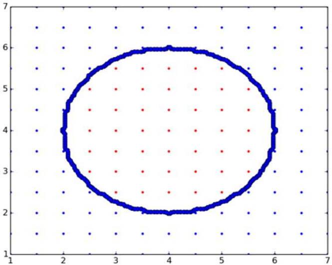



From [4],[5]

# Dual Contouring

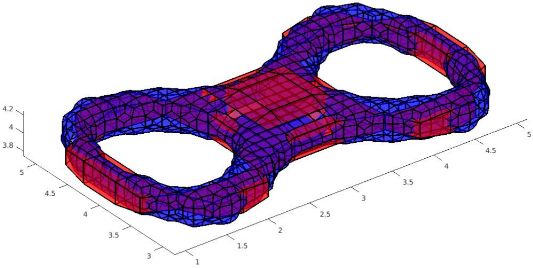
- Python implementation — Use of powerful libraries, including VTK
- Output: Closed surface made out of *quads*
- Coarsening is needed for surface fitting algorithms



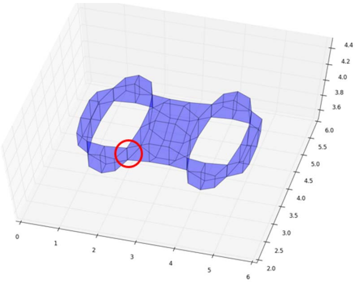
# Dual Contouring

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



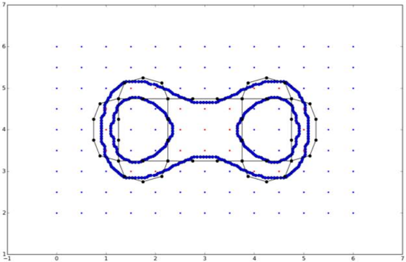
# Dual Contouring — Problems

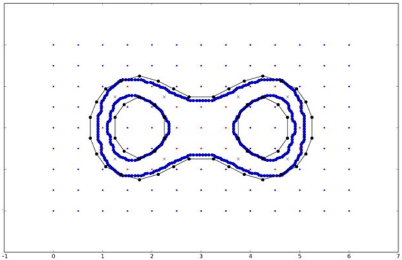
- **Non–manifold edges** appear
- One edge can only belong to two quads for the surface to be closed
- Special treatments in the implementation to avoid them



# Dual Contouring — Problems

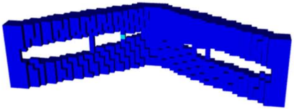
- **Non–manifold edges** appear
- One edge can only belong to two quads for the surface to be closed
- Special treatments in the implementation to avoid them

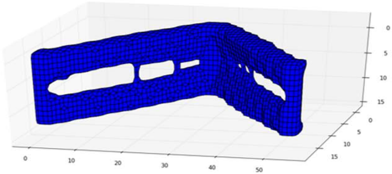




# Dual Contouring — Input

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

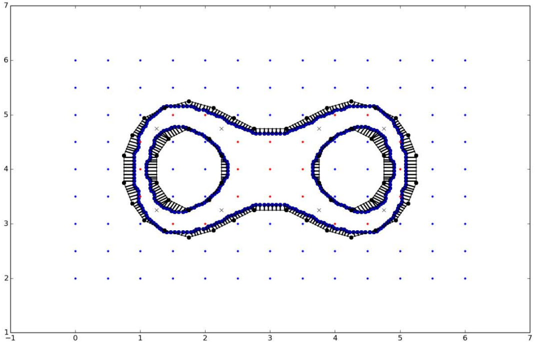




# Demo

# 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

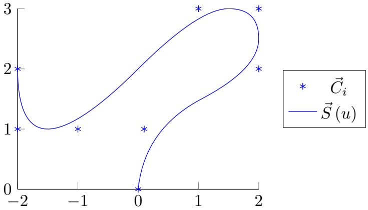
$$\vec{S}(u,v) = \sum_{i,j=1}^{n,m} \vec{C}_{i,j} N_i^p(u) N_j^p(v),$$

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

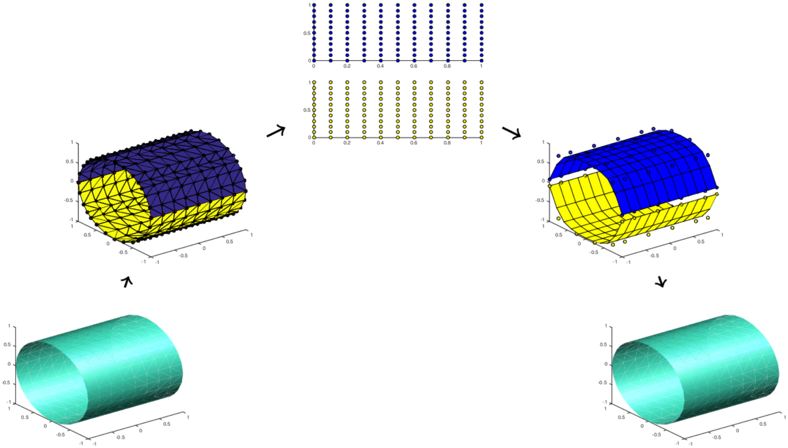
B–Splines

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

**Engineers are working with CAD**



# B–Spline Fitting Pipeline [2]



# Status

## Last milestone

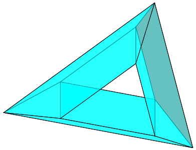
- ✗ Automatic patch selection
- ✗ Parametrization of obtained patches
- ✓ B–spline fitting using least squares
- ◷ Smooth connection of patches
- ✗ Conversion back to CAD

## 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

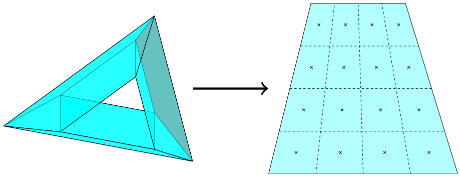
# Long way to smoothness – Peter's scheme

Control mesh



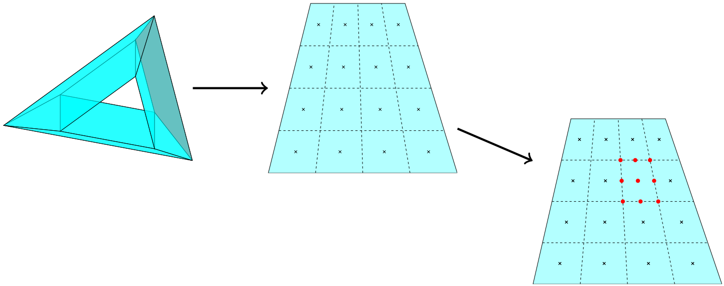
# Long way to smoothness – Peter's scheme

Refined control mesh



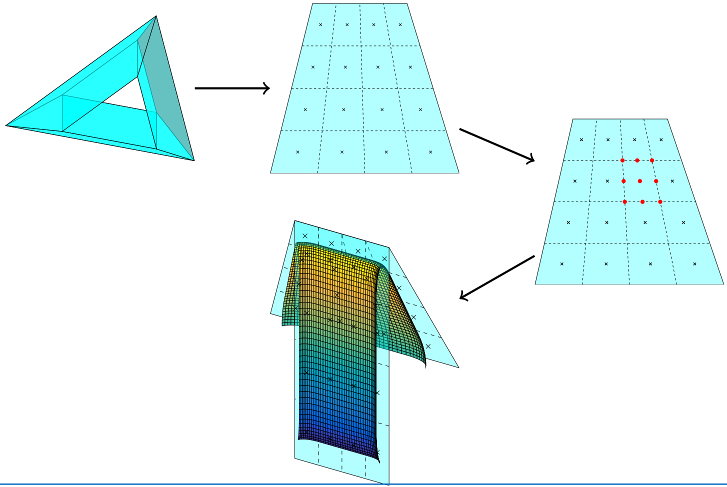
# Long way to smoothness – Peter's scheme

Bezier control points



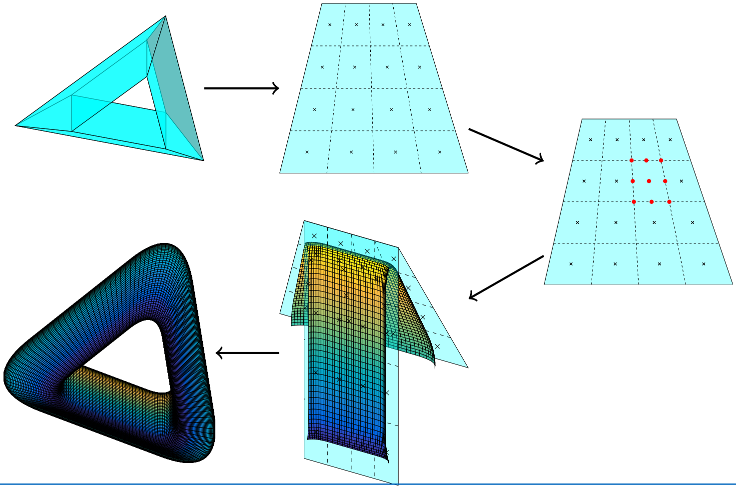
# Long way to smoothness – Peter's scheme

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**

$$\downarrow$$

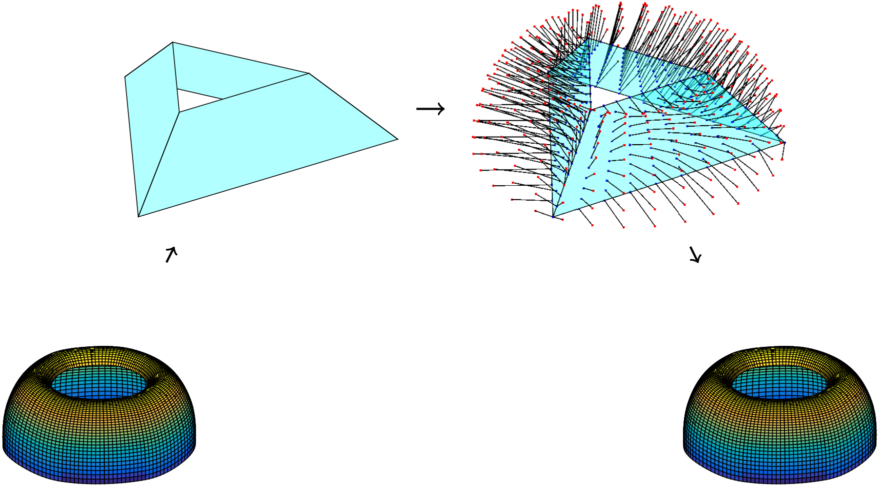
$$E_{dist}(V_x) = \sum_{i=1}^{N} \| P_i - y_i V_x \|_2^2 \rightarrow min,$$

$y_i$ - coefficients obtained from the Peters' scheme theory.

## What is achieved?

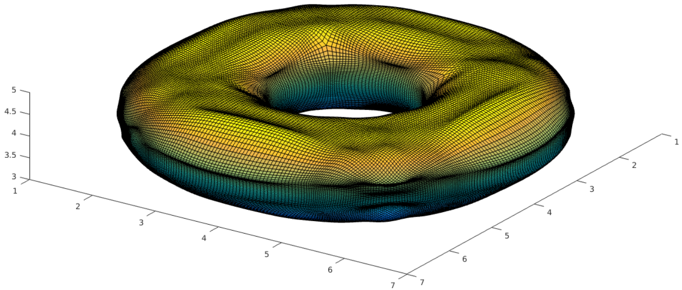
- Smoothness of the fitted surface is now guaranteed by construction
- Fitting of more complex shapes achieved

# 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

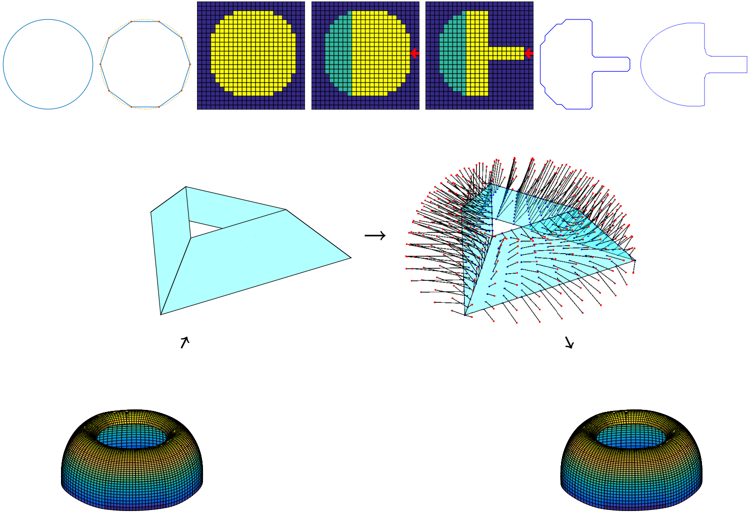
## C++

- ⊕ 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!

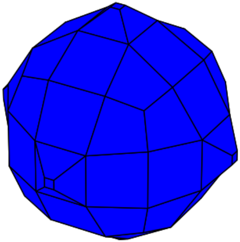


# Literature

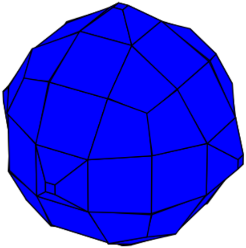

1. **William Hunter.** ”Predominantly solid-void three-dimensional topology optimisation using open source software”
2. **Gerrit Becker, Michael Schäfer, Antony Jameson.** ”An advanced NURBS fitting procedure for post-processing of grid-based shape optimizations”
3. **Matthias Eck, Hugues Hoppe.** ”Automatic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



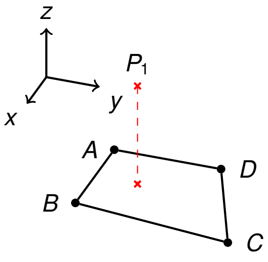
DC sphere



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

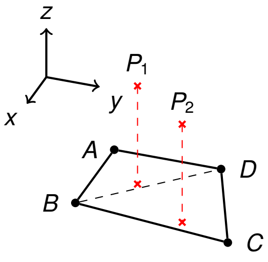
$$B_{BAD} = \begin{pmatrix} \vec{n} & \vec{AB} & \vec{AD} \end{pmatrix}$$

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 $u = 1 - u$, $v = 1 - v$