Recovering Primitives in 3D CAD meshes

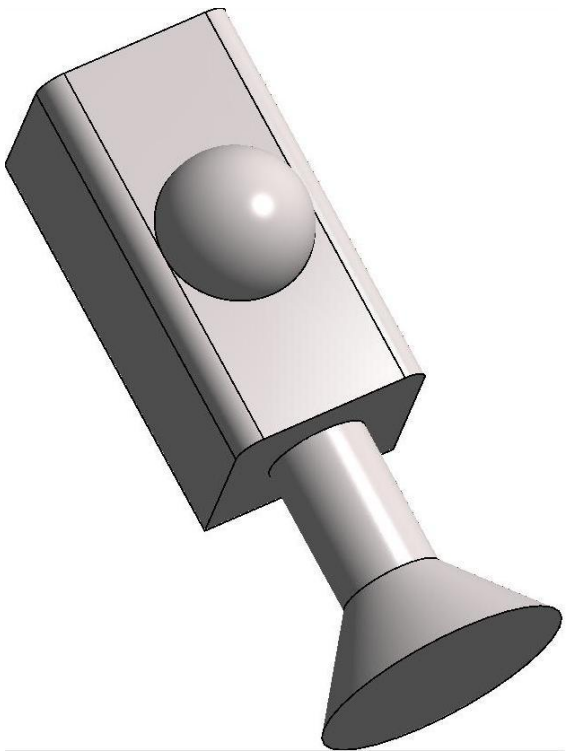
Roseline Bénière
G. Subsol, G. Gesquière, F. Le Breton and W. Puech

LIRMM, Montpellier, France
C4W, Montpellier, France
LSIS, Arles, France

14 octobre 2010
Réunion ICAR

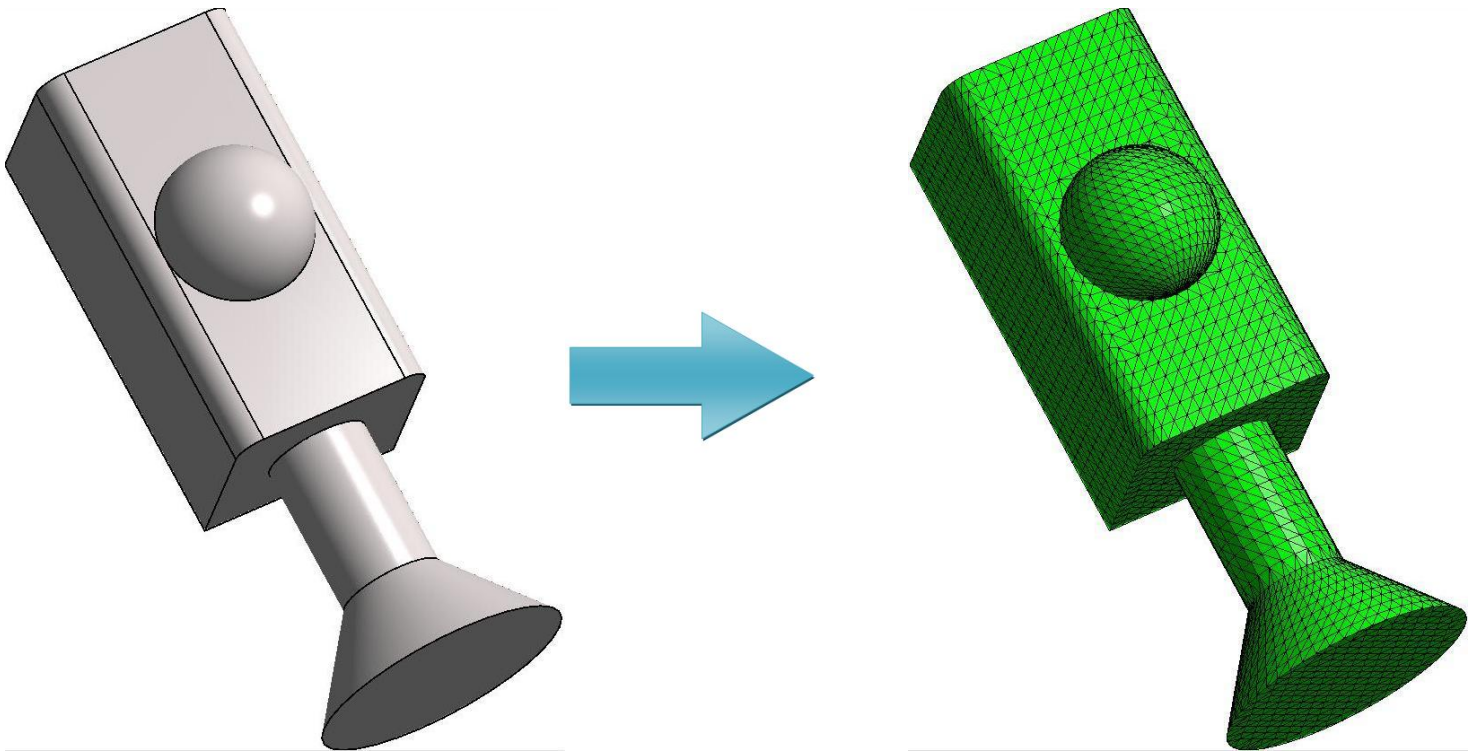
Objective

In CAD an object is usually modeled by a structured combination of primitives



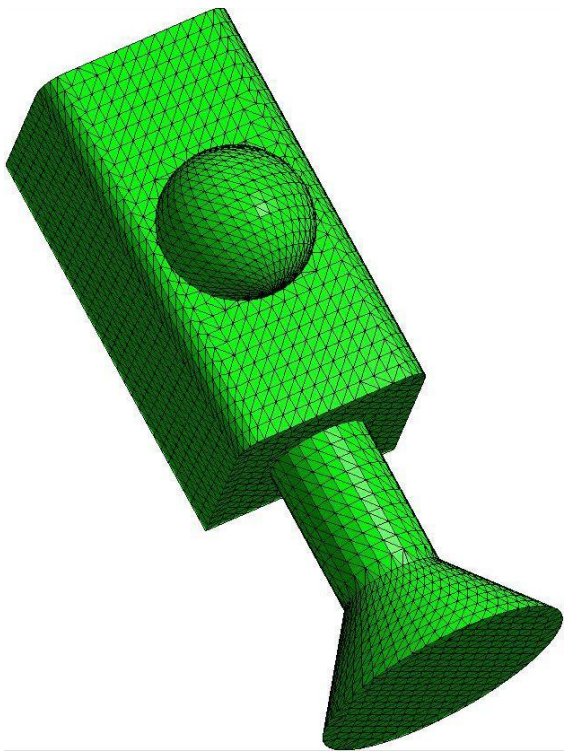
Objective

In CAD an object is usually modeled by a structured combination of primitives
But to use, we often need to discretized it into a 3D mesh



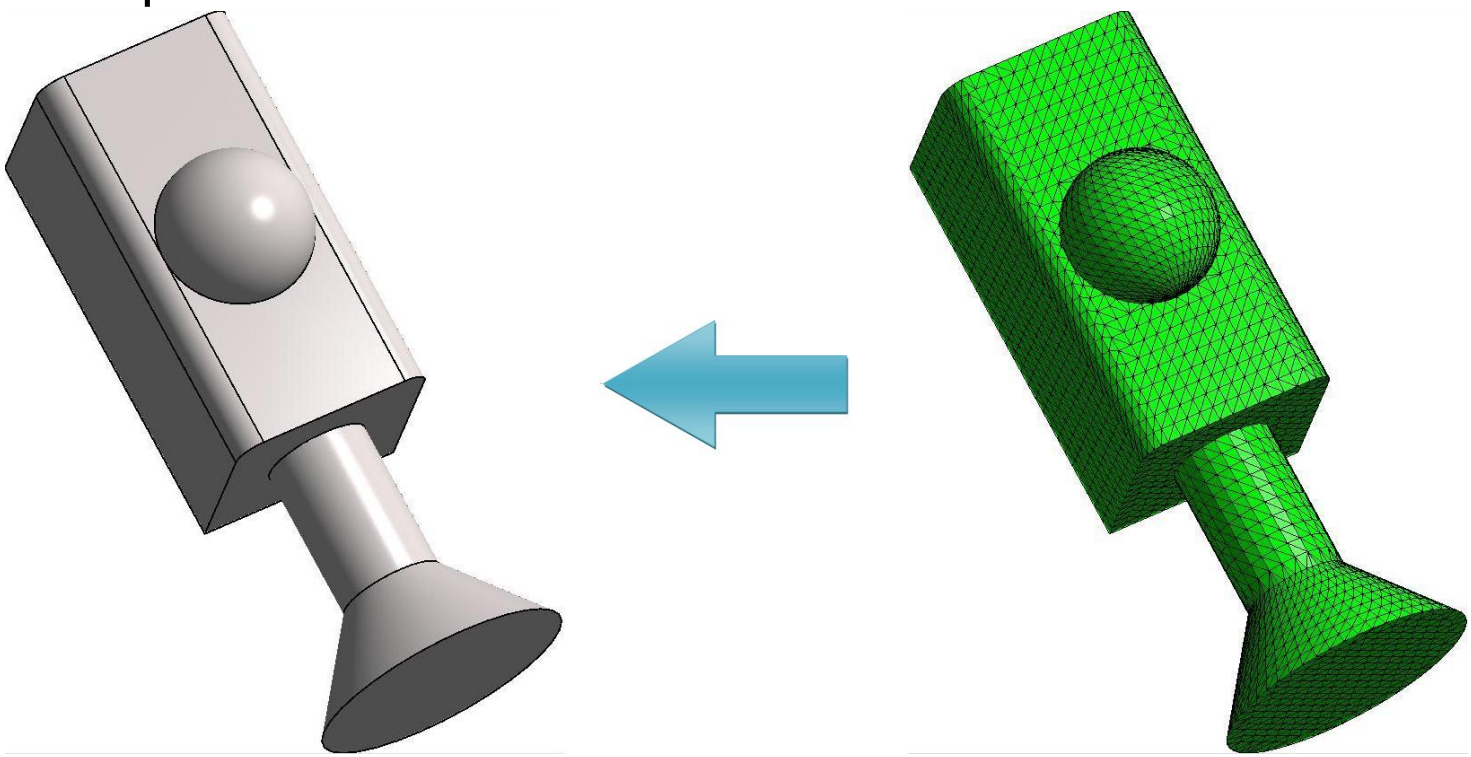
Objective

In CAD an object is usually modeled by a structured combination of primitives
But to use, we often need to discretized it into a 3D mesh
And the initial model can be lost or not correspond anymore



Objective

In CAD an object is usually modeled by a structured combination of primitives
But to use, we often need to discretized it into a 3D mesh
And the initial model can be lost or not correspond anymore
So a primitive extraction algorithm is needed to reconstruct the initial representation



Previews

Benkõ *et al.*
Algorithms for reverse engineering boundary representation models
Computer-Aided Design 33(11) : 839-851 2001

Bohm *et al.*
Curvature based range image classification for object
PROC SPIE INT SOC OPT ENG 4197 : 211-220 2000

Previews

Benkõ *et al.*
Algorithms for reverse engineering boundary representation models
Computer-Aided Design 33(11) : 839-851 2001

Bohm *et al.*
Curvature based range image classification for object
PROC SPIE INT SOC OPT ENG 4197 : 211-220 2000

Same Process :

Previews

Benkõ *et al.*
Algorithms for reverse engineering boundary representation models
Computer-Aided Design 33(11) : 839-851 2001

Bohm *et al.*
Curvature based range image classification for object
PROC SPIE INT SOC OPT ENG 4197 : 211-220 2000

Same Process :

1. Segmentation

Previews

Benkõ *et al.*
Algorithms for reverse engineering boundary representation models
Computer-Aided Design 33(11) : 839-851 2001

Bohm *et al.*
Curvature based range image classification for object
PROC SPIE INT SOC OPT ENG 4197 : 211-220 2000

Same Process :

1. Segmentation
2. Classification

Previews

Benkõ *et al.*
Algorithms for reverse engineering boundary representation models
Computer-Aided Design 33(11) : 839-851 2001

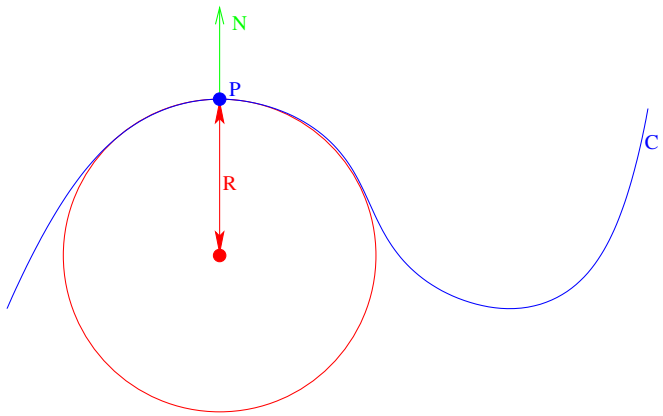
Bohm *et al.*
Curvature based range image classification for object
PROC SPIE INT SOC OPT ENG 4197 : 211-220 2000

Same Process :

1. Segmentation
2. Classification
3. Fitting

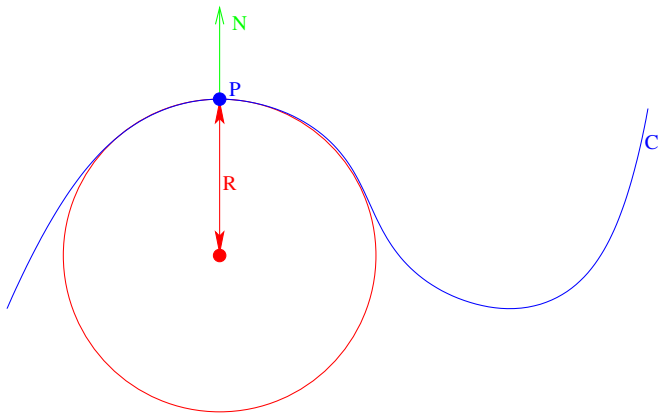
Curvature 2D and 3D

Curvature 2D :



Curvature 2D and 3D

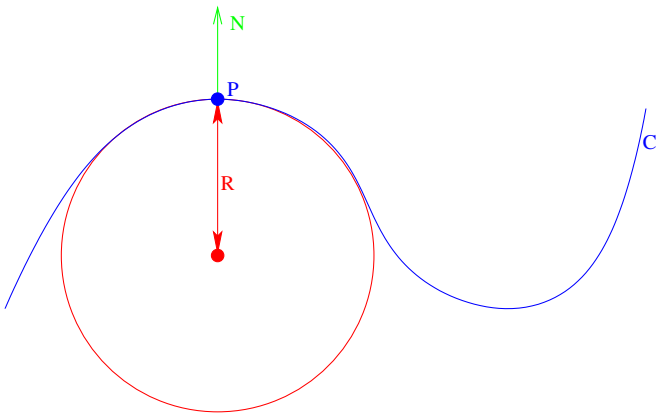
Curvature 2D :



$$\Rightarrow K_p = \frac{1}{R}$$

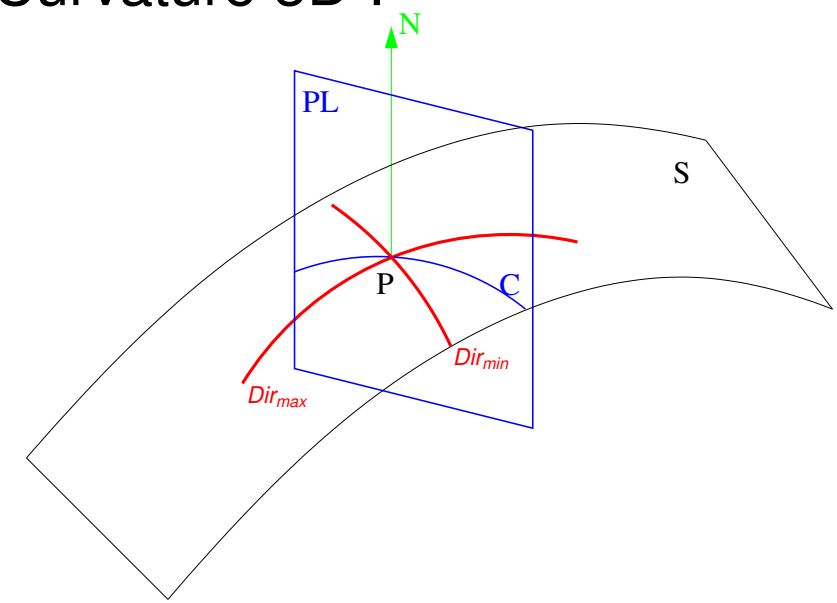
Curvature 2D and 3D

Curvature 2D :



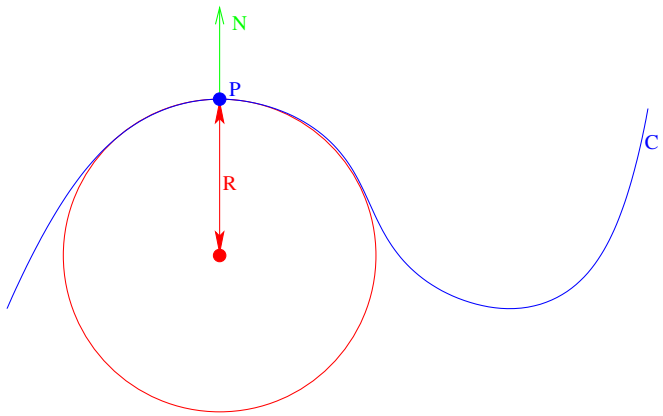
$$\Rightarrow K_p = \frac{1}{R}$$

Curvature 3D :



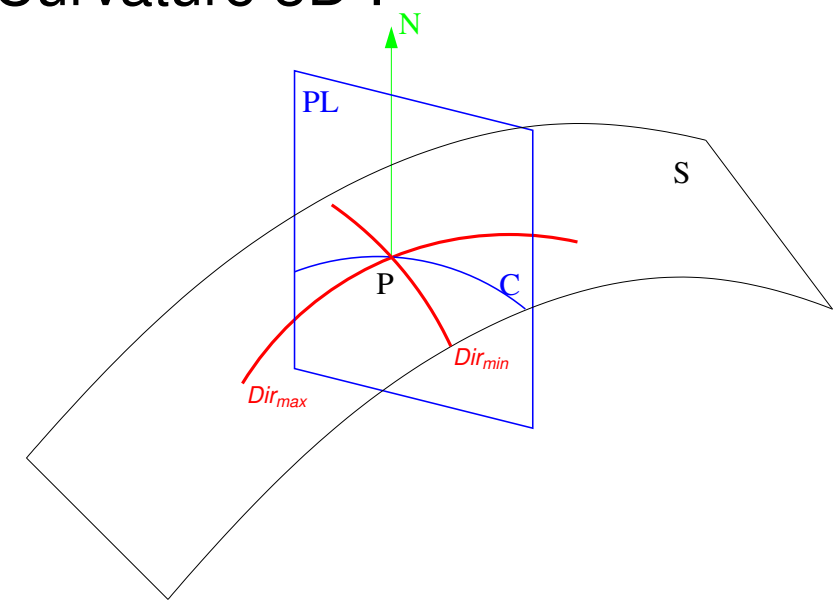
Curvature 2D and 3D

Curvature 2D :



$$\Rightarrow K_p = \frac{1}{R}$$

Curvature 3D :



- 2 Principal Curvatures (k_{max} et k_{min})
- 2 Principal Directions (Dir_{max} et Dir_{min})
- Normal

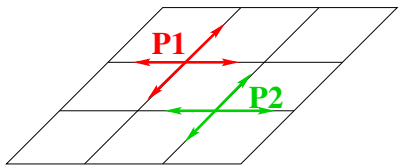
Primitive curvature features

The points contained in Plane, Sphere, Cone or Cylinder have specific features on curvature :

Primitive curvature features

The points contained in Plane, Sphere, Cone or Cylinder have specific features on curvature :

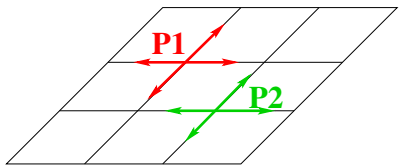
- Plane $\Rightarrow k_{max} = k_{min} = 0$

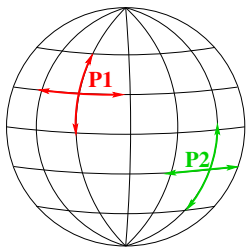


Primitive curvature features

The points contained in Plane, Sphere, Cone or Cylinder have specific features on curvature :

- Plane $\Rightarrow k_{max} = k_{min} = 0$
- Sphere $\Rightarrow k_{max} = k_{min} = \frac{1}{\text{rSp}} \neq 0$

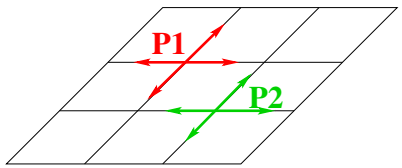




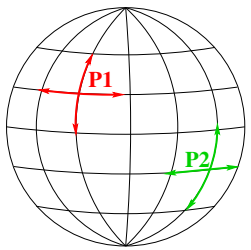
Primitive curvature features

The points contained in Plane, Sphere, Cone or Cylinder have specific features on curvature :

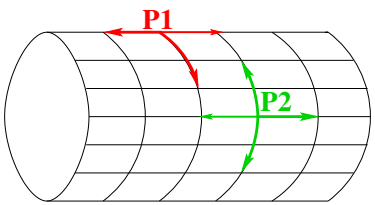
- Plane $\Rightarrow k_{max} = k_{min} = 0$



- Sphere $\Rightarrow k_{max} = k_{min} = \frac{1}{\text{rSp}} \neq 0$



- Cylinder $\Rightarrow k_{min} = 0$ et $k_{max} = \frac{1}{\text{rCy}}$

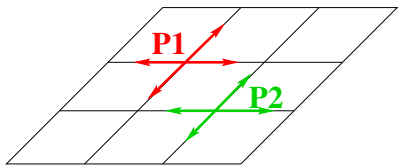


$Dir_{min} =$ Generating line

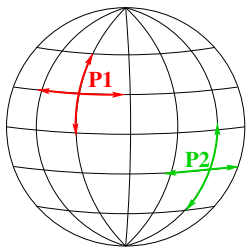
Primitive curvature features

The points contained in Plane, Sphere, Cone or Cylinder have specific features on curvature :

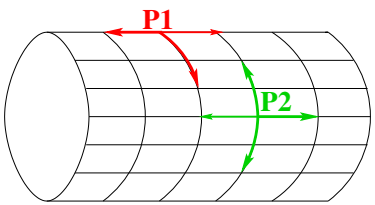
- Plane $\Rightarrow k_{max} = k_{min} = 0$



- Sphere $\Rightarrow k_{max} = k_{min} = \frac{1}{\text{rSp}} \neq 0$



- Cylinder $\Rightarrow k_{min} = 0$ et $k_{max} = \frac{1}{\text{rCy}}$



$Dir_{min} =$ Generating line

- Cone $\Rightarrow$ idem Cylinder but with a variable radius

Discrete Curvature

In a mesh, we compute a discrete curvature for each point.

Discrete Curvature

In a mesh, we compute a discrete curvature for each point.

We use the Euler formula

$$k_n = k_{max} \cos^2(\theta) + k_{min} \sin^2(\theta)$$

With θ the angle between n and Dir_{max}.
The neighbors are studied to approximate k_{max}, k_{min} and θ.

Discrete Curvature

In a mesh, we compute a discrete curvature for each point.

We use the Euler formula

$$k_n = k_{max} \cos^2(\theta) + k_{min} \sin^2(\theta)$$

With θ the angle between n and Dir_{max}.
The neighbors are studied to approximate k_{max}, k_{min} and θ.

To determine the point which will be used, we fix a *k-neighborhood*.

Discrete Curvature

In a mesh, we compute a discrete curvature for each point.

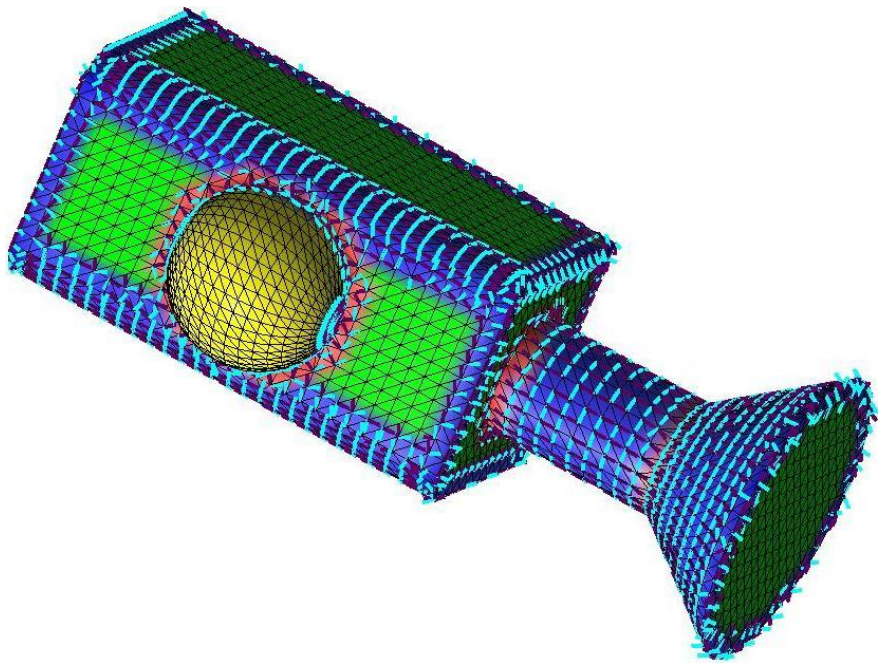
We use the Euler formula

$$k_n = k_{max} \cos^2(\theta) + k_{min} \sin^2(\theta)$$

With θ the angle between n and Dir_{max}.
The neighbors are studied to approximate k_{max}, k_{min} and θ.

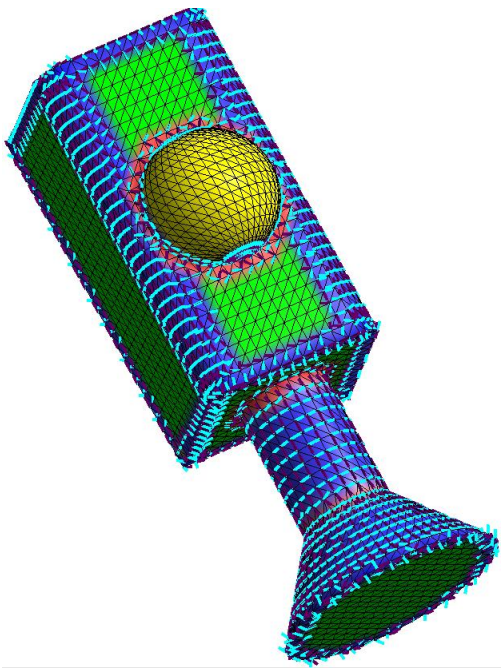
To determine the point which will be used, we fix a *k-neighborhood*.

Concave Point
Convex Point
Saddle Point
Plane Point
Spheric Point



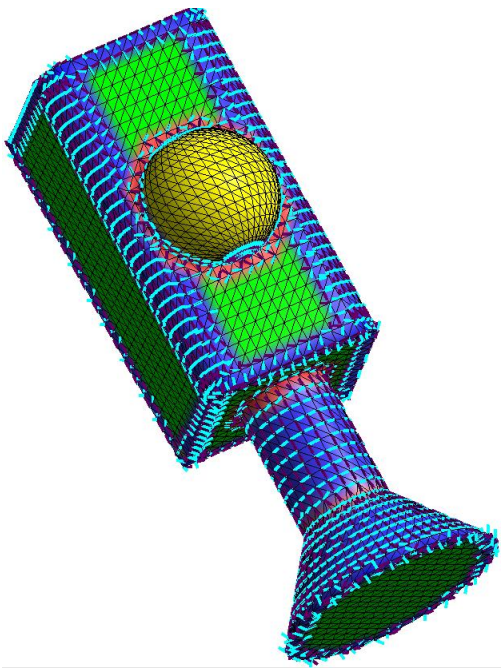
Planes Extraction

- From Curvatures



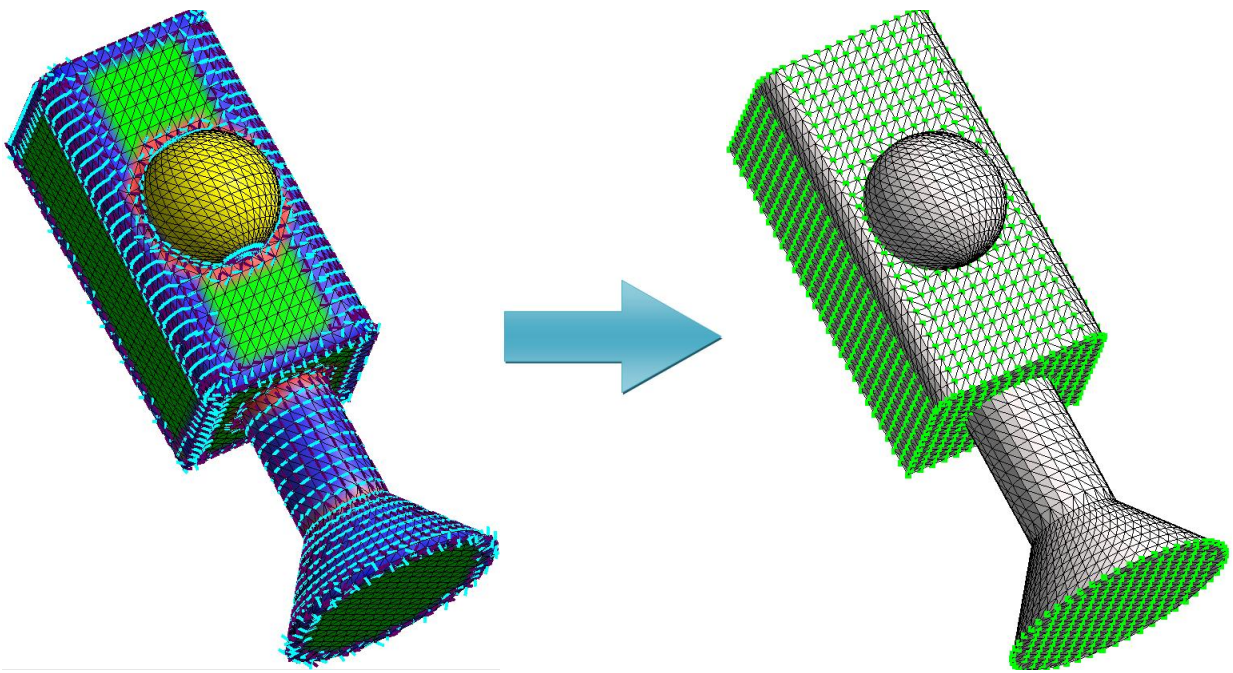
Planes Extraction

- From Curvatures
- Group all adjacent points with $k_{max} = k_{min} = 0$



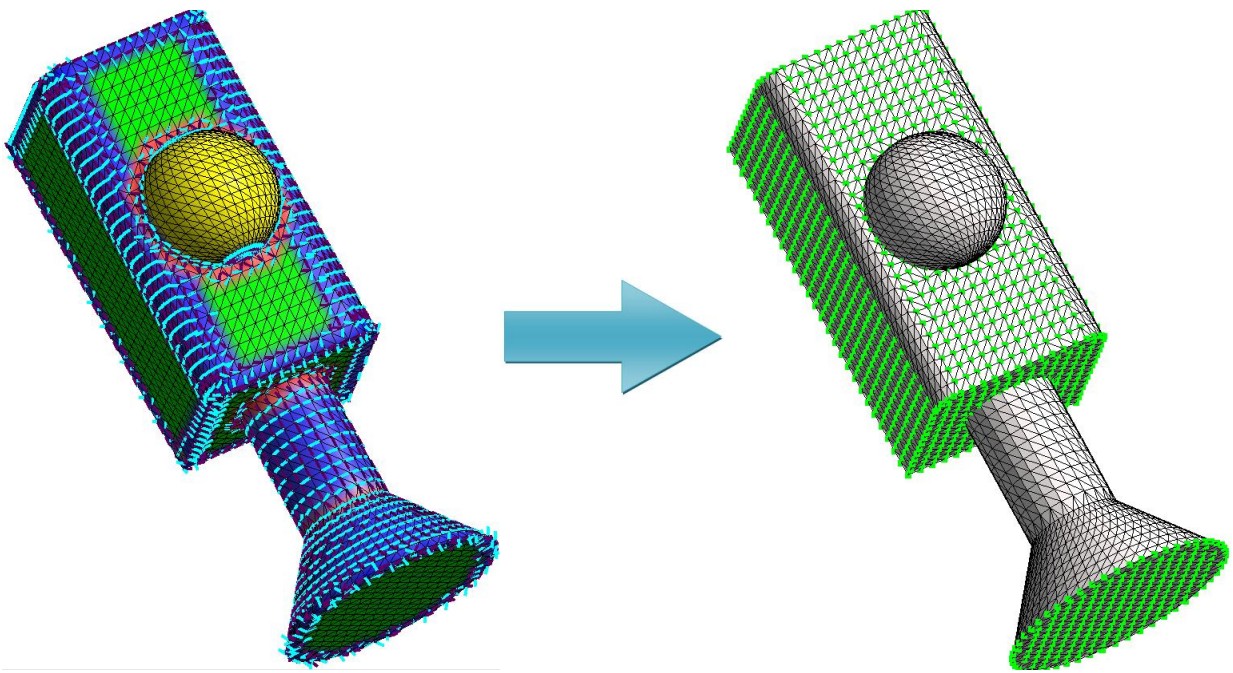
Planes Extraction

- From Curvatures
- Group all adjacent points with $k_{max} = k_{min} = 0$



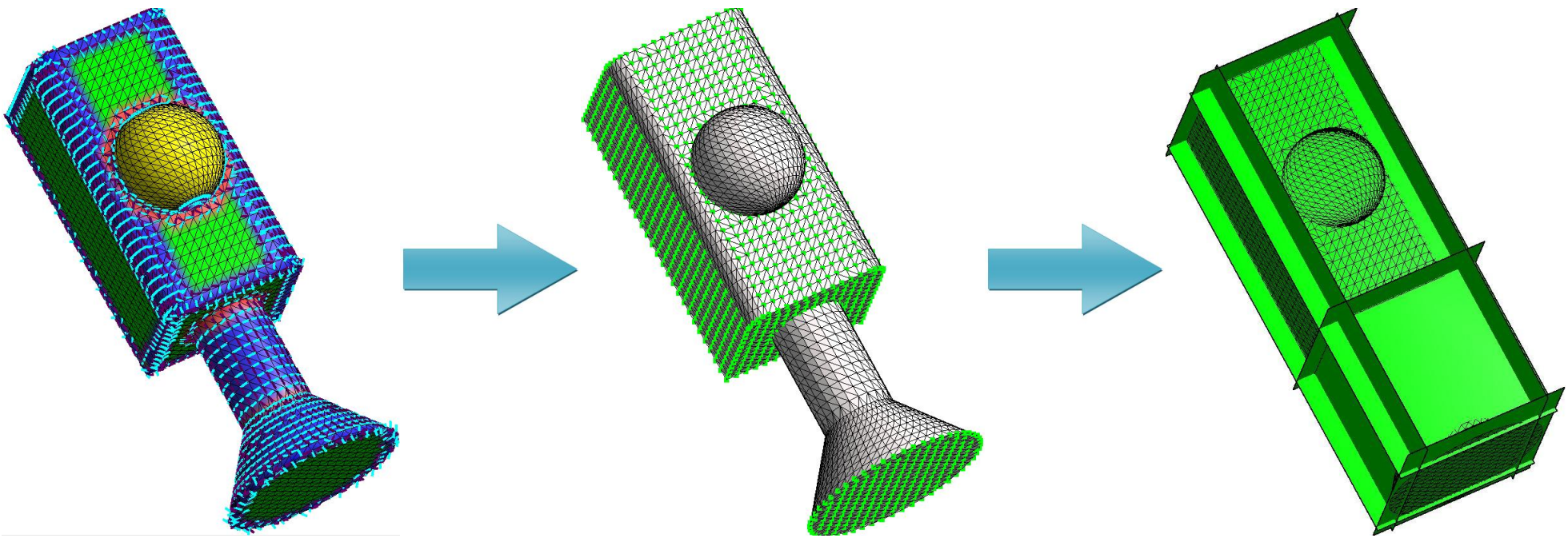
Planes Extraction

- From Curvatures
- Group all adjacent points with $k_{max} = k_{min} = 0$
- Equation Coefficients : $ax + by + cz + d = 0$ are approximated by a least square regression



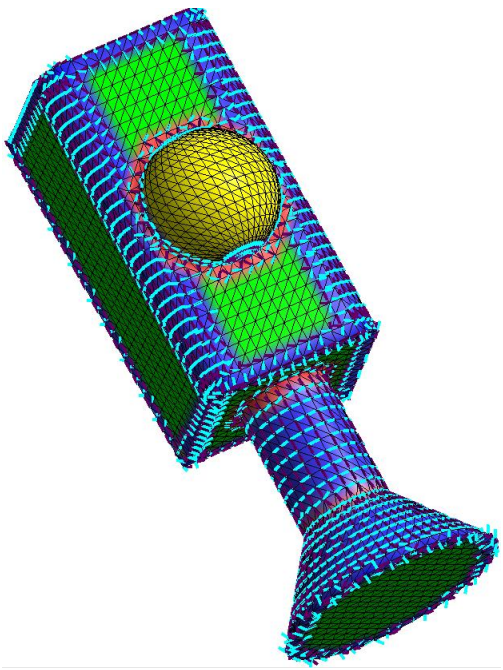
Planes Extraction

- From Curvatures
- Group all adjacent points with $k_{max} = k_{min} = 0$
- Equation Coefficients : $ax + by + cz + d = 0$ are approximated by a least square regression



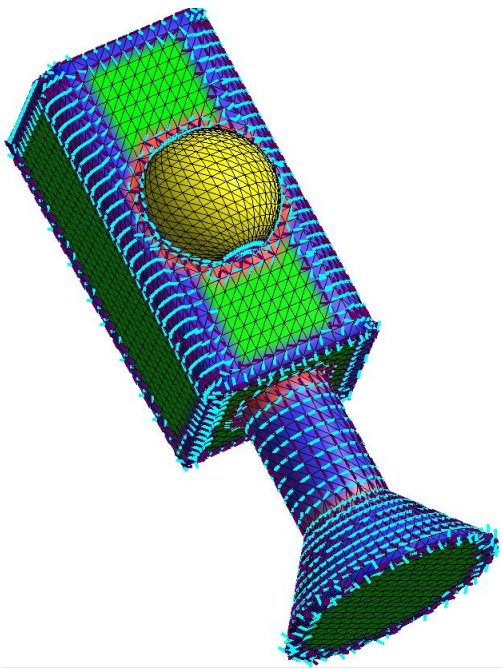
Spheres Extraction

- From Curvatures



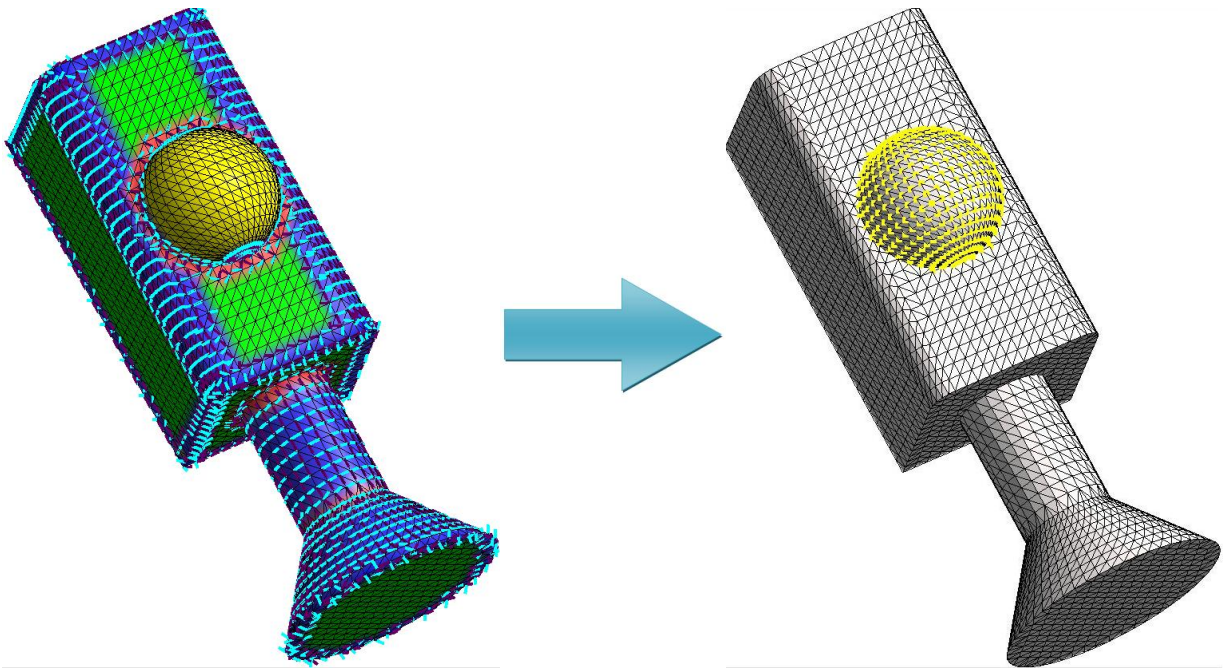
Spheres Extraction

- From Curvatures
- Group all adjacent points with $k_{max} = k_{min} \approx K$



Spheres Extraction

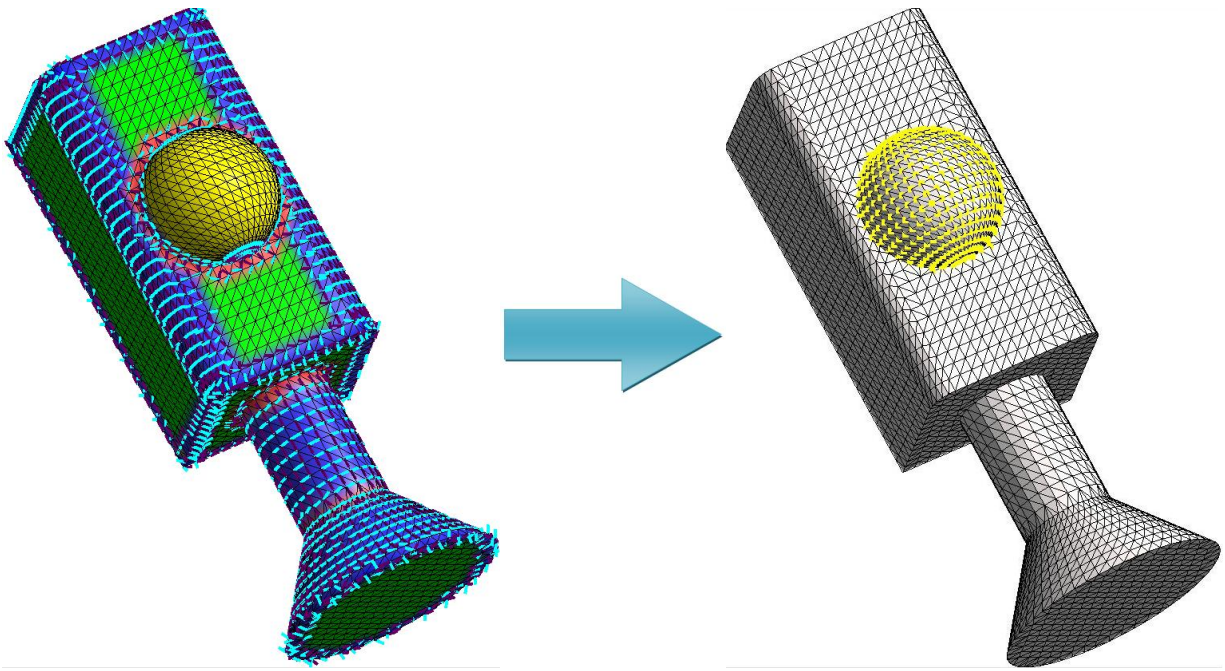
- From Curvatures
- Group all adjacent points with $k_{max} = k_{min} \approx K$



Spheres Extraction

- From Curvatures
- Group all adjacent points with $k_{max} = k_{min} \approx K$
- The radius and the center are approximated by a least square method

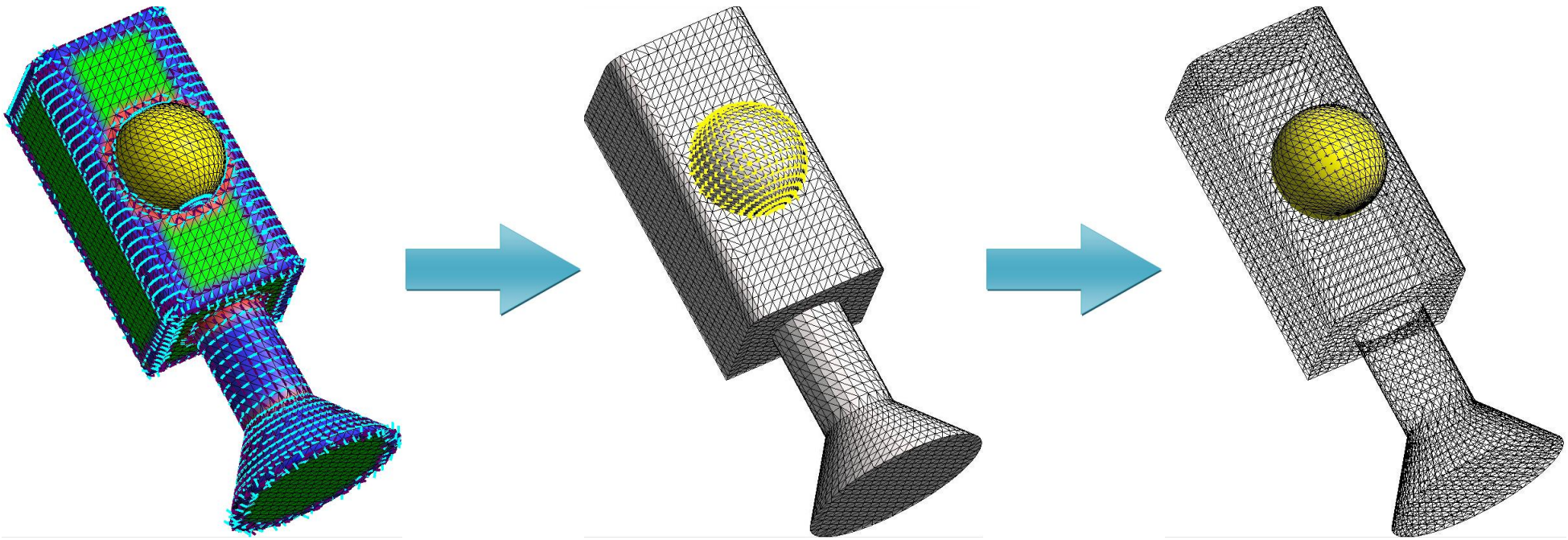
$\Rightarrow$ The average of the curvature inverse is used to validate it



Spheres Extraction

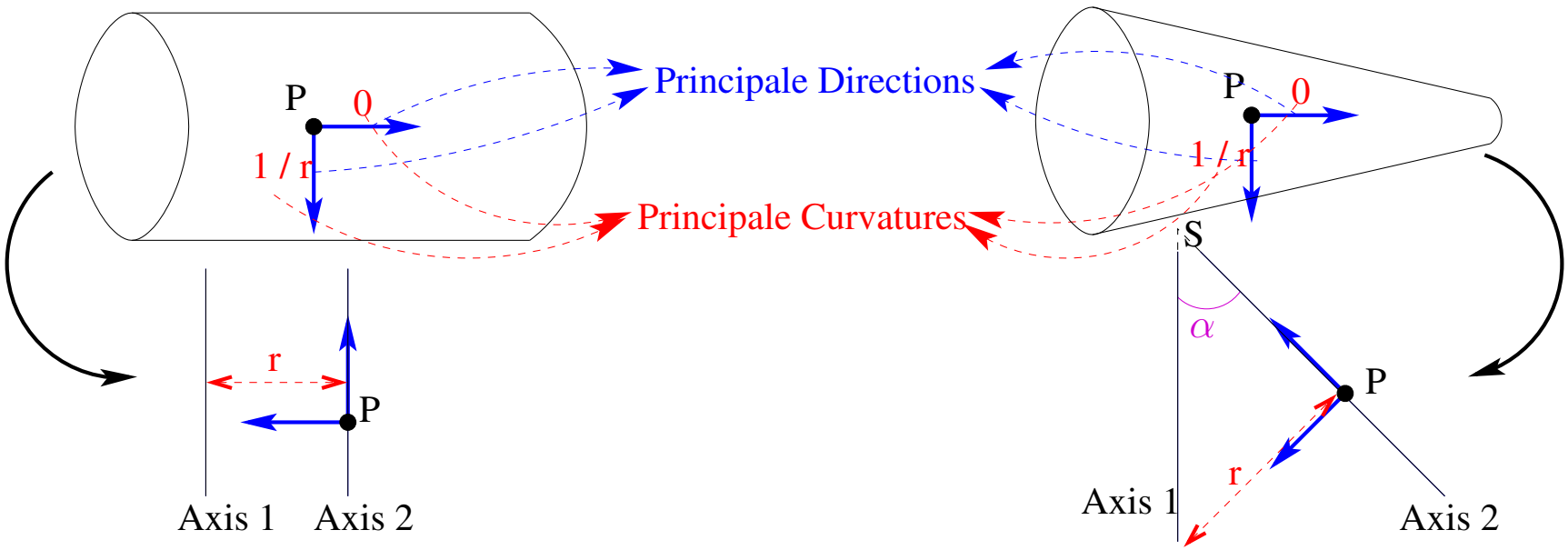
- From Curvatures
- Group all adjacent points with $k_{max} = k_{min} \approx K$
- The radius and the center are approximated by a least square method

$\Rightarrow$ The average of the curvature inverse is used to validate it



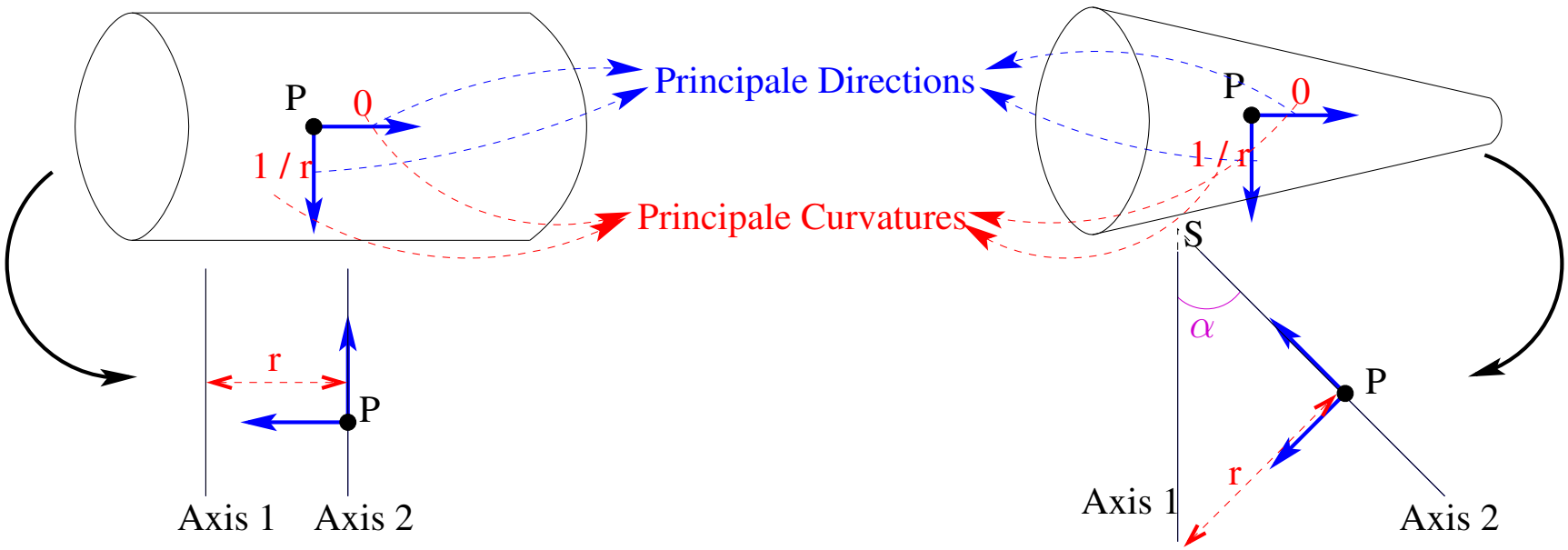
Cones/Cylinders Extraction

Features of Cones and Cylinders :



Cones/Cylinders Extraction

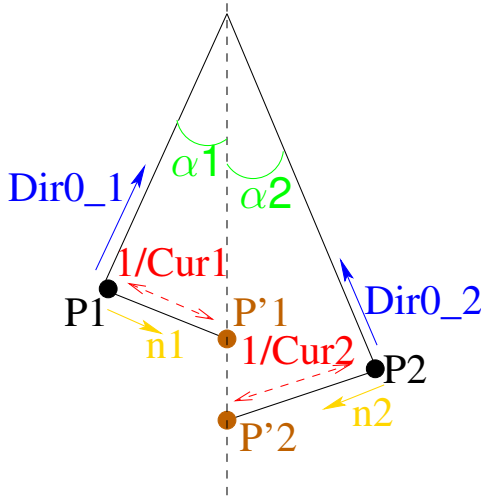
Features of Cones and Cylinders :



Criterion of belonging to a same cone :

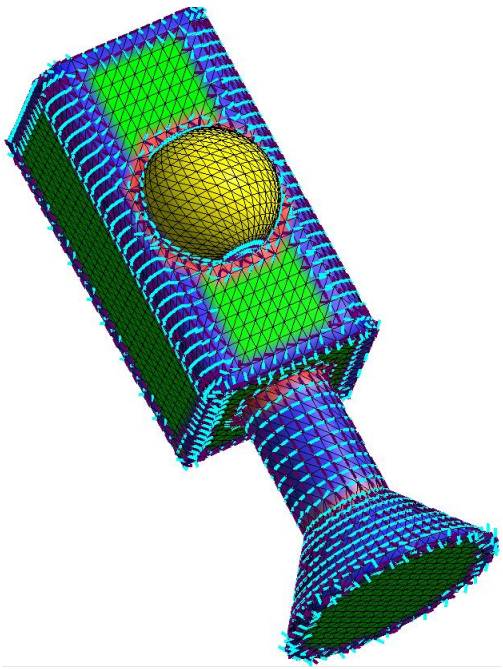
If P_1 et $P_2 \in$ same cone

$\Rightarrow \alpha_1 = \alpha_2$



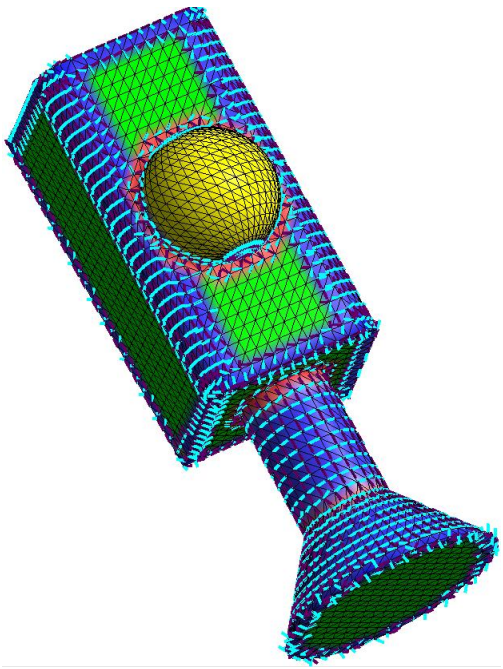
Cones/Cylinders Extraction

- From Curvatures



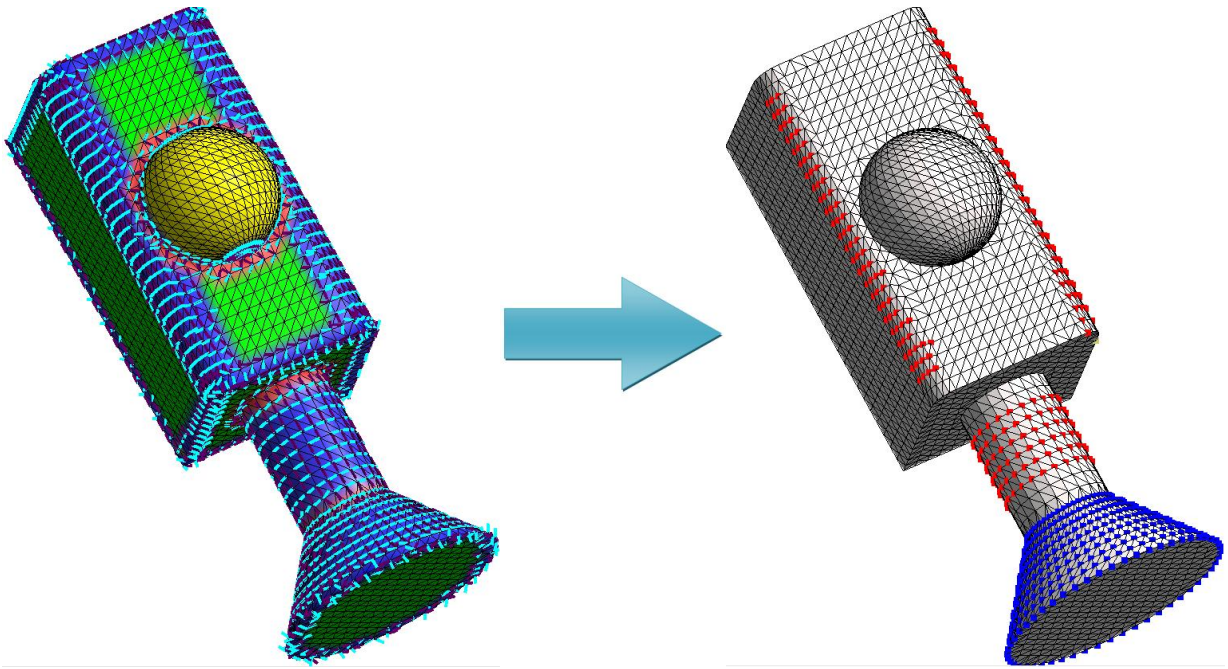
Cones/Cylinders Extraction

- From Curvatures
- Group adjacent points by the criterion of belonging



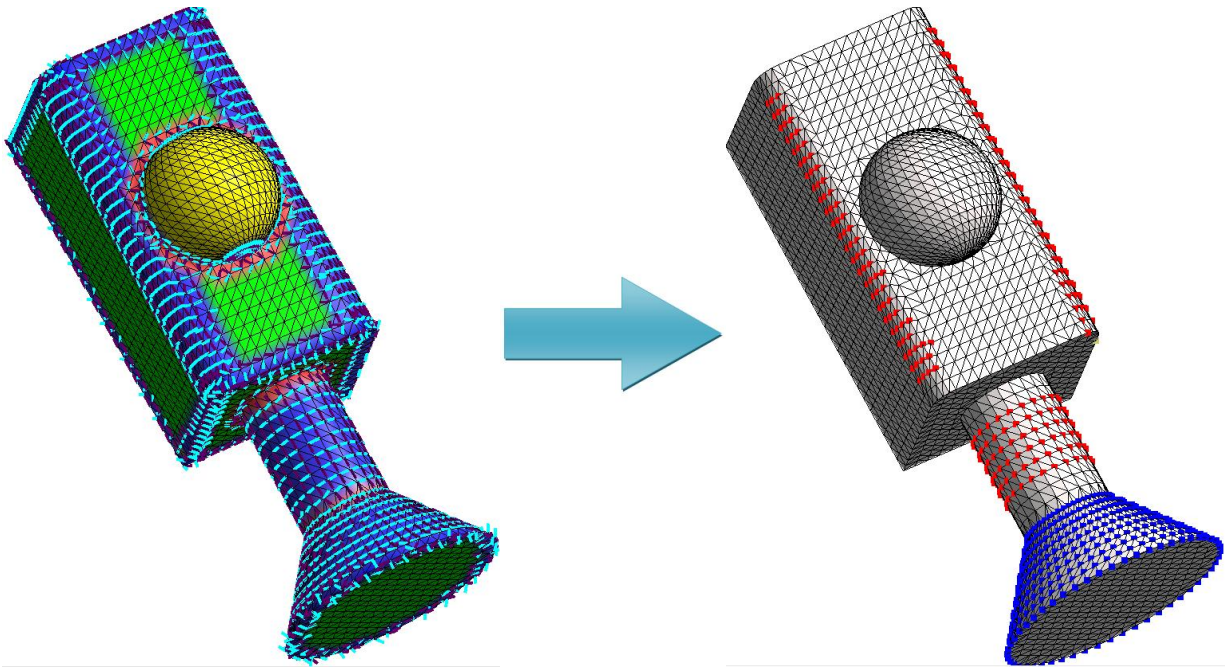
Cones/Cylinders Extraction

- From Curvatures
- Group adjacent points by the criterion of belonging



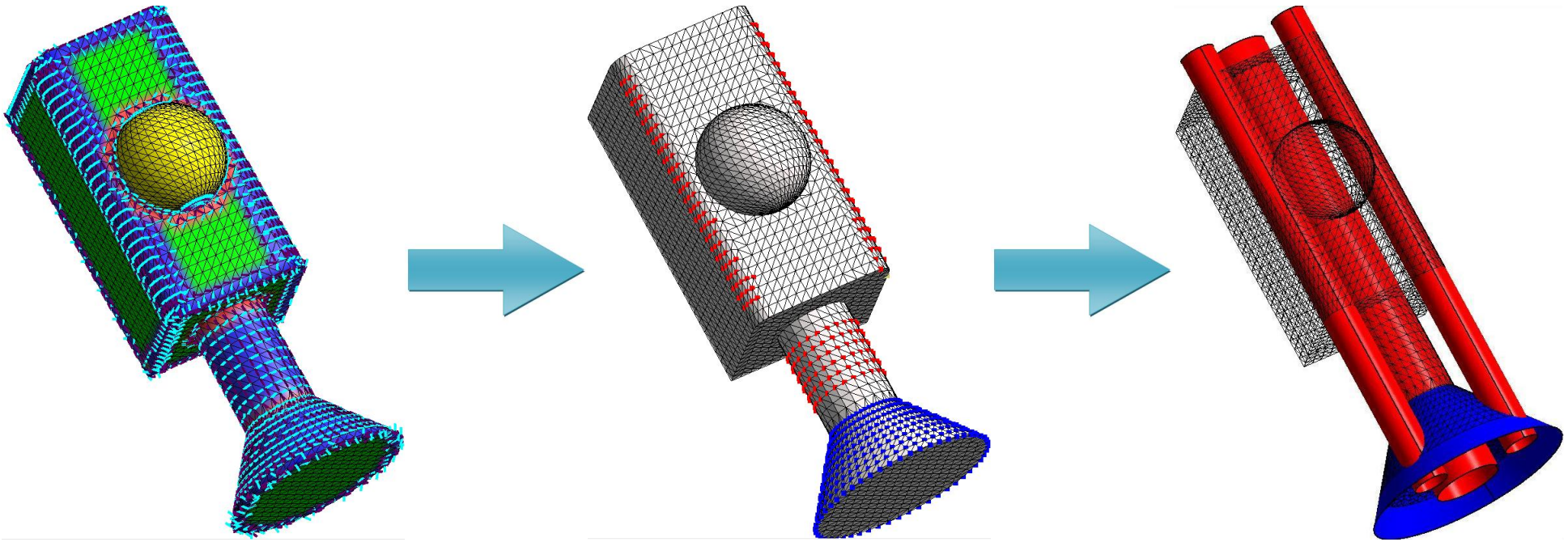
Cones/Cylinders Extraction

- From Curvatures
- Group adjacent points by the criterion of belonging
- For each point : $pointAxisTheo = point + normal * radius$
  $\Rightarrow$ Rotation Axis $\Rightarrow \alpha$ angle between Axis and $Dir_{=0}$
  - $\alpha = \pi \Rightarrow$ Cylinder : Average curvature $\Rightarrow$ Radius
  - $\alpha \neq \pi \Rightarrow$ Cone : Intersection between each plane created by the two principal directions of one point $\Rightarrow$ Vertex

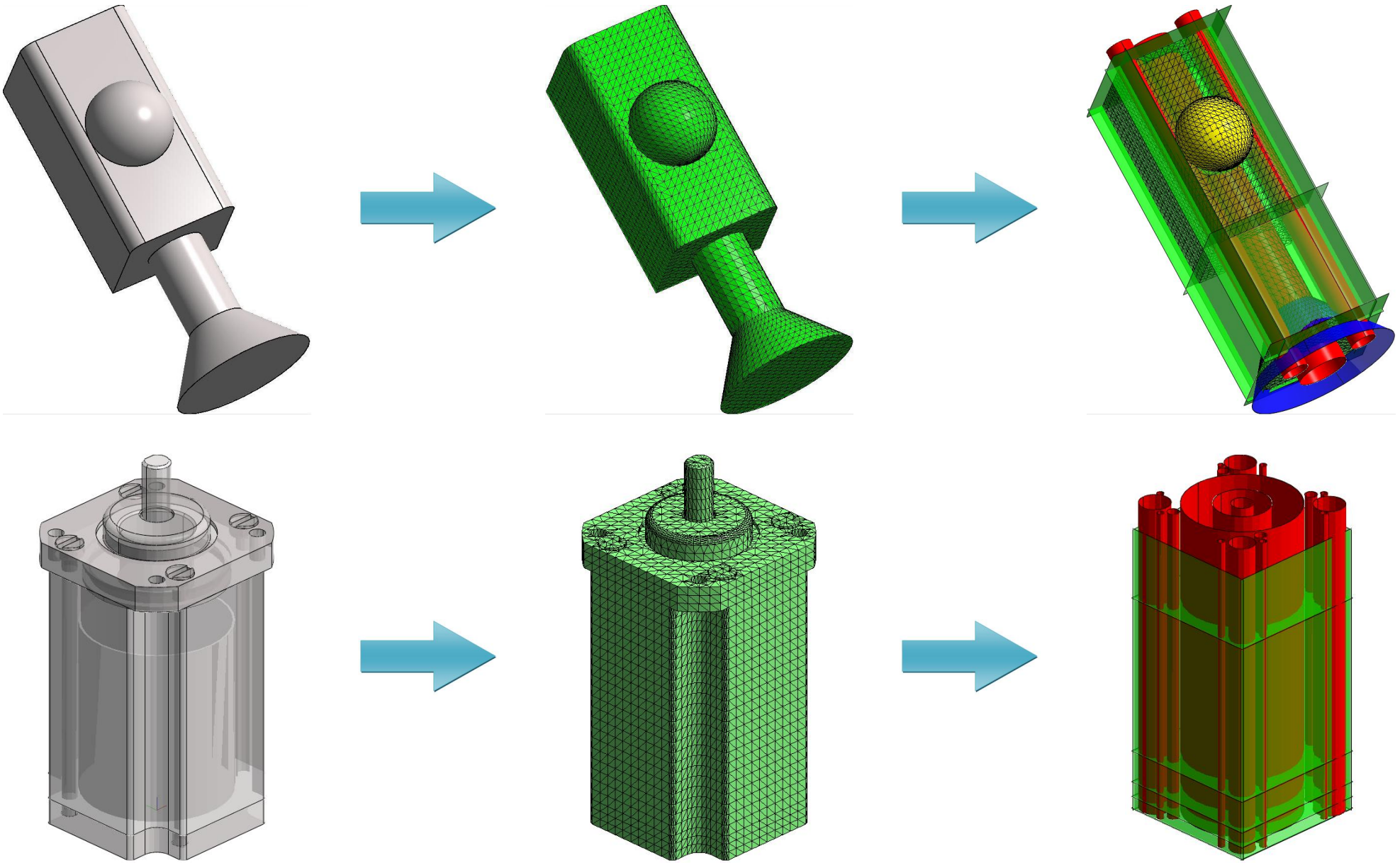


Cones/Cylinders Extraction

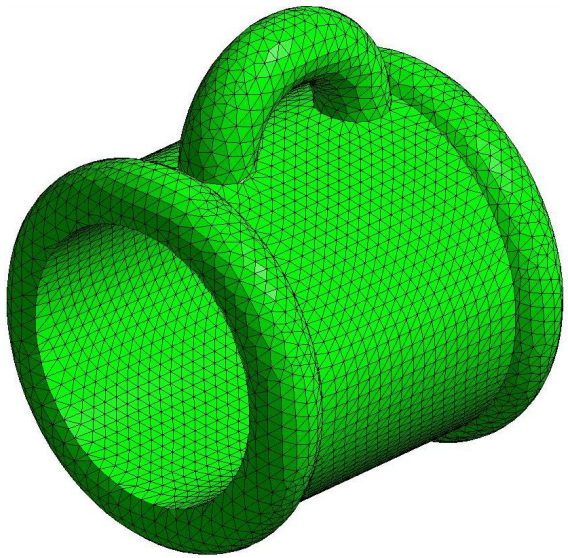
- From Curvatures
- Group adjacent points by the criterion of belonging
- For each point : $pointAxisTheo = point + normal * radius$
  $\Rightarrow$ Rotation Axis $\Rightarrow \alpha$ angle between Axis and $Dir_{=0}$
  - $\alpha = \pi \Rightarrow$ Cylinder : Average curvature $\Rightarrow$ Radius
  - $\alpha \neq \pi \Rightarrow$ Cone : Intersection between each plane created by the two principal directions of one point $\Rightarrow$ Vertex

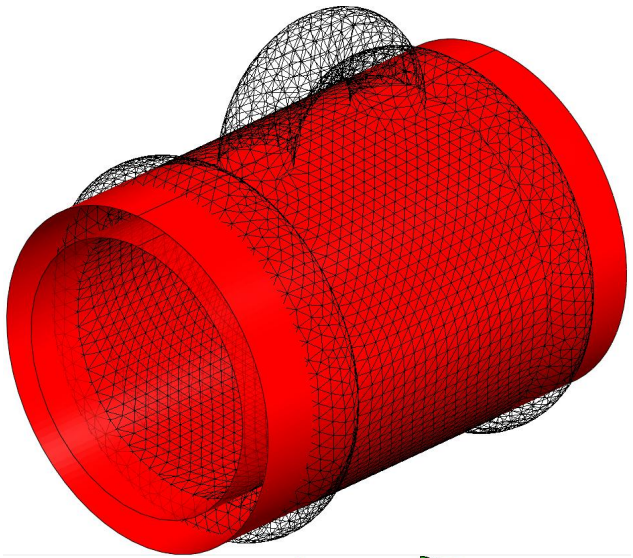


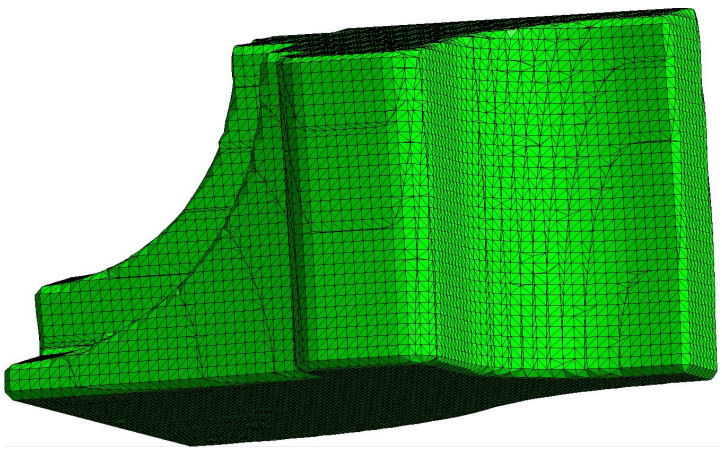
Results

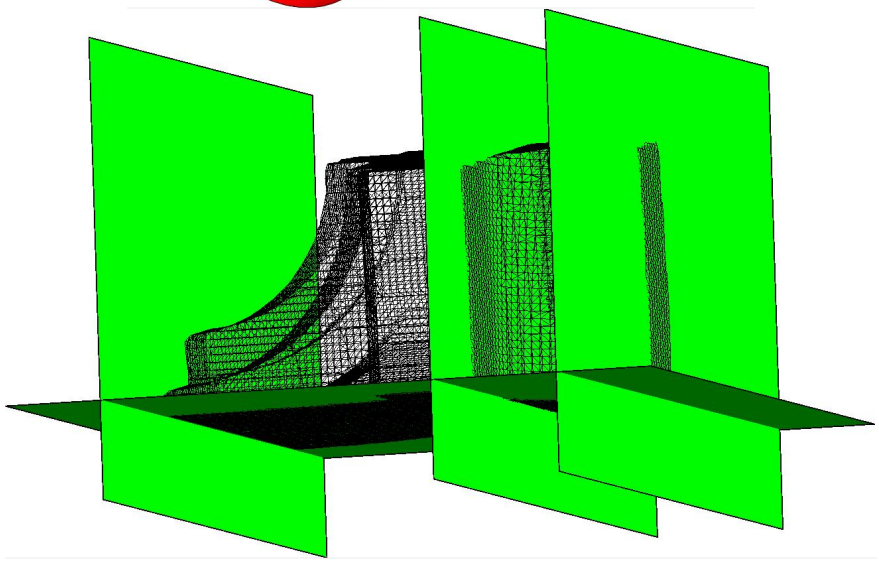


Results





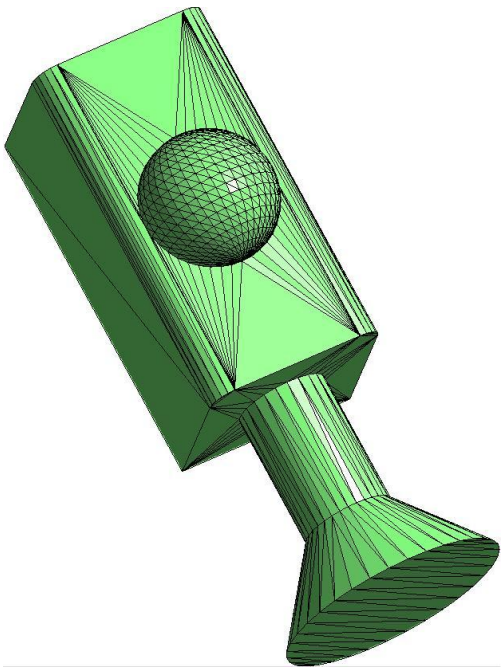




http ://shapes.aimatshape.net

Sparse Mesh

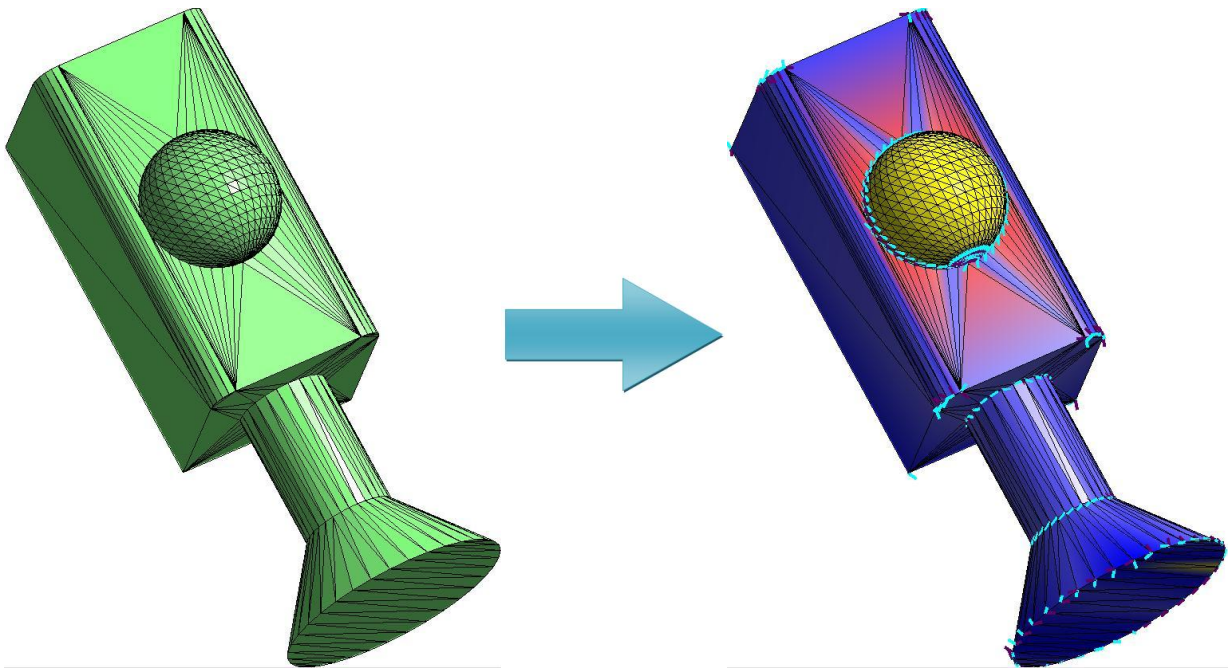
The mesh can not have many point



Sparse Mesh

The mesh can not have many point

$\Rightarrow$ The curvature computation is not correct

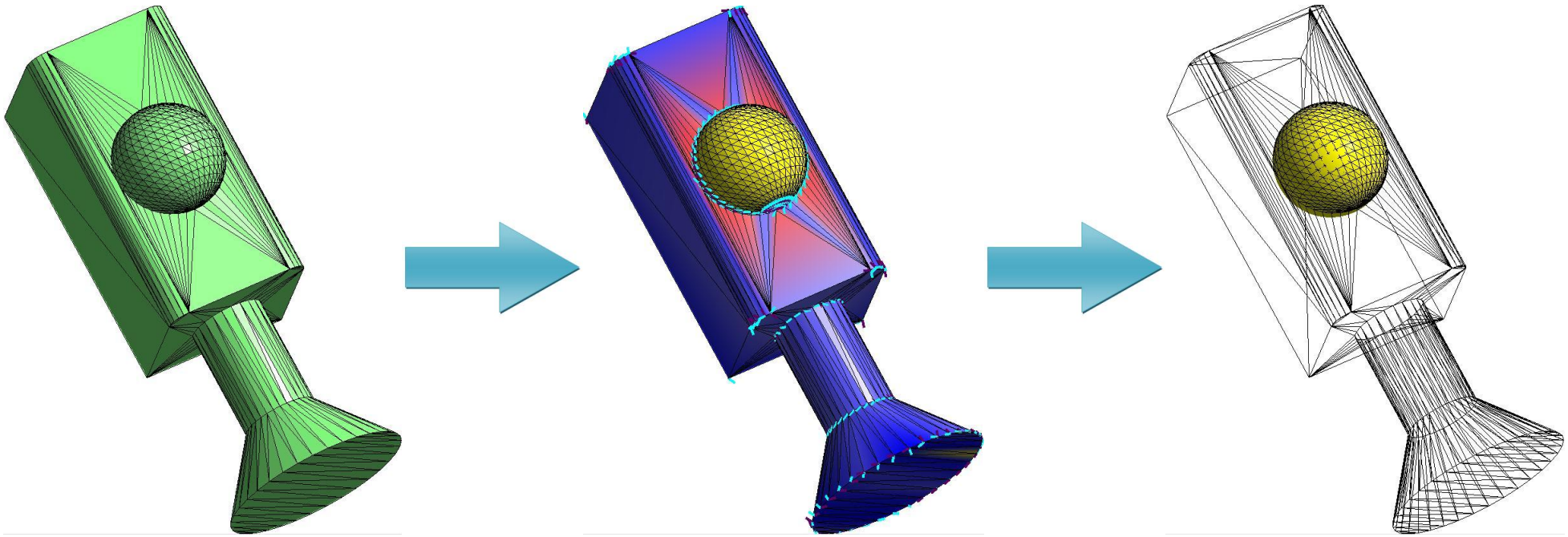


Sparse Mesh

The mesh can not have many point

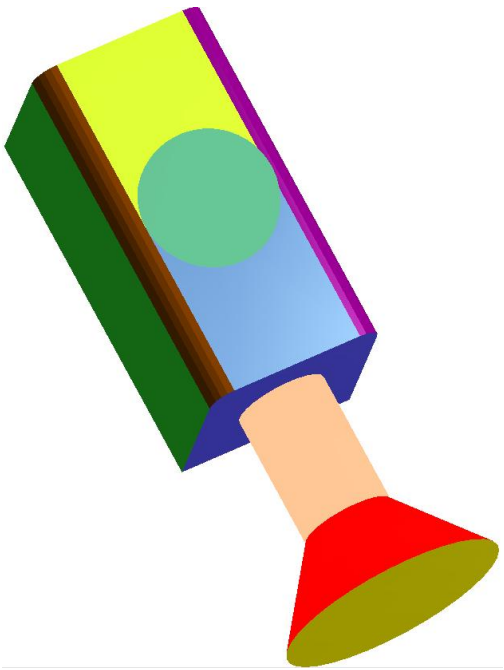
$\Rightarrow$ The curvature computation is not correct

$\Rightarrow$ The primitive extraction is disturbed



Sparse Mesh ⇒ Segmentation

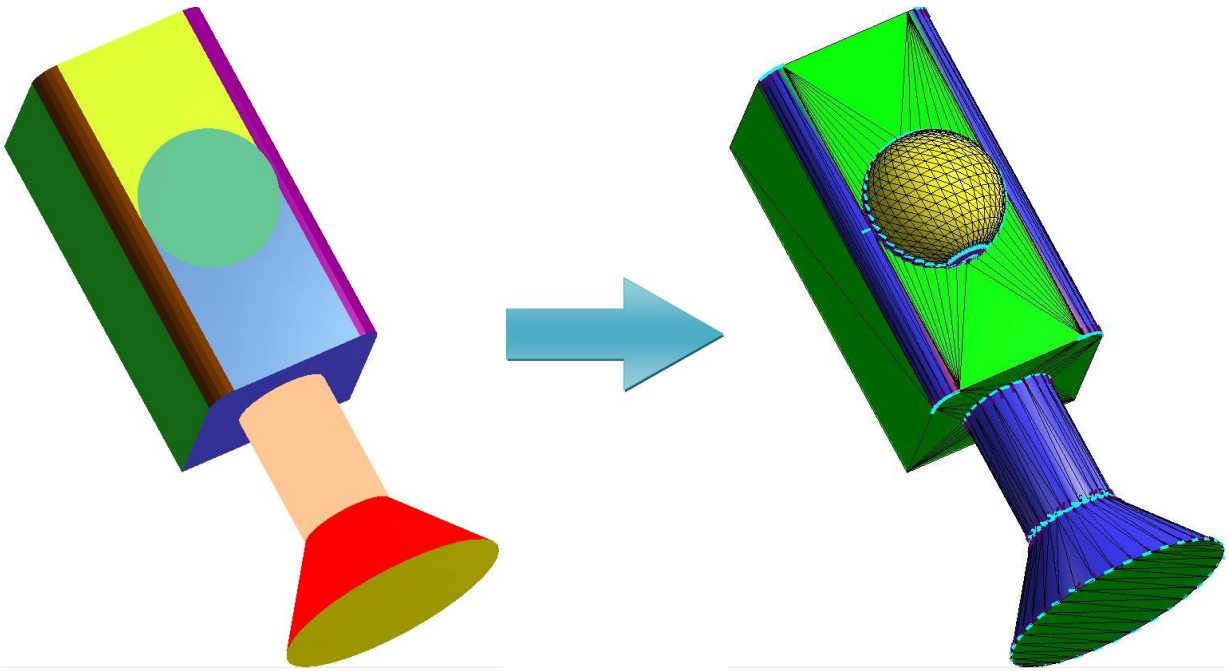
A solution can be to segment the mesh (by the dihedral angle for example)



Sparse Mesh ⇒ Segmentation

A solution can be to segment the mesh (by the dihedral angle for example)

⇒ The curvature computation is better

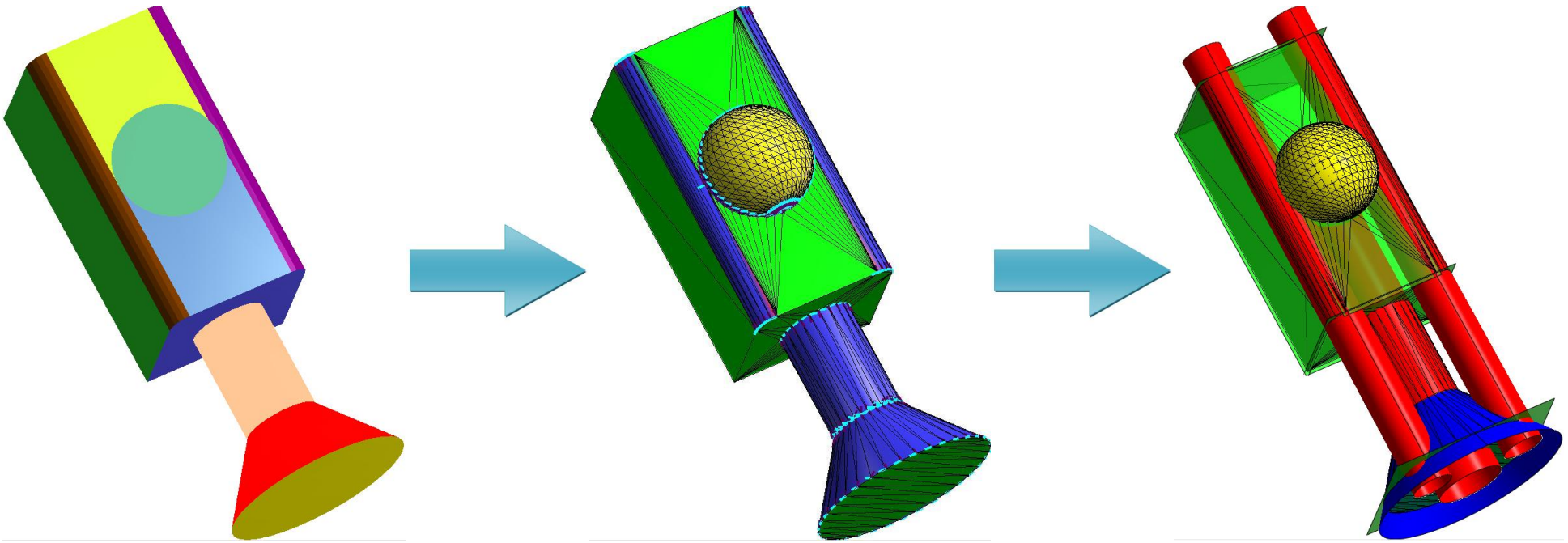


Sparse Mesh ⇒ Segmentation

A solution can be to segment the mesh (by the dihedral angle for example)

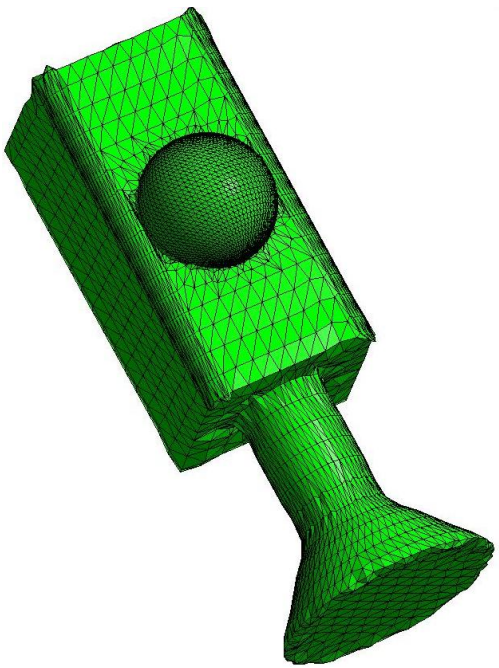
⇒ The curvature computation is better

⇒ The primitive extraction is improved



Noisy Mesh

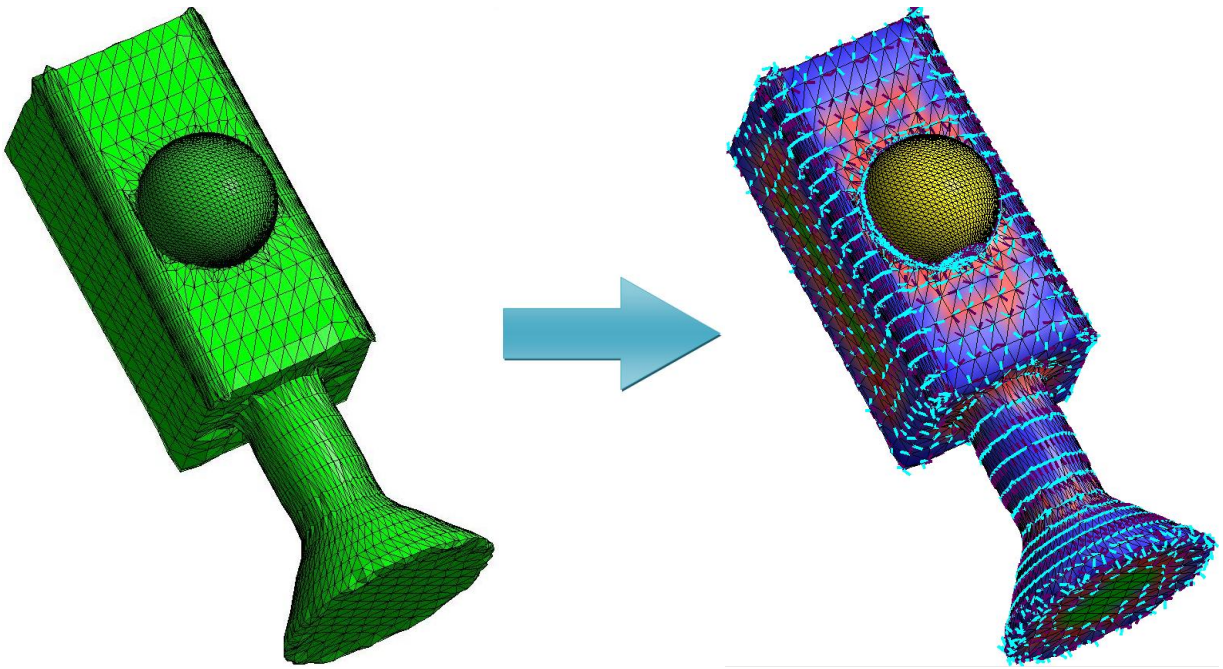
The mesh can not have many point



Noisy Mesh

The mesh can not have many point

$\Rightarrow$ The curvature computation is not correct

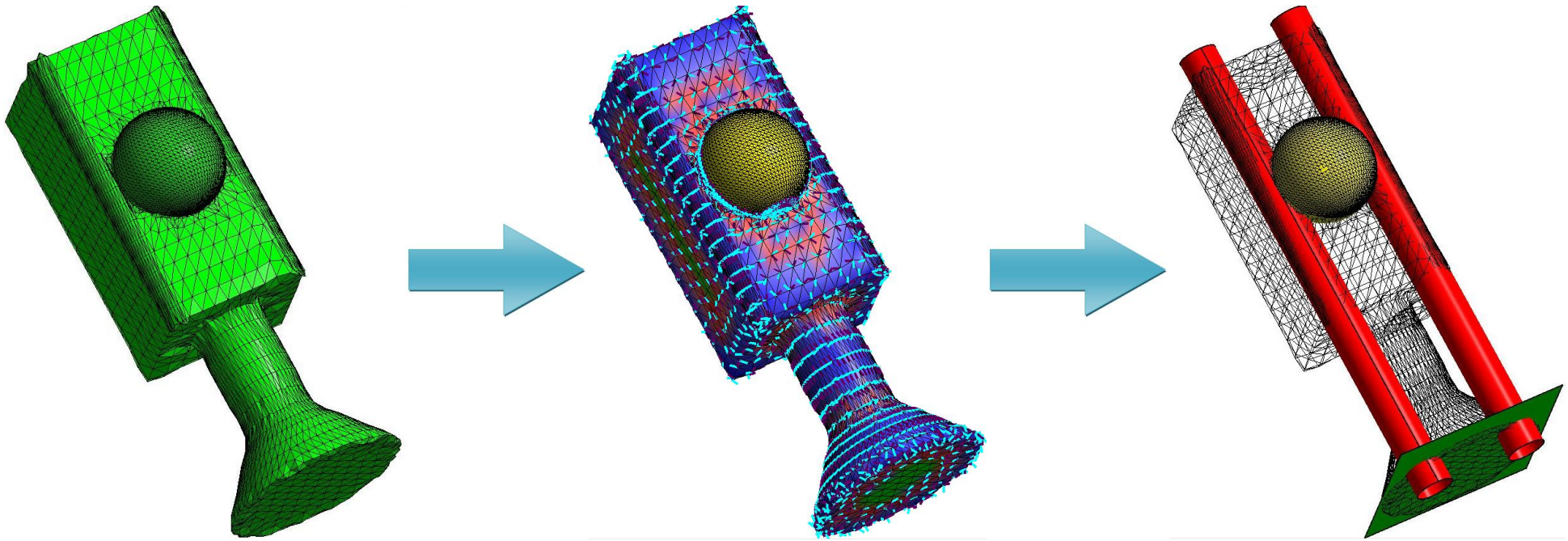


Noisy Mesh

The mesh can not have many point

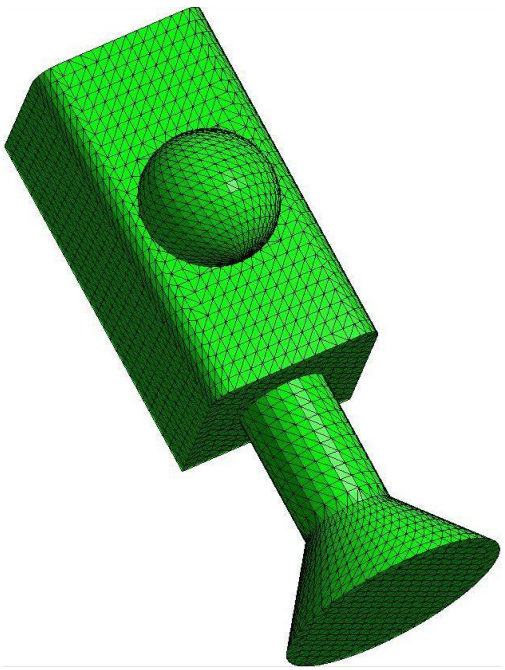
$\Rightarrow$ The curvature computation is not correct

$\Rightarrow$ The primitive extraction is disturbed



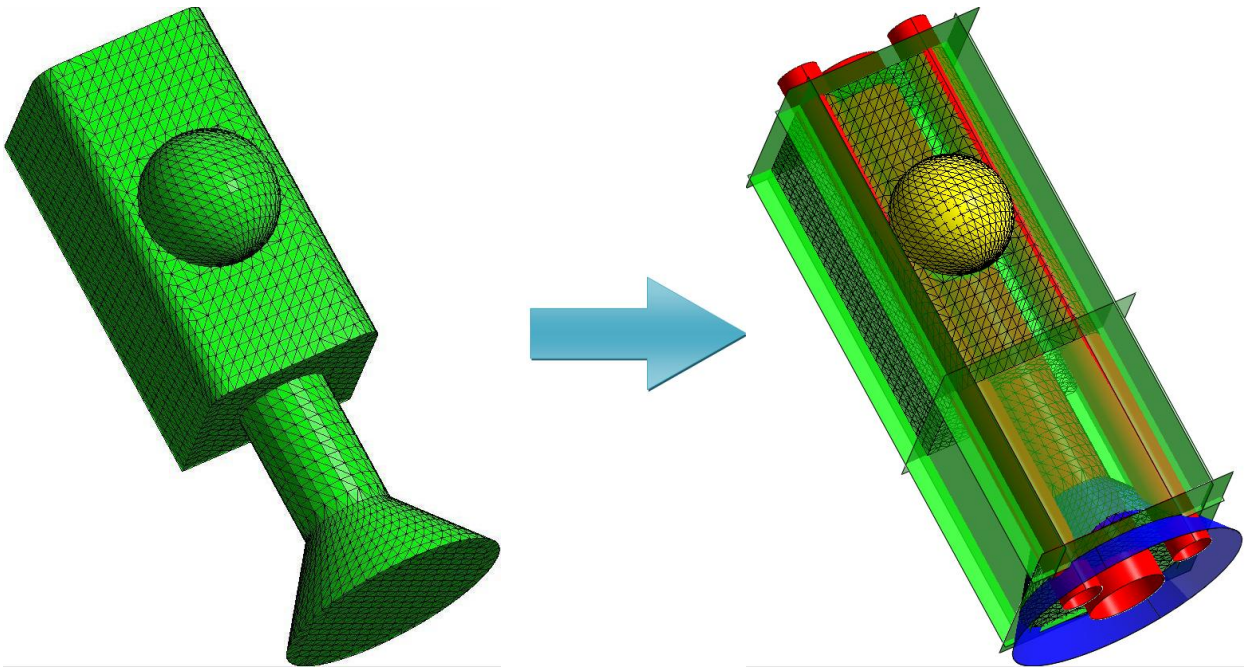
Conclusion and Future Work

Our method take a Mesh



Conclusion and Future Work

Our method take a Mesh $\Rightarrow$ extract primitives.

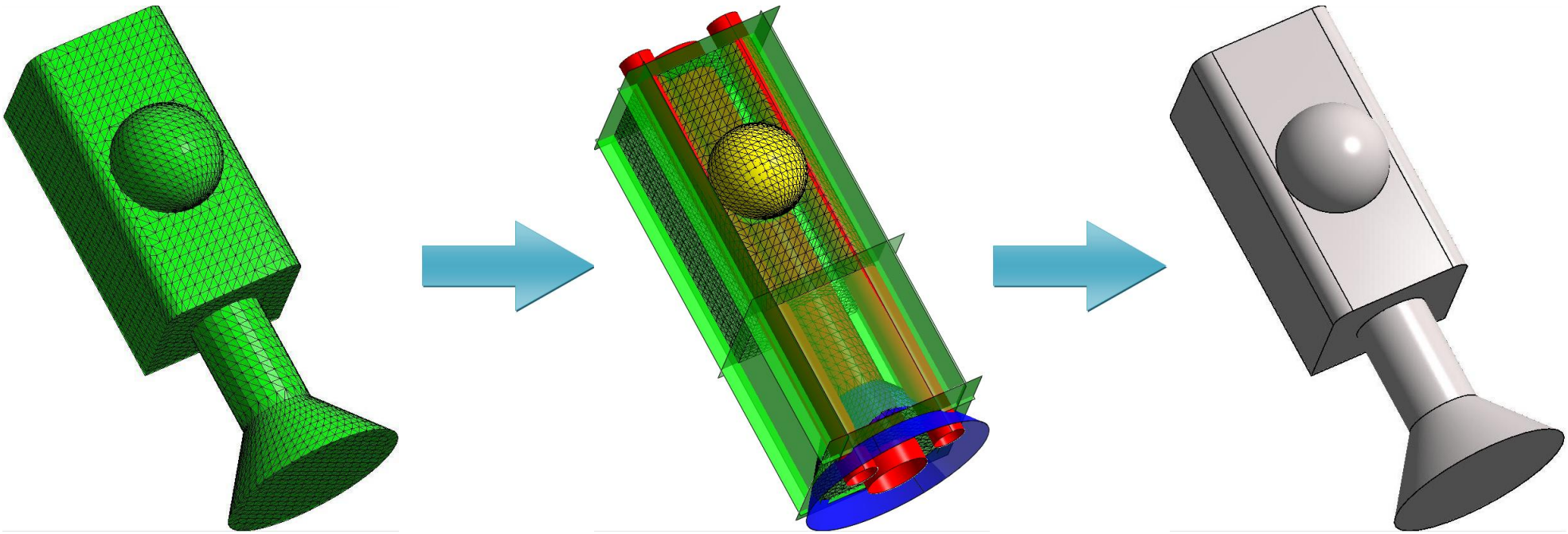


Conclusion and Future Work

Our method take a Mesh $\Rightarrow$ extract primitives.

Future Work

- Cut and Fuse Primitives to reconstruct the continue representation

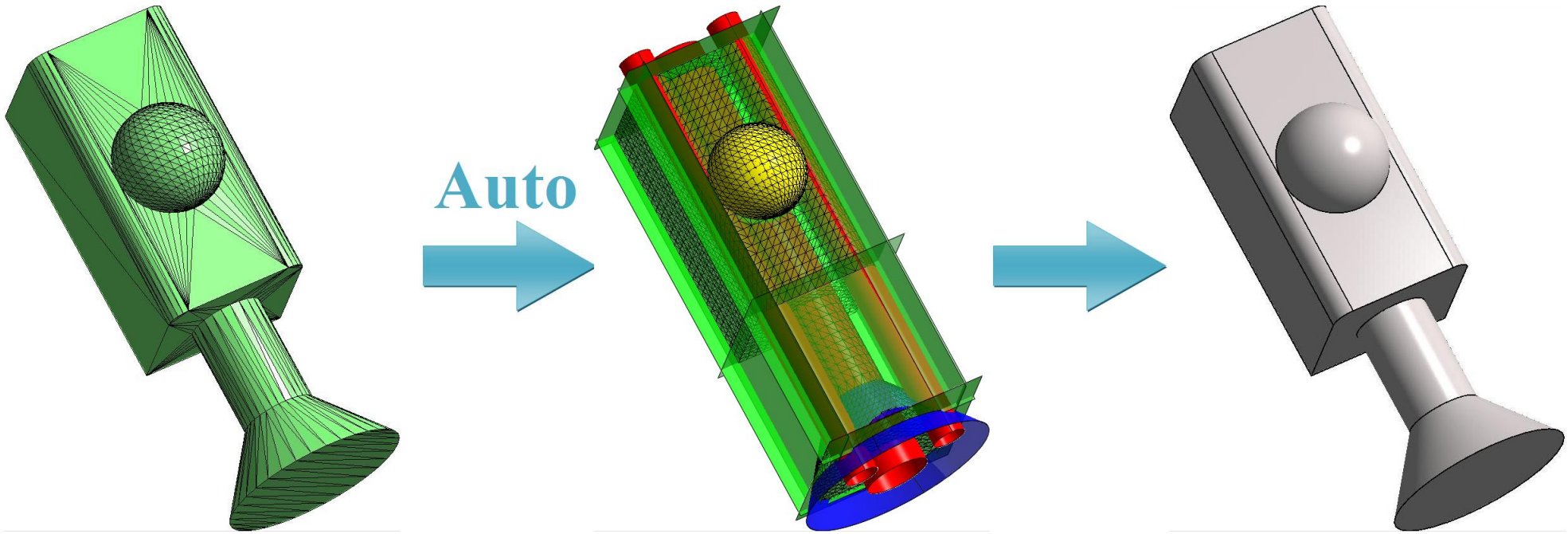


Conclusion and Future Work

Our method take a Mesh $\Rightarrow$ extract primitives.

Future Work

- Cut and Fuse Primitives to reconstruct the continue representation
- Add a Segmentation step to the method

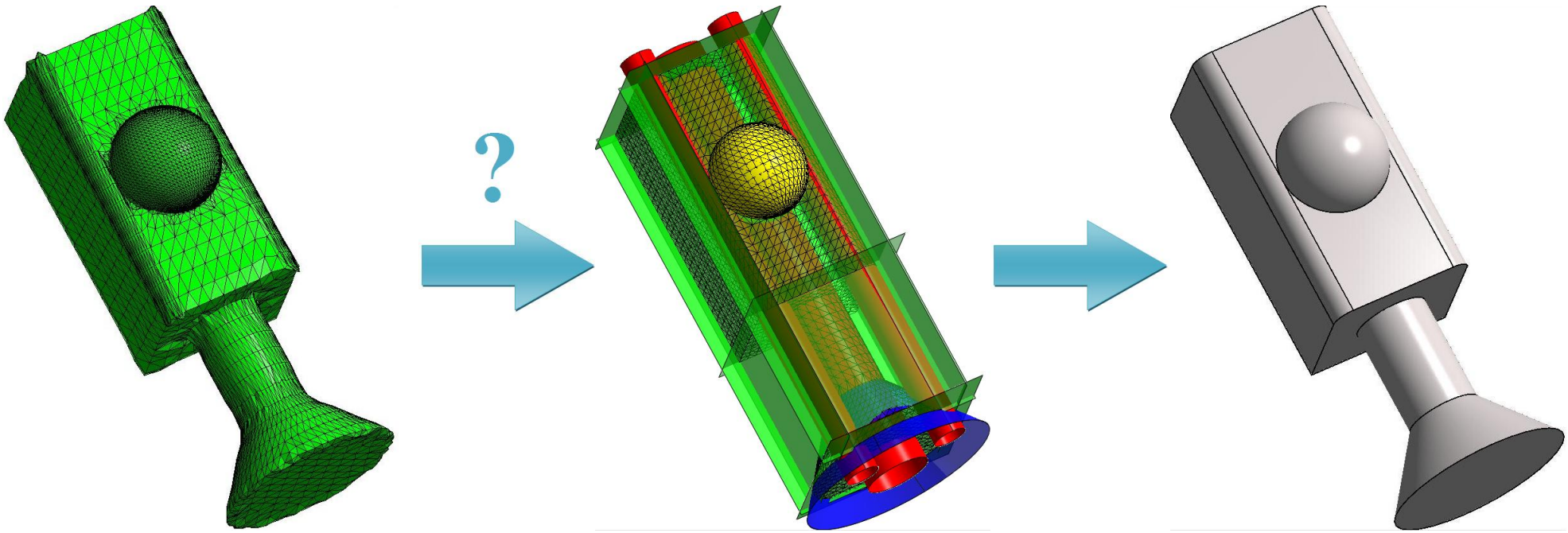


Conclusion and Future Work

Our method take a Mesh $\Rightarrow$ extract primitives.

Future Work

- Cut and Fuse Primitives to reconstruct the continue representation
- Add a Segmentation step to the method
- Deal with noisy mesh



Thanks for your attention

QUESTIONS ?

Site : www.lirmm.fr/~beniere
Mail : roseline.beniere@lirmm.fr
C4W site : www.c4w.com

Roseline Bénière, G. Subsol, G. Gesquière, F. Le Breton and W. Puech,
Recovering Primitives in 3D CAD meshes, SPIE, *San Francisco*, 2011

LIRMM

C4W
Custom CAD Software