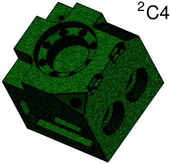
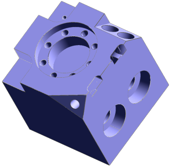
CAD-driven pattern recognition in reverse engineered models

S. Gauthier[1,2] **W. Puech**[1] **R. Bénière**[2] **G. Subsol**[1]

[1] Research-team ICAR, LIRMM, CNRS / Univ. Montpellier, France
[2] C4W, Montpellier, France

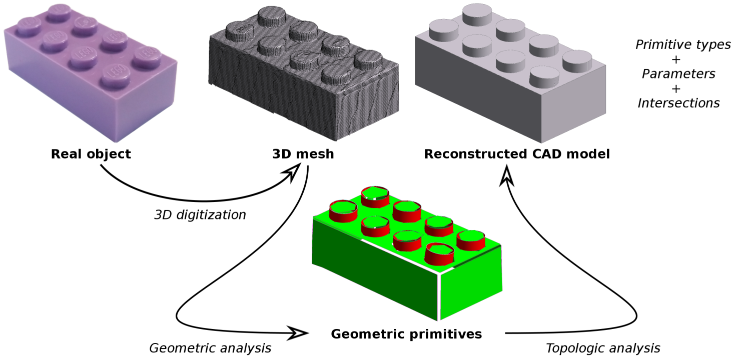
gerard.subsol@lirmm.fr





February 26, 2019

Reverse Engineering



Primitive types
+
Parameters
+
Intersections

Real object

3D mesh

Reconstructed CAD model

3D digitization

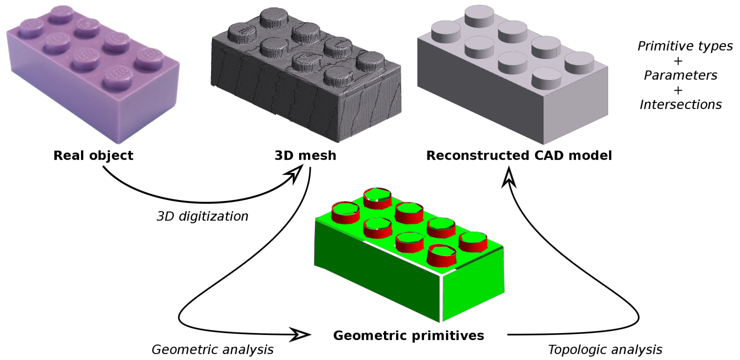
Geometric primitives

Geometric analysis

Topologic analysis

S. Gauthier et al. Analysis of digitized 3D mesh curvature histograms for reverse engineering. Computers in Industry, 2017.

Reverse Engineering



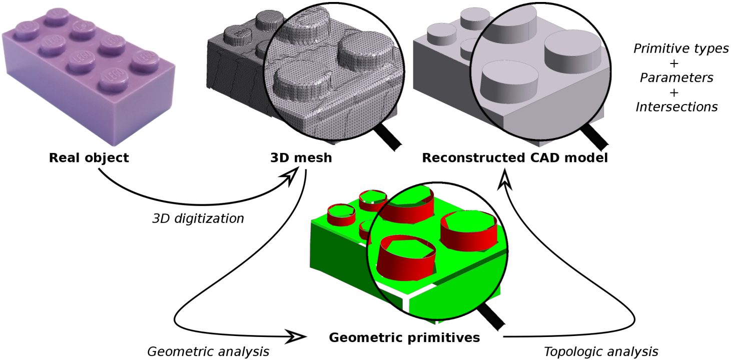
S. Gauthier et al. Analysis of digitized 3D mesh curvature histograms for reverse engineering. Computers in Industry, 2017.

⇒ (Re)create a CAD model (primitives, parameters, intersections)

⇒ Modify an existing object

⇒ Perform non-destructive control

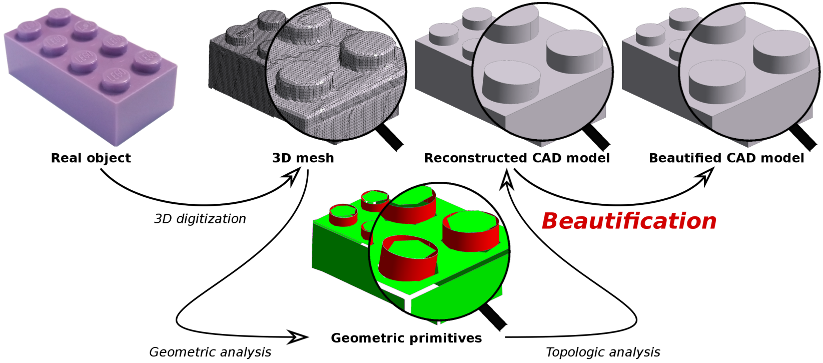
Reverse Engineering



Primitive types
+
Parameters
+
Intersections

⇒ (Re)create a CAD model (primitives, parameters, intersections)
⇒ Modify an existing object
⇒ Perform non-destructive control

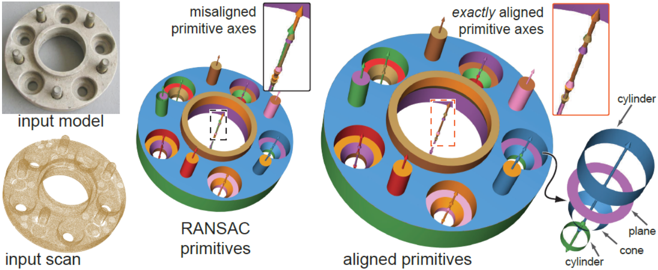
Reverse Engineering



⇒ (Re)create a CAD model (primitives, parameters, intersections)
⇒ Modify an existing object
⇒ Perform non-destructive control

Beautification

F. Langbein. Beautification of reverse engineered geometric models. PhD thesis, Cardiff University, 2003.
C. Gao et al.. Local topological beautification of reverse engineered models. Computer-Aided Design, 2004.
I. Kovács et al.. Applying geometric constraints for perfecting CAD models in reverse engineering. Graphical Models, 2015.
S. Oesau et al.. Planar shape detection and regularization in tandem. Computer Graphics Forum, 2016.
J. Chen and H. Feng. Idealization of scanning-derived triangle mesh models of prismatic engineering parts. International Journal on Interactive Design and Manufacturing (IJIDeM), 2017.

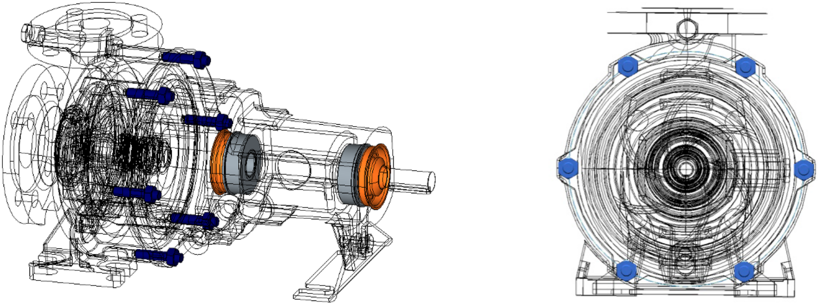


Y. Li et al.. Globfit: Consistently fitting primitives by discovering global relations. ACM Transactions on Graphics (TOG), 2011.

⇒ based on geometric **relationships** between primitives

S. Gauthier et al. Orientation Beautification of Reverse Engineered Models. GRAPP/VISIGRAPP, 2017.

CAD-driven Beautification

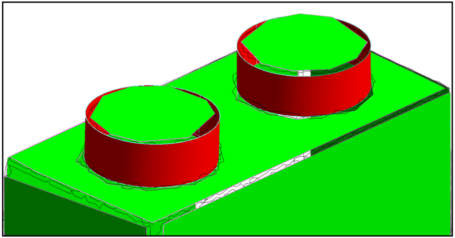


H. Vilmart et al.. From CAD assemblies toward knowledge-based assemblies using an intrinsic knowledge-based assembly model. Computer-Aided Design and Applications, 2018.

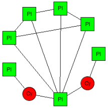
⇒ but beautification may also be based on **CAD knowledge**:

- **Feature**: fixed subset of primitives (*screw/nut*)
  → alignment, dimension constraints...
- **Pattern**: repetition of features (*circular repetition*)
  → position, dimension constraints...

Step 1: Construction of a Relationship Graph



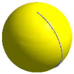
Graph:
- **node = primitive**
- **edge = relationship**



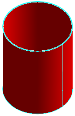
Neighborhood

Primitive parameters:
- **orientations**
- **dimensions**
- **positions**

Plane:
- **position** *(x, y, z)*
- **orientation** *(a, b, c)*



Sphere:
- **position** *(x, y, z)*
- **radius** *r*

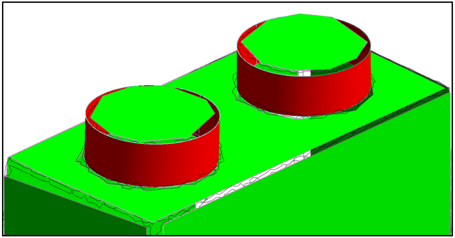


Cylinder:
- **position** *(x, y, z)*
- **orientation** *(a, b, c)*
- **radius** *r*

Cone:
- **position** *(x, y, z)*
- **orientation** *(a, b, c)*
- **angle** *α*

Set of primitives+parameters and the neighborhood relationships.

Step 1: Construction of a Relationship Graph

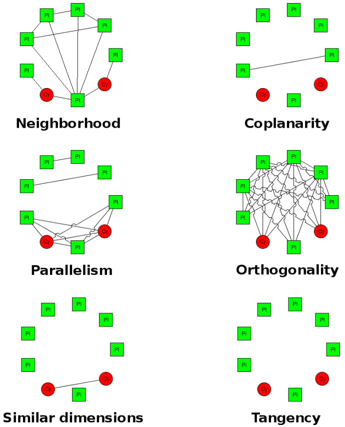


Primitive parameters:
- orientations
- dimensions
- positions

Tolerances:
- angle
- dimension
- distance

Relationships:
- angle between orientations
- difference between dimensions
- distance between positions

Graph:
- node = primitive
- edge = relationship



Neighborhood

Coplanarity

Parallelism

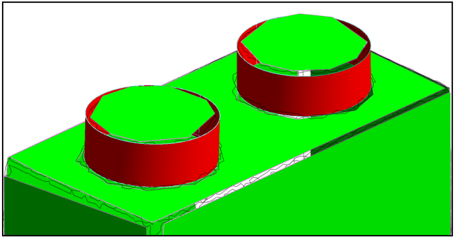
Orthogonality

Similar dimensions

Tangency

→ **Relationship Graph**: geometric relations between each pair of primitives (up to some tolerances).

Step 1: Construction of a Relationship Graph



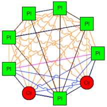
Graph:
- node = primitive
- edge = relationship

Primitive parameters:
- orientations
- dimensions
- positions

Tolerances:
- angle
- dimension
- distance

Relationships:
- angle between orientations
- difference between dimensions
- distance between positions

Relationship Graph

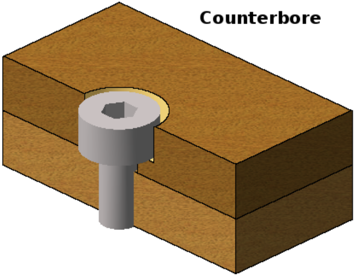


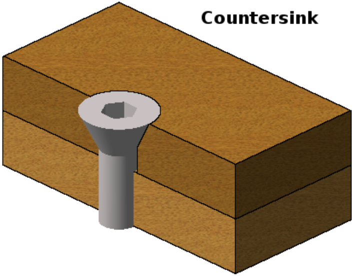
Neighborhood
Parallelism
Orthogonality
Coplanarity
...

→ **Relationship Graph**: geometric relations between each pair of primitives (up to some tolerances).

Step 2: Feature Recognition

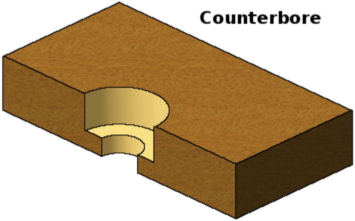
1) Definition of a **Relationship Sub-Graph** for each Feature;

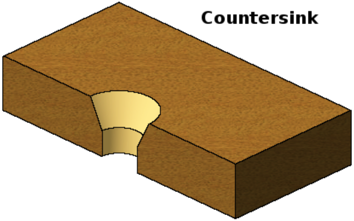




Step 2: Feature Recognition

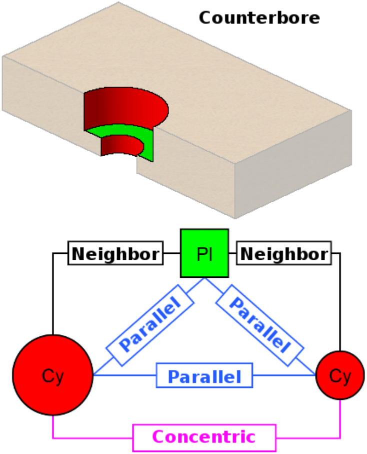
1) Definition of a **Relationship Sub-Graph** for each Feature;

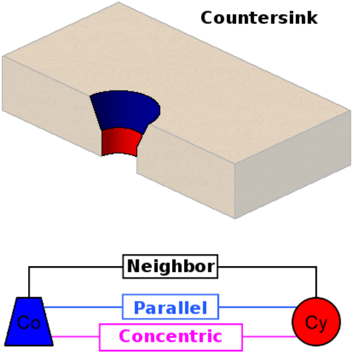




Step 2: Feature Recognition

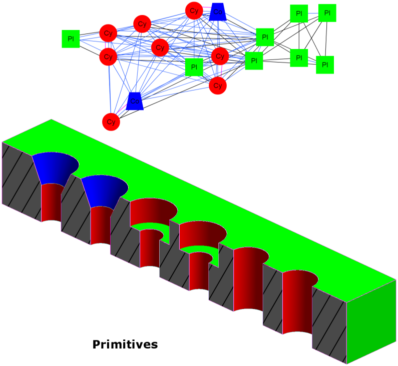
1) Definition of a **Relationship Sub-Graph** for each Feature;





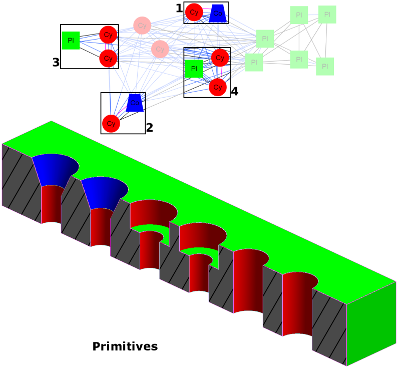
Step 2: Feature Recognition

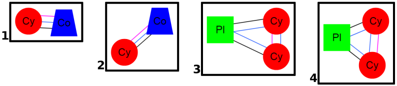
1) Definition of a **Relationship Sub-Graph** for each feature.
2) Recognition of the Sub-Graph in the overall Relationship Graph.



Step 2: Feature Recognition

1) Definition of a **Relationship Sub-Graph** for each feature.
2) Recognition of the Sub-Graph in the overall Relationship Graph.

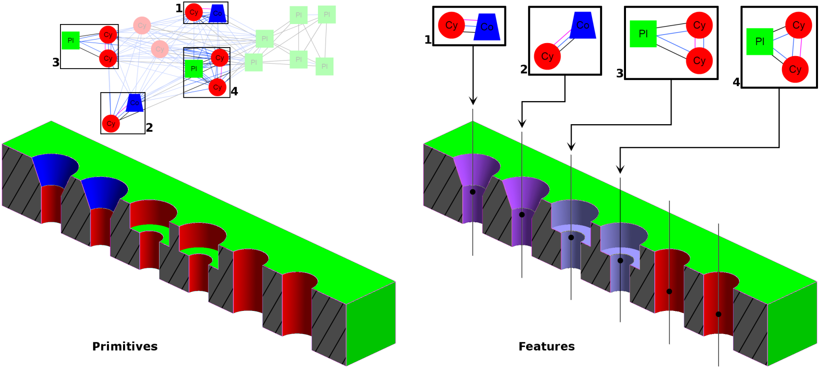




Features are recognized by finding one of their primitives (e.g. a cylinder) and then aligning their Sub-Graph with the Relationship Graph.

Step 2: Feature Recognition

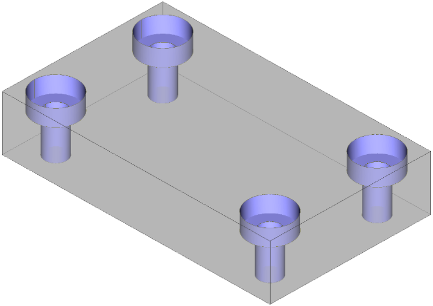
1) Definition of a **Relationship Sub-Graph** for each feature.
2) Recognition of the Sub-Graph in the overall Relationship Graph.

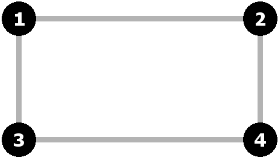


Features are recognized by finding one of their primitives (e.g. a cylinder) and then aligning their Sub-Graph with the Relationship Graph.

Step 3: Pattern Recognition

1) A Feature is then defined by its Sub-Graph type and some parameters (based on primitives constituting it).
2) Define a **Feature Graph** based on the recognized Features (types+parameters) in the Relationship Graph.
3) Based on this Feature Graph, recognize recursively a Pattern of 2 similar Features (same type and parameters (up to some tolerances)).

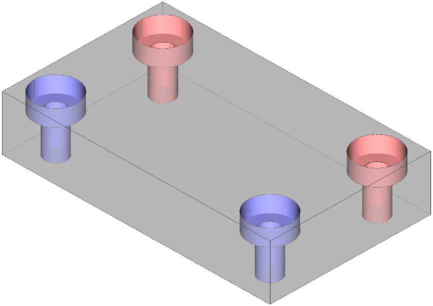


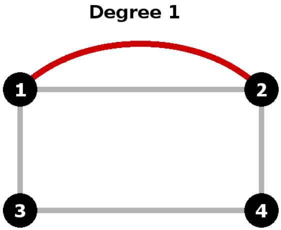


Feature parameters=axis+radius

Step 3: Pattern Recognition

1) A Feature is then defined by its Sub-Graph type and some parameters (based on primitives constituting it).
2) Define a **Feature Graph** based on the recognized Features (types+parameters) in the Relationship Graph.
3) Based on this Feature Graph, recognize recursively a Pattern of 2 similar Features (same type and parameters (up to some tolerances)).

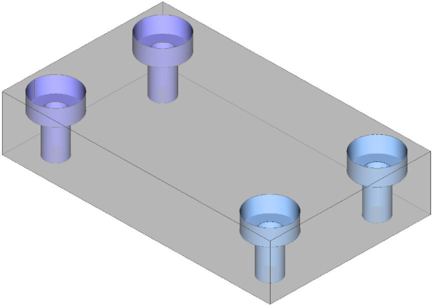


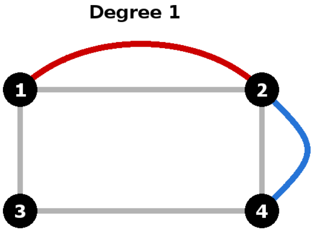


Feature parameters=axis+radius

Step 3: Pattern Recognition

1) A Feature is then defined by its Sub-Graph type and some parameters (based on primitives constituting it).
2) Define a **Feature Graph** based on the recognized Features (types+parameters) in the Relationship Graph.
3) Based on this Feature Graph, recognize recursively a Pattern of 2 similar Features (same type and parameters (up to some tolerances)).

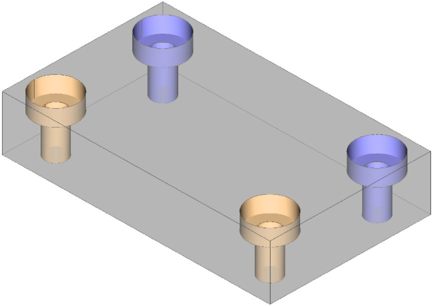




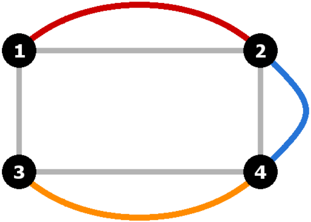
Feature parameters=axis+radius

Step 3: Pattern Recognition

1) A Feature is then defined by its Sub-Graph type and some parameters (based on primitives constituting it).
2) Define a **Feature Graph** based on the recognized Features (types+parameters) in the Relationship Graph.
3) Based on this Feature Graph, recognize recursively a Pattern of 2 similar Features (same type and parameters (up to some tolerances)).



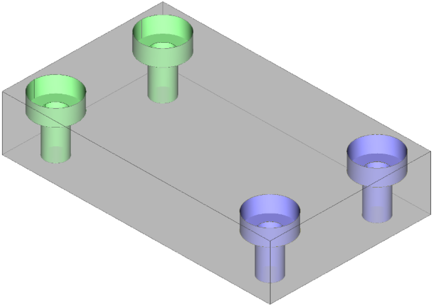
Degree 1

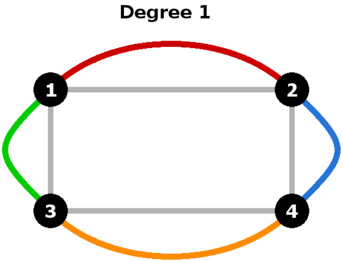


Feature parameters=axis+radius

Step 3: Pattern Recognition

1) A Feature is then defined by its Sub-Graph type and some parameters (based on primitives constituting it).
2) Define a **Feature Graph** based on the recognized Features (types+parameters) in the Relationship Graph.
3) Based on this Feature Graph, recognize recursively a Pattern of 2 similar Features (same type and parameters (up to some tolerances)).

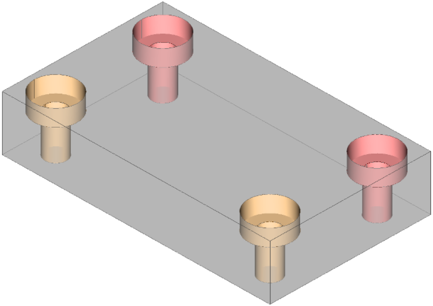


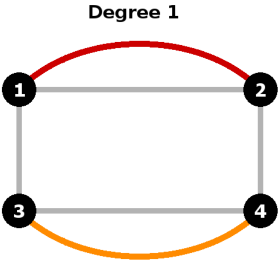


Feature parameters=axis+radius

Step 3: Pattern Recognition

1) A Feature is then defined by its Sub-Graph type and some parameters (based on primitives constituting it).
2) Define a **Feature Graph** based on the recognized Features (types+parameters) in the Relationship Graph.
3) Based on this Feature Graph, recognize recursively a Pattern of 2 similar Features (same type and parameters (up to some tolerances)).

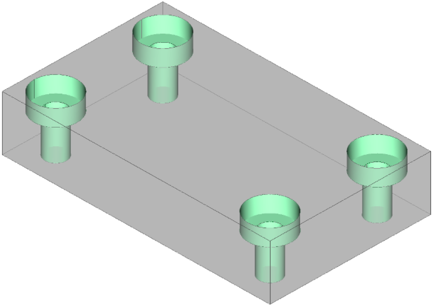


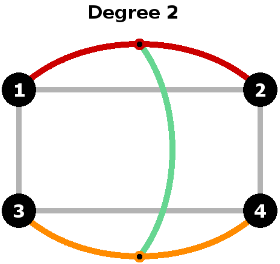


Feature parameters=axis+radius

Step 3: Pattern Recognition

1) A Feature is then defined by its Sub-Graph type and some parameters (based on primitives constituting it).
2) Define a **Feature Graph** based on the recognized Features (types+parameters) in the Relationship Graph.
3) Based on this Feature Graph, recognize recursively a Pattern of 2 similar Features (same type and parameters (up to some tolerances)).

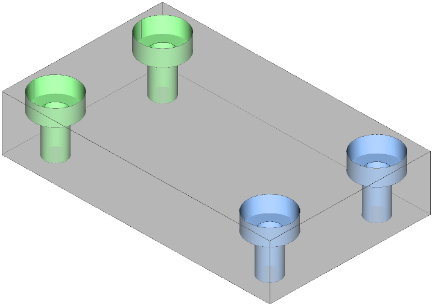


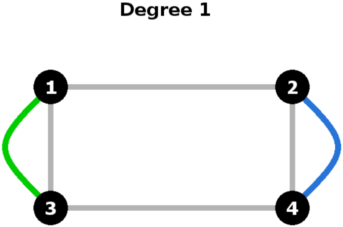


Feature parameters=axis+radius

Step 3: Pattern Recognition

1) A Feature is then defined by its Sub-Graph type and some parameters (based on primitives constituting it).
2) Define a **Feature Graph** based on the recognized Features (types+parameters) in the Relationship Graph.
3) Based on this Feature Graph, recognize recursively a Pattern of 2 similar Features (same type and parameters (up to some tolerances)).

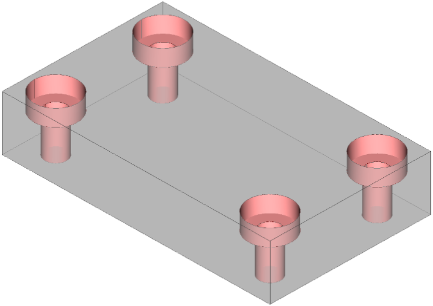


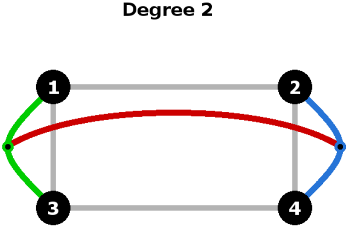


Feature parameters=axis+radius

Step 3: Pattern Recognition

1) A Feature is then defined by its Sub-Graph type and some parameters (based on primitives constituting it).
2) Define a **Feature Graph** based on the recognized Features (types+parameters) in the Relationship Graph.
3) Based on this Feature Graph, recognize recursively a Pattern of 2 similar Features (same type and parameters (up to some tolerances)).

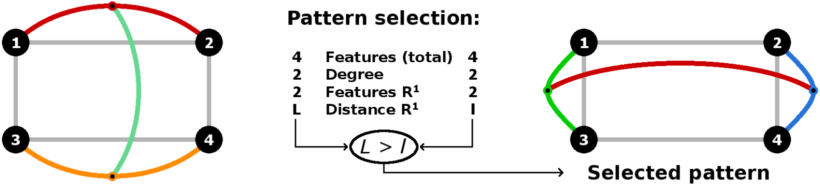




Feature parameters=axis+radius

Step 3: Pattern Recognition

4) If several Patterns are possible, select the optimal one according to some rules.



Pattern selection:

4	Features (total)	4
2	Degree	2
2	Features R^1	2
L	Distance R^1	l

$L > l$

Selected pattern

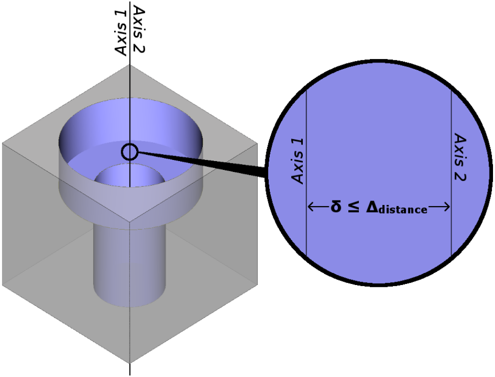
Selection rules:

1. maximize the total number of grouped Features;
2. minimize the degree;
3. maximize the number of Features in Sub-patterns;
4. minimize the distance between the two Features/Sub-patterns.

Application to Beautification

Once features and patterns have been recognized:

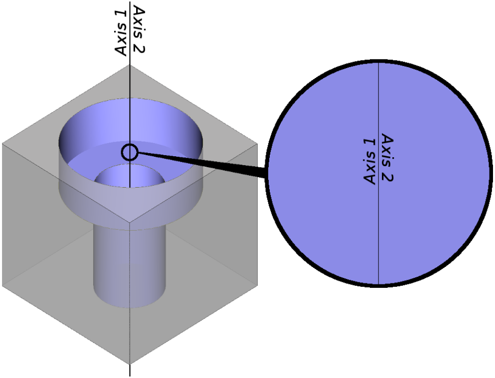
1) Feature: **relative regularization** of all the primitive parameters based on the constraints of the Relationship Subgraph.



Application to Beautification

Once features and patterns have been recognized:

1) Feature: **relative regularization** of all the primitive parameters based on the constraints of the Relationship Subgraph.

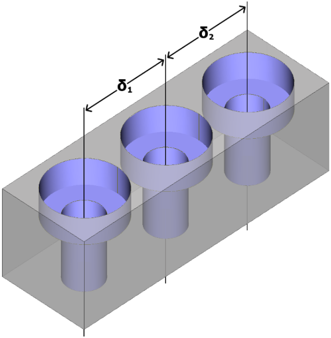


The axes of the two cylinders are aligned within the Feature.

Application to Beautification

Once features and patterns have been recognized:

1) Feature: **relative regularization** of all the primitive parameters based on the constraints of the Relationship Subgraph.
Definition of **Feature parameters**= common parameters.

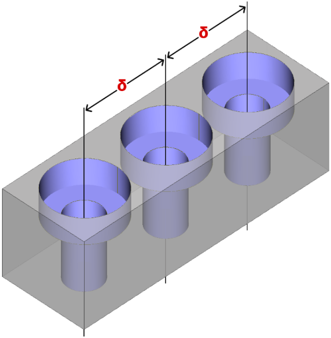


Feature parameters = position+orientation of the common axis.

Application to Beautification

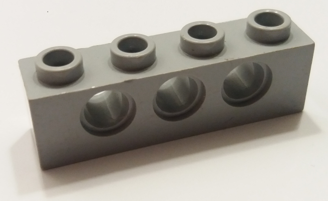
Once features and patterns have been recognized:

1) Feature: **relative regularization** of all the primitive parameters based on the constraints of the Relationship Subgraph.
   Definition of **Feature parameters**= common parameters.
2) Pattern = **global regularization** of the Feature parameters.



Feature parameters = position+orientation of the common axis.
The 3 axes are positioned in the same plane and equidistant.

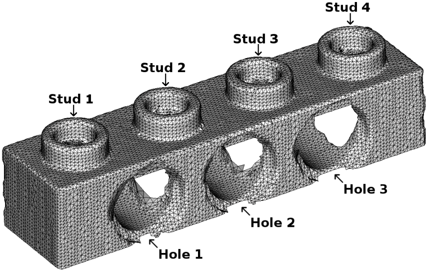
Lego bar



Lego **:**

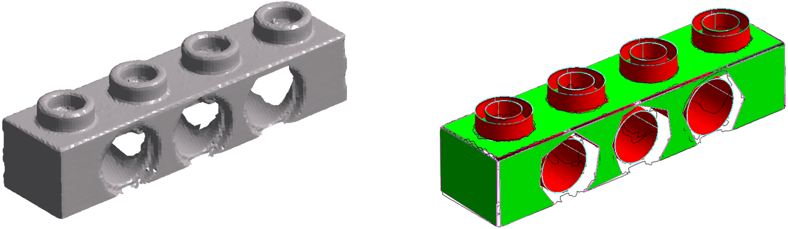
12 500 points
24 371 triangles

31,82 mm long
7,98 mm large
8,59 mm high



3D mesh of a Lego bar with a structured-light scanner (accuracy $\approx$ 100 μm).

Lego bar

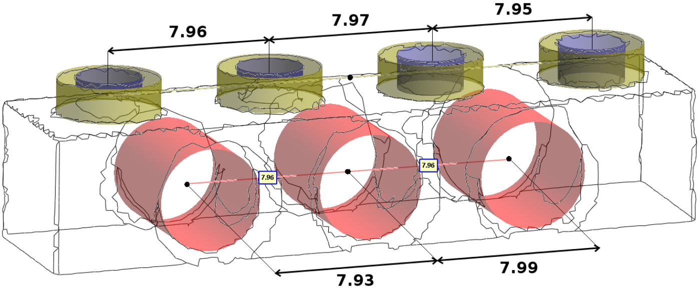


Planes: 5
Cylinders: 11

Detection of primitives and parameters

Plane: position+orientation *Cylinder: position+orientation+radius*

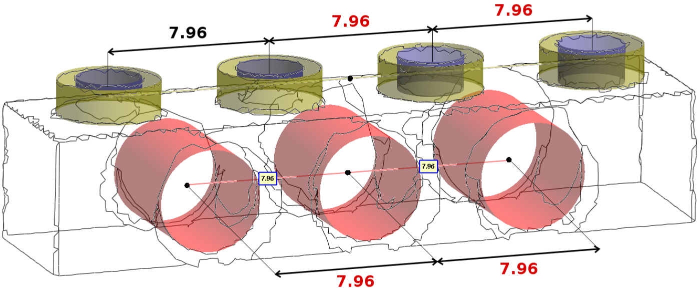
Lego bar



- *Feature 1: 2 concentric cylinders (violet+yellow)*
- *Feature 2: cylinder (pink)*

Feature beautification ⟶ concentric cylinder axes are aligned.

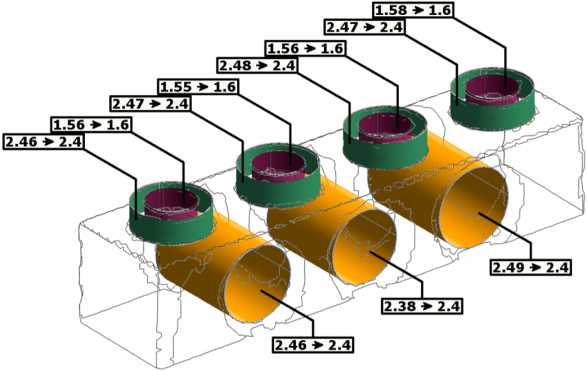
Lego bar



- *Pattern 1: 4 × Feature 1*
- *Pattern 2: 3 × Feature 2*

$\longrightarrow$ *Feature axes are aligned and made equidistant.*

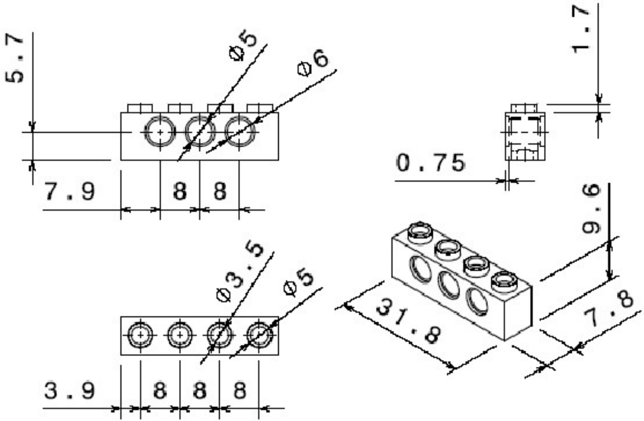
Lego bar



- *Pattern 1: 4 × Feature 1*
- *Pattern 2: 3 × Feature 2*

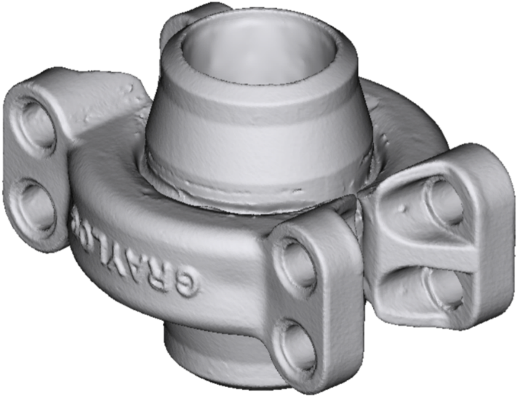
⟶ *Feature axes are aligned and made equidistant.*
⟶ *Feature radii are equalized inside a Pattern.*

Lego bar



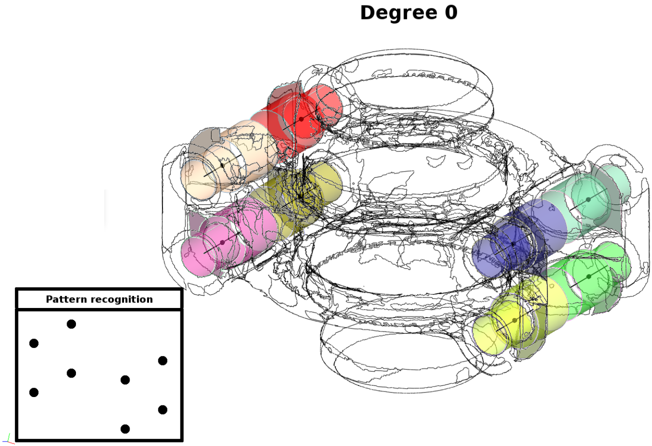
Can be compared with the CAD model for control.

Clamp connector



341 primitives (planes, cylinders, cones).

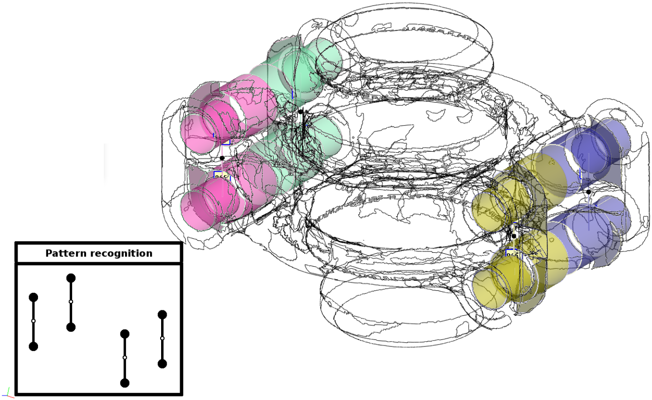
Clamp connector



8 Features ”counterbore”.

Clamp connector

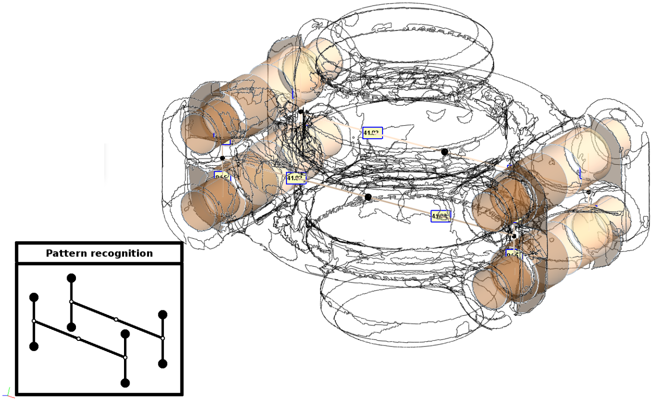
Degree 1



Sub-pattern of 2 Features.

Clamp connector

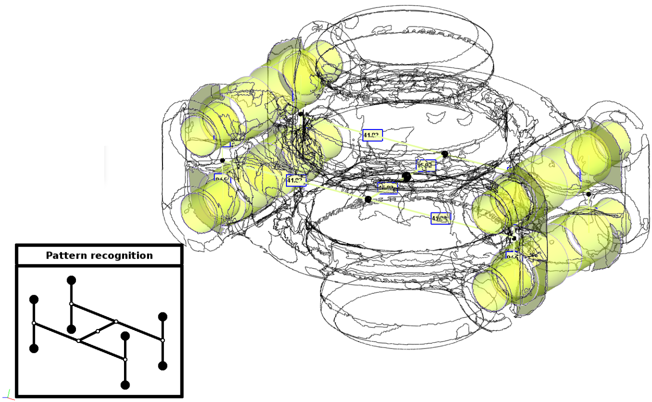
Degree 2



Sub-pattern of 2 Sub-patterns of 2 Features.

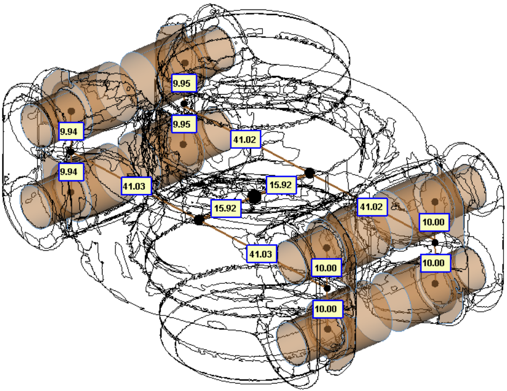
Clamp connector

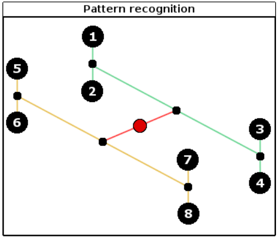
Degree 3



Pattern of 2 Sub-patterns of 2 Sub-patterns of 2 Features?

Clamp connector



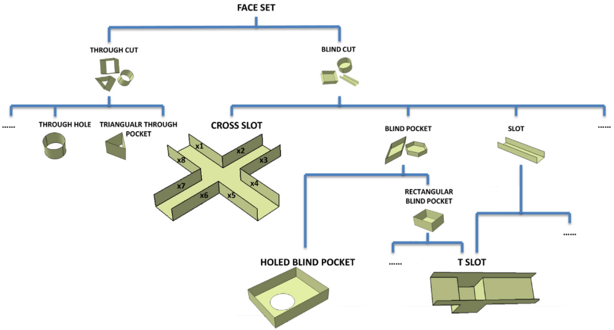


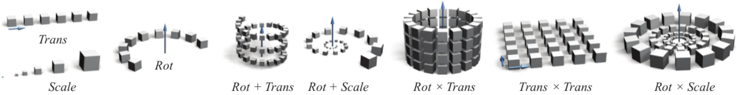
$$R^3(R^2(R^1(p_1^0, p_2^0), R^1(p_3^0, p_4^0)), R^2(R^1(p_5^0, p_6^0), R^1(p_7^0, p_8^0)))$$

Introduction of a new mirror-type Pattern rule
→ *Sub-patterns of opposite orientation.*

Perspectives

Recognize more types of Features and more configurations of Patterns.

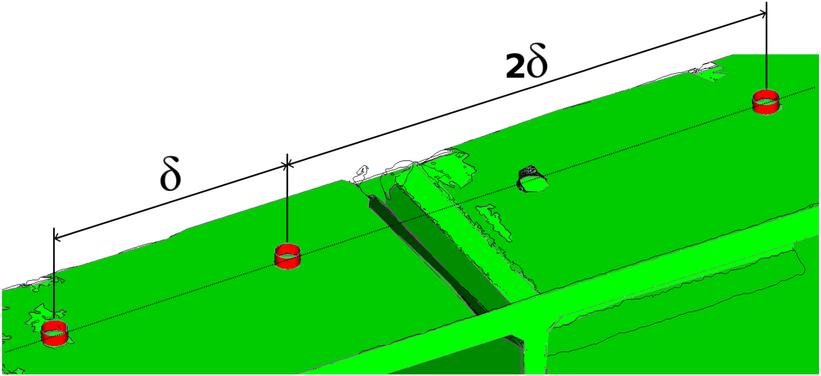


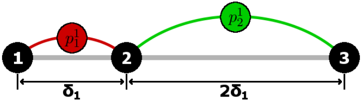


M. Pauly et al.. Discovering structural regularity in 3D geometry. ACM Transactions on graphics, 2008.
Q. Wang and X. Yu. Ontology based automatic feature recognition framework. Computers in Industry, 2014.

Perspectives

Formulate hypotheses to infer primitives.

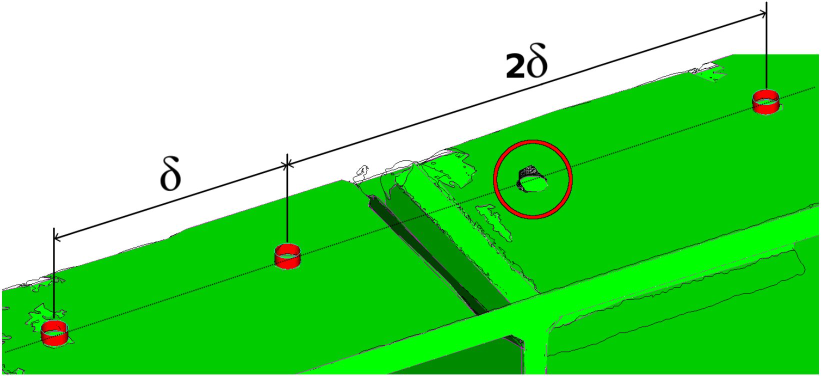


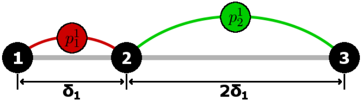


$$\left.\begin{array}{lcl} p_1^1 & = & \{p_1^0, p_2^0\} \end{array}\right\} \delta_1$$
$$\left.\begin{array}{lcl} p_2^1 & = & \{p_2^0, p_3^0\} \end{array}\right\} 2\delta_1$$

Perspectives

Formulate hypotheses to infer primitives.

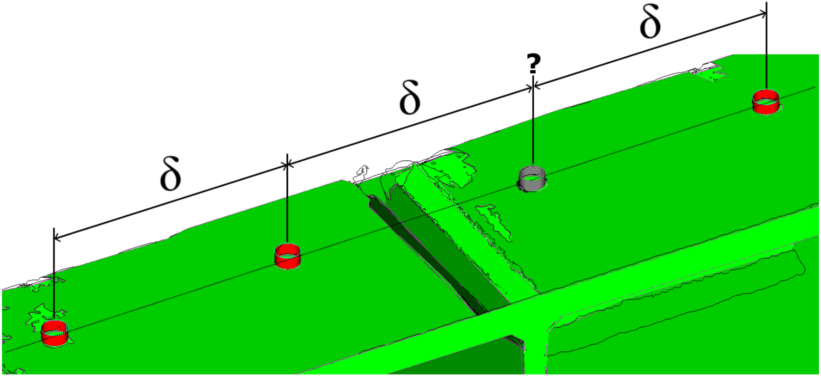


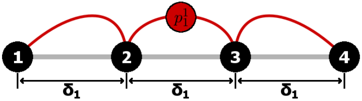


$$p_1^1 = \{p_1^0, p_2^0\} \Big\} \; \delta_1$$
$$p_2^1 = \{p_2^0, p_3^0\} \Big\} \; 2\delta_1$$

Perspectives

Formulate hypotheses to infer primitives.

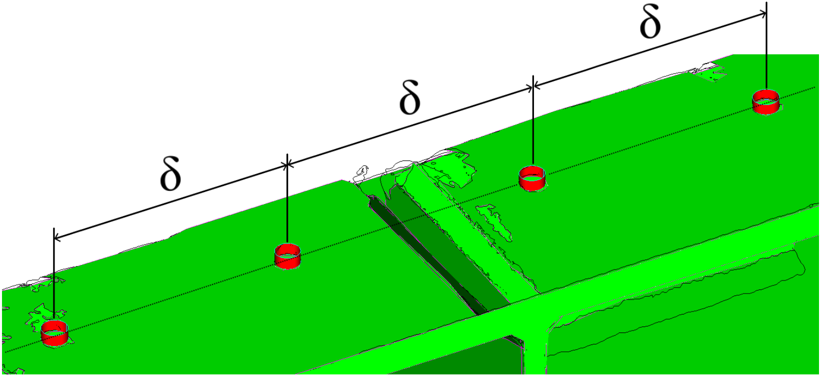


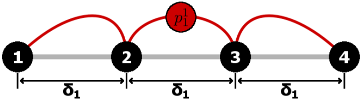


$$p_1^1 \;=\; \{p_1^0, p_2^0, p_3^0, p_4^0\} \;\Big)\; \delta_1$$

Perspectives

Formulate hypotheses to infer primitives.

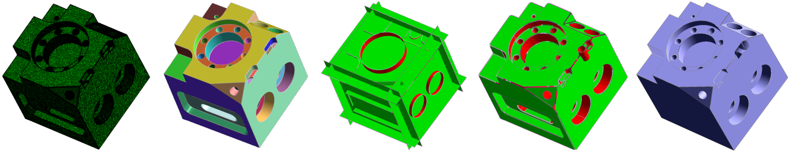




$$p_1^1 \;=\; \{p_1^0, p_2^0, p_3^0, p_4^0\} \;)\; \delta_1$$

Thank you

Some questions?



Email: gerard.subsol@lirmm.fr

Silvère Gauthier, W. Puech, R. Bénière, **G. Subsol**,
CAD-driven pattern recognition in reverse engineered models, 2019

UNIVERSITÉ DE MONTPELLIER LIRMM Icar cnrs C4W