Reusing heterogeneous data for the conceptual design of shapes in virtual environments — Source link ☒

Zongcheng Li, Franca Giannini, Jean-Philippe Pernot, Philippe Veron ...+1 more authors

Institutions: Arts et Métiers ParisTech

Related papers:

- Towards a new approach for the description of shapes from multimodal data

- Virtual reality based solid modeling

- Example-based conceptual styling framework for automotive shapes

- Function-based representation of complex geometry and appearance

- Scale-aware black-and-white abstraction of 3D shapes

Reusing heterogeneous data for the conceptual design of shapes in virtual environments

Zongcheng Li, Franca Giannini, Jean-Philippe Pernot, Philippe Veron, Bianca Falcidieno

Reusing heterogeneous data for the conceptual design of shapes in virtual environments

Zongcheng Li[1,2] · Franca Giannini[2] · Jean-Philippe Pernot[1] · Philippe Véron[1] · Bianca Falcidieno[2]

Abstract Today, digital data such as 2D images, 3D meshes and 3D point clouds are widely used to design virtual environments (VE). Most of the time, only one type of those multimodal data is used to describe and specify the shapes of the objects. However, a single object can be seen as a combination of components linked with constraints specifying the relationships and the rigid transformations defining their arrangement. Thus, the definition of new methods able to combine any kind of multimodal data in an easy way would allow non-experts of VE to rapidly mock up objects and scenes. In this paper, we propose a new shape description model together with its associated constraints toolbox enabling the description of complex shapes from multimodal data. Not only rigid transformations are considered but also scale modifications according to the specified context of the constraint setting. The heterogeneous virtual objects (i.e., composed by scalable multimodal components) then result from the resolution of a constraint satisfaction problem through an optimization approach. The proposed approach is illustrated and validated with examples obtained using our prototype software.

1 Introduction

Due to the great advances in acquisition devices and modeling tools, a huge amount of digital data (e.g., images, videos, 3D models such as point clouds, meshes or B-Rep models) is becoming now available in various application domains. In particular, virtual environments (VE) make use of those digital data allowing more attractive and more effectual communication and simulation of real or not (yet) existing environments and objects. Despite those available datasets and possibly associated metadata, the design of application-oriented VE still results from a long and tedious iterative modeling and modification process that involves several actors (e.g., experts of the application domain, 3D modelers and virtual reality (VR) programmers, designers or communications/marketing experts). Depending on the targeted application, the number and the profiles of the involved actors may change. Today's limitations and difficulties are mainly due to the lack of strong relationships between the expert of the domain having creative ideas, the digitally skilled actors, the tools and the shape models taking part to the VE development process. Actually, existing tools mainly focus on the detailed geometric definition of the shapes and are not suitable to

✉ Jean-Philippe Pernot
Jean-Philippe.Pernot@ensam.eu

Zongcheng Li
Zongcheng.Li@ensam.eu; Zongcheng.Li@ge.imati.cnr.it

Franca Giannini
Franca.Giannini@ge.imati.cnr.it

Philippe Véron
Philippe.Veron@ensam.eu

Bianca Falcidieno
Bianca.Falcidieno@ge.imati.cnr.it

[1] LSIS UMR CNRS 7296, Arts et Métiers ParisTech, Aix-En-Provence, France

[2] IMATI-CNR, Genoa, Italy

effectively support the collaboration between the experts in the creative process of the VR environment and the related assets specification. In addition, the huge amount of available digital data is not fully exploited. Clearly, those data could be used as a source of inspiration for new solutions, being innovative ideas frequently coming from the (unforeseen) combination of existing elements (Sawyer 2013). Therefore, the availability of software tools allowing the reuse and combination of such digital data would be an effective support for the conceptual design phase of both single shapes and VR environments.

The process of generating new ideas can be perceived as the first and most critical part of the creative design. Most of the time, the innovative ideas are produced by iterating back and forth between multiple sources. Smith (1998) has summarized 172 methods for generating ideas. Actually, the most creative ideas are coming from copying and combining existing things (Sawyer 2013). Earlier, Albert Einstein was used to say about his thoughts: "Words do not play any role in my thought; instead, I think in signs and images which I can copy and combine (Albert Einstein)."

However, even if taking ideas from different compositions, to mentally combine and rearrange them together with specific relations and structures is very common and popular in creative conceptual design, existing digital tools weakly support such a modeling process. Some approaches are appearing in 3D shape modeling. Jain et al. (2012) have set up a system to create new shapes by blending between shapes taken from a database. Similar approaches can also be found in sketch-based modeling and search system (Xie et al. 2013; Lee and Funkhouser 2008). However, such a combination is difficult when combining heterogeneous data being represented in terms of different elements and at different levels of information granularity. This paper addresses such a difficult problem of combining multimodal data, and possibly associated semantics, in a unified way so as to stimulate the creativity of the end users and ease the generation of conceptual models in the early design phases of the VE development process. The aim of our proposal is definitively not to replace the existing tools but rather to find a way to combine their outputs in an efficient and unified way so as to stimulate the creativity of the end users.

The paper is organized as it follows. Section 2 reviews the state of the art in this domain. The proposed generic shape description model is then introduced in Sect. 3 together with its constitutive elements. The GSDM modeler and its associated interface are presented in Sect. 4. This new modeling approach is then illustrated and validated in Sect. 5 through several examples mixing different heterogeneous data according to different scenarios. The last section concludes this paper and gives directions for future work.

2 Related works

2.1 Toward modeling with heterogeneous data

Overall, the idea of combing different types of documents and information is not completely new. In product modeling, PDM (product data management) systems aggregate different digital documents related to a specific product and its associated lifecycle. In construction industry, BIM (building information modeling) aims at including and formalizing all the data (possibly of different dimensionality) involved in the design, construction and usage of buildings and infrastructures. Similarly, GIS (geographical information systems) are sophisticated systems dealing with heterogeneous data, combining 2D vector and raster data with 3D data as well as text information to represent and analyze geospatial data. They are generally supported by database systems, and all the considered elements are georeferenced. Thus, their positioning is rather standard and because of their usage almost no interaction with the shape elements is applied. Unfortunately, the above systems and methods do not really refer to combining heterogeneous data in the sense that we intend to do, i.e., with each specific type of input considered as a part of the shape of a single object. Moreover, no specific attention is paid to give easy user interaction capabilities for the reciprocal adjustment of the elements and their simultaneous manipulation.

However, various research attempts (Pernot et al. 2008; Allègre et al. 2006; Antonelli et al. 2013) and current commercial 3D system developers are working on combining different 3D representations to take advantage of their properties for shape manipulation and to allow the exploitation and reuse of existing models. These systems deal with 3D data, while text is used mostly for annotation purposes, and 2D images are often applied as textures to 3D models. Texture mapping has made it possible to simulate near-photorealistic 3D models in real time. This is a very common way to represent 2D and 3D shapes altogether. In most 3D video games, 2D planar surfaces with transparent texture mapping are usually used together with closed 3D surfaces to simulate objects (e.g., grass, leaves or trees). Similarly, CAD (computer-aided design) systems accept the simultaneous usage of images and 3D models either for rendering purposes or for simulating a shape. For example, to model buildings or industrial installations, images can be used to represent internal or environmental furnishings of 3D building components. Image-based modeling (IBM) techniques are also used for reconstructing architectural models (Jiang et al. 2009; El-Hakim 2002; Deluca et al. 2006; Panchetti et al. 2010). Other approaches try to improve the way shapes can be defined interactively

in a VR environment (Wendrich 2009–2016). However, they do not necessarily focus on the conceptual design phase where the user is more interesting in the combination of several multimodal data rather than on the generation of a final 3D model to be used for further animations. Finally, texts can be used today to design new shapes, but their semantic meaning is somehow treated as descriptive sentences to describe new shapes (Décriteau et al. 2016).

Actually, today's limitations are due to the lack of generic shape description models able to support the combination and simultaneous modification and interaction of heterogeneous data in a unified environment. As such, the purpose of this work is to propose a new shape description model able to handle heterogeneous data and easy to be generated, manipulated and modified.

2.2 Structure-based shape descriptors

Most shape description techniques consider information related to the contours and/or regions of the shape. Some of these techniques may transform 2D/3D space coordinates into another space to get useful information. Color and light information are also used to describe a shape. Additional algorithms have been developed providing structure-based and more meaningful descriptors. As those techniques are typically used for extracting information from well-defined shapes, their main applications are shape classification and retrieval. However, some of these techniques can be potentially used for modeling new shapes, especially graph- or structure-based techniques, which might turn out to be very useful to align or assemble several shapes (Mitra et al. 2013).

A shape skeleton (or topological skeleton) is a thin version of a shape, obtained from points, which hold the same distance to its boundaries. There are several mathematical definitions used in the literature to define a skeleton. Different algorithms have been applied to compute it. The concept of skeleton is also interchangeable as "medial axis" and "thinning." Reeb graphs represent the evolution of the level sets of a real-valued function defined over the surface bounding the object (Reeb 1946; Biasotti et al. 2008). A Reeb graph, as it strongly preserves the topological information of a shape, has been widely used in different areas. If the function used to calculate Reeb graph is on a special flat space, then the results form a polytree which is also named contour tree. As skeletons, Reeb graphs are also helpful for image segmentation.

Those graph-based shape descriptors have a strong potential usage to define or align shapes. For example, the one straight segment of a skeleton may represent the major orientation axis of this shape. If the shape is used and relocated in another 3D space, then its skeleton is very useful to set the orientation of this shape.

As a consequence, the proposed approach should take advantage of such potentially available structure-based descriptors to ease the generation, modification and interaction of the combined heterogeneous data. Technically, it means that the proposed framework should support the specification of relationships either at the level of the shapes themselves (i.e., contours and regions) or at the level of the associated shape descriptors. How to do it in a unified way is explained in the next section.

2.3 A multilayered shape understanding paradigm

Shape representations and description techniques have shown different ways of capturing information from shapes with different aspects. Those features can also be considered as different characteristics for understanding the information associated with shapes. With the development of computer graphics and its application domains, the meaning of "Shape" has become richer.
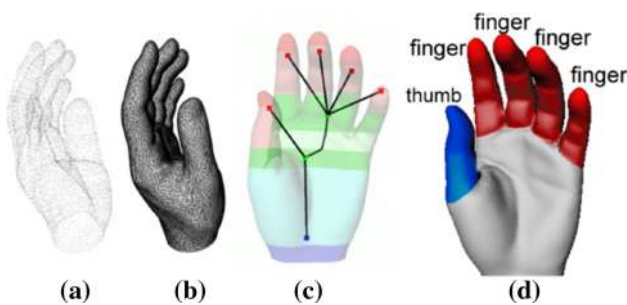
A shape can be defined by "Parts" and "Relations" (Luciano da Fontoura and Roberto Marcondes Cesar 2000). The shape is seen as a set of parts that are spatially arranged through the spatial relations among them. These relations among the shape parts can be classified in different ways (Bloch 1999; Hudelot et al. 2008). A possible classification includes the following three types of relation: topological, distance and directional (Takemura 2008). Topological relation, such as "inside," "outside" and "adjacent," is invariant to rotation and scaling transformation. Distance relation is linked to quantitative measures. If two shapes are "far from" or "close to," each other needs to be further specified. Directional relation is characterized by the orientation of angle-based aspects following some reference such as the medial axis, or the segments of the border of a 2D shape.

In 2004, the European project AIM@SHAPE (Falcidieno et al. 2004; Repository 2011–2015) proposed a new way of understanding shapes. Shape is any individual object having a visual appearance, which exists, in some (two-, three- or higher-dimensional) space such as pictures, sketches, images, 3D objects, videos, 4D animations. Shapes are characterized by several properties. They have a geometry (the spatial extent of the object), they can be described by structures (object features and part-whole decomposition), they have semantics (meaning, purpose), and they may also have some interaction with time (e.g., history, shape morphing, animation, video). Finally, they are endowed with attributes (colors, textures, names, attached to an object, its parts and/or its features).
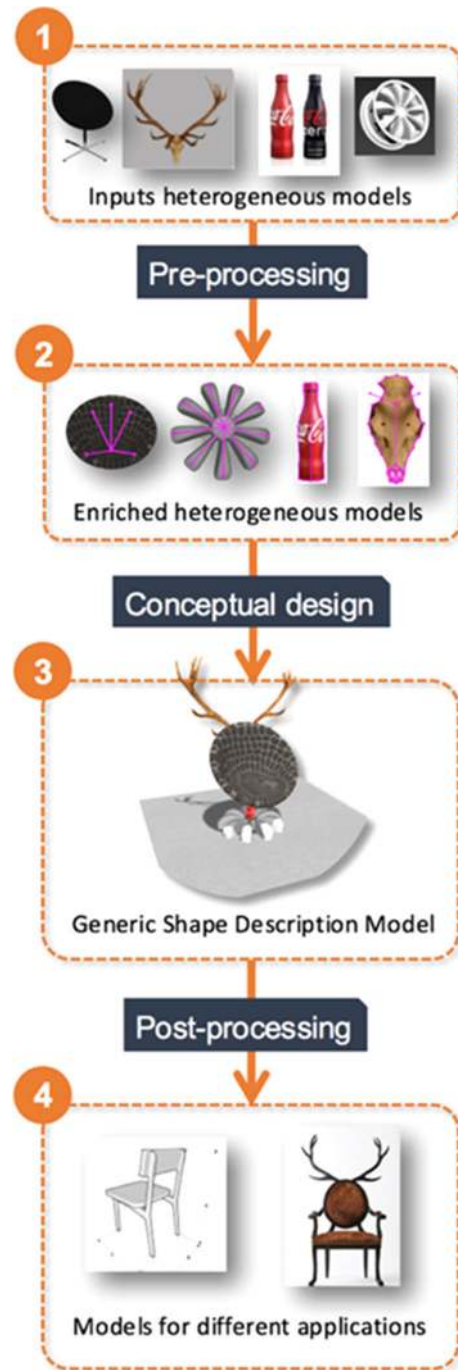
Compared with the definition of Luciano da Fontoura and Roberto Marcondes Cesar (2000), this interpretation

corresponds to a broader view of shape. The shape parts and their relations can be considered as the structure of shape. Their appearance features such as their colors and textures are grouped as the attributes of the shape. This definition also associates semantics to shapes, which can be used for semantic-based retrieval processes. With this definition, the information associated with shapes can be structured into three different layers including geometric, structural and semantic information levels (Fig. 1).

The generic shape description model (GSDM) proposed in this paper has been designed based on those three layers. This enables the possibility to describe multimodal data in a same structure, so as to be able to combine together all the data whatever their representations are. The GSDM is detailed in the next sections. It is part of a framework presented in Fig. 2 and used to generate objects for VR applications. Starting from a set of heterogeneous models found on Internet or in any available database (1), a preprocessing phase enriches the models using existing approaches such as skeletonization, cropping, contouring (2) and prepare them for the conceptual design phase. During the conceptual design phase, the GSDM (3) is generated, modified and manipulated using models, methods and tools presented in this paper and detailed in the next sections. Then, the GSDM, which combines enriched heterogeneous models, can be transformed and adapted for different applications (4) using existing approaches such as the creation of meshes from images or the merging of meshes and so on. Again, the aim of the proposed approach is not to replace existing modeling tools used during the pre- and post-processing phases but to advantageously make use of their potential in a unified way so as to foster the emerging of new ideas by a combination of existing objects. Thus, the pre- and post-processing phases will not be part of this paper.

Fig. 2 Generic shape description model (GSDM) as part of a modeling framework enabling conceptual design of objects from enriched heterogeneous data

3 Generic shape description model (GSDM)

The so-called generic shape description model (GSDM) has been designed to support the specification and modification of shapes composed by several heterogeneous data. This section introduces this new model together with a set of models and tools used to generate and manipulate it.

Fig. 1 Digital shape represented by two different geometric descriptions: a point cloud (**a**) and a triangular mesh (**b**); the structure of the hand model, defined as the configuration of main body with protrusion-like features (**c**); the corresponding semantically annotated model exploiting its structure (Repository 2011–2015)

3.1 Overview

The GSDM is structured in three levels of information: conceptual level, intermediate level and data level.

At the conceptual level, three basic elements are defined to describe the object's constitutive parts as well as the underlying relations: Component, Group and Relation. Components correspond to the raw and potentially enriched heterogeneous data used as input of the conceptual design phase. A Group indicates a shared behavior or meaning among Components, while Relation explains the topological, distance and directional relations between either Components or Groups or Components and Groups. They help the non-expert user to provide an overview of what is going to be described, without requiring a precise specification. Thus, the idea is to work with high-level functionalities that will act on lower levels (intermediate and data levels). The conceptual level is further described in Sect. 3.3.

To have a detailed description of each part, the data level is needed. This level describes a part of an object through three types of information: Geometry, Structure and Semantics. The first two provide information concerning the appearance of a part, whereas the third refers to its meaning. This level is only partially handled by the non-expert user. This level is detailed in Sect. 3.2. In this paper, it is assumed that the heterogeneous models used as input of the conceptual design phase went through a preprocessing phase wherein such data level is set up.

The intermediate level is then introduced as a mean to specify the relation between each part. At this level, the specific geometric and structural information of the constitutive Components and Groups are exploited and linked with specific constraints. For example, two Components/Groups are connected by indicating one's location related to the other. At this level, the whole geometry or the whole structure of each part is necessarily accessible to the user. To set up those relations, some Key Entities have to be specified to identify the anchorage elements where restrictions on the related locations are defined. Limitations on reciprocal locations between Key Entities are indicated as Constraints. All those information are gathered together at the intermediate level that is further detailed in Sect. 3.4. Ideally, the end user does not access it in a direct manner but through the specification of Relations at the conceptual level. The different levels and associated concepts are represented in Fig. 3. From the users' point of view, the modeling approach is top-down in the sense that they prefer to work at the conceptual level with Components, Groups and Relations. The user can also operate at the intermediate level to detail the object' sub-parts arrangements. To ease the instantiation at the different levels, specific mechanisms have been imagined and will be
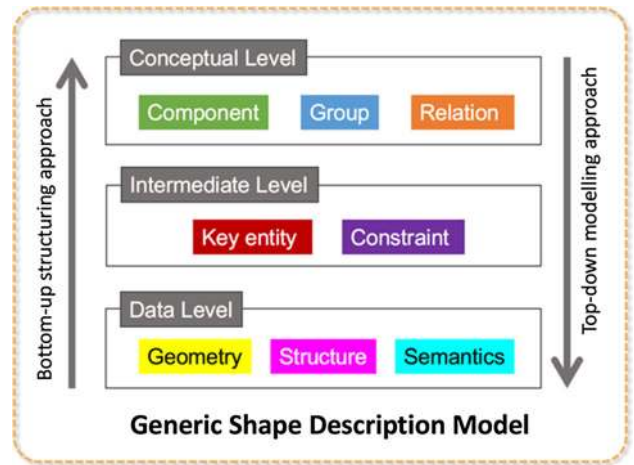
Fig. 3 Bottom-up structure of the GSDM associated with a top-down modeling approach

presented in the next sections and notably the so-called smart manipulation/positioning and smart constraining functionalities. The data level is used to represent the heterogeneous information, to specify Key Entities, to visualize Components and to modify the shapes of Components if needed. Constraints and Key Entities are based on the data level and have their own structures. Overall, the structure of the GSDM is bottom-up since lower levels are used as input of higher levels.

3.2 Data level (Geometry, Structure and Semantics)

3.2.1 Geometry

The shape of an object can be represented by different geometric representations. Heterogeneous geometric representations can be used when specifying the GSDM of an object. No assumption is done on the type of data that can be used to represent a shape. This means that at this level, vector and raster 2D and 3D data are addressed altogether, which is different from existing techniques. All those representations are put in the same 3D reference frame where all the manipulations (translation, rotation and scaling) are performed as described in the next sections. Moreover, the GSDM supports multi-representations and multi-resolutions. It means that a given object can be represented by continuous or discrete representations, in 2D or 3D and at different levels of details, all those representations being stored in the GSDM. Examples of 2D and 3D geometric representations are given in Fig. 4.

3.2.2 Structure

Structures can be defined as the relationships between the different parts of an object and are represented by
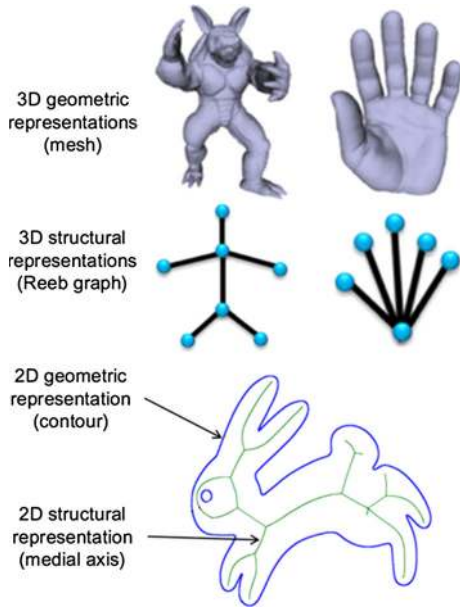
Fig. 4 Examples of geometric and structural representations

structure-based shape descriptors. In the proposed approach, such descriptors are used to help the user to position different parts possibly defined by heterogeneous data. Therefore, not only users can define relations between the parts and their associated geometric representations, but they can also make use of the available structures to specify how the parts behave in relation to each other.

Different kinds of structural representations, such as the medial axis, the symmetry axis, the Reeb graph, the skeleton, are suitable for helping the user to specify the relative positioning of the heterogeneous parts constituting an object. Compared to what exists in the CAD domain, our approach is not restricted to the use of a limited set of entities such as the axis of a cylinder or the center point of a circle that can be constrained together to perform the assembly of different parts. In our approach, more complex descriptors can be used to specify the relations between the components to be combined/assembled. Actually, our approach is not limited to the use of geometric entities defining the geometric model itself. It can use any structure that can be extracted from a given geometric representation. This is a strong difference when compared to other existing approaches.

During the conceptual design phase, the easy manipulation of the structure of an object is very helpful to sketch rapidly how the different components taking part to the GSDM have to be organized and connected. Figure 4 illustrates some examples of 3D and 2D geometric representations (Geometry) as well as possible associated structure-based descriptors (Structure). In this way, the geometric models are segmented and their different parts can easily be constrained through their structures.

Finally, the Structure defined in the GSDM also has an associated Geometry that can be also heterogeneous. At the data level, the Structure of the GSDM can be a combination of different shape descriptors, such as the medial axis or any segmentation obtained from the corresponding geometric representations, either 2D or 3D.

3.2.3 Semantics

Semantics is the purpose and meaning of an instance or an action in a specific context. For example, it can be used to either define the names of the parts used in the conceptual design or specify the intention of an instance or an action. Considering our GSDM, different data are reorganized together following user-specified rules. Most of the time, these rules are associated with meanings explaining why this action is done. For example, the user wants to put two parts together. This action of putting things together can be associated with the purpose of geometrically merging the two parts into one, or it can have the purpose of assembling them such that each part maintains its individuality.

Semantics can also be used for further design phase or information retrieval. There are two kinds of semantics: intrinsic and extrinsic. Intrinsic semantics express the meaning of something that can be obtained directly from it. For example, a surface can be considered as cylindrical if the distances between all the points on the surface to an axis are equal. The intrinsic semantics in the GSDM can be the "type" of the geometry or structure. This intrinsic information can be extracted from the original data with more or less complex computations. Extrinsic semantics refers to additional information independent of the original data under a specific context. For example, some additional information such as the color, the material, the name, the function, the role in the overall object (e.g., chair, seat, legs) can be added to the representations depending on the context. This information is not contained in the instance and therefore has to be attached/added to it. Both intrinsic and extrinsic semantics are stored in the GSDM and can be manipulated following specific rules (e.g., propagation, inheritance), which are not detailed in this paper.

3.3 Conceptual level (Component, Group, Relation)

As explained in Sect. 3.2, the data level contains the low-level information needed for the definition, structuring, understanding as well for the visualization and manipulation of shapes. However, in our system, the data level is not directly operated by the users who are focusing on the overall conceptual specification of the object to be designed and not in fine-tuning the underlying final geometric models. The user is more focusing on the part-whole decomposition of the object to which behaviors can be

directly associated. This motivates the need of a conceptual level manipulated directly by a non-expert to specify the decompositions into Components, Groups and Relations between them.

3.3.1 Component

In the design context, a Component is a part of an object which reorganizes together some (possibly one) Geometric and/or Structural representations so as to represent a part with a basic semantic meaning.

Components are indivisible parts in the sense that they will not be further decomposed. The user can define a part according to its functions or any other purpose. A part can also be split into more parts. When the decomposition of a part offers enough information, or when a further decomposition is meaningless, the user stops decomposing it. In this case, this part can be considered as an indivisible part. Figure 5 details an example of two possible decompositions, which depend on the design context and user intent. If the user is interested in providing a global description of the object in relation to the way it is assembled, then the teapot is described by only two parts: the main body and the cover. When the user wants to decompose it according to the functional characteristics of the components, the teapot is decomposed in a container, a spout, a handle and a cover. If further details are required so as to compare this teapot to another one, then the cover can be decomposed in two parts: the spot-like handler and the disk-shaped surface. Consequently, the number of parts highlights both the complexity of the object and how precise a user wants to be.
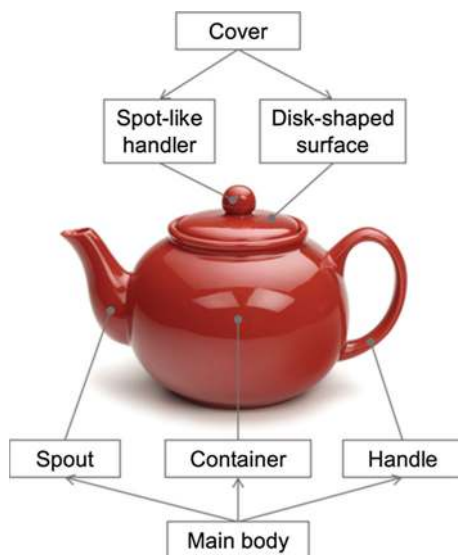

Fig. 5 Two possible decompositions of an object with respect to the design context and user intent
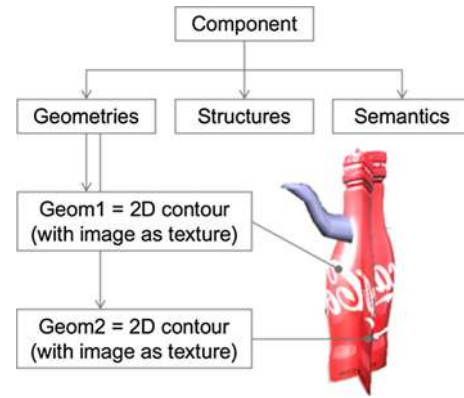

Fig. 6 Component with multiple geometric representations

Components own specific properties. They are representation independent. The meaning conveyed by a specific Component can originate from various ideas and semantics linked to heterogeneous data. In this definition, the number of geometric or structural representations of a Component is not limited. In other words, a Component can have multiple geometric or structural representations as represented in Fig. 6. Components are also context oriented. An object can be decomposed differently depending on the context as in the example presented in Fig. 5. Finally, every Component could be in turn described according to the three previously defined layers of information: Geometry, Structure and Semantics.

3.3.2 Group

A Group gathers together several Components associating them either a specific meaning, or a behavior, or attributes. For instance, a user may want to group a set of Components so as to select, modify or search them as a single one. In the following, the general term Element will be used to indicate either a Component or a Group.

Groups own specific properties. A Group is constituted by at least two Elements, i.e., both Components and Groups can be part of other Groups. All the elements in a Group should have a specified semantic meaning to indicate the purpose of being a Group. It explains why different Elements have to be considered as one. For example, they have a similar function, or the same color. A Group can be an Element of another Group and an Element can belong to several Groups. This is the non-exclusive inclusion property.

Examples of Groups are presented in Fig. 7 representing an office room. All the books on the desk can be clustered in a Group called "Books." The laptop and the mouse form a Group called "PC." The "PC" and the "Books" can be also considered as a Group sharing the fact that they all are on the desk. Another Group called "Furniture" refers to all
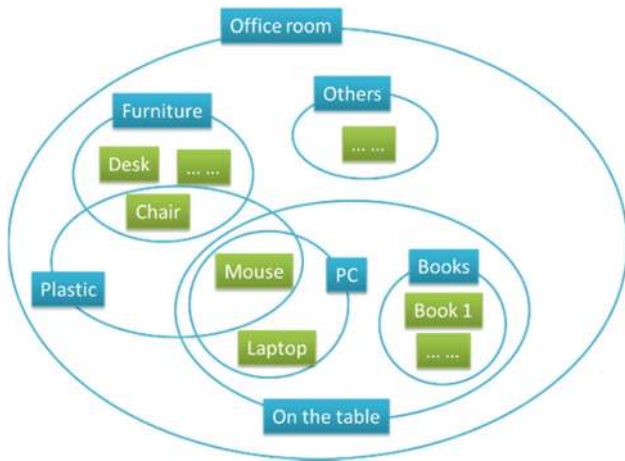
Fig. 7 Example of groups (Office Room): *blue circles* and blocks for Groups and *green blocks* for Components

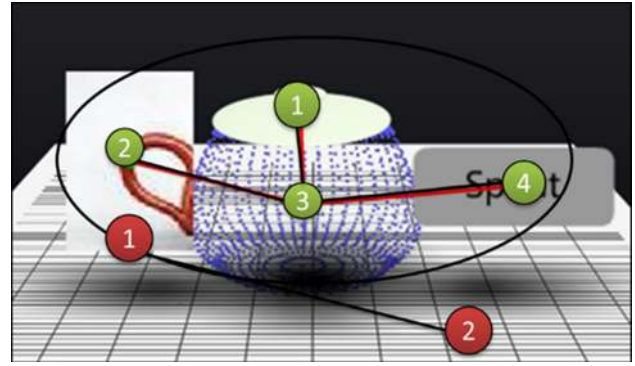Fig. 8 Examples of Relations between heterogeneous data

the furniture in this office room including the desk and the chair. The chair and the mouse can be also considered as a Group as they are both made by plastic.

In this example, it can be noticed that the "Mouse" is shared by four Groups: "PC," "Plastic," "On the table" and "Office room." To reach the element "Mouse" of the Group "Office room," a minimum of one Group ("Plastic"), and a maximum of two Groups ("On the table" and "PC") need to be traversed. The number of Groups to traverse to locate an Element is defined as the depth of the Element in this Group. Thus, for an Element in a Group, there might be a minimum depth and a maximum depth. Depth characterizes the complexity of a Group and is transparent to the user. However, it could be used by the algorithms for the manipulation of Groups.

3.3.3 Relation

Relations are used to describe the way two Elements (either Components or Groups) are connected together.

The links between different Elements may express very complex relations relying on complex operations/

algorithms. For example, to satisfy the relation expressing the geometric merging of two Components together, the related location of the two Components needs to be specified, as well as their geometric representations and the parameters related to the merging algorithm properly saying. For a non-expert in computer graphics, describing these complex links can be very difficult. Thus, the approach is top-down, i.e., from the purpose to the specification of the Geometry or Structure and associated rules. Additionally, at the conceptual level, Relations are simple semantic indications characterizing the type of relation/operation independently of the representations or of the associated evaluation rules. For the example presented in Fig. 8, the user wants to merge the spout (textual representation) and the container (point cloud representation) of a teapot together as these two Components are connected because water can flow from the container in the spout. Although the two Components have different representations, the Relation can be specified at a top level. In this paper, we do not address the way the Relation is satisfied at the low-level and we stay at the top level while manipulating the conceptual model. Of course, at the low-level, specific algorithms should perform the merging between the point cloud and the textual representation but this is not part of this work. In our approach, Relations can be of four different types: Merging, Assembly, Shaping and Location. The Merging Relation links two elements that have to be geometrically operated to obtain a unique geometric model. This Relation can be compared to the Boolean union operation on geometric models. For example, in Fig. 8, the teapot is composed by four different Components each of them having a different geometrical representation: a 2D image for the handle, a point cloud for the container, a 3D mesh for the cover and a text mapped on a plane for the spout. In real life, even if the container and the spout may be produced separately, a merging operation between them can be required such that the spout and the container create a unique continuous volume. Similarly, the container and the handle can be merged. At

the conceptual design level, this is acceptable since the aim is not to create the final shape of the object but to express all the information needed to fully specify this Relation. Distinguishing what happens at the conceptual level from what happens at the geometric level allows a complete shape definition not restricted to the use of limited, time-consuming and sometimes unpredictable modeling details and operations. As already stated, the purpose of the GSDM is to provide the representation of how an object should be created by combining subparts, possibly not completely defined. The Relations aim at specifying the links between them. The real merging operation does not take place at this stage but it can be obtained by processing the GSDM once the geometric description of each constituting Component is completely specified and harmonized (i.e., compatible geometric representations on which Boolean operations can be applied). All those post-treatments of the GSDM are performed during the post-processing step as mentioned in Fig. 2.

The Assembly Relation is a notion similar to what exists in CAD systems. Different Elements are connected together without fusing them into a unique geometric representation but simply linking them with different joints. In the example of Fig. 8, the container (point cloud) and the cover (3D mesh) are linked with an Assembly Relation.

The Shaping Relation is used to indicate the intent to modify the shape of an Element, i.e., to reshape it. It is not simply merging the overlapping area of two Elements by cutting the useless areas, but restyling one Element while taking into account the characteristics of another. These two elements may come from different objects. An example is presented in Fig. 9 where specific egg-like chairs could be obtained while combining a traditional chair with the shape of an egg.

Finally, the Location Relation is used to position an object with respect to the others. On the example of Fig. 8, the group composed by the container, the spout, the handle and the cover is located with respect to the table on which they lie. As for the Assembly Relation, the Location Relation does not affect the shapes.
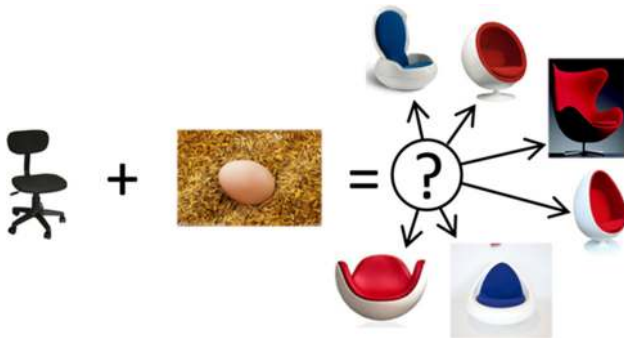
Relations own specific properties. A Relation is only built between two Elements. Actually, a Relation can be built between more than two elements by exploiting the Group notion. For example, if four legs of a table need to be assembled to a desktop, a Group "support" can be created including the four legs and it can be linked to the desktop through an Assembly Relation. As already stated, a Relation aims at specifying the purpose and rules of the link between Elements, and it is independent from the actual representation of each Element. If there is a Relation between Group A and Element (Group) B, then this Relation explains that all the Elements in Group A should have the same kind of relation with B (or with the Elements in B if B is a group). This is the inheritance property. For example, considering the Group of four legs assembled with a desktop, it is not necessary to indicate that each leg is assembled with the desktop. However, it could be necessary to have some Relations between the legs inside of the Group of legs. This inheritance property is managed at programming level, and it is not specified in the data structure. Finally, the uniqueness property indicates that there is only one kind of Relation between two Elements, including the inherited Relation.

3.3.4 Smart positioning

For positioning an element, a smart manipulation system has been designed and is accessible through a specific selector. Actually, repositioning a 3D object in professional software requires the user to specify in which direction or on which plane the positioning is applied. In our implementation, this decision is automatically made by the system. If the user wants to reposition an object on a plane, naturally, he/she will prefer to turn the viewer to face this plane so that the movement of the object can be clearly seen. Based on this consideration, two possibilities have been considered to automatically specify planes. One is parallel to the global reference plane crossing the pivot point of the selected element (Fig. 10a). The other one is perpendicular to the global reference plane crossing the pivot point of the selected element and facing the user (Fig. 10b). The specified plane is highlighted by an orange and transparent color.
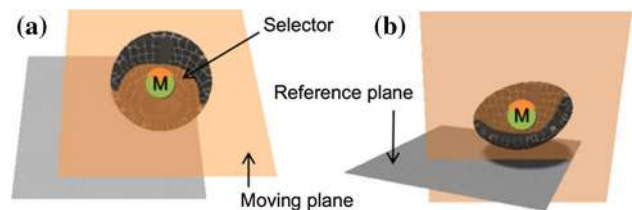

Fig. 9 Example of possible intents expressed by a Shaping Relation

Fig. 10 Positioning of a component: **a** on a plane parallel to the camera, in a *top view*; **b** on a perpendicular plane in a *side view*

| Table 1 Classification of the proposed Key Entities (KEs)	Classification of Key Entities (KEs)	Point	Line	Oriented point	Array
	Geometric KE	$\mathbf{E_P}$	$\mathbf{E_L}$	$\mathbf{E_F}$	\
	Parametric indirect KE	$\mathbf{E_{PP}}$	$\mathbf{E_{LP}}$	$\mathbf{E_{FP}}$	$\mathbf{E_A}$
	Parametric direct KE on a 2D contour	\	$\mathbf{E_{LC}}$	$\mathbf{E_{FoC}}$, $\mathbf{E_{FiC}}$	\
	Parametric direct KE on a 3D mesh	\	$\mathbf{E_{LM}}$	$\mathbf{E_{PM}}$	\
	Parametric direct KE	$\mathbf{E_{PW}}$	$\mathbf{E_{LW}}$	\	\

3.4 Intermediate level (Key Entity and constraint)

The intermediate level specifies how the Components are located the ones with respect to the others in the global reference frame of the virtual environment. In the proposed approach, the location and size of each Component is defined by nine variables specifying the position (x_c, y_c, z_c), orientation $(\alpha_c, \beta_c, \gamma_c)$ and scale (s_x, s_y, s_z) of its associated reference frame. To give more freedom in the definition of the conceptual heterogeneous shapes, and to prevent over-constrained configurations, three scale factors are used, one along each direction. All those variables (nine for each Component) form the unknowns of an optimization problem where Constraints are set up between Key Entities linking the different reference frames and associated variables. The way the different Components are constrained is explained in the next section, whereas the way the optimization problem is solved is detailed in Sect. 3.5.

3.4.1 Key Entity

A Key Entity is a geometric primitive (point, line or oriented point) associated with either the geometric or structural representations of a Component, or simply located in its local reference frame. In the proposed approach, the idea is to make use of Key Entities to constrain directly the geometric and structural representations of the Components taking part to the object definition. This is a much more meaningful and natural way of specifying the relative positioning than to do it indirectly through the reference frames.

There exist two categories of Key Entities: Geometric and Parametric. A Geometric Key Entity of a Component is only related to the local reference frame of the Component, and it is not modified when its geometric or structural representations evolve. For example, a Geometric Key Point defined for a mesh corresponds to a position in the local reference frame of the mesh. Thus, if the mesh is scaled, this point will not move. A Parametric Key Entity can be represented by a point, a line or an oriented point so as to represent a plane. These Key Entities can be associated directly with the geometric or structural representations of a Component such as a vertex of a mesh with its normal. In addition, a Parametric Key Entity can also be
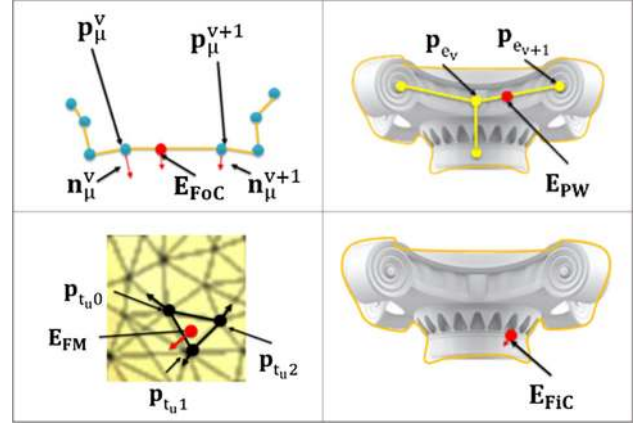
Fig. 11 Examples of Key Entities

created by building rules between other Key Entities. For example, a line can be defined by two points which can either be Geometric or Parametric Key Entities. In this case, it is called a Parametric Indirect Key Entity.

A Key Entity can be associated not only with a point, a line or an oriented point but also with a combination of them (indicated as an array). It is defined by some parameters from which the coordinates of the represented geometric primitives are obtained. All the proposed Key Entities are listed in Table 1, and some of them are illustrated in Fig. 11. In this figure, $\mathbf{E_{FoC}}$ is an oriented point on a contour, $\mathbf{E_{FiC}}$ is a pixel on an image oriented by the normal of the image, $\mathbf{E_{PW}}$ is a point on the structure of an image, and $\mathbf{E_{FM}}$ is a point on a mesh.

The parameters of a Parametric Direct Key Entity include the related Component, the related representation (geometric. such as a 3D mesh, or structural, such as a skeleton) and some numerical parameters necessary to compute the coordinates of the associated geometric element. The parameters to define a Parametric Indirect Key Entity contain the already specified Key Entities and some numeric parameters explaining their relations.

All the key entities are represented in a 3D space. A Key Entity owns specific properties. It can be represented by a geometric element such as a point, a line, an oriented point or a combination of them (i.e., an array). This indicates the dimension of the Key Entity. A point is a one-dimensional entity, a line and an oriented point are two-dimensional

Table 2 Dimensions of the Key Entities

Point	1D Key Entities Line	2D Key Entities Oriented point	n-D Key Entities Array
$\mathbf{E_P}$	$\mathbf{E_L}$	$\mathbf{E_F}$	$\mathbf{E_A}$
	$\mathbf{E_{LC}}$	$\mathbf{E_{FoC}}$	
$\mathbf{E_{PW}}$	$\mathbf{E_{LW}}$	$\mathbf{E_{FiC}}$	
$\mathbf{E_{PP}}$	$\mathbf{E_{LM}}$	$\mathbf{E_{FM}}$	
	$\mathbf{E_{LP}}$	$\mathbf{E_{FP}}$	

Table 3 Constraints and associated combinations of KEs

Name	Acceptable combinations of KEs
Distance (C_D)	(Pt, Pt), (Pt, OPt), (OPt, OPt)
Angle (C_A)	(Li, Li), (Li, OPt), (OPt, OPt)
Coincidence (C_{Co})	(Pt, Pt), (Pt, OPt)
Parallelism (C_{Pa})	(Li, Li), (Li, OPt), (OPt, OPt)
Perpendicularity (C_{Pe})	(Li, Li), (Li, OPt), (OPt, OPt)
Colinearity (C_{Cl})	(Pt, Li), (OPt, Li)
Coplanarity (C_{Cp})	(Pt, OPt), (OPt, OPt)
Coaxiality (C_{Ca})	(Li, Li)
Tangency (C_T)	(OPt, Li), (OPt, OPt)
Insertion (C_I)	(Li, Li)
Contact (C_{Ct})	(OPt, OPt)
Pattern (C_{Pt})	(Pt, Array), (Li, Array), (OPt, Array)

entities, and an array is an n-dimensional entity where n is the sum of its key entities' dimensions. The dimension of a Key Entity corresponds to the number of $\mathbb{R}^3$ elements used to specify its representation. For example, $\mathbf{E_{LC}}$ (an edge of a 2D contour) is represented by a line defined by two points (each of them is an $\mathbb{R}^3$ space). In other words, each dimension corresponds to an $\mathbb{R}^3$ instance, which is named as the "dimensional characteristic" of this key entity. A table classifying all the key entities by their dimensions is presented in Table 2.

Finally, from the presented specification of Key Entities, it can be noticed that the Parametric Key Entity can be associated with the geometric and/or the structural representations of a Component, while in traditional CAD systems, the key entities used to specify constraints in assemblies are only located on its geometric layer. This is the multi-modality property used to enable the simultaneous manipulation of heterogeneous data.

3.4.2 Constraint

Constraints limit the relative location of two Key Entities (KEs), and consequently, they constrain the relative positioning of the underlying representations in the virtual environment. If more than two KEs are involved, an array $\mathbf{E_A}$ of KEs is to be used. Constraints are defined by equations or inequalities linking the KEs. The equations and inequalities depend on the type of Constraint. Even for the same Constraint, different combinations of two KEs may require different equations or inequalities. Table 3 gathers together all the considered Constraints as well as the related possible combinations of KEs (Pt = Point, OPt = Oriented Point, Li = Line).

Coincidence is between two points. Colinearity is used to limit the position of a point along a line. Coplanarity is used to keep a point on a surface. Coaxiality forces two lines to be coincident. Insertion constrains two lines to be coaxial; then, it limits the distance between them. Contact is used to put two surfaces touching each other, and tangent is used to constrain a line and a plane or two planes to be

tangent. Pattern is used to distribute points along a line or around a point.

For example, the Contact constraint between two KEs $\mathbf{E_1}$ and $\mathbf{E_2} \in \{\mathbf{E_F}, \mathbf{E_{FoC}}, \mathbf{E_{FiC}}, \mathbf{E_{FM}}, \mathbf{E_{FP}}\}$ is defined as follows:

$$\mathbf{C_{Ct}}(\mathbf{E_1}, \mathbf{E_2}) = e_0(\mathbf{p_{E1}}, \mathbf{p_{E2}}, 1) \&\& e_3(\mathbf{n_{E1}}, \mathbf{n_{E2}})$$

where $\mathbf{p_{Ei}}$ and $\mathbf{n_{Ei}}$ are, respectively, the geometric point and the normal associated with $\mathbf{E_i}$. This constraint is defined by two sets of equations driven by two generalized functions e_0 and e_3 so that:

$$e_0(\mathbf{V}_1, \mathbf{V}_2, \alpha) \rightarrow V_1 == \alpha \cdot \mathbf{V}_2 \rightarrow \begin{bmatrix} x1 \\ y1 \\ z1 \end{bmatrix} == \alpha \cdot \begin{bmatrix} x2 \\ y2 \\ z2 \end{bmatrix}$$

$$e_3(\mathbf{V}_1, \mathbf{V}_2) \rightarrow \begin{cases} x1y2 == x2y1 \\ y1z2 == y2z1 \\ z1x2 == z2x1 \\ x1x2 \leq 0 \\ y1y2 \leq 0 \\ z1z2 \leq 0 \end{cases}$$

In other words, e_0 is used to scale a vector with respect to another one using linear equations. When used with $\alpha \cdot = 1$ it corresponds to a strict equality of two vectors, which can be used to impose a coincidence constraint. e_3 imposes that the two vectors are collinear using nonlinear equations coming from a vector product. The inequalities are also used to further constrain the two vectors. When considering the Contact constraint, the inequalities are used to specify the orientation of the normals.

The constraints that have been considered were thought to be meaningful for users. As a consequence, semantically, some of them can be special cases of others just putting a specific different value. For example,

"Coincidence" between two points can be considered as a special case of "Distance" between two points equal to zero. However, the equation to compute distance is not linear. Thus, to maximize the use of linear equations and linear inequalities, "Coincidence" and "Distance" are differently formulated. At the end, six generalized functions e_i have been defined, each of them assigning specific equations and/or inequalities. As mentioned before, e_0 is used to scale a vector with respect to another one. e_3 to impose that a vector has to be opposite to another one. Actually, the implementation of e_3 makes use of e_0. Then, e_1 generates one dot product equation so as to define a perpendicularity between two vectors. e_2 imposes two vectors to stay parallel. It is a specific configuration of e_0. e_4 assigns a single nonlinear equation while considering the distance between two points. And, finally, e_5 is used to impose an angle between two vectors. Due to space limit, the six generalized functions are not detailed and all the constraints (Table 3) which have been built on top of those equations are also not detailed.

As the other constitutive classes of the GSDM, Constraints own specific properties. The value of each Constraint is true or false; in other words, it is a Boolean-valued formula. Therefore, conditional operations can be applied between Constraints, such as conditional equal ("=="), conditional AND ("&&") and conditional OR ("||"). The results of the conditional operations are still Boolean valued. Finally, because of the specification of KEs, the constraints are built both at the structural and the geometric levels.

3.4.3 Smart constraining

Clearly, the specification of KEs requires the access to the data level. This step can be tedious and time-consuming for a non-expert user more interested in working at the conceptual level. Thus, specific smart functionalities have been designed so as to automatically identify potential KEs involved in a Relation specified between two Elements and to automatically select the types of Constraint to be specified between those KEs. When two Elements have to be constrained with a specific Relation and a set of Constraints, the system first detects the closest points on each Element and defines Key Entities on the underlying geometric or structural representations. Then, depending on the type of KEs, the Constraints can be automatically defined. All the configurations have not been detailed in this paper. For example, when the two automatically created KEs correspond to two lines, the system computes the angle between them and automatically chooses the closest configuration between either a parallelism (angle smaller than 45°) or a perpendicular (angle greater than 45°). This is illustrated in Fig. 12. Practically, the smart positioning
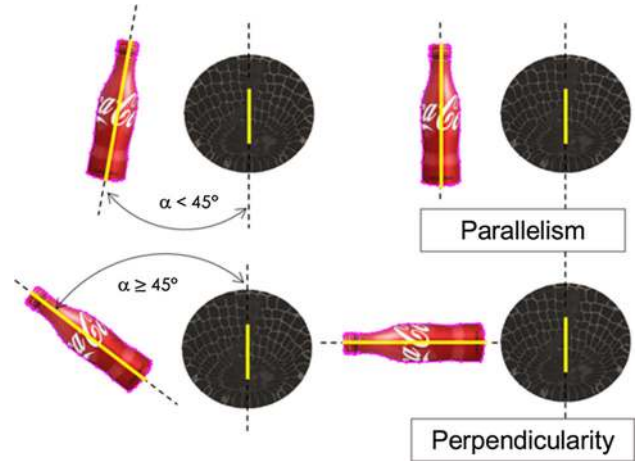

Fig. 12 Automatic specification of Constraints depending on the type of the identified KEs

system (Sect. 3.3.4) is first used to preposition the Elements which are then automatically constrained using the smart constraining approach.

3.5 Constraint satisfaction problem solving

The final positioning of all the Elements taking part to the heterogeneous object definition requires the fulfillment of all the Constraints. This corresponds to the resolution of a constraint satisfaction problem (CSP) either at the end of the specification process, or every time a new Constraint is added.

3.5.1 Optimization problem formulation

In our approach, the optimization problem is decomposed in:

- a set of $9 \times N_c$ unknown variables whose values have to be found. N_c is the number of Components taking part to the object definition. Each Component i has a local reference frame whose position with respect to the global reference frame of the virtual environment is defined by 9 parameters: 3 parameters for the position $\mathbf{P}_i = (x_i, y_i, z_i)$, 3 parameters for the orientation $\mathbf{R}_i = (\alpha_i, \beta_i, \gamma_i)$ and 3 parameters for the scaling $\mathbf{S}_i = (s_{xi}, s_{yi}, s_{zi})$. Each variable has its own definition domain characterizing its possible values. The position and scaling are in $\mathbb{R}^3$, whereas the orientation is in $[-\pi, \pi]^3$.
- a set of constraints/equations limiting the values that the variables can take. Those equations correspond to the ones generated when specifying the previously introduced Constraints.

- an objective function to be minimized and used to select one among the multiple solutions which satisfy the constraints.

In our approach, the resolution of the optimization problem is performed in Mathematica9 (Mathematica9 2016) where several numerical algorithms can be used. For linear problems, simplex algorithms, revised simplex algorithms, interior point algorithms can be used (Vanderbei 2001). For nonlinear local optimization, the interior point algorithm (Mehrotra 1992) can also be used. For nonlinear global optimization, Nelder and Mead (1965), differential evolution (Price and Storn 1997), simulated annealing (Ingber 1993) and random search can be used.

3.5.2 Objective function to be minimized

Since the CSP problem is often under-constrained, an objective function has to be added and minimized/maximized. In comparison with traditional CAD systems, which also have to deal with such a freedom, our approach gives the user the possibility to define his/her own functional to be minimized. Thus, the user can have access to a wider variety of shapes satisfying the same set of constraints. Actually, as it is in real life, the idea is to try to minimize the energy used to move the components between their initial locations and the ones satisfying the constraints. Here, the energy to be minimized takes into account the energy required to move, rotate and deform the different Components belonging to the heterogeneous object definition. Basically, for each Component, this energy is composed of:

- a position energy w_{pi} characterizing the amount of energy needed to translate the ith Component between a position $\mathbf{P}_i^k$ and another one $\mathbf{P}_i^{k+1}$:
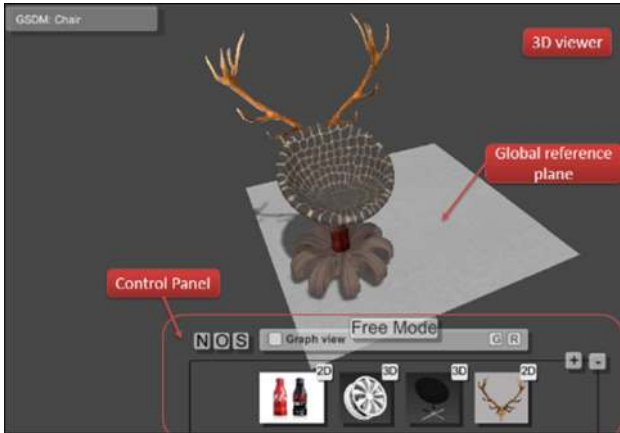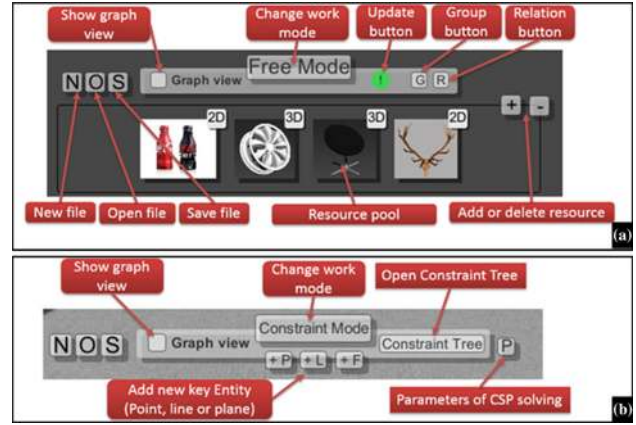

Fig. 13 Interface of the developed prototype

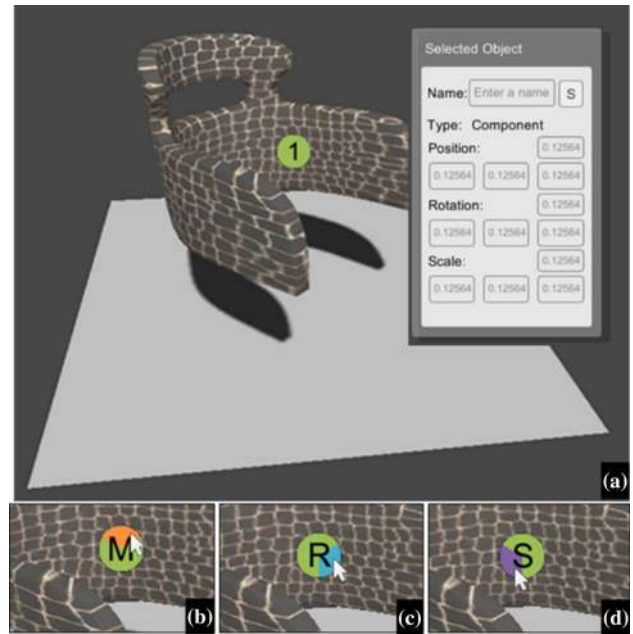Fig. 14 Control panel modes: free mode (**a**), constraint mode (**b**)

Fig. 15 Manipulation of a Component (or Group): **a** when a Component is selected. **b**, **c**, **d** When the mouse moves over different zones of the selection handle

$$w_{pi} = \mu_{pi}\left\|\mathbf{P}_i^{k+1} - \mathbf{P}_i^k\right\|$$

where μ_{pi} stands as a factor that can be easily computed from the volume and density factor of the Component.

- a rotation energy w_{ri} characterizing the amount of energy needed to rotate the ith Component from an orientation $\mathbf{R}_i^k$ to another one $\mathbf{R}_i^{k+1}$:
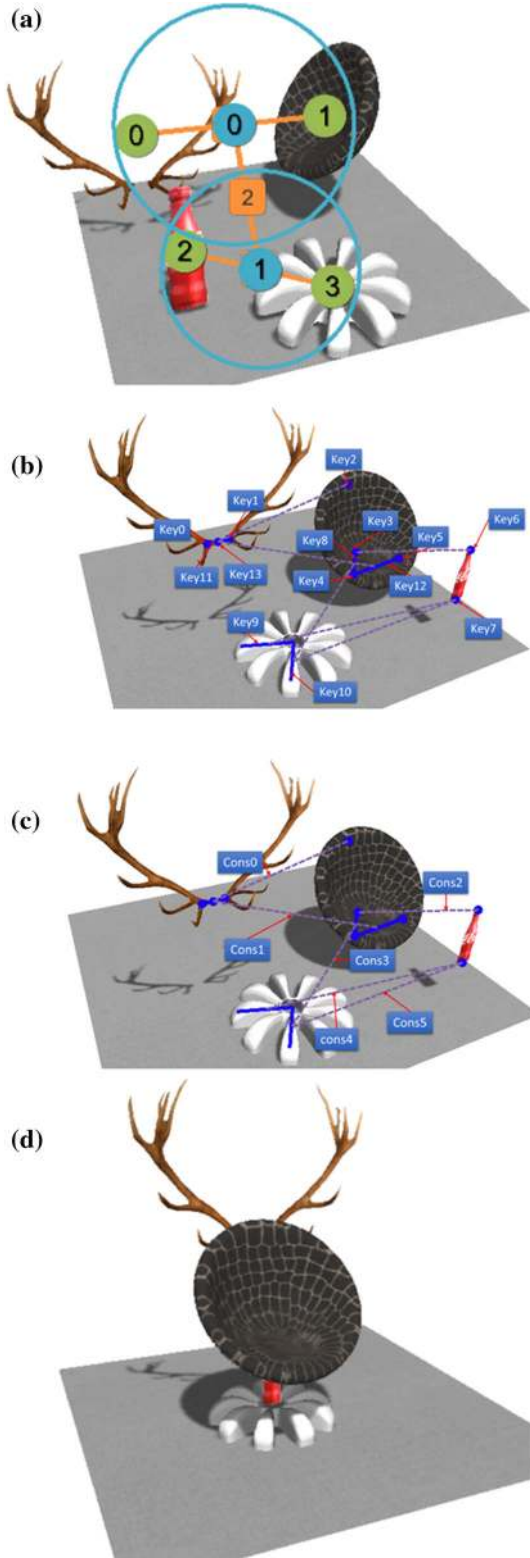
$$w_{ri} = \mu_{ri}\left\|\mathbf{R}_i^{k+1} - \mathbf{R}_i^k\right\|$$

where μ_{ri} stands as a factor that can be easily computed from the volume and density factor of the Component as well as from the radius of rotation, i.e., the distance between the gravity center and the rotation center.

Fig. 16 Conceptual design of a crazy chair

- a scaling energy w_{si} characterizing the amount of energy needed to scale the ith Component from a scale $\mathbf{S}_i^k$ to another one $\mathbf{S}_i^{k+1}$:

$$w_{si} = \mu_{si}\left\|\mathbf{S}_i^{k+1} - \mathbf{S}_i^k\right\|^2$$

where μ_{ri} can be easily computed from a stiffness coefficient describing how rigid the transformation is. The square comes from the use of the Hooke's law.

From those definitions, a global energy can be defined and used as the objective function W to be minimized during the resolution of the optimization problem:

$$W = \sum_{i=1}^{N_c}\left(\mu_{pi}\left\|\mathbf{P}_i^{k+1} - \mathbf{P}_i^k\right\| + \mu_{ri}\left\|\mathbf{R}_i^{k+1} - \mathbf{R}_i^k\right\| + \mu_{si}\left\|\mathbf{S}_i^{k+1} - \mathbf{S}_i^k\right\|^2\right)$$

N_c is the number of Components involved in the GSDM definition. Our objective is to minimize the sum of these three energies. If the position of a Component i should not change too much, the μ_{pi} parameter can be set up to a very large value. In this sense, a link between the relocations of each Component and a semantic meaning is set up. In other words, the proposed resolution strategy is more meaningful compared with the one integrated in traditional CAD modelers. The importance of semantics for the constraints can be found in Tutenel et al. (2008). Thus, different energy factors μ_{pi}, μ_{ri} and μ_{si} can be used for different Components. The energy factors actually limit the flexibilities of positioning, rotating and scaling each Component.

4 The GSDM modeler

The GSDM introduced in this paper has been implemented in a user-friendly system totally developed by the authors using the C# language based on Unity3D (Unity3D 2016). The adopted mathematical tool for solving the CSP is Mathematica.NET/Link 9 (Mathematica9 2016). The prototype includes three main modules: modeling of the GSDM, visualization of the GSDM and controllers for the graphic user interface (GUI). GSDM modeling deals with the data structures of the different notions of GSDM, together with the initialization (e.g., of a Component), manipulation (e.g., rotate all Components inside a Group), modification (e.g., change the parameters of a Constraint) and CSP solving of the GSDM. GSDM visualization is necessary for the representation of the GSDM (e.g., how to represent the Geometry and the Structure, how to show the Group). GUI controllers are mainly for developing easy and friendly interfaces for non-expert users and for working both with simple mouse and with touch screen modalities.

The developed system has been conceived to reduce the users' effort needed to specify the various elements of the GSDM, as shown in the associated video. It includes the smart capabilities defined in Sects. 3.3.4 and 3.4.3 for positioning components, either by simple drag and drop capabilities or by expressing constraints among them. The

Table 4 Summary of the GSDM characteristics for the different examples

	Crazy chair (Fig. 16)	Crazy chair (Fig. 17a)	Crazy chair (Fig. 17b)	Reverse (Fig. 18)	Power plant (Fig. 19)
Components	4	4	4	4	6
Groups	2	2	2	0	0
Relations	3	3	3	4	5
Key Entities	13	13	13	16	24
Constraints	6	6	6	11	20
Linear equations	18	18	18	20	40
Position factor	500	500	500	500	500
Rotation factor	500	5	5	5	5
Scaling factor	10^5	1	100	10^5	10^5
CSP solving time	12.5 s	10.5 s	10.2 s	39.7 s	57.9 s
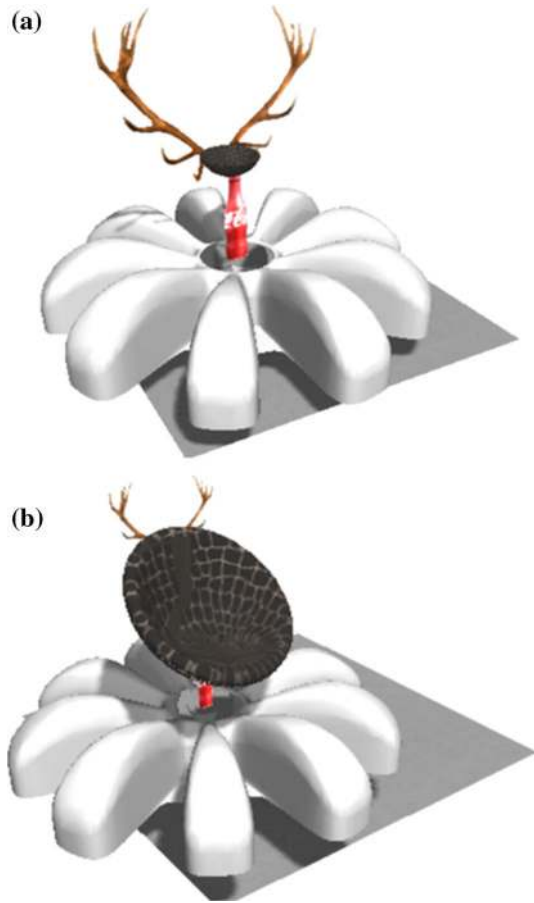
Fig. 17 Conceptual design of a crazy chair using different energy factors as mentioned in Table 4
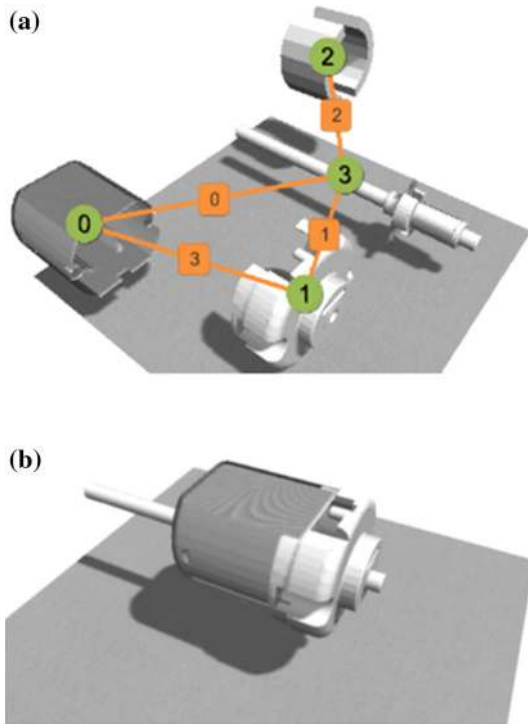
user interface has mainly two areas consisting of a 3D viewer and a control panel as shown in Fig. 13, and it is designed to be as simple as possible.

The 3D viewer is the main workspace to select, manipulate and modify the different notions of the GSDM. The control panel includes the main controllers that execute the complex functions of the GSDM. The user can choose between two work modes as shown in Fig. 14. One is called "Free mode," which is conceived for the manipulation of the conceptual level of the GSDM. The other is the "Constraint mode," which is designed for working on the intermediate level. A mode switch button allows changing from one mode to another. Contextual menu and interaction (mouse and/or touch) behavior are available according to the selected mode. Thus, for instance in the "Constraint mode," to simplify their specification, the various possible Key Elements are sensible when the mouse moves over. Default constraints are set automatically as described in Sect. 3.4.3. If the user is not satisfied, the button "Constraint tree" allows him/her to select the whished constraint among the ones defined in Table 3. Analogously, in the "Free mode," to ease the positioning and sizing of the Components, a new way to manipulate the objects and the viewer in a 3D scene has been designed, using only drag and drop. When an Element is selected, a round spot appears that specifies the number of the element (Fig. 15). It is divided into three zones. When the mouse moves over one of the three zones, its color changes and a letter appears: Orange and the letter "M" is for moving/positioning; blue and the letter "R" is for rotating, and purple and "S" is for scaling. Then, if the user presses the selection handler to realize a drag action, then different types of operations are carried out according to the pressed zone.

5 Results

To illustrate the potential of the proposed approach, various examples have been tested.
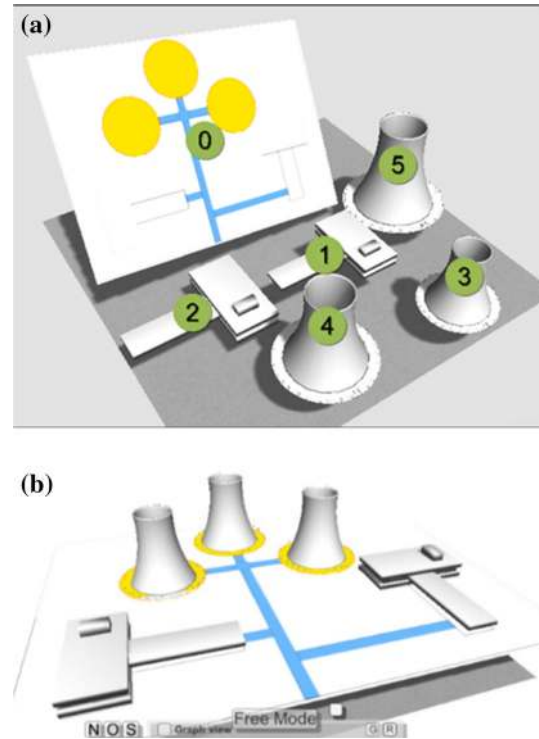
Fig. 18 Reverse engineering of a mechanical engine

Fig. 19 Mixing 3D models of a power plan with 2D plans

The first example aims at validating the proposed approach while demonstrating its capacity to manage heterogeneous models in the conceptual design phase. The idea is to design a so-called crazy chair mixing a set of 2D pictures and 3D textured meshes found on Internet. From a set of inputs (Fig. 16a) and user-specified Relations linking Key Entities (Fig. 16b) with Constraints (Fig. 16c), our system generates the solution presented in Fig. 16d. The values of the energy factors are specified in Table 4 together with some figures characterizing the complexity of the examples. Here, 4 Components have been used and split in 2 Groups. Three Relations have been defined and make use of 6 Constraints involving 14 Key Entities. Overall, this generates 18 linear equations which can be solved in 11.5 s when using a positioning factor set up to 500, a rotation factor set up to 500 and a scaling factor set up 10,000. Setting up a large scaling factor helps keeping the initial size of the heterogeneous models that were initialized using our smart positioning interface.

If the energy factors are modified, other solutions are found (Fig. 17).

The second example focuses on the reverse engineering of a mechanical engine (Fig. 18). It illustrates the possibility to assemble scanned parts without necessarily reconstructing the CAD models as it is traditionally done in commercial CAD software. Here, Relations and Constraints are specified between discrete representations.

The third example is a configuration of a nuclear site, demonstrating the capacity to rearrange 3D models on a 2D plan, which is useful for architectural design (Fig. 19). Here again, in contrast to what is possible in commercial software, our system really solves a set of Constraints specified between the Key Entities of the image and the Key Entities of the CAD models, thus exploiting heterogeneous data.

Table 4 shows the number of Elements defined in the GSDM of these three examples. It shows that the simultaneous manipulation of heterogeneous data has been made possible and is quite fast with respect to the time the user would have to spend to do it manually.

6 Conclusion and perspectives

This paper has introduced the so-called generic shape description model (GSDM) together with its general structure and associated concepts and definitions. This is the first step for describing shapes with heterogeneous data using a unified approach. Heterogeneous objects are obtained while constraining different components within a unique reference frame. Position, orientation and scale of the components are considered as unknowns of an optimization problem. An extended constraints toolbox has been developed together with the mechanisms to specify

them in an easy way. The resolution of the optimization problem tends to minimize a deformation energy involving position, orientation and scaling factors. The proposed approach has been illustrated through several applications requiring the simultaneous manipulation of heterogeneous models in different context. It is clear that using such an approach is more efficient and accurate than what exists in commercial CAD software. However, the development of an effective conceptual design tool based on the GSDM requires the resolution of some research and implementation issues.

The semantics associated with the current version is mainly used to store information for initializing different constituents of the GSDM, such as the "type" or "reason." For some specific design contexts and applications, it can be further specified and extended together with the related mechanisms to treat such high-level information.

The concepts of geometry and structure have been included in the GSDM. In principle, they encompass any geometric and structural representation. In this work, not all the geometric and structural representations have been treated. To effectively exploit all the existing resources, additional representations should be considered. This could be done through the development of new plug-ins. Moreover, even if there exists plenty of algorithms for shape segmentation and structural descriptors' computation, most of the data available are still containing only pure geometric information. Actually, most of the resources require some human intervention to be used in our system for the component selection. This is a limitation. Additionally, for input data missing structural information, the system automatically creates a structure that is the bounding box, which might limit the specification of the relations between components.

Of course, the constraints toolbox can also be extended to consider new constraints useful in specific applications not yet treated. User-specified constraints can also be considered to extend even more the capacity of the system.

Moreover, the relation type of "Shaping" is not fully expressed in this paper, while just a general concept has been proposed. However, such a relation can be of real interest for design and creativity issues. In the post-processing, a fully 3D representation should be generated from the GSDM with its 3D structure and semantics. This requires more advanced techniques in mesh merging and 3D reverse engineering from images. New research on structure and semantics merging is required for their correct updating according to the achieved 3D object model. With both the preprocessing (shape segmentation and structuring) and post-processing phases, we believe that the GSDM can be used in the whole 3D object design process while strongly improving the collaborative conceptual design phase.

Finally, we can imagine to use GSDM in other application contexts, such as the medical analysis domain for representing different medical data and diagnostic results (CT images, type-B ultrasonic images, etc.) in a unified 3D environment, aligned to a 3D model of a human body. GSDM could also be used as a plug-in for a 3D presentation tool such as Microsoft's PowerPoint but in 3D. In this case, text and animation abilities should be further developed.

References

Allègre R, Galin E, Chaine R, Akkouche S (2006) The HybridTree: mixing skeletal implicit surfaces, triangle meshes, and point sets in a free-form modeling system. Graph Models 68(1):42–64

Antonelli M, Beccari C, Casciola G, Ciarloni R, Morigi S (2013) Subdivision surfaces integrated in a CAD system. Comput Aided Des 45(11):1294–1305

Biasotti S, Giorgi D, Spagnuolo M, Falcidieno B (2008) Reeb graphs for shape analysis and applications. Theoret Comput Sci 392(1–3):5–22

Bloch I (1999) Fuzzy relative position between objects in image processing: a morphological approach. IEEE Trans Pattern Anal Mach Intell 21(7):657–664

Décriteau D, Pernot J-P, Daniel M (2016) Towards a declarative modelling approach built on top of a CAD modeller. Comput Aided Design Appl 13(6):737–746

Deluca L, Véron P, Florenzano M (2006) Reverse engineering of architectural buildings based on a hybrid modeling approach. Comput Graph 30(2):160–176

El-Hakim SF (2002) Semi-automatic 3D reconstruction of occluded and unmarked surfaces from widely separated views. Int Arch Photogr Remote Sens Spatial Inf Sci 34(5):143–145

Falcidieno B, Spagnuolo M, Alliez P, Quak E, Vavalis E, Houstis C (2004) Towards the semantics of digital shapes: the AIM@-SHAPE approach. EWIMT

Hudelot C, Atif J, Bloch I (2008) Fuzzy spatial relation ontology for image interpretation. Fuzzy Sets Syst 159(15):1929–1951

Ingber L (1993) Simulated annealing: practice versus theory. Math Comput Model 18(11):29–57

Jain A, Thormählen T, Ritschel T, Seidel H-P (2012) Exploring shape variations by 3D-model decomposition and part-based recombination. Comput Graphics Forum 31(2):631–640

Jiang N, Tan P, Cheong LF (2009) Symmetric architecture modeling with a single image. ACM Trans Graph 28(5):1–8

Lee J, Funkhouser T (2008) Sketch-based search and composition of 3D models. In: EUROGRAPHICS workshop on sketch-based interfaces and modeling, 2008

Luciano da Fontoura C, Roberto Marcondes Cesar J (2000) Shape analysis and classification: theory and practice. CRC Press, Boca Raton

Mathematica9 (2016) Available: http://www.wolfram.com/mathematica/new-in-9/

Mehrotra S (1992) On the implementation of a primal-dual interior point method. SIAM J Optim 2:575–601

Mitra N, Wand M, Zhang H, Cohen-Or D, Kim V, Huang Q-X (2013) Structure-aware shape processing. In: SIGGRAPH Asia 2013 courses. ACM, New York

Nelder J, Mead R (1965) A simplex method for function minimization. Comput J 7:308–313

Office Room (2016) Available: http://www.decosee.com/2014/04/07/modern-office-room-minimalist-idea-23394.html

Panchetti M, Pernot J-P, Véron P (2010) Towards recovery of complex shapes in meshes using digital images for reverse engineering applications. Comput Aided Des 42(8):693–707

Pernot J-P, Falcidieno B, Giannini F, Léon J-C (2008) Hybrid models deformation tool for free-form shapes manipulation. In: ASME 2008 international design engineering technical conferences & design and automation conference, New-York

Price K, Storn R (1997) Differential evolution. Dr. Dobb's J 264:18–24

Reeb G (1946) Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique. Comptes-rendus de l'Académie des Sciences, pp 848–849

Repository TS (2011–2015) Shape repository. http://visionair.ge.imati.cnr.it/ontologies/shapes/

Sawyer K (2013) Zig Zag: the surprising path to greater creativity. Jossey-Bass, New York

Smith G (1998) Idea-generation techniques: a formulary of active ingredients. J Creative Behav 32(2):107–133

Takemura CM (2008) Modelagem de posições relativas de formas complexas para análise de configuração espacial. Doutorado em Ciências da Computação, Universidade de São Paulo

Tutenel T, Bidarra R, Smelik RM, de Kraker KJ (2008) The role of semantics in games and simulations. ACM Comput Entertain 6(4):1–35

Unity3D (2016) Available: http://www.unity3.com

Vanderbei R (2001) Linear programming: foundations and extensions. Springer, Berlin

Wendrich R (2009–2016) Raw shaping form finding project. www.rawshaping.com

Xie X, Xu K, Mitra NJ, Cohen-Or D, Gong W, Su Q, Chen B (2013) Sketch-to-design: context-based part assembly. Comput Graphics Forum 32(8):233–245