



HAL
open science

Conceptual design of shapes by reusing existing heterogeneous shape data through a multi-layered shape description model and for VR applications

Zongcheng Li

► **To cite this version:**

Zongcheng Li. Conceptual design of shapes by reusing existing heterogeneous shape data through a multi-layered shape description model and for VR applications. Mechanics [physics.med-ph]. Ecole nationale supérieure d'arts et métiers - ENSAM; Università degli studi (Gênes, Italie), 2015. English. NNT : 2015ENAM0025 . tel-01346606

HAL Id: tel-01346606

<https://pastel.hal.science/tel-01346606>

Submitted on 19 Jul 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

PhD THESIS in cotutelle

to obtain the degrees of

Docteur de

l'Arts et Métiers ParisTech

Spécialité “Conception”

École doctorale n°432 “Science des Métiers de l’Ingénieur”

and

Dottore di Ricerca della

Università degli Studi di Genova

Specialità “Ingegneria Meccanica”

Della Scuola di Dottorato in “Scienze e tecnologie per l’ingegneria” ciclo XXVI

Presented and defended publicly by

Zongcheng LI

September 28th, 2015

Conceptual design of shapes by reusing existing heterogeneous shape data through a multi-layered shape description model and for VR applica-

Directors of thesis: **Jean-Philippe PERNOT**

Co-director of these: **Bianca FALCIDIENO**

Co-supervisors of the thesis: **Philippe VÉRON, Franca GIANNINI**

Jury

M. Frédéric NOEL, Professor, G-SCOP, Grenoble-INP

Reviewer

M. Umberto CUGINI, Professor, KAEMaRT, Politecnico di Milano

Reviewer

M. Simon RICHIR, Professor, LAMPA, Arts et Métiers ParisTech

Examiner

Mme Rezia MOLFINO, Professor, PMAR, Università degli Studi di Genova

Examiner

Arts et Métiers ParisTech - Centre d’Aix-en-Provence, LSIS, France
Università degli Studi di Genova, CNR-IMATI. Ge, Italia

*To those who love me,
and those who I love.*

- ACKNOWLEDGMENTS -

First of all, I would like to thank all the members of the jury for the attention they have paid to my work and the defense presentation, and for their constructive remarks and questions. More precisely, I thank:

- Simon RICHIR, with his great sense of humor, who made me the honor of being the examiner and the president of this jury,
- Frédéric NOEL and Umberto CUGINI, who have made great remarks as being the reviewers of this manuscript,
- Rezia MOLFINO, who has examined carefully this manuscript.

I would like to thank all my dear supervisors. I had a great pleasure to work with them, although we are from different culture and it took time to have a deep understanding of each other at the beginning. On the other hand, these sparks of culture brought the opportunity of combining fantastic ideas, which opened the view of thinking.

For Jean-Philippe PERNOT and Philippe VÉRON, co-director and co-supervisor of my thesis in France, I would like to express my deep recognition, not only for the scientific quality of their suggestive remarks which directly hit the point and opened my view, but also for their passion and motivation during those long discussions we had.

I would like to thank Franca GIANNINI and Bianca FALCIDIENO, respectively my co-supervisor and co-director of my thesis in Italy, not only for the scientific quality of their remarks but also for the help during each time of my stay in Genova. Their kindly and warm cares helped me pass through those difficulties not only in the scientific work but also in my foreign student life.

I would like also to thank Samir GARBAYA, for the time he spent during the discussions we had together with my four supervisors.

I would like to thank all the members of the Laboratory of LSIS in France and the Laboratory of CNR-IMATI in Italy, not only for the sharing of scientific ideas during my work but also for the kindness and friendly time we have spent together. I would like to thank Aleksandar PETROV, Imon BANERJEE, Romain PINQUIÉ, Yósbel GALAVÍS ACOSTA, Hao Hu, Ferial KOCHTBENE, Nouha HICHRI, Meriem HAYANI MECHKOURI, Widad ES-SOUFI, Mohamed Aymen SAHNOUN and Ahmed AHMED.

And finally, I would like to thank my parents and friends in China, for their endless love, patience and support all along those years during my studding in Europe.

Zongcheng LI

TABLE OF CONTENTS

INTRODUCTION	1
- PART A: BACKGROUND AND STATE-OF-THE-ART -	5
1 RESEARCH METHODOLOGY AND BACKGROUND.....	7
1.1 <i>Research methodology</i>	8
1.1.1 Qualification.....	8
1.1.2 Research method and process.....	8
1.2 <i>Context</i>	10
1.2.1 What is design?	10
1.2.2 Conceptual design in user-centered design process	11
1.2.3 Creative conceptual design.....	13
1.2.4 What is Virtual reality?.....	16
1.2.5 Conceptual design for VR applications	18
1.2.6 Shape modeling for developing VR applications	20
1.3 <i>Definition of research problem</i>	20
2 SHAPE REPRESENTATION, DESCRIPTION AND MODELING TOWARDS REUSING HETEROGENEOUS DATA.....	23
2.1 <i>Heterogeneous shapes data resources</i>	24
2.2 <i>Toward shape modeling by reusing heterogeneous data</i>	28
2.2.1 Traditional modeling and representation	28
2.2.2 Modeling with heterogeneous data.....	31
2.3 <i>Brief introduction to shape description and useful shape descriptors</i>	35
2.4 <i>A multi-layered shape understanding paradigm</i>	40
2.5 <i>Modeling tools</i>	43
2.6 <i>Conclusion and remarks</i>	46
- PART B: CONTRIBUTION -	49
3 TOWARDS A NEW APPROACH FOR SHAPE SPECIFICATION AND CREATION.....	51
3.1 <i>Introduction</i>	52
3.2 <i>An Industry example: Lookx</i>	52
3.2.1 description	52
3.2.2 analysis	54

3.3	<i>Two projects related to this Ph.D. subject</i>	58
3.3.1	The VISIONAIR project	58
3.3.2	The Co-DIVE project	60
3.4	<i>Purpose of this Ph.D.</i>	63
3.5	<i>Conclusion and remarks</i>	64
4	GENERIC SHAPE DESCRIPTION MODEL	67
4.1	<i>GSDM – A multi-layered framework</i>	68
4.2	<i>Data level (Geometry, Structure and Semantics)</i>	69
4.2.1	Geometry	70
4.2.2	Structure	73
4.2.3	Semantics	75
4.3	<i>Conceptual level (Component, Group and Relation)</i>	76
4.3.1	Component	77
4.3.2	Group	80
4.3.3	Relation	83
4.4	<i>Intermediate level (Key Entity and Constraint)</i>	87
4.4.1	Basics	87
4.4.2	Key entity	90
4.4.3	Constraint	101
4.5	<i>General overview</i>	110
4.6	<i>Conclusion and remarks</i>	110
5	MODELING WITH GSDM	111
5.1	<i>Workflow of the modeling process with GSDM</i>	112
5.2	<i>Working at the conceptual level</i>	113
5.2.1	Creation of <i>components</i>	113
5.2.2	Make groups and <i>relations</i>	115
5.3	<i>Working with the intermediate level</i>	118
5.3.1	Automatic identification of <i>key entities</i>	119
5.3.2	Automatic selection of the type of <i>constraint</i> and assign it to the right <i>relation</i>	120
5.3.3	Specification of <i>key entities</i>	121
5.4	<i>Constraint Verification</i>	122
5.4.1	Introduction to numerical optimization	122
5.4.2	Specification of the objective function	124

5.5	<i>Conclusion and remarks</i>	128
6 IMPLEMENTATION AND RESULTS		129
6.1	<i>The adopted development environment</i>	130
6.2	<i>Overview of the implementation</i>	132
6.2.1	Overview of the user interface	132
6.2.2	Visualization of heterogeneous data	134
6.2.3	Manipulation of a component (or group)	136
6.2.4	Visualization of Relation and Constraint	142
6.2.5	Symbolic representation and graph view of GSDM	142
6.2.6	CSP solving	144
6.3	<i>Examples created using the implemented approach</i>	145
6.3.1	Example one: Crazy chair	146
6.3.2	Example two: Assembly scanned pieces	153
6.3.3	Example three: power plant configuration	157
6.4	<i>Conclusion and remarks</i>	160
SYNTHESIS, CONCLUSIONS AND PERSPECTIVES		161
REFERENCES		165



INTRODUCTION

It is complex, time-consuming, and costly to develop innovative and creative products, especially for people who have never been involved in a Product Development Process (PDP). The formalization and manipulation of new ideas and concepts is not straightforward and it is sometime difficult to come out with concrete and explicit representations of those imaginations. Most of the time, they evolve over and over again through different stages (e.g. research, analysis, test, prototyping, re-test) before they reach a satisfactory compromise. However, it has been recognized that over 80% of the total life cycle cost of a product is committed at the very early design stage and it is therefore crucial to focus on it [1] [2] [3].

An outstanding concept development is crucial. During this stage, market research, product specifications and an economic analysis are carried out. At the end of it, a development project is outlined. **Conceptual design**, as an early design phase, plays a very important role for both generating new ideas and reducing costs [4]. With the current intense market competition, **creativity** and **innovation** are more and more urgently needed in the conceptual design phase. To support those creative and innovative tasks, **Virtual Reality (VR)** can play a central role. As the objects designed for a VR application do not need to be manufactured immediately, for example video games, films, etc., but to be visualized and integrated by the user, the design of the **shapes** of these virtual objects becomes the key to achieve a competitive application. Thus, new paradigms have to be foreseen to really take advantage of such new technologies.

Methods and tools supporting human creativity with both manual and computational means have been proposed in the past. Those methods and tools for the design of shapes generally require the use of very specialized computer graphics knowledge. However, creativity does not. Many new ideas come from our daily life. Any non-expert can come up with fantastic ideas for new shapes by usually **reusing and mixing existing shapes**. With the development of Internet and smartphones, many different digital data can be easily generated (e.g. by the digital camera of a smartphone) or found (e.g. through a search in a database). However, few

tools can combine and integrate them together so as to generate a new shape, as they are in different dimensions and stored in different data structures. There is always a gap between the users with creative ideas and the conceptual design tools, a gap between the conceptual model and the final product.

Moreover, the meanings of “shape” are different depending on the application contexts. Today, the information associated with shapes has become broader than ever. Mere geometric descriptions (e.g. CSG¹, B-Rep²) cannot satisfy the needs for modeling shapes in different applications. Structure-based shape descriptors show a higher level of shape features explaining the part-whole relation or topological relation of the geometry that can be used to analyze and potentially generate shapes. In the last few years, there has been a considerable increase of interest in the techniques used to extract and stream knowledge embedded into shapes. The semantics [5] of shape enables to represent shapes that are both machine-understandable and human-understandable. In many applications such as medicine, biology, etc., the meaning of the shape is often more important than the geometry itself. However, the knowledge conveyed by digital shapes is quite limited as the modeling tools and the data formats are quite centered on the geometric aspects and have few relations with data interpretation, or optimized only for one targeted application. Other aspects such as the structural information or the semantic based information on shapes provide a different point of view of shapes and are very useful in specific areas. A multi-layered shape description model should be developed from the very beginning of any shape design process.

Thus, all the above-mentioned statements prove one fact : a more general and efficient way for the **creative conceptual design of shapes by reusing existing heterogeneous shape data with user-centered approaches through a multi-layered shape description model** is much needed.

This is the general context for the subject of this Ph.D. thesis. The work done in this thesis has been carried out within the framework of a French national project named Co-DIVE (Conceptual Design In virtual Environment) funded by the ARTS Carnot Institute of Technology and a European Infrastructure project called VISIONAIR³ (VISION Advanced Infrastructure for Research).

To better introduce the different parts of the proposed shape description model and the related methods and tools, this document is organized in two parts:

Part A: Background and state-of-the-art which corresponds to the explanation of the background together with the state-of-the-art of the existing approaches related to shape modeling, representation and description. It has been divided into two chapters:

¹ Constructive Solid Geometry

² Boundary Representation

³ Project web page: <http://www.infra-visionair.eu>

-
- The first chapter (**Chapter 1**) aims at presenting the background for this thesis, the description of the problem and the objectives as well as the adopted research approach. The basic definitions of some key concepts are also introduced in this chapter.
 - The second chapter (**Chapter 2**) gathers together a presentation and analysis of the existing shape representations, shape descriptors, shape modeling approaches and tools, trying to highlight the issues and limits existing today in the scientific context of the subject presented in the previous chapter.

Part B: Contribution that focuses on the proposed shape description model as well as on the user-centered processing of conceptual design while using this model. It consists in four chapters:

- The third chapter (**Chapter 3**) presents the Co-DIVE and VISIONAIR projects, showing a real example of a virtual reality application that has been developed using the classical shape modeling process. By throwing away the bricks in the classical process, the first idea of the proposed shape description model is presented.
- The fourth chapter (**Chapter 4**) introduces the definitions and the specifications of the new shape description model through three levels: geometry, structure and semantics together with the data structure of each definition in UML⁴.
- The fifth chapter (**Chapter 5**) investigates the user-centered modeling workflow using the proposed shape description model, which enables the storing of shape information in the new model in a correct and non-ambiguous way.
- The sixth chapter (**Chapter 6**) ends this part while presenting a user-centered implementation as well as some experimentations in different contexts.

A synthesis of the key points in the proposed shape description model and user-centered modeling process ends this document. Since the proposed concepts and tools are part of a first exploration step towards solving the problem of user-centered creative conceptual design, they possess their own limits, but they are a way to start new developments, to open new perspectives and discussions as sketched in the last part of this manuscript.

⁴ Unified Modeling Language



- PART A: BACKGROUND AND STATE- OF-THE-ART -

THE FIRST PART presents the background for this Ph.D. thesis project and the state of the art of the existing approaches that can be used for shape representation, description and modeling. It consists of two chapters.

The first chapter starts by introducing the research methodology used in this work. According with the research method chosen, a first step is presented in this chapter that aims at clearly defining the research problem.

The second chapter collects together the introduction of the existing shape models, representations, modeling approaches, shape descriptors and modeling tools used to create digital shapes, pointing out the direction towards a multi-layered shape representation.



Words do not play any role in my thought; instead, I think in signs and images which I can copy and combine.

Albert Einstein

1.

RESEARCH METHODOLOGY AND BACKGROUND

CHAPTER OVERVIEW

THE OBJECTIVE OF THE PRESENT CHAPTER is to introduce the methodology applied during this Ph.D. thesis (**Section 1.1**), a brief introduction and analysis of the background (**Section 1.2**) in order to clearly define the objectives, the research problem and the literature review directions (**Section 1.3**).

1.1 RESEARCH METHODOLOGY

This manuscript aims at presenting the research results and proposed solutions for the defined scientific problem. In order to better introduce this work, it is necessary at first to present the applied methodology.

1.1.1 QUALIFICATION

Depending on its different aspects, research can be qualified in different ways [6], [7], [8] [9] [10]:

- **Quantitative and Qualitative:** Quantitative research means gathering objective numerical data. Statistical models are applied to analyze and explain the collected information. Qualitative research, instead of numbers, is to describe or interpret what is being researched. This type of research is using words or visual representations to provide information.
- **Observational and Experimental:** Observational research is to directly collect information without the input from other previous researches. Experimental research sets the parameters or conditions and is able to change them so as to determine their effects. It helps researchers understand the meaning of certain variables depending on different conditions.
- **Basic, Applied and Developmental:** Basic research is to just determinate what is true. Applied research is to solve problems using what has been found. Developmental research is more or less the same as applied research while more focusing on improving existing techniques to generate new ones.

Mainly, in this Ph.D., the research works are of a **qualitative, observational and developmental** nature. In the introductory part on implementation, an **experimental** type of research can be found comparing different solutions with different parameters (**Chapter 6**).

1.1.2 RESEARCH METHOD AND PROCESS

As the work presented in this document is not just pure research analyzing existing related works, but it also includes a development stage, the research method applied in this thesis is illustrated in **Figure 1.1**, which is based on the research method introduced by [8]:

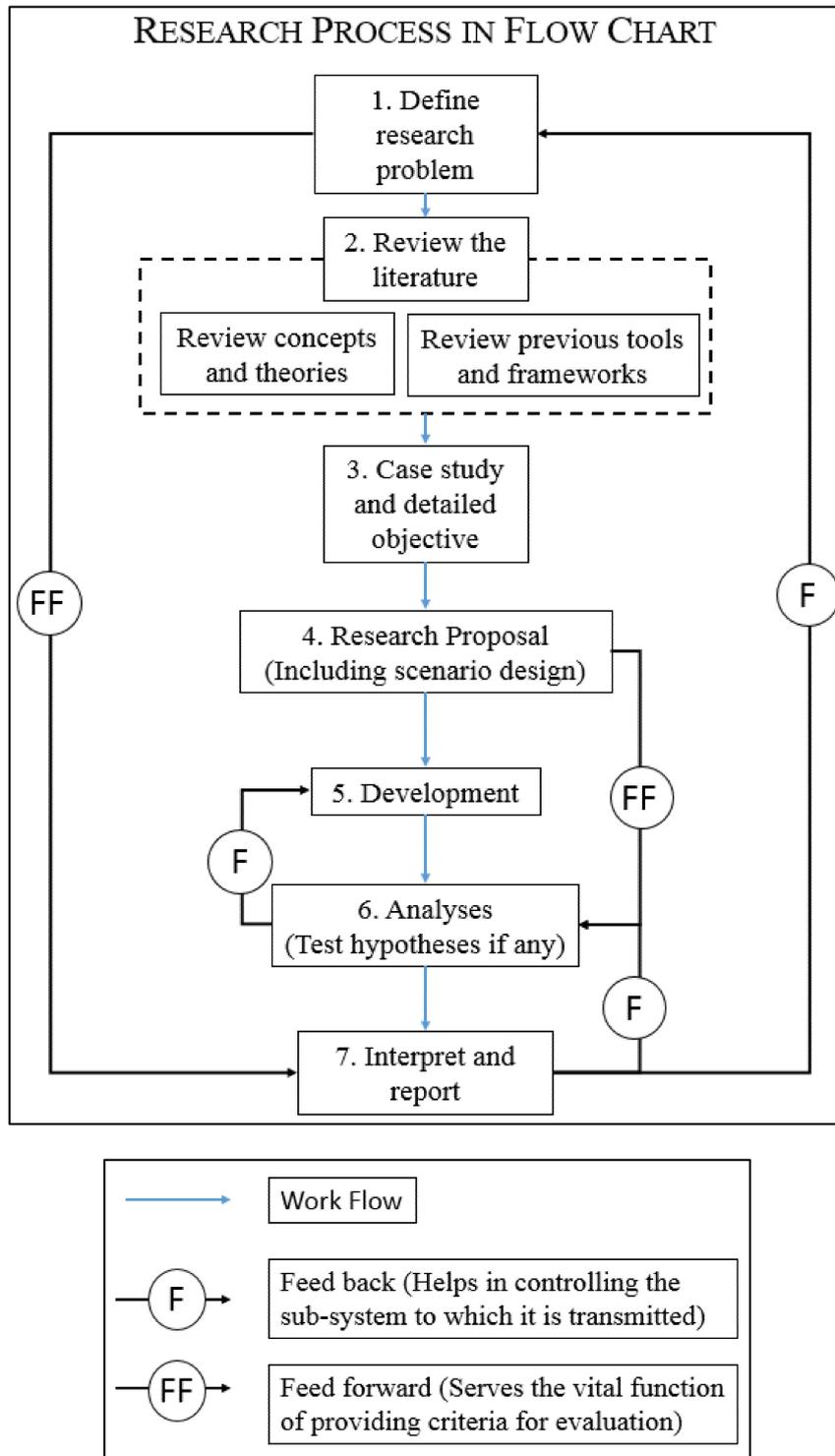


Figure 1.1 Research process in flow chart from [8]

Step 1: **Define research problem** corresponds to two phases. The first one is to understand the problem by selecting the general area of interest or aspect of the subject. The second one is to rephrase the problem as well as the objective into meaningful terms from an analytical standpoint. This step is detailed in this first chapter (**Chapter 1**).

Step 2: **Review the literature** aims at finding the existing solution (if any) corresponding

to the problems indicated in step 1 or presenting the issues which blocks the development of suitable solutions; a case study is also introduced at this stage. Finally, some criteria are proposed for evaluating the suggested solution. This step corresponds to **Chapter 2**.

Step 3: **Case study and detailed objectives** where one or more specific case(s) are analyzed in order to clearly define the position of the work and list the detailed objectives considering the results of the literature reviews in step 2. This step is detailed in **Chapter 3**.

Step 4: **Research proposal** describes the solutions proposed to answer the defined problem. A scenario is also designed in order to demonstrate the capacity and feasibility of the proposed solution. This is developed in **Chapter 4**.

Step 5: **Development** designates the related activities designed to implement the proposal into a real application, ready to be tested. The process of implementation is not wholly introduced in this document, but some interesting results are presented in **Chapter 5**.

Step 6: **Analysis** corresponds to the step in which the results are analyzed with the criteria proposed in step 2. This is part of the final section on **Chapter 6** and the final **Conclusions**.

Step 7: **Interpret and report** is about writing the final document, the Ph.D. manuscript in the present case.

1.2 CONTEXT

As stated in the introduction, this Ph.D. thesis undertakes to find a more general and efficient way for a **creative conceptual design of shapes for virtual reality** applications by integrating and merging existing **heterogeneous shape data** with **user-centered** approaches through a multi-layered **shape description model**.

This project is related to many different areas including conceptual design, creativity, virtual reality, heterogeneous shape data, user-centered design and shape description models. A brief introduction of these areas and their relations is presented in the following subsections.

1.2.1 WHAT IS DESIGN?

In our modern world, there exist lots of applications of design. “Design” is seen from many different points of view when trying to give its definition. Design can be considered as both a verb — the *process* of design — and a noun — the *product* resulting from the design process, as Kathryn Best noted in her book [11]:

“*Design* describes both the process of making things (designing) and the product of this process (a design). ... The *activity* of designing is a user-centered, problem-solving process....”—Kathryn Best [11]

There are also some further definitions:

“Design is the craft of visualizing concrete solutions that serve human needs and goals within certain constraints.”—Kim Goodwin [12]

“Design is directed toward human beings. To design is to solve human problems

by identifying them, examining alternative solutions to them, choosing and executing the best solution.”—Ivan Chermayeff, confounder of Chermayeff & Geismar

“Design is what links creativity and innovation. It shapes ideas to become practical and attractive propositions for users or customers. Design may be described as creativity deployed to a specific end.”—Sir George Cox [13], former Chairman of the UK’s Design Council

“*Design* is an integrative process that seeks resolution—not compromise—through cross-disciplinary teamwork. Design is intentional. Success by design simply means prospering on purpose.”—Michael Smythe, winner of the DINZ (Designers Institute of New Zealand) Outstanding Achievement Award for 2004. “Design, in its broadest sense, is the enabler of the digital era—it’s a process that creates order out of chaos that renders technology usable to business. Design means being good, not just looking good.”—Clement Mok [14]

Design should not be considered as a methodology but a process. *Methodology*, referring to a body of knowledge comprising the principles, guidelines, best practices, methods and processes relating to a particular discipline [15], is a much broader concept than a *process* which is a systematic series of actions directed to some end [16].

1.2.2 CONCEPTUAL DESIGN IN USER-CENTERED DESIGN PROCESS

There are many ways of describing the whole product design process depending on its different applications. This Ph.D. thesis focuses on the user-centered design (UCD) approach, which is a broad term to describe design processes where end-users influence how a design takes shape [17]. The term “user-centered design” originated in Donald Norman’s research laboratory at the University of California San Diego (UCSD) in the 1980s and became widely used after the publication of a co-authored book entitled: *User-Centered System Design: New Perspectives on Human-Computer Interaction* [18].

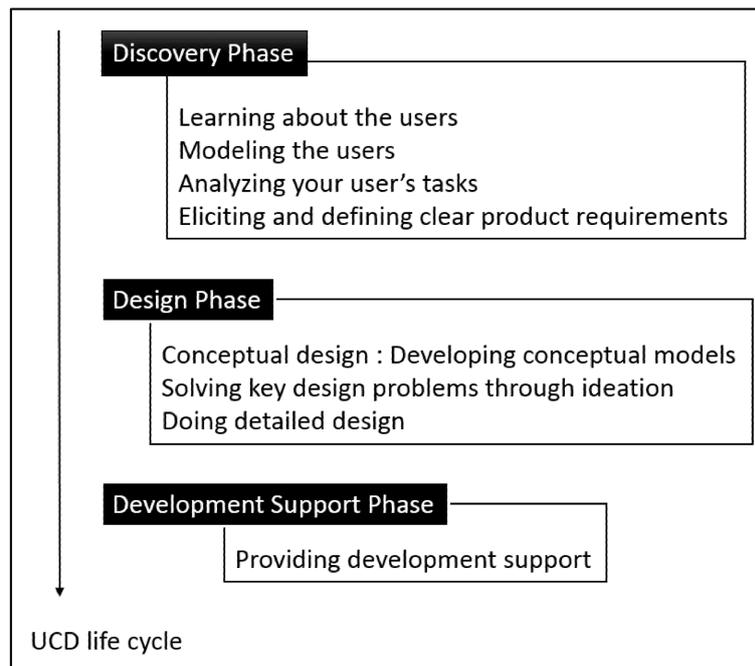


Figure 1.2 User-centered Design life cycle

It is important to think carefully about who the users are and how to involve them in the design process. Eason [19] has identified three types of users: primary, secondary and tertiary. The primary users are people who actually use the artifact; the secondary users will occasionally use the artifact or use it through an intermediary; and the tertiary users are the persons who will be affected by the use of the artifact or make decisions about its purchase. In the book written by [20] some ways of involving users in the design and development of a product/artifact have been suggested, still it must be noticed that the involvement of users mainly occurs in the early phase of the design cycle.

According to [21], the UCD lifecycle can be described as in **Figure 1.2** (which is an abstraction of the idea presented in [21]). This life cycle comprises three phases: Discovery, Design and Development support. Conceptual design is located at the early Design phase. Once the target users' current tasks and workflows are understood, it is important to do conceptual design (or conceptual modeling) before launching into ideation and detailed design. Conceptual design is a preliminary design activity that helps to see the product's concepts, workflows, features and language from the users' viewpoint so as to guarantee that the designed products may be more consistent, simpler and easier for them to understand and use.

Statement 1: The conceptual design phase is a critical stage in the development of new products.

At this stage, the designers and other members of the development team brainstorm product ideas based on research of customers' needs. They aim to produce initial concepts in the form of sketches or outline specifications for commercial and technical evaluation. Following a detailed review, the most promising concepts move forward to detailed design and development. It is often a collaborative process, involving people in a multidisciplinary domain.

At the end of the conceptual design phase, a conceptual model is created. Jeff Johnson has given this definition of a “conceptual model”, which shows a general definition but also adaptable for shapes:

“A conceptual model ... [expresses] the concepts of the intended users’ task domain: the data that users manipulate, the manner in which the data are divided into chunks, and the nature of the manipulations that users perform on the data. It explains, abstractly, the function of the product and what concepts people need to be aware of in order to use it.”—Jeff Johnson [22]

Statement 2: Developing a high-level conceptual model expressing its tasks is an essential part of the conceptual design phase.

Together, the conceptual model and task scenarios help to generate efficient workflows, which enable users to successfully complete their tasks and accomplish their actual goals. Product development outline becomes clear once the conceptual model has been defined.

Conceptual design is difficult, in particular for complex objects such as aircraft, ships, and industrial appliances, especially when the requirements are from mixed cultures. The list of requirements is generally quite long and it often takes a fully-detailed design to assess whether the proposed solution meets the functional requirements, only to learn in the end that it does not. Therefore a system that can quickly generate a proposed solution and evaluate whether it satisfies the defined requirements without extensive detailing effort is strongly needed.

1.2.3 CREATIVE CONCEPTUAL DESIGN

Creativity in general and in design or conceptual design in particular has been a topic of major attention over the years. With the foreseen economic value and the fascination of creativity, it is no surprise that a lot of researchers should attempt to understand it, seek factors that enable or associate creativity in different settings and develop methods to enhance it.

Idea generation techniques can be perceived as the first and most critical part of creative design. The most innovative ideas are generated by iterating back and forth between multiple sources. Smith [23], summarized 172 methods for generating ideas. Six of them are very often used in our daily life including: decomposition, search by relation, structure, combination, rearrangement and special resource-based strategies.

Decomposition methods are reductive; they convert an undifferentiated stimulus into one rich in detail, offering cues for idea generation. It can be from wholes into parts and attributes, and from ends into means, among others. *Relation search* methods consciously look for relations between two or more things, one of which is part of the problem, in hopes of finding solution ideas. *Structure* methods organize information to reveal relationships. *Combination* asks one to combine elements and attributes together in search of ideas. A less systematic variation of *combination*, *rearrangement*, alters the structure of a situation by rearranging its parts. *Special resource strategies* use a specialized outside resource, enabling idea generation to reach

beyond a person's normal mental repertoire.

One well-known example of using these six methods to create new ideas is the LEGO® brick games as presented in **Figure 1.3**.

This example shows a part of a farm built from pieces of bricks. Some bricks are different from one another and they are rearranged and combined together to simulate different objects depending on the various relations among them. LEGO® brick building is a very good way for children to develop their creativity.



Figure 1.3 Example of LEGO brick game: Farm⁵

These methods are also applied to modern conceptual design especially in the furniture and architecture fields. The **Figure 1.4** show some “crazy” designs of furniture combining different ideas together.



Figure 1.4⁶ A few examples of crazy furniture combining ideas. The combined ideas in each example are: A. lamp with horse, B. orange with table, C. coat hanger with chair, D. octopus

⁵ Image downloaded from: <http://uk.ign.com/articles/2014/08/19/lego-minecraft-sets-to-release-later-this-year>

⁶ A. Designed by Front©: <http://www.atomicinteriors.co.uk/product/moooi-horse-lamp>

B. Downloaded from http://freshome.com/2007/08/24/orange-slice-table/orange_slice-tablejpg/

C. Designed by Baita Design, downloaded from: <http://www.trendhunter.com/trends/reindeer-coat-hanger-chair>

D. and E. Downloaded from: <http://forum.xcitefun.net/creative-and-modern-chair-designs-t71749.html>

F. Design by graemebettlesdesign: <http://graemebettlesdesign.blogspot.fr/p/lamps.html>

with chair, E. flower with chair, and E. child skeleton with lamp.

Actually, the most creative ideas are coming from copying and combining existing things [24] in an unexpected manner, as Albert Einstein always said 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

From all these examples and the analysis from [23], one can formulate the following statement:

<p>Statement 3: Taking ideas from different resources then combining and rearranging them together with specific relations and structures is a very common and popular way in creative conceptual design.</p>
--

Shai and al. in [25] proposed a multidimensional group of criteria to classify creativity studies, including:

- Actor complexity
- Product complexity
- Cognitive style or trait
- Cognitive process
- Problem structure
- Design processes, practices, culture or tools.

Actor complexity is showing the number of actors working on the design, their organization, e.g. whether collocated or distributed, etc.

Product complexity explains whether the design is using many complex building blocks or with few simple ones.

Cognitive style or trait is for designers. A cognitive style could be viewed as a cognitive structure; its complement is the *cognitive process* which is the process used by designers to address design problems such as the 172 methods analyzed by [23]. More creative cognitive styles and processes help to create more creative designs.

Beside the designers' contribution, the design context plays a major role in design. *Problem structure* shows whether the problem is defined for a single device or a complex system.

The design processes, practices, culture or tools reflect the experiences of the designers' organization. A mature design company has a well-structured organization and powerful tools which help to develop their creativity. A less experienced design group may see its creativity limited by its own organization or by the use of less appropriate tools.

Thus, based on these criteria, some symbols to describe the gradation and meaning of them are proposed by the author as in **Table 1.1**.

Criteria relative to creative conceptual design	Degree of criteria		
	★	★★	★★★
Actor complexity	Single actor	Small group (Up to 5 people)	Well-structured team
Product complexity	Simple (Up to 3 building blocks, e.g. shape of a cup of tea, shape of a mouse of a PC)	Medium (Up to dozens building blocks, e.g. shape of furniture, shape of a bicycle)	Complex (e.g. airplane, vehicle)
Cognitive style or trait	Less design knowledge	With design knowledge but few experience	Well experienced designer
Cognitive process	Simple (few idea generation techniques)	Medium (group of ideas generation techniques)	Well-defined (Suitable and well-structured idea generation techniques)
Problem structure	Simple (with few requirements)	Medium (With reasonable groups of requirements)	Complex (Complex defined structure)
Design processes, practices, culture or tools.	Simple tools for personal design	Small studio with mature design tools	Big company, well organized with powerful tools

Table 1.1 List of criteria relative to the creative of conceptual design

For example, the design of an aircraft is classified with three stars for all these criteria. A logo of a small company designed by one designer is classified with only one star for all criteria.

1.2.4 WHAT IS VIRTUAL REALITY?

Conceptual Modeling has become popular in different areas such as Information Systems, Web Information Systems, User Interface Modeling, and Software Engineering. However, it has been less used in domains like 3D Modeling and Virtual Reality (VR).

Often, the virtual world is so close to the real one that people cannot even find the differences between them. This applies primarily to the various social network phenomena, such as the Internet – an online network, so firmly established in a person’s everyday life that we can say that a man today does not live one life, but two, and often even more lives, online (for e.g. a role in a computer game). Communication, information retrieval, and entertainment are transferred from the real world into a virtual world. VR’s techniques are applied to the reduction of the distance between the real world and the virtual one.

The term “virtual reality” was coined by Jaron Lanier in the late 1980s, but the origin of VR technologies can be traced back to Ivan Sutherland’s work on interactive computing and head-mounted displays in the mid-1960s [26] [27]. In a paper [28] he contributed to the international Federation of Information Processing Congress in 1965, entitled “The ultimate display”, he outlined the model for a human-computer interface that has continued to inspire the thinking about computer-generated virtual environments ever since. In the late 1980s [29], Jaron Lanier was the first to start attaching the label “virtual reality” to interactive computer-generated three-dimensional immersive displays.

A. Sevalnikov, a Russian researcher about virtual reality gives his own definition of what VR is, as seen below:

“A special system of information reflection, which makes the user feel as he or she is inside the special world created by specific devises” —Sevalnikov [30]

Other definitions can be found as below:

“Virtual reality – is 3D artificial space created by computer means, in which the user can get into, can change it and fell the real emotion in real time” —Bychkov [31]

“ A medium composed of interactive computer simulations that sense the participant’s position and actions, providing synthetic feedback to one or more senses, giving the feeling of being immersed or being present in the simulation” —William R. Sherman and Alan B. Craig [32]

There are four key elements of virtual reality experience [32]: virtual world, immersion, sensory feedback, interactivity. Sherman and Craig have also proposed two definitions of the virtual world:

“virtual world 1. an imaginary space often manifested through a medium. 2. a description of a collection of objects in a space and the rules and relationships governing those objects.” —William R. Sherman and Alan B. Craig [32]

A VR application comprises different components [33], which can be summarized as:

- **The scene and the object.** The scene is the world where objects are located together with other virtual elements such as lights, viewpoints and cameras.
- **Behaviors.** The object may have various behaviors such as moving, rotating and so on. They also have their own way of actions defined to realize specific tasks.
- **Interaction.** The users have to be able to interact with the virtual world. For example, pick up an object, drag an object to finally realize virtual tasks.
- **Communication.** Communication corresponds to the interaction between different users in a similar virtual world.
- **Sound.** Sound is also involved in VR applications that simulate the real world.

Here are four technologies that are crucial for VR [34] [35]:

- The visual (aural and haptic) displays that immerse the user in the **virtual world (virtual environment)** and that block out contradictory sensory impressions from the real world.
- The graphics rendering system that generates, at 20 - 30 frames/second, the ever-changing images.
- The tracking system that continually reports the position and orientation of the user's head and limbs.
- The database construction and maintenance system for building and maintaining detailed and realistic models of the virtual world.

Among the various elements required for VR experience and the crucial technologies for VR, one can state that:

Statement 4: A virtual world (namely a virtual environment), which is the content of a given medium, is the first element that needs to be created for VR applications.

With the increasing development of VR techniques, to represent or create a 3D object as well as the whole scene will necessarily be more efficient so that other elements can be quickly developed based on the digital representation of the virtual world. The representation or creation of the virtual world starts with the design, which has to be improved from the very early phase: the conceptual design (as presented in the previous section). On the other hand, VR is a kind of human communication medium in the sense that human will experience in VR application interacting with the virtual environment. Therefore the design of the virtual environment should be driven by the final user. In other words, it should be **user-centered**.

1.2.5 CONCEPTUAL DESIGN FOR VR APPLICATIONS

As for other domains, introducing a conceptual design phase in the development process of a VR application may help the VR community in several ways. As conceptual modeling will introduce a mechanism to abstract from implementation details, it will reduce the complexity of developing a VR application (for e.g. if the conceptual model is not satisfied then modifications can be done at this conceptual design stage, which will be much easier than change the final VR environment). In addition, if well done, such an abstraction layer can also hide the specific jargon used in VR and then no special VR knowledge will be needed for making the conceptual design. Therefore, non-technical people (like the customer or the end-user who's going to use the VR application to realize their tasks) can also be involved in the development and this will improve communication between the developers and the other stakeholders. In addition, by involving the customer more closely in the design process of the VR application, earlier detection of design flaws is possible. All this could contribute to realize more VR applications in a shorter time.

Statement 5: Customers should be involved in the conceptual design phase.

However, conceptual design for VR faces a lot of challenges since a number of aspects such as the five components of VR and the crucial techniques presented previously, are absolutely essential. Classical modelling languages such as ER [36], ORM [37] and UML [38] are too limited for modelling a VR application in an appropriate way as they are lacking modelling concepts [39] [40] in terms of expressiveness towards VR modelling.

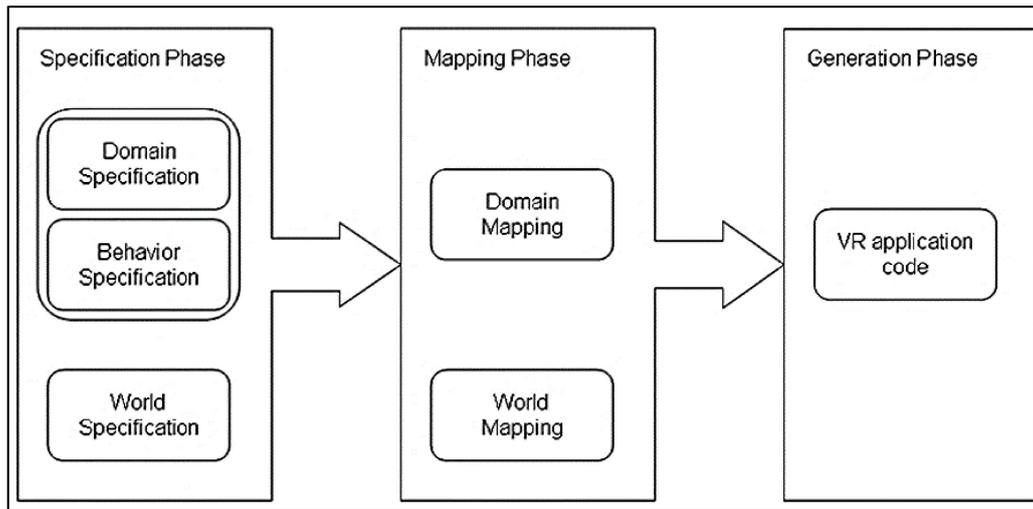


Figure 1.5 Process of conceptual modeling for VR application [41].

In 2007, [41] proposed a general conceptual modelling approach for VR including three phases: the conceptual specification phase, the mapping phase and the generation phase as presented in **Figure 1.5**. The specification phase corresponds to the specification of high-level intuitive modeling concepts.

Statement 6: There is a gap between the conceptual specifications and their VR implementations

The mapping phase tries to bridge the gap. One concept can consist in mapping to different VR models. With this step, VR-specialists are needed, especially when 3D models need to be created. The generation phase is to generate the final VR data format such as X3D or VRML. For example the concept of a “building” can be mapped to a 3D box which is finally structured with a mesh in VR application with texture information.

X3D is an xml based open file standard to represent a 3D scene or 3D objects. The development of real-time communication of 3D data across all applications and network applications has evolved from its beginnings as the Virtual Reality Modeling Language (VRML) to the considerably more mature and refined X3D standard. X3D has a broad structure for saving different types of VR components which includes 2D/3D graphics, animations, audio and video, user interactions, navigation, networking, particle systems and so on. There are more than 40 different definitions of the architecture and the base components of X3D standards. It is a highly

specified standard for VR applications, which is not suitable for conceptual design.

1.2.6 SHAPE MODELING FOR DEVELOPING VR APPLICATIONS

Designing a virtual world comprises the design of each object and the design of the virtual scene. However, the existing user-centered conceptual design approaches (such as [42]) cannot really satisfy the increasing needs of a fast virtual world generation system, especially for VR shape modeling due to the high knowledge requirement of Computer Graphics (CG), VR and product design.

However, the users, who usually lack this knowledge, are not directly integrated in the conceptual modeling process.

Statement 7: There is a gap between the user with creative ideas and design tools.

Due to this gap, the time spent on modeling and modifying shapes is much longer compared with other processes. Shape modeling is the crucial process of the user-centered conceptual design for VR applications.

Today, digital 3D models used for VR applications contain much broader information than just the geometry, in order to simulate not only the appearance but also the behavior of the object. This information can be summarized in four categories:

- Geometry
- Appearance (e.g. color, texture, etc.)
- Physics (e.g. gravity, material, etc.)
- Behavior (e.g. animation, deformation, etc.)

Therefore, different aspects need to be considered in the shape modeling process for developing VR applications. A better shape description structure will reduce the gap between the conceptual model and the final implementation; this will also reduce the gap between the user with creative ideas and the design tools.

1.3 DEFINITION OF RESEARCH PROBLEM

In accordance with the analysis of the background presented in the previous subsection, the research problem lies in the definition of interest areas and their relations from general to specific, the description of problems and their objectives, which can be illustrated by **Figure 1.6**.

Conceptual design, which is an early design phase, plays a very important role in creating new products. It is much needed by virtual reality applications as there is greater flexibility of shapes in a virtual environment. Creativity is one of the key requirements of the conceptual design phase which will help the final design be more attractive. One technique for non-expert people to create new ideas is to combine existing ones. However there is a gap between these

non-expert users with creative ideas and the current design tools which require designers with a thorough knowledge of computer graphics (CG). On the other hand, the information associated to shapes in virtual reality applications covers a large range of aspects. There is another gap between the conceptual specification of shapes and the final data structure in implementations.

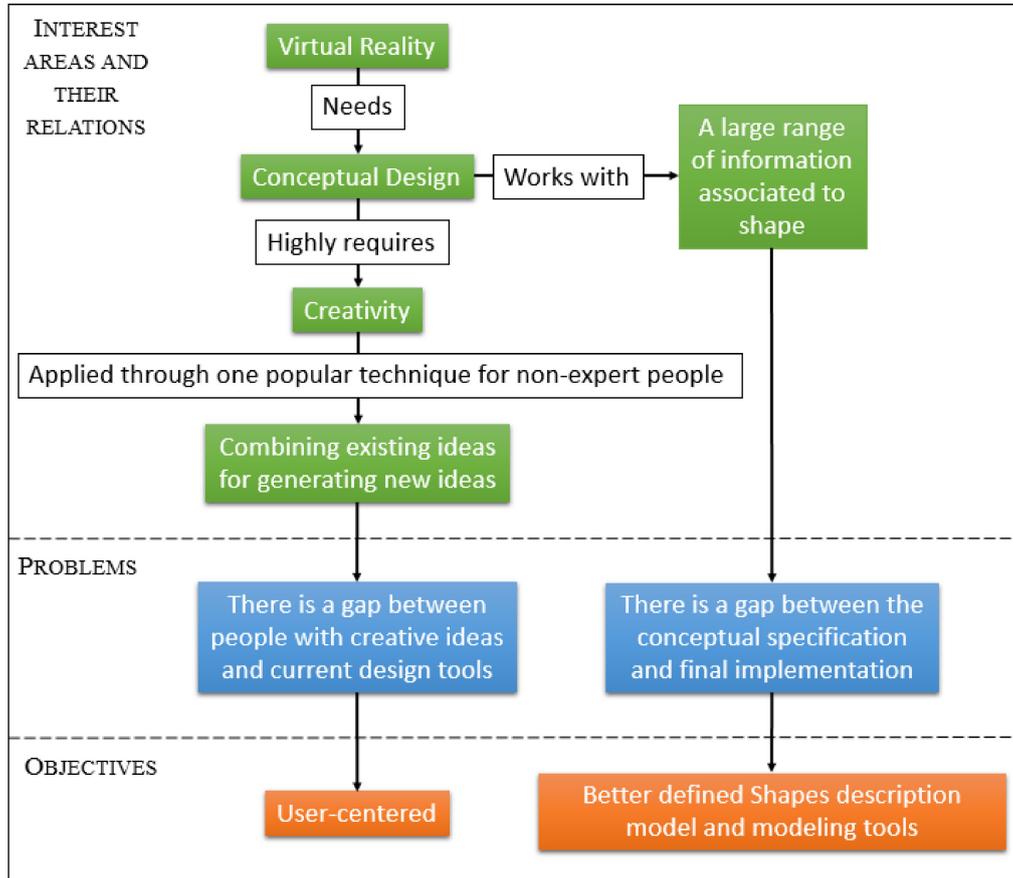


Figure 1.6 Research problem

Taking into consideration all the statements above, from **Statement 1** to **Statement 7**, the problems raised by this Ph.D. thesis are listed here:

- | |
|--|
| <p>Problem 1: There is a gap between:
 the non-expert people with creative ideas who do not possess the know-how for reusing existing (by combining or rearranging them) shapes to create new ones, and the design tools currently available.</p> |
| <p>Problem 2: There is a gap between:
 the conceptual specification of shapes enriched with added information, and the final data structure in the implementations.</p> |

Thus, the main objective of this PhD thesis is to **find solutions for a user-centered shape description model as well as the related methods and tools to reduce the two gaps presented in Problem 1 and Problem 2.**

To review the existing situations so as to shed light on the objective, three literature review directions (described in **Chapter 2**) have been chosen:

- Shape modeling / representation approaches for reusing heterogeneous shape data. It focuses on the existing shape modeling / representation techniques towards the reuse of different shape resources to generate new shapes
- Shape descriptors and information stored in shapes. It does not focus on how to extract information of shapes but tries to understand what kinds of information are associated to shapes, what is the use of these information, what kinds of information extracted are needed to describe a shape or be reused.
- Shape modeling tools using heterogeneous data input

In this chapter, a general description of this subject has been introduced. With the analysis of the reviewed results to be presented in **Chapter 2**, a more specific description of some research directions and scientific issues will be highlighted.

The noblest pleasure is the joy of understanding.

Leonardo da Vinci

2.

SHAPE REPRESENTATION, DESCRIPTION AND MODELING TOWARDS REUSING HETEROGENEOUS DATA

CHAPTER OVERVIEW

THE OBJECTIVE OF THE PRESENT CHAPTER IS to review the related techniques and tools defined in the three literature review directions indicated at the end of the previous chapter. It includes four sections. Starting with an introduction about heterogeneous shape data (**Section 2.1**), this chapter reviews the shape modeling approaches by using heterogeneous data (**Section 2.2**). Then, shape descriptors are introduced (**Section 2.3**), pointing out a multi-layer shape information paradigm for understanding shapes (**Section 2.4**). A comparison of existing 3D modeling tools is highlighted (**Section 2.5**). Finally, a conclusion and some remarks will show the results of this review (**Section 2.6**).

2.1 HETEROGENEOUS SHAPES DATA RESOURCES

Shape holds a mysterious power for both understanding the world and changing it. As with one of the earliest cave paintings in the Indonesian island of Sulawesi that dates back to about 39,900 years, shapes can store information and pass it from generation to generation (**Figure 2.1**).



Figure 2.1 Cave paintings on the Indonesian island of Sulawesi⁷

On the other hand, using shapes is also a very common way to express ideas, especially from the point of view of creativity and innovation, as we can see in Leonardo DA Vinci's 1488 design for a flying machine (**Figure 2.2**).

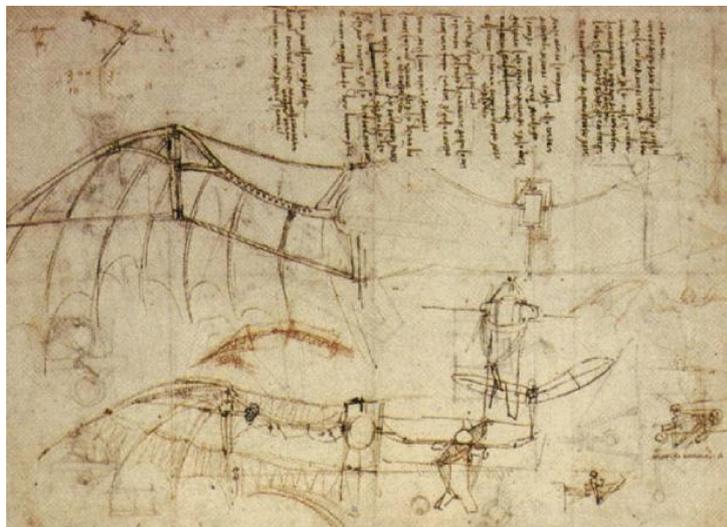


Figure 2.2 Design of a flying machine by Leonardo da Vinci, 1488⁸

⁷ Image downloaded from: <http://images.nationalgeographic.com/>

⁸ Image downloaded from: <http://www.drawingsofleonardo.org/>

With the development of computer graphics, digital shapes have become today another way of representing the real world as well as the imagined virtual world, especially in the field of conceptual design, virtual arts and film. Compared with typical two- or three- dimensional shapes, films and animations extend the dimension of shapes to four, adding one timeline- based dimension. Thus, the information provided by shapes becomes richer and richer.

As a kind of media resource for storing information, shapes today can be seen almost everywhere in our daily life. Some of them can easily be found or even created without a vast knowledge of design or computer graphics. As stated in **Statement 3** and **Problem 1**, the combination of existing shapes so as to imagine new ones is one of the very common ways of applying creativity to shape design, as targeted in this thesis. These varied range of shape resources are what to be reused.

In this manuscript these resources on shapes in the media are referred to as “Heterogeneous Shapes data” or “Heterogeneous Data”. A definition is given below:

Definition 2.1: Heterogeneous Shape Data or Heterogeneous Data = the varied range of media containing shape information.

Some examples are listed below:

- Drawings / Sketches

Drawings or sketches are traditional ways of representing or describing a shape. Especially in an early design phase, sketches are used to describe the concept of products. They are flexible so that some innovative and creative ideas are usually expressed by them.

- Digital images

With the increasing development of smartphones, it has become very common today to take pictures at any time, in any place with a digital camera or a smartphone. Those images are usually used to represent human faces, foods, animals, attractive places or other objects that relate to our daily life. The digital camera is one way to generate a digital image, but there are also many software programs (such as Adobe Photoshop, Corel Painter, etc.) to directly create digital images, thus apparently replacing traditional drawings/sketches on papers. Scanning paper sheets is another means of obtaining digital images. There are mainly two different types of digital images: first the *raster* or *pixel* images which contain a finite set of digital values, then the *vector* images which result from the use of geometric primitives, such as points, lines, curves and polygons. .

- 3D meshes

3D meshes can be considered as the most common 3D data used by designers and artists for developing 3D games, 3D movies or other visualization-based applications. 3D meshes are also used for mechanical simulations with Finite Element approaches. One way to create a 3D mesh is the use of meshing tools such as Autodesk© 3ds Max®, Autodesk Maya 3d®, Blender®, etc. Another way, which has become extremely popular, is through the use of 3D scanning

systems. Indeed a 3D scanner firstly generates point clouds of the surface of the object then a mesh process is applied to produce a 3D mesh. There are also many other converters, which can convert other 3D data into a mesh. A 3D surface mesh is stored in a boundary representation (B-Rep) structure.

- CAD (Computer-Aided Design) modeling

CAD models are generated by CAD software such as Dassault Systèmes© Solidworks®, Autodesk© AutoCAD®, Dassault Systèmes© CATIA®, etc. Compared with 3D meshes, CAD models contain information on the mathematical model of geometry. CAD models are mainly used for industrial product design and manufacturing.

- Texts

There are two different ways to consider texts. First, the text itself is a kind of abstraction invented by human beings. Second, it can also be used to describe shapes. Currently, texts are an important way of storing information, including shape information. Depending on the accuracy of text, a shape can be very well defined or maybe just merely expressed by some few key features.

- Audio tracks

Audio is not a direct media of shape resource. However, if the audio document is the oral description of a shape, it can be considered as an indirect resource of shape information which is similar to a text.

- Video tracks

Video or animations are higher dimensional resources (2D images plus “time” as the third dimension, or capture of 3D object plus “time” as the fourth dimension). They can represent the morphing or the movement of an object along the time. Using digital cameras is the main way to create videos. They could also be generated via special software programs such as Adobe© Flash®, Maxon© Cinema 4D®, etc.

The above-mentioned types of data are some examples of media resources containing shape information. They are considered as “heterogeneous” shape data as they represent or describe shapes by different ways, different dimensions, and with different data structures. However, all these kinds can be used to represent or describe shapes with their advantages and drawbacks.

Criteria for comparing heterogeneous data

To compare heterogeneous data, a list of criteria is proposed below:

- **Dimension:** This criterion refers to the dimensional space in which the shape is represented and manipulated.
- **Type of Information:** some data may contain only geometrical information on a shape, others will include other information such as assembly, constraints, etc. This criterion refers to the different types of information stored in the heterogeneous data.

- **Data structure:** refers to the structure used to store the information.
- **Descriptive power:** shows to what extent the object represented in the heterogeneous data is well described. (“well” in the sense that if the descriptive power is high, then the object is easier to recognize and be reconstructed).
- **Generation:** This criterion defines the way the data are produced.

To illustrate the criteria proposed above, the **Table 2.1** draws a comparison between some examples of heterogeneous data:

	Drawing /Sketches	Digital image	3D Mesh	CAD Model	Text	Audio	Video /Animation
Dimension	2D	2D	3D	3D	/	/	2D or 3D plus time
Type of information	Geometry, color, texture, composition lines, etc.		Geometry with topological information	Geometry with mathematical definition, assembly, constraints, etc.	Linguistic expressions		Geometry, texture, color, morphing, audio, time, etc.
Data structure	No	Simple (Pixels, vector)	Medium (B-rep)	Compact and complex (B-rep, feature-based, parametric, etc.)	Linguistic grammar		Complex (image sequences plus time line)
Description power	Medium	Medium	High	High	Low		High
Ways to generate	Pen plus paper	Digital camera, scanner, software, etc.	Software, scanner, etc.	software	Hardware (Pen or recorder), software		Digital camera, software
Advantages	Easy to create, flexible	Easy to be obtained by cameras. Huge data available on line	High dimension with not complex structure	High dimension with product level's information	Easy to be generated, flexible		Rich information

Draw-backs	Low dimension, special skills are needed (drawing, photographing, etc.), the whole object cannot be visualized	3D modeling skills are needed, needs to be enriched by other resources such as texture information.	No visual representation	Special skills are needed
------------	--	---	--------------------------	---------------------------

Table 2.1 Comparison of some heterogeneous shape data resources

Back to **Definition 2.1**, the term “information on shapes” can be considered as a criterion designed to check if the data are ‘heterogeneous shapes’ data. However, what the “information on shapes” designates needs to be further specified (**Section 2.4**). In the following section, a review of the existing shape modeling approaches toward the modeling by combining heterogeneous data.

2.2 TOWARD SHAPE MODELING BY REUSING HETEROGENEOUS DATA

2.2.1 TRADITIONAL MODELING AND REPRESENTATION

Traditional shape modeling starts from several basic geometric elements, then applies different modeling techniques to define complex geometries and finally represents them in a digital representation.

GEOMETRIC ELEMENTS

As introduced in [43], four categories of these basic geometric elements based on the dimension of manifold are listed as below:

- Punctual element (dimension 0): Point and discrete sets (for e.g. voxel, pixel, etc.).
- Linear elements (dimension 1): straight lines or curves (for e.g. Bézier, B-Spline, etc.)
- Surface elements (dimension 2)
 - Implicit surfaces: surfaces defined by an implicit equation.
 - Parametric surfaces: surfaces defined by parameters such as cylinder surface, sphere surface, Bézier surface, B-Spline surface, NURBS surface, and so on.
 - Subdivision surfaces: a mesh subdivided by a refinement scheme.
- Volumetric elements (dimension 3): A closed surface without self-intersection such as a cube, a sphere, and so on.

MODELLING AND REPRESENTATION APPROACHES

Depending on the adopted representation, modelling approaches can be classified into three categories:

- Collection of discrete primitives. Depending on the resolution, a shape can be simply represented or highly detailed such as pixel images, point clouds and voxels.

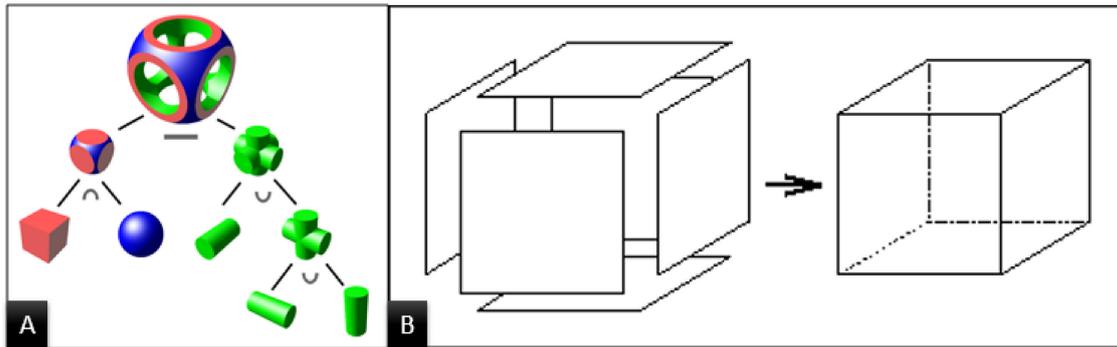


Figure 2.3 A. Example of C.S.G modeling, where \cap for intersection, \cup for union and $-$ for subtraction, B. Example of B-Rep modeling⁹

- C.S.G modelling enables the definition of a geometric model as the result of a succession of topological operations (union, subtraction and intersection) applied to elementary primitives such as cube, sphere, cylinders, etc. (e.g. **Figure 2.3.A**)
- B-Rep modelling is used to express a solid through the set of surfaces defining its closed boundaries. (e.g. **Figure 2.3.B**)

Based on the modelling process direction, modelling approaches can also be classified into other two categories:

- Top-down modelling, [44] it starts with building an overview such as a plan or sketch that defines component locations, key dimensions, etc. Then the assembly of the different parts creates the final shape.
- Bottom-up modelling, it starts with lower dimensional geometric elements from which higher dimensional shapes are created. Four techniques are widely used (**Figure 2.4**):
 - Extrude: two-dimensional closed geometric elements can be extruded to define a solid model with a specific distance (**Figure 2.4.A**).
 - Sweep: a solid model is created by sweeping the cross sections along a specific trajectory (**Figure 2.4.B**).
 - Revolve: a two-dimensional closed geometric element is rotated around an axis with a specific angle to form a solid model (**Figure 2.4.C**).
 - Loft: surface created by cross several sections and possibly with some guides to be followed in between the sections (**Figure 2.4.D**).

⁹ Image downloaded from : http://en.wikipedia.org/wiki/Constructive_solid_geometry

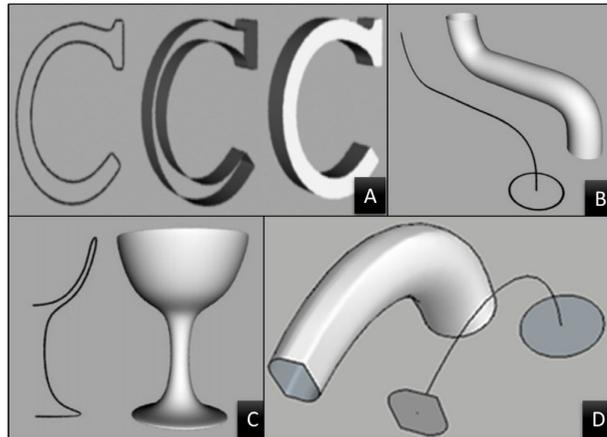


Figure 2.4 A: Example of extrude. B: Example of sweep. C: Example of revolve. D: Example of loft

To provide more advanced, user friendly way to create shapes which can also better encapsulate the intension of the design other approaches have been introduced in CAD design tools for engineering, styling and animation, such as:

- Feature-based modelling [45], [46] creates shape with feature elements such as slots, holes, pockets, etc. A form feature is defined as a specific geometric configuration formed on the surface, edge or corner of a work piece [47] [48].
- Constraint-based modelling refers to specifying shapes with the help of constraints, when defining shape parameters of each part or assembling different parts. A computation is needed to verify if the shape so defined is over-constrained.
- Morphing-based modelling is another technique for manipulating shapes with specific rules [49]. Free-form deformation (FFD) is one of the most popular techniques for aesthetic designs [43]. A totally morphing based modelling example is illustrated in **Figure 2.5**, which is created by a modelling tool called ZBrush.

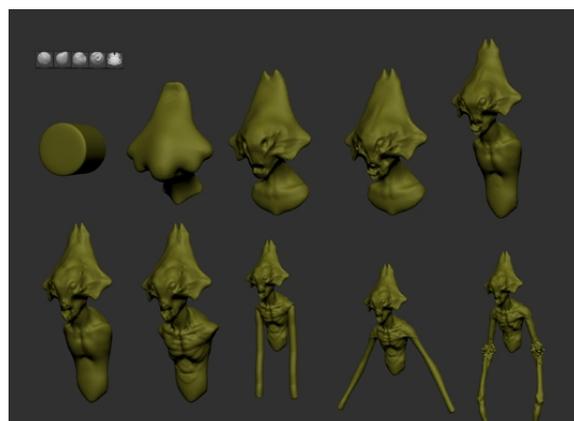


Figure 2.5 Example of morphing based modeling¹⁰

¹⁰ Image downloaded from: <http://www.zbrushchina.com/>

In today's manufacturing-based CAD tools (e.g. Solidworks, CATIA, AutoCAD, etc.), these approaches are mixed together to design a new shape. Some primitives or parts are firstly generated with bottom-up approaches then top-down approaches are applied to assemble them together.

The modelling approaches introduced in this subsection are considered as 'traditional', as they do not use heterogeneous shape resources. There are few approaches today that combine 2D and 3D shape resources to define 3D shapes (such as loft using 2D curves to generate 3D shapes).

DECLARATIVE AND SEMANTIC BASED MODELLING

To reduce the complexity of traditional procedural modelling approaches, declarative and semantic modelling approach starts with stating what has to be created instead of how to be modeled. Some predefined and semantic oriented features are usually used in those systems to help the user bridge the gap between the declarative model (and the semantic model) and the final 3D geometry. Such as in the system mentioned in [50], a sketch based interface with high level features is used to speed a terrain model generation.

2.2.2 MODELING WITH HETEROGENEOUS DATA

Traditional modeling approaches do not refer to really "combining" heterogeneous data, in the sense that each heterogeneous input is considered as a part of the shape. Some of them directly define the mathematical models of shape without any other shape resources or "use" lower dimensional shapes as parameters to specify higher dimensional shapes, as mentioned previously. Others "reconstruct" lower dimensional input to achieve higher dimensional shapes. Heterogeneous data are represented at different stages in these approaches. Only in a few situations they are visualized at the same time to represent the different parts of a shape.

POINT CLOUD AND HYBRID GEOMETRY MESHING

With the development of 3D scanners and 3D printers, efficient techniques related to point cloud meshing are highly required. Point clouds are considered as input, and then different meshing approaches are applied to generate a mesh (e.g. **Figure 2.6**).

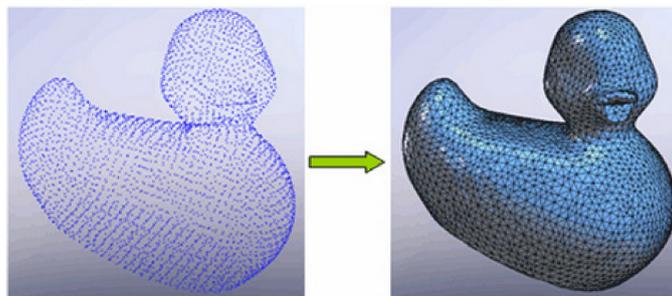


Figure 2.6 An example of point cloud meshing¹¹

¹¹ Image downloaded from: http://rd.newdimchina.com/expertise/mesh_processing.html

The first algorithm for meshing point clouds was described by [51]. However the problem deserved little consideration until [52], who mentioned two trends in surface reconstruction: the Delaunay-based methods where a sub-complex of the Delaunay complex is used to approximate the surface (e.g. [53], [54]), and the volumetric methods, where surfaces are approximated as a zero-set of scalar 3D function (e.g. [55]).

Meshing techniques based on hybrid geometry (a combination of point clouds, polygons, etc.) are developed on the basis of point cloud meshing. However, as the process of converting an unstructured input polygon soup into a consistent polygonal model requires the solution of several sub-problems (such as the calculation capacity of huge data, problems with noise, outliers and under-sampling, etc.), this problem is really complex. To produce a manifold mesh with the corresponding topology and geometry is still one of the main issue of surface reconstruction.

IMAGE-BASED MODELING

Another way to model shapes using other types of data is image-based modeling, which reconstructs a 3D mesh from 2D images. Due to the fact that image-based modeling has a good potential for generating very realistic images, it has gained a lot of attention in graphics community.

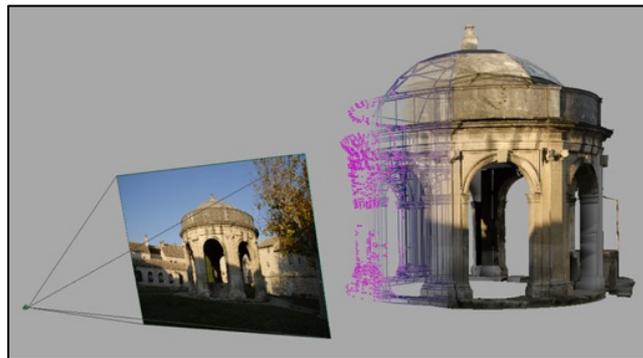


Figure 2.7 Example of image-based modeling

Image-based modeling techniques are usually used on constrained problem of reconstructing architectural models ([56] [57] [58], e.g. **Figure 2.7**). In many cases, the most impressive and accurate results come from those achieved with interactive approaches. Those techniques have been used in the context of Reverse Engineering to fill in holes in meshes [59] or to simplify triangle meshes [60].

Some techniques take more than one image. These images can be un-calibrated images (also called “shapes from video”) such as introduced in [61], [62], or oriented images taken from different views as explored in [63]. Range images are also used to reconstruct 3D models ([64], [65]). There are also some approaches trying to reconstruct a 3D mesh from hybrid images (e.g. both range and color images) such as introduced in [66], [67].

2D IMAGES CONSIDERED AS TEXTURES OF 3D IMAGES

One situation in which both 2D and 3D shapes are used together in the same model consists in taking a 2D image as texture for 3D models. The related techniques are named as texture mapping, which is a way of adding surface details, texture (a bitmap or a raster image), or color to computer generated graphics or 3D models. Its application to 3D graphics was pioneered by Edwin Catmull in 1974 [68] [69]. Texture mapping has made it possible to simulate near-photorealistic 3D models in real time.

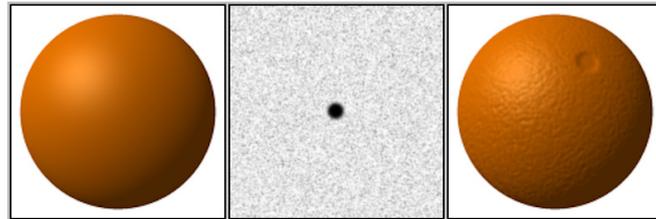


Figure 2.8 3D shape (left) with texture – bump map (middle) to simulate an orange (right)¹².

As an example, a bump mapping is shown in **Figure 2.8**, where the surface of a sphere is simulated as it was an orange.

As with bump mapping, texture mapping is a solution for reusing 2D image and 3D models. However, these two types of data are not all used to define the geometry of the shape. Texture improves only the appearance of a shape such as in the example in **Figure 2.8**, the surface mesh of the sphere is not modified by adding mottled notches.

2D PLANAR SURFACES USED IN VIRTUAL ENVIRONMENT

In most 3D video games, 2D planar surfaces with transparent texture (image with RGBA format) mapping are normally used together with closed 3D surfaces to simulate objects. For example grass, leaves or trees (e.g. **Figure 2.9**).

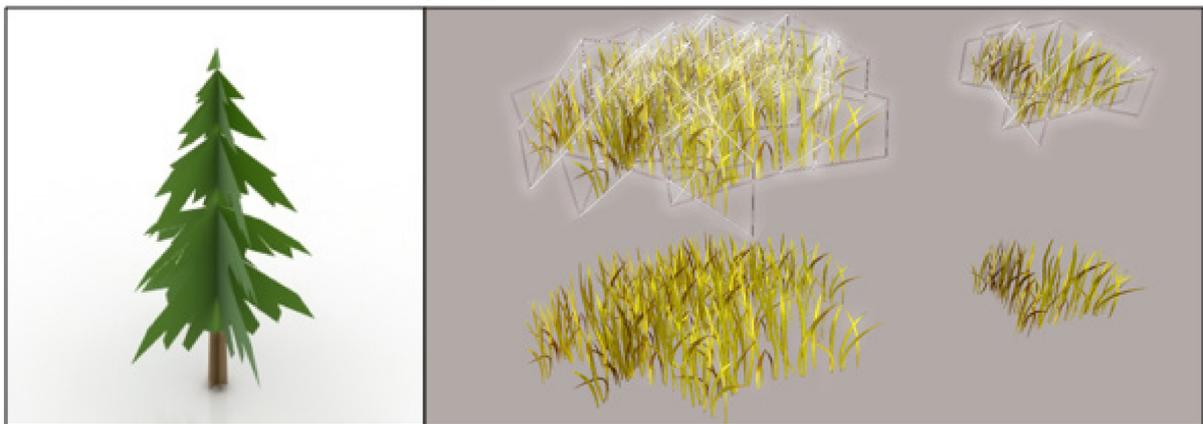


Figure 2.9 Example of tree and grass simulated in 3D environment by 2D planar surfaces¹³

¹² Image downloaded from: http://en.wikipedia.org/wiki/Bump_mapping

¹³ Image downloaded from: <http://image.baidu.com/i?tn=baiduimage&ipn=r&ct=201326592&cl=2&lm=-1&st=->

For CAD usage, 2D planar surfaces are also used to present different views of the designed product.

These 2D planar surfaces are positioned freely without defining any specific relations between them (For CAD 2D views, they are positioned in specific view directions). They are not usually used to design a shape but simulate it in an approximate way.

TEXT WITH 2D AND 3D SHAPES

Text as a kind of shape is also used together with 2D or 3D shape in VR applications. For 2D shapes, Microsoft Word and Microsoft PowerPoint are very common examples, which enable to manipulate texts with 2D shapes. However, these texts are not used to define a shape but to present information in addition to the underlying context.

3D multiplayer games usually put a text above a character as its name or conversations as presented in **Figure 2.10**.



Figure 2.10 Examples of texts used in 3D virtual environment¹⁴

They can also be considered as a planar surface that always faces the viewer. However, similar to other situations presented previously, texts are not used today to design new shapes.

1&fm=result&fr=&sf=1&fmq=1424953536720_R&pv=&ic=0&nc=1&z=&se=1&show-tab=0&fb=0&width=&height=&face=0&istype=2&ie=utf-8&word=3d+%E8%8D%89

¹⁴ Image downloaded from: http://image.baidu.com/i?tn=baiduimage&ipn=r&ct=201326592&cl=2&lm=-1&st=1&fm=result&fr=&sf=1&fmq=1424954101973_R&pv=&ic=0&nc=1&z=&se=1&show-tab=0&fb=0&width=&height=&face=0&istype=2&ie=utf-8&word=3d+game+rpg

2.3 BRIEF INTRODUCTION TO SHAPE DESCRIPTION AND USEFUL SHAPE DESCRIPTORS

In the previous sections, some heterogeneous shape data resources have been presented, from the type of media point of view. This section focuses on reviewing shape descriptors in order to find out what kind of information are stored in shapes and how to describe them and which of them are important, as the three literature review directions defined at the end of **Chapter 1**. To distinguish among “shape representation”, “shape description” and “shape descriptor”, here their definitions are proposed in this manuscript:

Definition 2.2: Shape representation = a computational model that represents an object in the digital world which is quantitatively and qualitatively similar to the ‘real-life’ object.

Definition 2.3: Shape description = expression of the meaningful characteristics of a shape in a specific context.

Definition 2.4: Shape descriptor = meaningful numbers or characters produced to describe a shape in a specific context.

Compared with the **Representation** of shapes, the **Description** of shapes is only qualitatively similar. A representation is more detailed and accurate than a description, but without any meaningful high-level information. A description focuses on the meaning or characteristics of an object from a specific point of view and it is always calculated based on certain shape representations. The importance of the digital representation of an object does not only lie in the possibility of displaying it - a display can also consist in a particular simulation which is a simulation of the final appearance of the object- but also in the fact that we can draw some analyses, carry out simulations and extractions for further data/information-storing purposes, while the point of using descriptions is to find some meaningful information on the shape that can be used to classify objects or retrieve shapes. Shape feature **descriptors** are quantities produced so as to describe a given shape. Descriptors can be considered as a higher-level means of description, which uses only numbers to describe a shape in a specific application context (e.g. volume, area, etc.). From the point of view of the structure of the data, a shape representation has a fully defined structure called a digital model to save the geometric information of a shape, while shape descriptions or descriptors use simple data structures to code meaningful information.

There is no clear quantitative boundary between shape representation and shape description. They both are ways of describing shapes, and the technologies related to them are often similar. Therefore, in the current literature, they are usually considered as belonging to the same research area of shape analysis.

2-D shape descriptors		3-D shape descriptors	
Contour Based Descriptors	Fourier Descriptors (FD)	View Based	Adaptive Views Clustering
	Wavelet Descriptors (WD)		Compact Multi-View Descriptor
	Curvature Scale Space (CSS)		LightField Descriptor (LFD)
	Shape Context (SCD)	Histogram Based	Shape Spectrum
Region Based Descriptors	Zernike Moment Descriptors (ZMD)		Generalized shape distributions
	Scale Invariant Feature Transform (SIFT)		Bag-of-Features (BoF)
	Angular Radial Transform (ART)	Transform Based	Spherical Harmonics Descriptor
Hybrid Descriptors	FD + ART		PCA Spherical Harmonics Trns.
	FD + ZMD		Spherical Trace Transform
		Graph Based	Skeletal Graph Based
			Reeb Graph Based
		Hybrid 3D Descriptors	CMVD + STT
			Depth-Buffer + Silhouette + REXT
			SIFT + Bag of Features
			Depth-Buffer + Spherical Harmonics

Figure 2.11 Classification of 2-D and 3-D shape descriptors by [70]

The review in this section is not focusing on the techniques of how information are extracted from shapes but what kind of information can be extracted and what kind of information people are focusing on. Therefore, some existing surveys ([71], [72], [73], [74], [75], [76], [77], [70]) on shape analysis and shape descriptors have been reviewed from the very beginning of 1970s until 2013.

There are some general descriptors, such as perimeter [78], area [79] [80] [78] [81], statistical moments [82] [83] [84] [85], shape signatures [86] [87] [88], polygonal approximation-based shape descriptors [82] [89], complexity [90]. A recent classification of 2D and 3D shape descriptors can be found in [70] as presented in **Figure 2.11**.

Most shape description techniques are extracting information from the contours or the regions of the shape. Others may transform 2D/3D space coordinates into other spaces to get useful information. Color and light information are also used to describe a shape. The following **Table 2.2** shows other ways to classify shape description and representation techniques for each ten years starting from the 1970s. (Time for the “proposed classification” in this table is not referred to the time when the specific technique appeared but the time when it became popular and considered as a “class”)

From this table, it can be noticed that different shape information was focusing in different time. Those related algorithms are been developed towards graph based and more meaningful descriptors. They have grown from simple algorithms dedicated to 2D silhouette feature extraction only, to multi-scale transforms, graph- based feature extraction for various application domains. As those techniques are typically used for extracting information from a well-defined shape, their main applications are shape classification and retrieval. However, among these techniques, there are some whose results can also be potentially used for modeling new shapes especially graph- or structure - based techniques, which might turn out to be very interesting for this thesis.

Time	Proposed classification	Mainly Targeted shape type	Applications	Issues	Perspectives
70s	-External / Internal -Scalar Transform / Space domain -Preserving / Non-Preserving	-2D silhouettes	-Recognition of Characters, waveforms, chromosomes, cells and machine parts	-Time of computation	-The possibility to develop algorithms for analyzing shapes compatible with human intuition
80s	-Transform schemes / Spatial techniques / Global technique	-2D silhouettes -2D image of 3D object	-Document processing, -Microscopy, -Radiology, -Remote sensing, -Navigation, -Reconnaissance	-No general theory of control in image analysis -Simple domain -Only 2D	-Those techniques was potentially considered as a science to be continued to provide a theoretical background for the design of application-oriented system
90s	-Contours / Regions/ Transforms -Single Transform / Multiscale Transform	-2D shapes -Few 3D shapes	-Neuroscience, -Document Analysis, -Visual Arts, -Internet, -Medicine, -Biology, -Physics, -Engineering, -Security -Agriculture.	-Inherent inter and multidisciplinary -incomplete representation -Lack of guidelines for choosing suitable descriptor -Few of 3D descriptors -No standard criteria for validation	-Board application potential (Internet, Virtual Reality, etc.) -Integration with graph theory and complex networks -Towards the combination of descriptors -Content Based retrieval -Semantic Based retrieval
00s - now	-Contours / Regions /hybrid -Structural / Global -Feature based / Graph based / view based -Shape features / Appearance feature	-2D shapes -3D shapes	-Neuroscience, -Document Analysis, -Visual Arts, -Internet, -Medicine, -Biology, -Physics, -Engineering, -Security -Agriculture. -Virtual Reality -3D films	-3D feature extraction -Diversity of shape formats -Not well-defined representation -Appearance features and semantic retrieval	-Benchmarks for domain specific domain -Feature extraction from compressed formats -Computation complexity and efficiency for graph based approaches -Artist friendly and intuitive tools -Shape retrieval from scenes containing multiple object.

Table 2.2 Comparison of shape representations and description techniques

SHAPE SKELETON

Shape skeleton is a thin version of that shape, which is the locus of points that 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. In some papers, skeleton is also named as “topological skeleton” or related to “Voronoi diagram”. The concept of a skeleton is also interchangeable with “medial axis” and “thinning”. Here some major algorithms for extracting shape skeletons are listed.

Topological thinning based methods [91] extract a skeleton by removing its outer layer over and over again until a specific thinness. This kind of methods is sensitive to noise and orientation, however it is quite simple and keep the topological relation of the shape.

Distance transform base methods [92], [93] obtain skeletons by computing for each point/voxel the distance from the background. However the results are not always connected and additional methods are needed to get a connected skeleton. However, this kind of method can always provide a centered skeleton.

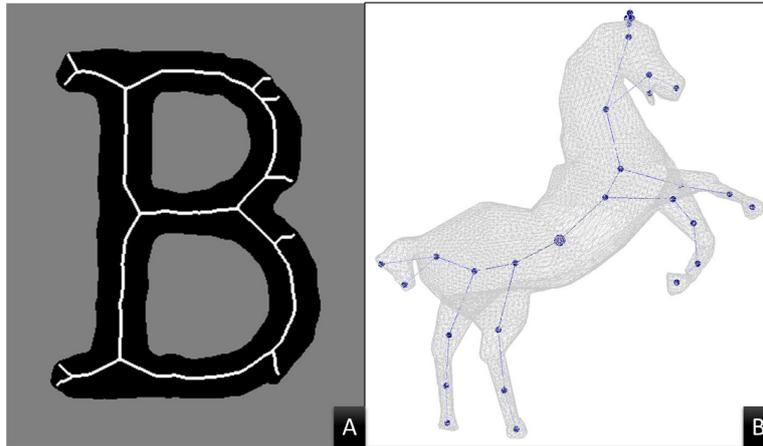


Figure 2.12 Examples of skeletons: A – 2D, B-3D¹⁵

Wavefront propagation based methods [94] find the skeleton by propagating a wavefront from the root to the outer boundaries of the shape. Skeletons generated using this kind of method are usually continuous and smooth while very sensible to noise as well. Moreover, the results are not usually located at the center of the shape.

Voronoi diagrams [95] are generated by partitioning a shape into different regions by a set of chosen points/ voxels and then linking these points through other algorithms (for e.g. heuristics based) to get the final skeleton.

REEB GRAPH

Reeb graph [96] [97] calculates the skeleton through the evolution of the level sets of a real-valued function. Reeb graphs, as they strongly preserve the topological information of a shape, have been widely used in different areas. If the function used to calculate a Reeb graph is on a special flat space, then the result forms a polytree, which is also named contour tree. Reeb graphs are also helpful for image segmentation. An example of a Reeb graph is presented in **Figure 2.13**.

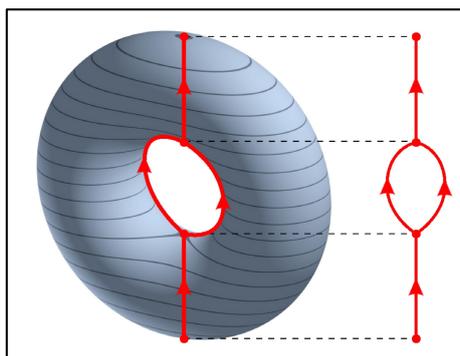


Figure 2.13 Example of a Reeb graph calculated from a height function¹⁶

¹⁵ Image downloaded from: http://en.wikipedia.org/wiki/Topological_skeleton and http://www.sci.utah.edu/~jtierny/img/pacific_material6.png

¹⁶ Image downloaded from : http://en.wikipedia.org/wiki/Reeb_graph

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 this shape will be used and relocated in another 3D space, then this skeleton is very useful to set the orientation of this shape.

BOUNDING BOX

Beside of the graph-based shape descriptors, bounding box is another very interesting shape descriptor, which is useful to merge heterogeneous shape representations.

Bounding box is an enclosing box where a shape can be put inside. Minimum bounding box is the smallest enclosing box of a shape. Oriented bounding box is one of the enclosing box of the shape with respect to an orientation constraint. The minimum oriented bounding box follows the major orientation of a shape, which is useful to locate this shape in a 3D space.

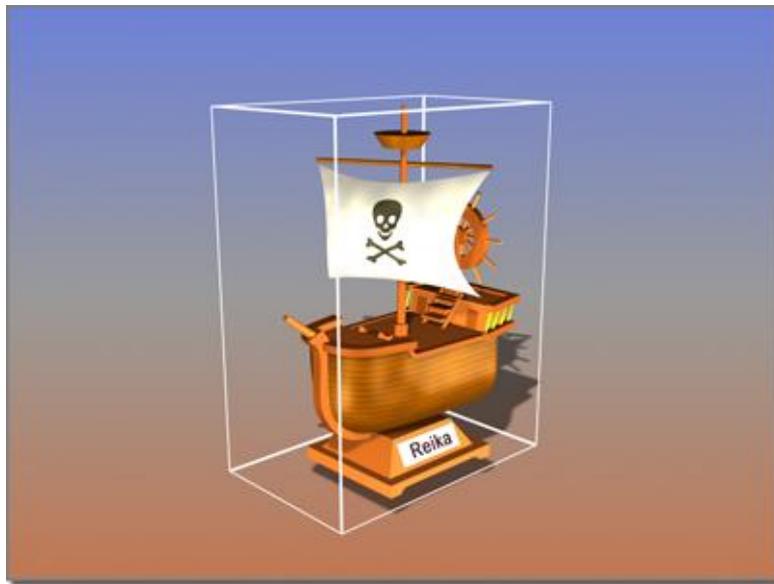


Figure 2.14 Example of a minimum bounding box of a ship¹⁷

The word “semantics” has become popular in recent years, and to indicate is a key issue in image analysis techniques, which is called the “Semantic Gap”. In [98], it is defined as: “the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation”. Semantic-based retrieval tries to extract the cognitive faculties of human beings to map the low level image features to high level concepts so as to reduce the semantic gap. One possible solution is to represent image content with semantic terms, which allow users to access images through text queries, which are more intuitive, easier and preferred by end users to express their intent compared with using images [99]. Researchers are moving towards intelligent image retrieval, which supports more abstraction by understanding the image content in terms of high level concepts.

¹⁷ Image downloaded from : http://www.3dmax-tutorials.com/graphics/il_bounding_box.jpg

2.4 A MULTI-LAYERED SHAPE UNDERSTANDING PARADIGM

From the feature-extraction point of view, 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 (CG) and its application domains, the meaning of “Shape” has become richer.

In [74], a shape is defined by “Parts” and “Relations” which can be summarized as.

Definition 2.5 : Shape = parts + relations

The approach of decomposing shapes into a set of parts before undertaking other shape analysis tasks was proposed early in the 1960s. A shape is then seen as a set of parts that are spatially arranged through the spatial relations among them. A key-issue of applying this definition is to store the information on the shape parts as well as the relationships among them. Two solutions could be found in the literature. One is shape grammars [100], which represent the shape parts with terms and symbols from formal grammars. A shape is then represented as a string of such symbols. The relations are implicitly represented as juxtapositions of these symbols. The other solution is to use graphs. The shape parts are associated with graph vertices while the relations among such parts are represented as edges between vertices.

The spatial relations among the shape parts can be classified in different ways [101] [102]. A possible classification proposed by [103] includes the following three classes:

- Topological relation: this kind of relationship is invariant to rotation and scaling transform, such as “inside”, “outside” and adjacent.
- Distance relation: this kind of relation is linked to quantitative measures. The meaning that two shapes being “far from” or “close to” each other needs to be further specified. Fuzzy modeling plays an important role in this issue.
- Directional relation: this kind of relationship 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 [104] proposed a new way of understanding shapes as can be seen below:

Definition 2.6: Shape = 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 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).
- They have **attributes** (colors, textures, names, attached to an object, its parts and/or its features).

Compared with **Definition 2.5**, this definition shows a broader view of shape. The shape parts and their relations in **Definition 2.5** can be considered as the structure of shape. Their appearance features such as their colors, textures etc. 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 is mainly structured into three different layers including geometric, structural and semantics-based levels (see an example presented in **Figure 2.15**).

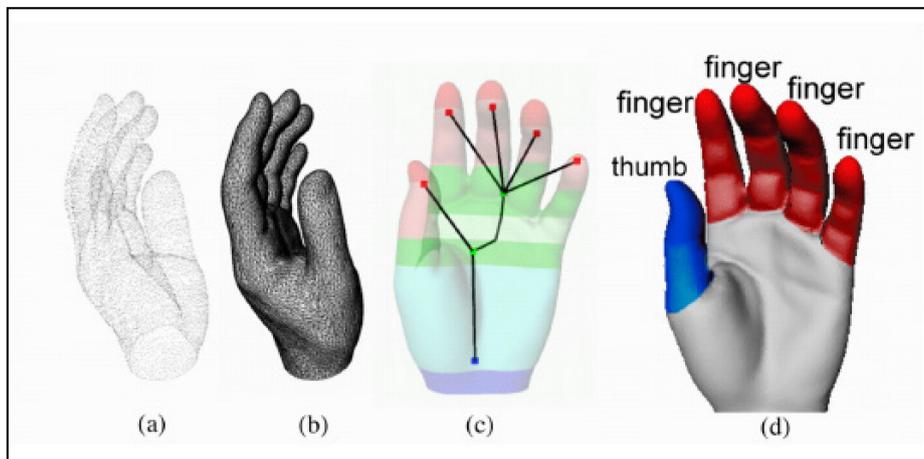


Figure 2.15 Form [104]. A digital shape represented by a point cloud (a); a geometric model of the point cloud, defined as a triangle mesh (b); the structure of the hand model, defined as the configuration of protrusion-like features (c); the model has been semantically annotated, using its structure.

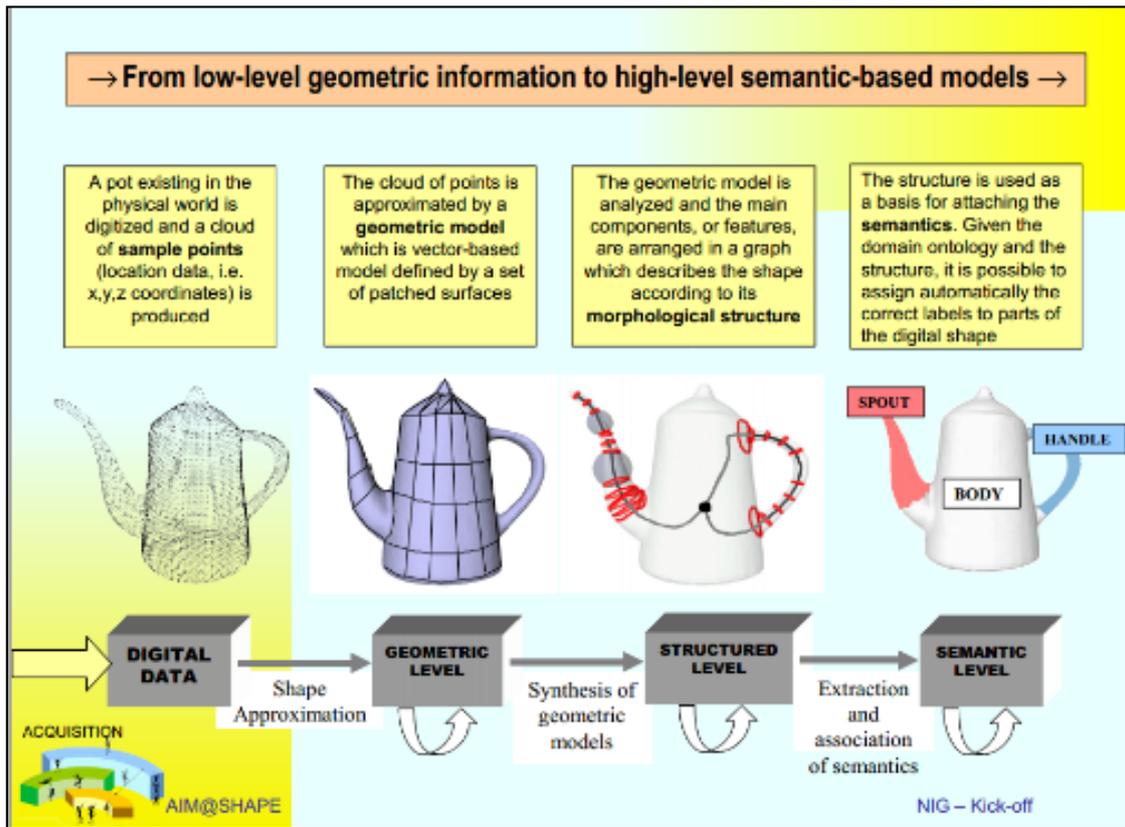


Figure 2.16 AIM@SHAPE Bottom-Up approach [104]

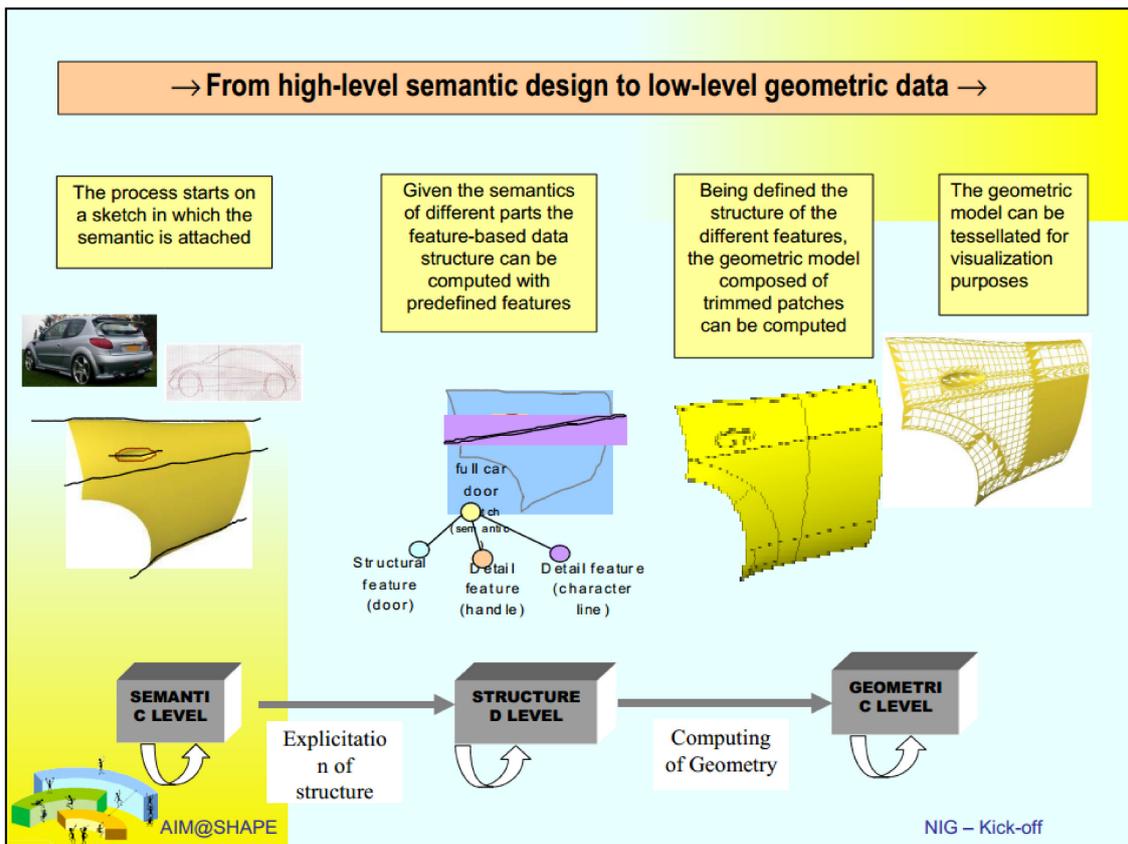


Figure 2.17 AIM@SHAPE Top-Down approach [104]

The general AIM@SHAPE modelling approach is catalogued into two classes. One is a bottom-up approach that extracts morphological structures from low-level geometry and captures the implicit semantic information digital shapes (**Figure 2.16**).

The other one is a top-down approach, which computes the feature-based structure from given high-level semantics to finally reshape the geometric models (**Figure 2.17**).

If all the shape data in the database are structured using this paradigm, then content-based image retrieval and semantic-based shape retrieval will be easier as it groups the information related to shape into different layers. However, today there is no standard format that can support this kind of shape paradigm since different tools are used to capture the information on different layers and store them separately while no one can combine all this information into a single data structure. This is mainly because the shape-generation approaches used today are almost based on the geometric layer only. As a result of the process of generation, the formats typically include geometric information only with some simple semantic information. The standard VRML (a subset of X3D) [X3D, 2013] for virtual reality models can be considered as a most popular standard for 3D applications that embeds additional information along with geometry into 3D models including color, texture, material, environment, etc. However, no morphological structural information has been included. New modeling concepts should be developed to support this shape definition from the very early phase of shape modeling, leading to conceptual design.

2.5 MODELING TOOLS

The objective of this subsection is to review existing 3D shape modeling or VE modeling tools to check how heterogeneous data are used and how the user is involved. To this goal, six criteria are proposed to compare different modeling tools or modeling processes:

Criteria	Gradation of criteria			
	☆	★	★★	★★★
End-user driven	No	Less	In some cases (e.g. for texture provided by the end-user)	Totally
Adapted to non-expert user	No	Less	For simple design	Well suited
Workflow complexity	Non defined	Linear	Concurrent	Complex, Mixed
Input heterogeneous data	No	Simple type	2 or 3 types such as text with image or image with 3D mesh	Multiple resources (text, image, 3D mesh, video, audio, etc.)

Use of heterogeneous data	Non directly used	Simple use (e.g. texture)	Not all used to describe shapes	All used to describe shapes
Time consuming	Very high	High	Medium	Low

Table 2.3 Criteria for modeling tools or process

Based on these criteria, some tools are here reviewed. Most of the values the criteria are set according to the experience of the author while some of them are given by the comparison between different tools¹⁸. The comparison is presented below:

	End User driven	Suited to a non-expert user	Work flow complexity	Input of heterogeneous data	Use of heterogeneous data	Time consuming
Not-manufacturing design based (for modeling, animation, video games, lighting, rendering)						
3ds Max ¹⁹	★	☆	★★	★★	★	★
Blender ²⁰	★	☆	★	★★	★	★
Cinema 4D ²¹	★	☆	★★	★★	★	★
Maya3D ²²	★	☆	★★	★★	★	★
Swift 3D ²³	★	★	★★	★★	★	★
ZBrush ²⁴	★	★	★	★★	★	★
LightWave 3D ²⁵	★	☆	★★	★★	★	★
CAD based						

¹⁸ <http://software.toptenreviews.com/>

¹⁹ <http://www.autodesk.com/products/3ds-max/overview>

²⁰ <http://www.blender.org/>

²¹ <http://www.maxon.net/products/cinema-4d-studio/who-should-use-it.html>

²² <http://www.autodesk.com/products/maya/overview>

²³ <http://www.eraim.com/>

²⁴ <http://www.zbrush.com/>

²⁵ <https://www.lightwave3d.com/>

Sketchup ²⁶	★	★	★	★	★	★★
Solid-Works ²⁷	★	☆	★★★★	★	★	★
CATIA V5 ²⁸	★	☆	★★★★	★	★	★
AutoCAD ²⁹	★	☆	★★★★	★	★	★
Pro/ENGINEER ³⁰	★	☆	★★★★	★	★	★
VE modeling based						
Unity3D ³¹	★	★	★★★★	★★★★	★★	★★
GameWare ³²	★	★	★★	★★★★	★★	★
GameMaker ³³	★	★	★★	★★★★	★★	★
Unreal ³⁴	★	★	★★	★★★★	★★	★
Torque 2D/3D ³⁵	★	★	★★	★★★★	★★	★
Flash ³⁶	★	★	★★	★★★★	★★	★
Reverse engineering based						
Geomagic Design X ³⁷	★★★★	★★	★	★★	☆	★
Pilot3D ³⁸	★★★★	★★	★	★	☆	★

²⁶ <http://www.sketchup.com/>

²⁷ <http://www.solidworks.com/>

²⁸ <http://www.3ds.com/products-services/catia/products/v5/>

²⁹ <http://www.autodesk.com/products/autocad/overview>

³⁰ <http://zh-cn.ptc.com/product/creo>

³¹ <http://unity3d.com/>

³² <http://gameware.autodesk.com/>

³³ <http://www.yoyogames.com/studio/>

³⁴ www.unrealengine.com/

³⁵ www.unrealengine.com/

³⁶ <https://www.adobe.com/products/flash.html>

³⁷ <http://proto3000.com/rapidform-xor-reverse-engineering-software-benefits.php>

³⁸ <http://pilot3d.com/>

ReShaper ³⁹	★★	★★	★	★	☆	★
VRMesh ⁴⁰	★★★★	★★	★	★	☆	★

Table 2.4 Comparison of modeling tools and systems

Aesthetic design-based tools focus on the digital representation of shapes. These shapes are going to be used in VE such as films, video games. In general, they are not end-user driven as they face different types of end user. CG and design knowledge are also required. Workflows are not very complete as CAD based tools, as they do not define assembly or constraints. They use both 2D images and 3D models. Most of the time 2D images are considered as texture for 3D surfaces.

However, CAD-based models do not really focus on the “virtual appearance” of shapes. They are suited for industrial products, which need to be manufactured in the future. Therefore, not many heterogeneous data are used. At times, 2D images are used to present different views of a product. The working process in the CAD model is complex, as not only the parts of a product need to be defined but also assembly, constraints and manufacturing properties.

VE modeling-based tools are not aimed at designing a single shape, but at rearranging different 3D models in a virtual scene. Thus, multiple resources are involved such as image and video textures, texts and audio.

Reverse engineering-based tools focus on the reconstruction of specific resources. The input is totally provided by the user. However, this input is not used to directly modelling more complex shapes, but is then processed and converted into another type of model for the same object.

2.6 CONCLUSION AND REMARKS

This chapter reviewed previous works on shape modeling / representation approaches and tools towards the modeling by reusing heterogeneous data. Then some useful shape descriptors are also listed in this chapter. Finally, comparisons on modeling tools are presented. These reviews enables us to highlight the following facts:

Statement 8: Existing modeling approaches define a 3D shape in 3 different ways:

Traditional modelling approaches

Feature-based modeling

Reverse engineering (from lower dimensional input to higher dimensional output)

Statement 9: Reusing heterogeneous data in the shape modeling process works in 3 different ways:

2D image as texture

³⁹ <http://www.pcdmis.com/products/pc-dmis-reshaper>

⁴⁰ <http://vrmesh.com/>

Text, image, video as 2D planar surface

Point cloud, 2D image as input for reverse engineering (for point cloud meshing, image-based modeling)

Statement 10: Each shape representation or descriptor is used for specific application domains. There is no representation or description of shape that can be considered as a common input for all applications.

Statement 11: Graph-based or structure-based descriptors provide high-level information about a shape as opposed to low level geometry, which could be potentially used for defining new shapes.

Hence we can conclude that:

Conclusion 1: Existing modeling approaches do not reuse heterogeneous data to define new shapes.

This is due to mainly two reasons:

- 1 Representing heterogeneous data in the same environment asks for extracting information from these multiple structured inputs, which increases the complexity.
- 2 There is no standard structure supporting multiple resources in the same environment where they need to be rearranged all together.

Conclusion 2: As Shape is about its parts and relations (topological, distance, directional) its representation should contain information on geometry, structure and semantics.



- PART B: CONTRIBUTION -

THE SECOND PART AIMS AT the proposed models and related tools which have been developed to meet the objective specified at the end of **Chapter 1**. It is composed by four chapters.

The first one (**Chapter 3**) starts with a case study of a traditional VE generation process, leading to the presentations of two projects (VISIONAIR and Co-DIVE) on which this Ph.D. thesis is based. Inspired from these two projects together with the results from **- Part A: Background and state-of-the-art -**, this chapter ends with a proposed VE modeling process defined in the Co-DIVE project.

The second chapter (**Chapter 4**) is focusing on the introduction of the proposed shape description model, which enables to distribute the information related to a shape at three levels of geometry, structure and semantics.

The third chapter (**Chapter 5**) presents the user-centered process to use the proposed shape description model and the solutions for the related issues during this process.

The last chapter (**Chapter 6**) introduces the implementation developed based on the proposed shape description model including the user graphical interface, solutions for interaction and three examples made using this system.



Start by doing what's necessary, then do what's possible, and suddenly you are doing the impossible

Saint Francis of Assisi

3.

TOWARDS A NEW APPROACH FOR SHAPE SPECIFICATION AND CREATION

CHAPTER OVERVIEW

THIS CHAPTER STARTS WITH an example of real industry VR application after a brief introduction (**Section 3.1**). This example highlights the problems existing in today's shape specification and creation process for developing a Virtual Environments (VE) (**Section 3.2**). Towards the solution of these problems, two projects are introduced (**Section 3.3**) within which the work done in this thesis is carried out. One is the VISIONAIR European infrastructure project, a world-class infrastructure for advanced 3D visualization-based research, and the other is the Co-DIVE French national project, which addresses the development of models, methods and tools to support the conceptual design of VE. With the results of the literature reviews stated in the previous - **Part A: Background and state-of-the-art** - and the arguments presented in this chapter, **Section 3.4** summarizes the location of this Ph.D., while the last section lists a conclusion and some remarks (**Section 3.5**).

3.1 INTRODUCTION

The term *Information Technology* (IT) in its modern sense first appeared in an article published in the Harvard Business Review in 1958. Since that time, many applications have been developed to store, to retrieve, to transmit and to manipulate data⁴¹. This leads to new challenges not only to visualize, to capture, to search, to share, to store, to transfer but also efficiently re-use this huge amount of data.

More recently, Virtual Reality (VR) applications have been developed to solve some of those issues in many fields including industrial design, civil engineering, architecture, marketing, museums, video games, etc. Together with the development of dedicated models, methods and tools, such technologies are very promising.

In this context, the European project called VISIONAIR⁴² (VISION Advanced Infrastructure for Research) has been proposed to create a world-class research infrastructure that is open to research communities across Europe and around the world providing access to advanced visualization capabilities for conducting state-of-the-art research. In conjunction with VISIONAIR, the French national project called Co-DIVE (COnceptual Design In Virtual Environment) has been conceived to develop tools for Virtual Reality environment generation exploiting the huge amount of already created graphical resources as well as the most advanced geometric processing tools for shape understanding and representation.

Within the scope of these two projects, this thesis addresses the representation and modeling of digital objects using existing graphical resources to be reused and combined by neophyte users during the development of new Virtual Environments.

To better introduce the background of some issues tackled by these two projects, this chapter starts with a case study of a traditional VE generation process for a real VR application.

3.2 AN INDUSTRY EXAMPLE: LOOKX

Today, the VE specification and shape modelling process is long and tedious. To illustrate this issue, this section analyses a real example of a Virtual Reality (VR) application developed for a site protection company called Dirickx⁴³.

3.2.1 DESCRIPTION

The French company Dirickx is specialized in site perimeter protection offering fences, gates, access control and security solutions. Their VR application is called “Lookx”⁴⁴ aiming at personalizing fences and gates of a site (**Figure 3.1**).

⁴¹ Daintith, John, ed. (2009), "IT", A Dictionary of Physics, Oxford University Press, retrieved 1 August 2012

⁴² Home page <http://www.infra-visionair.eu>

⁴³ Home page : <http://www.dirickx.com/>

⁴⁴ Web page: <http://www.dirickx-cloture-lookx.fr/configurez-votre-cloture.php>

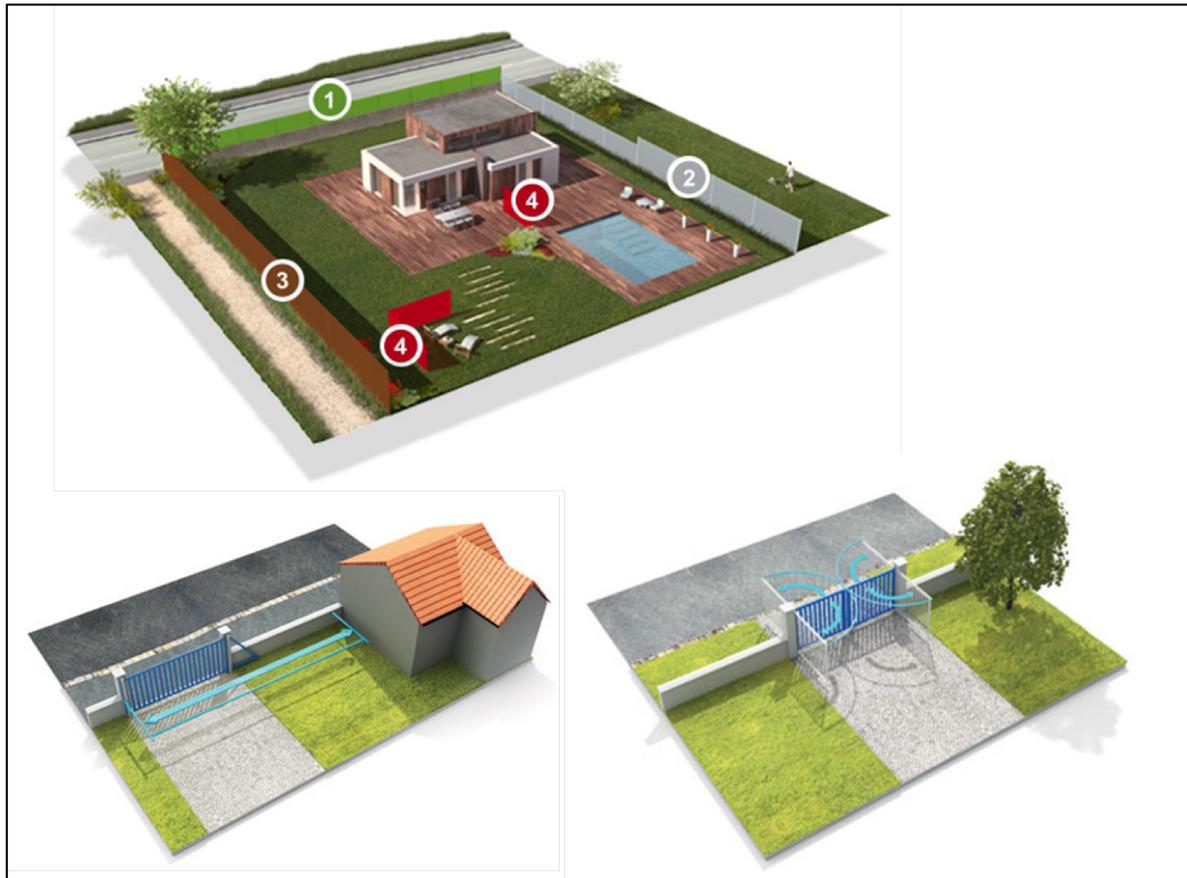


Figure 3.1 Using the application “Lookx” to personalize fences and gates of a site

Therefore, the presented use case can be characterized as follows:

- Application domain: site protection
- Application usage: Personalization of the products for clients
- Product categories: home protection, property protection, self-designed space protection, privacy protection
- Desired functions:
 - Presenting the predefined products in a 3D environment
 - Configuring each product including:
 - Location
 - Number
 - Size
 - Texture
 - Specific configuration depending on different products (e.g. angle between two connected fences)
 - Capture of a picture of the configured site
 - Price estimation
- Operation environment: PC (Windows + Mac), Smartphones and tablets

Using the criteria defined in **Subsection 1.2.3**, the design of this application can be categorized as below:

- Actor complexity: ★★
- Product complexity: ★
- Cognitive style or trait: ★★
- Cognitive process: ★
- Problem structure: ★★
- Design processes, practices, culture or tools: ★★

3.2.2 ANALYSIS

This application is developed under a traditional VE modelling process. Different actors are working together to develop this application. The workflow of this traditional process is presented in **Figure 3.2**.

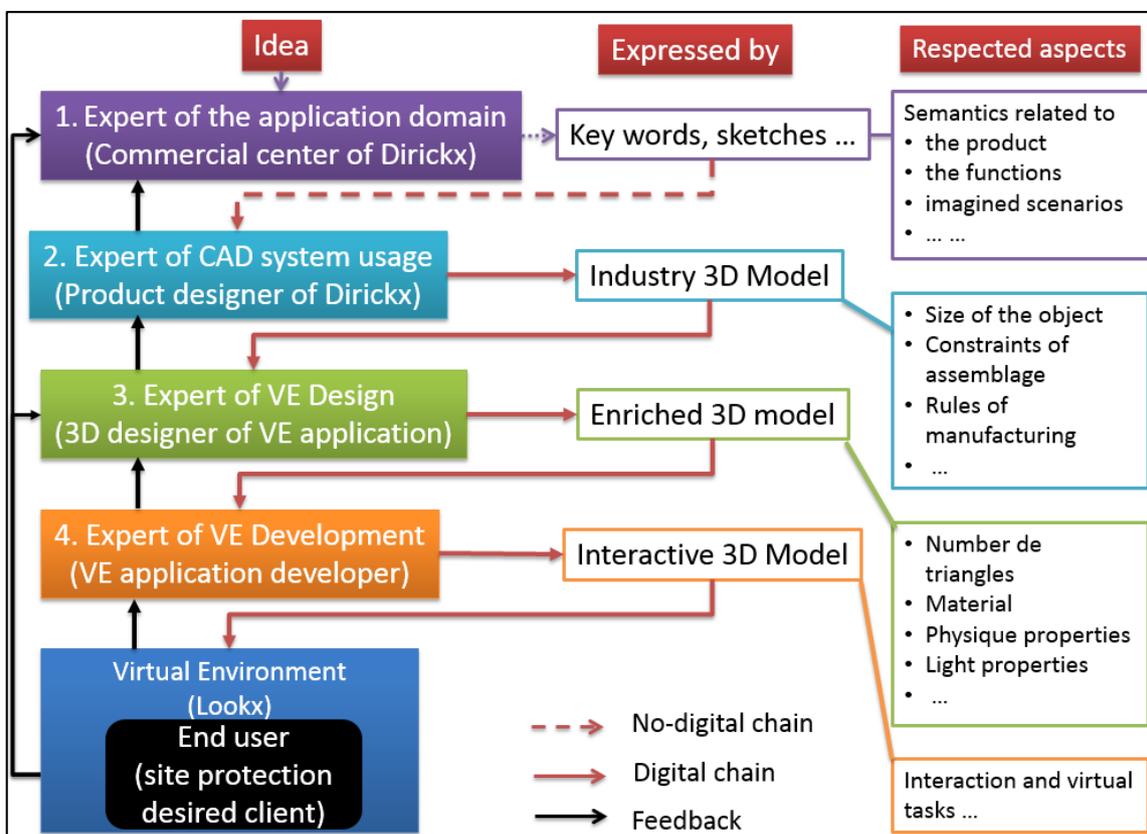


Figure 3.2 Traditional VE modelling approach

Each expert is going to be introduced through the example of “Lookx”:

- Expert of the application domain: These are persons who want to create a VE with a specific usage in mind related to their specific professional domain. In the example of “Lookx”, they belong to the commercial department of the Dirickx Company. Starting from the idea of creating an application for the configuration of fences and gates, they

express their needs by using keywords or sketches as shown in **Figure 3.3**. Here, they sketch a simple 2D plan of the VE to be built using some blocks showing the different virtual objects associated with some keywords. As they master the selling of products, but their computer graphics and design knowledge is limited, they give the plan to another actor in the VE modelling process.

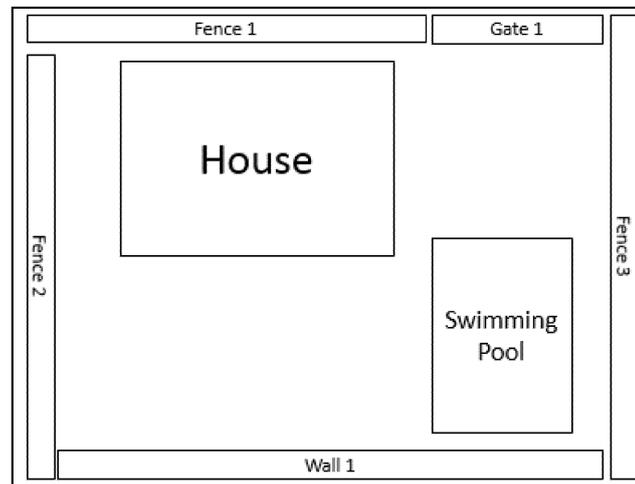


Figure 3.3 A sketch designed for “Lookx”

- Expert of CAD system usage: The CAD experts are those who build the CAD models for product specification and manufacturing. They are mainly considering the aspects related to the final manufacturing of this model, such as the exact size of the object, the assembly information, shape and position tolerances, etc. They are neither working on the behaviors of the models nor on the interactions with the users. They may use various approaches to build the 3D models, such as feature-based modelling approaches or reverse engineering techniques. They may also need a 3D data management system and a system for the maintenance of the consistency between the different building blocks, to ease possible modifications and avoid time-consuming manual manipulations. In our example, the expert of CAD system usage is the person who designs the industrial 3D CAD model of the fences and gates. Each model is fully defined with dimensions, assembly constraints, manufacturing tolerances⁴⁵. However, these models are too detailed and heavy to be directly manipulated in a VE, where real time interaction is required. Thus, they require an adaptation to meet the VE system characteristics which are normally unknown to the CAD expert.
- Expert of VE design: For a VE application, some of the information contained in a CAD model are not useful within the VE. For example, if an object has not to be decomposed in the VE application, the information related to its assembly are useless. For instance,

⁴⁵ For the reason of protection, these industrial 3D models cannot be presented in this manuscript.

in the case of a TV, just a 3D model of a textured parallelepiped can be enough. The inside parts of the TV, while fundamental for its production, are not necessary if there are no actions on them in the VE application. On the other hand, extra information not existing in the CAD model can be required in the VE. Thus, the VE designers have to modify the object representation. Being the CAD and VR systems often based on different shape models (B-Rep vs. tessellated models), VE designers first have to convert and simplify the CAD model to obtain a lighter and simpler VE model. Then, they do have to enrich it with some additional properties such as textures and materials. The **Figure 3.4** shows an example of an enriched 3D model of a fence designed in the “Lookx” application. In this example, two different textures (the red one and the wood colored one) have been added to the fence geometrically defined by simple parallelepipeds. Another task for the VE designer is to create the user interaction graphical tools, such as menus, buttons and other controllers.

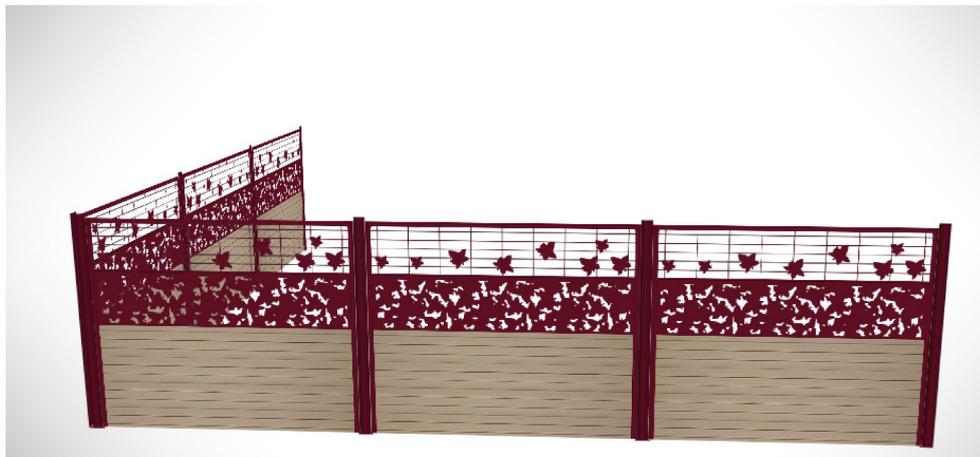


Figure 3.4 Enriched 3D fence

- Expert of VE development: The VE developers deal with the user-application interaction. Based on the requirements of the foreseen application, they need to consider the software and the hardware environment for its usage and development. In the example of “Lookx”, the application has to work on the web on smart phones and tablets. Therefore, considering the graphic characteristics of these devices, this application does not consider a high-level visualization environment. This is the reason why Flash player is chosen as the software environment. After choosing the hardware and software environment, the VE developer needs to select a tool and a programming language to create the application. During the development phase, behaviors will be added to the enriched 3D models obtained from the expert of VE design. These behaviors specify the user’s interaction modalities and model reactions to the user’s actions. For the example of “Lookx”, the end user can change the texture or the size of the chosen fence as presented in **Figure 3.5**. The code written

by the expert of VE development will finally check that each 3D model behaves correctly to realize the different virtual tasks defined by the expert of the application domain.

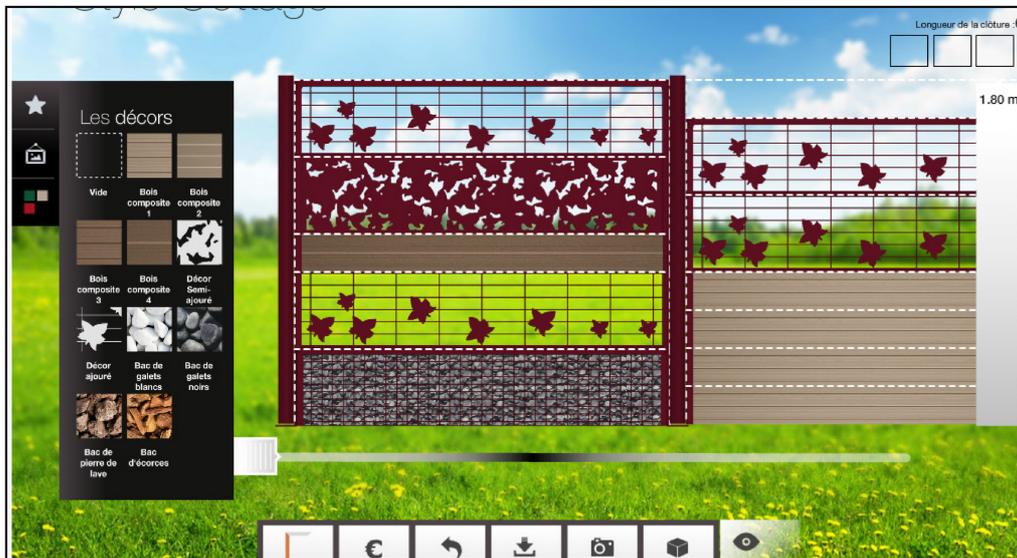


Figure 3.5 Configuration of a fence from the application of “Lookx”

In this process, as presented in **Figure 3.2**, there are multiple feedback loops. If there is something wrong in the final VE, it may require the intervention of all the experts involved and the repetition of various steps, possibly all. For example, if two objects are colliding, which is not allowed, the expert of VE development could design a warning box, or change the movement trajectory of the two objects. If these solutions do not solve the problem, then this problem will be passed to the expert of the VE design who may change the size of the two objects. There could be also other reasons that cause this problem. For example, one of the objects has not been well assembled by the CAD expert, or the scenario is not well designed by the expert of the application domain and these two objects should never be put together. Therefore, solving one problem may cause the intervention of all actors, thus requiring several modifications, which are usually long and tedious. In addition, multiple solutions could be found making it difficult to decide which the most suitable one is. The main reason for this is that they are sequential and with multiple feedbacks. Therefore, the inefficiency of the traditional VE modeling process is the main issue that needs to be solved.

Another important limit of this process is related to the loss of information between the successive steps. It can be noticed that the output (keywords, sketches, sentences, etc.) of the first actor (expert of the application domain) is not coded in the digital chain as presented in **Figure 3.2**. However this information directly conveys the needs and characteristics of the future VE. In other words, there exists a gap between the description of the VE by the application domain expert and the digital chain. This is due to the fact that the experts of the application domain are often not experts in computer graphics or digital design. They cannot directly express their ideas in the digital chain.

In addition, the data the experts of the application domain are using to express their ideas are not in the same data format. Actually, people are very frequently taking inspiration or are describing things through analogies, i.e. taking something existing and describing what they want to preserve or change (see **Subsection 1.2.3** about idea generation techniques for creative conceptual design). There are multiple heterogeneous resources that people can use to describe new shapes, such as key words, 2D images, 2D drawings, scanned 3D objects (point clouds), or existing 3D models. For example, in **Figure 3.3** there are some 2D blocks and some key-words to describe a VE. Although different data might be used for describing the same object, they are coded in different ways and today there are no user-friendly tools that can integrate all these data to make them compatible in the same digital modeling chain, without converting every representation into the same model, especially for a creative design phase.

Finally, the way the experts interact with the system is also important. In this sense, the definition of the specific behaviors (assembly constraints, displacement laws, etc.) has to be thought in such a way that it can be defined and modified by all the experts of the modeling process.

In this actual VE modeling process, several key problems and characteristics can be summarized as below:

- The inputs and outputs of each actor are different. Starting from an idea, the successive steps are so that the output of an actor become the input of the following actor, and so on. This process makes use of different types of digital models adapted to the various needs at the different steps:
 - Different actors need to consider different aspects related to their own domain of expertise;
 - This is a sequential process with multiple feedbacks and loops which slow down the design of VE (the black arrows in **Figure 3.2**);
 - The initial input heterogeneous data are not preserved and directly used in this actual modeling process.

The way experts interact with the system and specify the behaviors is not enough user-friendly and it is hard to interact with it for neophyte users.

3.3 TWO PROJECTS RELATED TO THIS PH.D. SUBJECT

3.3.1 THE VISIONAIR⁴⁶ PROJECT

The aim of VISIONAIR (VISION Advanced Infrastructure for Research) is to establish a European infrastructure for high-level visualization facilities that is open to research communities across Europe and around the world. By integrating existing facilities of a pan-European

⁴⁶ Home page: <http://www.infra-visionair.eu/>

network, it creates a world-class research infrastructure enabling to conduct cutting edge research. On many sites across Europe, it is infeasible to have at disposal the necessary visualization facilities needed to tackle high fidelity, large screen and/or immersive visualization. VISIONAIR is targeted to fill in this gap by providing access to the partner facilities, opening its doors for interested researchers to use the multitude of services available across the European visualization facilities. After submitting a successful research proposal, international researchers are invited to visit the partner's facility that best fits the scientific goals of their research. Users are not only given access to the top visualization facilities in Europe, but they are also supported in their experiments by funding their living and travel expenses. Researchers can choose from over 20 facilities located in 12 countries in Europe and Israel.

The project targets different fields of visualization and offers access to methods, software and hardware needed for successfully visualizing scientific data in various application fields such as engineering, medical, biology, chemistry or physics. Ultra-High-Definition facilities connected by high-speed networks are targeted at users who want to create high-resolution, high quality images (up to 8k) and possibly access those by high-speed networks. VISIONAIR provides the hardware and the unique network distribution services needed for the transmission of images to their end-points. The network services enable multiple high-resolution digital-media streams to be transferred, using dynamically available optical light paths across multiple domains, which can be used on scheduled or on-demand basis. While Scientific and Ultra-High-Definition Visualization can be done in any environment, researchers specifically targeting Virtual Reality can also apply to a multitude of facilities. Here, the focus is on immersive – possibly also haptic – experiences in virtual environments. Equipment available for researchers, ranges from head mounted displays to fully fledged stereoscopic PowerWalls and CAVEs. Further specialized equipment available allows users to carry out research by using advanced interactive facilities, such as multi-touch displays and Augmented Reality, a technique that enables users to overlay the real environment with context dependent computer generated images. Researchers also have access to the latest developments in display technology, like holographic displays or the above mentioned 8k displays.

The project maintains also an already huge database of visualization software and graphical models that is available to all researchers for free. Thus, experts can explore the multitude of visualization packages that are already available. Software covered here ranges from processing filters, converters and readers to fully-fledged modelers and visualization packages.

In order to improve the quality and variety of the services offered to external hosts, the project includes also some joint research activities. Among them, one is related to the development of tools for simplifying the collaborative creation of virtual environments also for non-expert users. This action is partially carried out by AMPT and CNR-IMATI and it is strongly related with the objectives of the project Co-DIVE.

3.3.2 THE CO-DIVE PROJECT

PROJECT DESCRIPTION

The Co-DIVE (COnceptual Design In virtual Environment) project aims at defining, developing and testing a set of models, methods and tools to overcome the limits of the actual VE modelling process while bringing together most advanced results at the level of the geometric modelling and at the level of the VE development process. At the end, the newly defined approach should enable the collaboration between experts in the fast definition and direct generation of the final VE as presented in **Figure 3.6**.

As described before, the involved actors, i.e. the VE application developer (expert of VE development), the VE end-user (expert of the application domain) and the geometric designer (expert of CAD usage and expert of VE design) have different views of the VE. This is because they have different needs. To better satisfy all their needs on information sharing, the Co-DIVE project tries to find a conceptual model (in the middle of **Figure 3.6**) linking the actors, tools and models interacting within the VE modelling process.

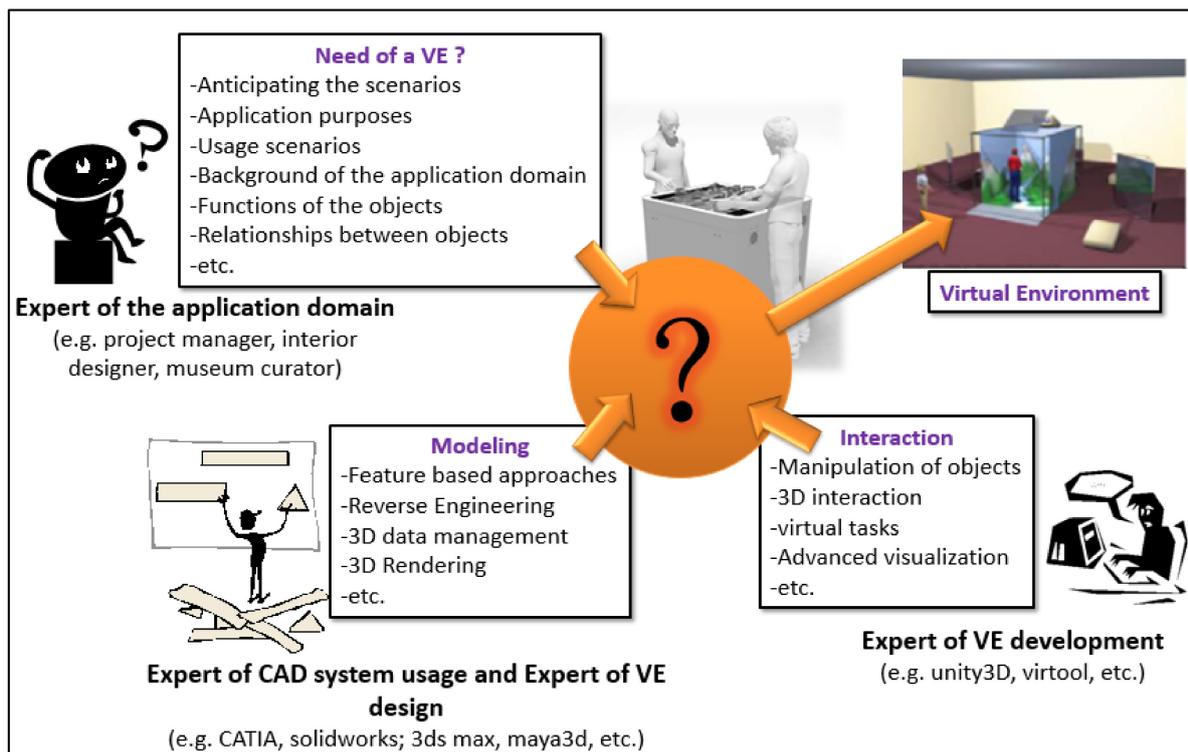


Figure 3.6 The different actors involved in the VE modelling process

In addition, to define the conceptual model, this project is also targeting at a creative idea generation technique (as mentioned in **Subsection 1.2.3**) by using existing resources to describe and characterize the virtual objects and their relations.

Moreover, to make easier the user interaction with the system, as in the case of smart phones technology, which pushes the use of touch screen interfaces, also multi-touch devices

can be used during the design process. Therefore, this project is also targeting a 3D multi-touch table (from a French company called “Immersion”⁴⁷) in order to make people collaborate around the same conceptual model. This table allows multiple users working together on a 3D screen, which gives them a better experience of visualization.

As a conclusion, here some scientific challenges targeted by the Co-DIVE project for the optimization of the VE modeling process are listed:

- Integration of the end-user(s) within the development process of the VE;
- Setting-up of a simplified model (the so-called conceptual model) of the VE from a set of user-specified functional requirements;
- User-friendly interface such as a touch screen modelling;
- Use of heterogeneous data in the digital chain;
- Optimization of the data exchanges between different domains and the VE modelling environment;
 - Semi-automatic generation of the VE;
 - Maintenance of the consistency between the building blocks, to ease the possible modifications and avoid time-consuming manual manipulations;
- Possibility of directly providing a behavior code.

In order to improve the actual modelling process and solve some of the above mentioned issues, the Co-DIVE project has been decomposed into two work packages: object modelling and scene modelling. The tasks of developing the two models and the related tools are distributed to two different Ph.D. subjects. **This Ph.D. thesis focuses on the object modelling phase and associated models, methods and tools.** The other Ph.D. thesis addresses the way the experts interact with the objects during the design phase as well as during the use of the virtual environment.

PROPOSED APPROACH

Within the Co-DIVE project, a new integrated VE modeling framework is proposed. It is not sequential and it allows the use of heterogeneous data shared between the different actors and experts whose roles have been previously introduced. This approach is represented in **Figure 3.7**. The proposed framework is built on top of a new shape description model (set in the middle of **Figure 3.7**) that is used as a common reference for the different actors for the VE specification since the initial idea description.

This new shape description model can be generated and manipulated by anyone, who could be a CAD expert or who doesn't have any knowledge in CAD or Computer Graphics.

To this aim, this new model should be able to deal with the different types of information used for the object and environment specification at the various phases. It should be generic enough to be linked to various authoring systems and to be suitable for indexing to facilitate a

⁴⁷ Home page: <http://www.immersion.fr/table-ilight-3d-touch/>

re-use of the created resources.

To guarantee the effectiveness of such an integrated collaborative design process, this intermediate model should be conceived to be a suitable basis for a future development of new methods and tools to:

- 1) Generate new 3D geometric models and assemblies (feature-based approaches, deformation techniques, etc.) from the multiple inputs;
- 2) Search similar models (including heterogeneous ones) within an existing database and re-use them for meeting new needs (addition/suppression, simplification, deformation, etc.) so as to reduce the gap between the wished objects and the ones found in the database;
- 3) Integrate different actors (sharing information, data exchange, etc.) in the framework and ease the co-modelling of the VE application so as to reduce the development time.

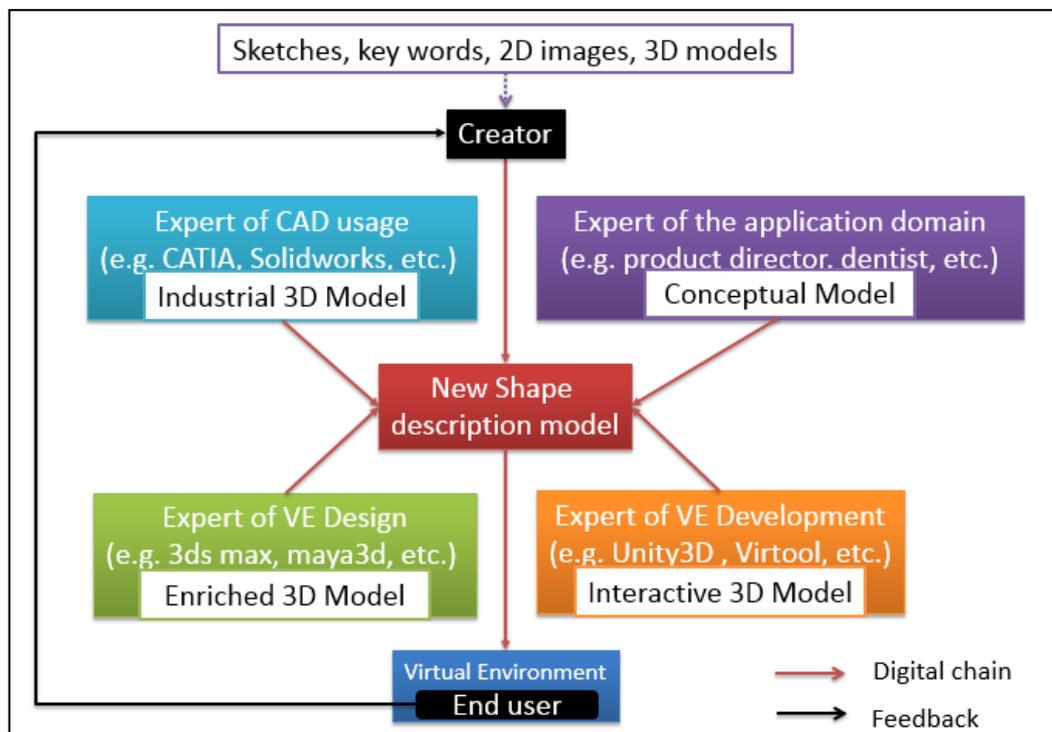


Figure 3.7 The proposed process for a VE modelling system

In other words, the model is the container of the needed information. What have to be developed are the methods and tools that provide the capabilities of storing and exploiting information for the realization of the actual models needed by the VE application.

Based on the criteria identified in **Section 2.5**, the two approaches (traditional VE modelling approach and the proposed one) are compared in **Table 3.1**:

	End User driven	Adapted to non-expert users	work flow complexity	Input heterogeneous data	Use of heterogeneous data	Time consuming
Traditional process	★,or ★★★	★	★	★,or ★★	★,or ★★★	★
Proposed process	★★★	★★★	★★★	★★★	★★★	★★★ or ★★★

Table 3.1 Comparison between the two processes

This PhD thesis addresses the specification of such common model and framework as well as the development of easy-to-use software tools for its creation supporting the conceptual design of the virtual assets to be inserted in a VE application.

3.4 PURPOSE OF THIS PH.D.

Within the scope of these two projects and the analysis of the state-of-the-art presented in - **Part A: Background and state-of-the-art** -, this Ph.D. thesis is not aiming at developing the related approaches and tools for the whole proposed modelling process (**Figure 3.7**) but the centered “New shape description model”. In this manuscript, this model is named as Generic Shape Description Model (GSDM). A detailed view and the location of this Ph.D. thesis is illustrated in the following **Figure 3.8**.

A pre-processing stage is needed to add additional information on the raw input heterogeneous data using shape representation / description and reverse engineering techniques (as presented in **Section 2.2.2**), such as segmentations, skeletons, Reeb graphs, etc., which can be useful to structure and define new shapes. The GSDM should take this enriched input and reorganize it in three layers of “geometry”, “structure” and “semantics” to explain what the parts and relations of a shape are, as defined in **Definition 2.5**. This model should also be able to be finally transformed to different digital representations by a post-processing stage using reverse engineering and geometrical manipulation techniques (e.g. image-based modelling, morphing, etc.).

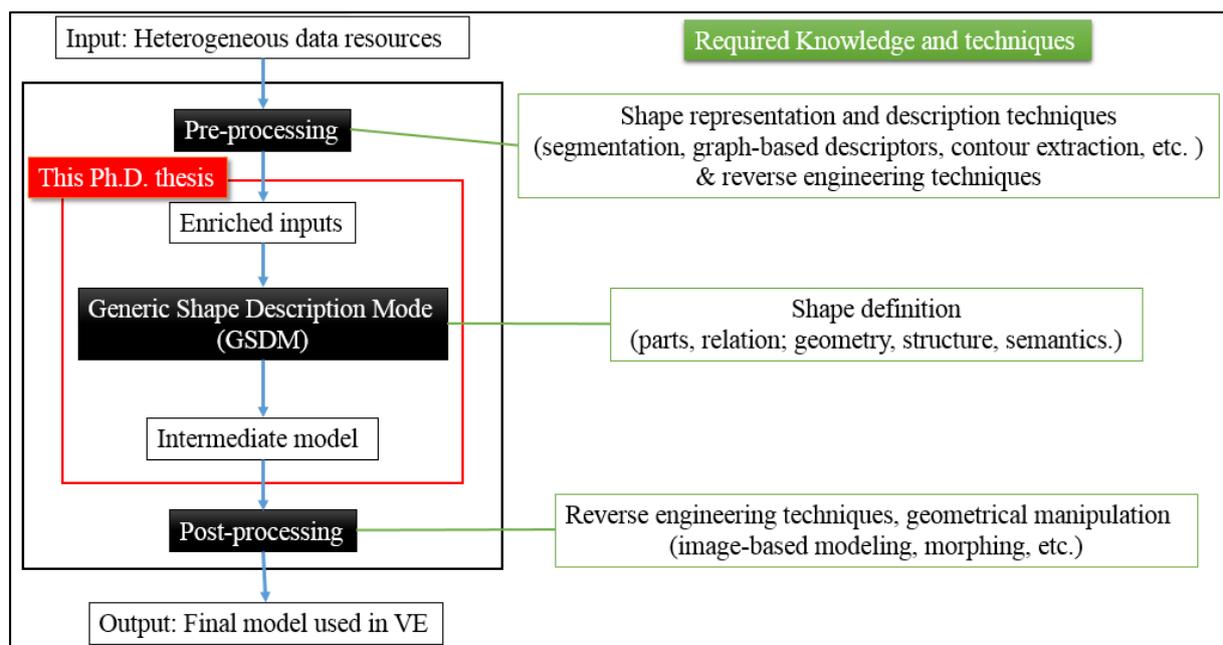


Figure 3.8 Location of this Ph.D. thesis

The red box in the center of **Figure 3.8** is where this thesis is located. The objective introduced in **Section 1.3** can be further detailed as below:

Objective: Development of the Generic Shape Description Model (GSDM) and the mechanisms used to reduce gaps:

- Gap 1 (between people with creative ideas and conceptual design tools):
 - By describing the concepts of “part” and “relation” (topological, distance and directional) of shape (introduced in **Definition 2.5**)
 - Through simple and smart user interfaces for non-expert users
- Gap 2 (between conceptual specification and VR implementation) by
 - Extracting information from heterogeneous input data
 - Reorganizing information into three layers of geometry, structure and semantics (introduced in **Definition 2.6**)

To make the conceptual design phase more natural and easy for non-expert users, multi-touch capability is also considered as a way to better interact with the conceptual model and related virtual objects.

3.5 CONCLUSION AND REMARKS

This chapter has positioned the work of this Ph.D. thesis within the context of two projects in which I have been involved. One is a European project called VISIONAIR aiming at creating a world-class research infrastructure. The other one is called Co-DIVE and it aims at bridging

the gap between different actors during the VE modelling process using heterogeneous data. The traditional VE modelling process has been presented through an example of a VR application called “Lookx” developed by the “Dirickx” French company. Through this example, several issues have been highlighted to justify the need for developing a completely new innovative integrate VE modelling framework. This new approach is no longer linear and sequential but concurrent and collaborative thanks to a new shape description model shared by the multiple actors. Based on the objective of this Ph.D. specified above, a new shape description model has been designed together with the related methods and tools. In the following, **Chapter 4** will introduce the definition of this conceptual model; **Chapter 5** will explain how to use this model and **Chapter 6** will present an implemented environment for evaluating the proposed model through a set of examples.



Design is not just what it looks like. Design is how it works.

Steve Jobs

4.

GENERIC SHAPE DESCRIPTION MODEL

CHAPTER OVERVIEW

THIS CHAPTER PROPOSES a new Generic Shape Description Model (GSDM), which may not only be used for the Co-DIVE approach but can also be considered as an independent model for the conceptual design of digital shapes. First an overview of the GSDM is described detailing its three different information levels (**Section 4.1**): data, intermediate and conceptual level. The following three sections (**Section 4.2, 4.3 and 4.4**) give a detailed definition and explanation of the three levels. An overview of the whole data structure of GSDM is presented in **Section 4.5**. Finally, the last section contains a conclusion and some remarks (**Section 4.6**).

4.1 GSDM – A MULTI-LAYERED FRAMEWORK

To clarify the specified objective defined at the end of **Chapter 3 (Section 3.3)**, the GSDM is structured with three levels of information: **Conceptual level**, **Intermediate level** and **Data level**.

On the *conceptual level*, three basic elements are defined to describe the meaningful object constituents and their relations: **Component**, **Group**, and **Relation** (the details of these three notions will be presented in the following sections). They are the three elements directly manipulated by the user. The conceptual level is aimed at reducing the gap defined in **Problem 1**. At this level, two questions arise:

- What are the different parts of this object?
- What are the relations between them?

Component & Group are designed to answer the first question and *Group & Relation* are developed to answer the second question (*Group* explains the topological relation of parts while *Relation* explains the distance and directional relation of parts). They help the non-expert user to have an overview of what is going to be described, but not a precise description. There are still two questions that need to be answered to clearly understand what the user has in mind:

- What does each part look like and what is its meaning?
- How are the different parts connected?

This second question finds its answer in the second level of the GSDM. On this level, the local geometric and structural information of a part are addressed. For example, two parts are connected by indicating the location of one with respect to the other. At this stage, the whole geometry or the whole structure of each part does not need to be accessed by the user, unlike some of the **Key Entities** that represent the anchorage element where restrictions on the related locations should be specified. Limitations on reciprocal location between *key entities* are labeled as **Constraint**. This information level is referred to as **Intermediate Level**.

To answer the first question requires knowing the **Geometry**, **Structure** and **Semantics** of each part; these three aspects constitute the so-called *data level* of the GSDM. This information provides both the appearance and the meaning of a part. They are also used to indicate the real location of the *key entities* for the application of the *constraints*. This level is only partially handled by the non-expert user and tackles the heterogeneous inputs.

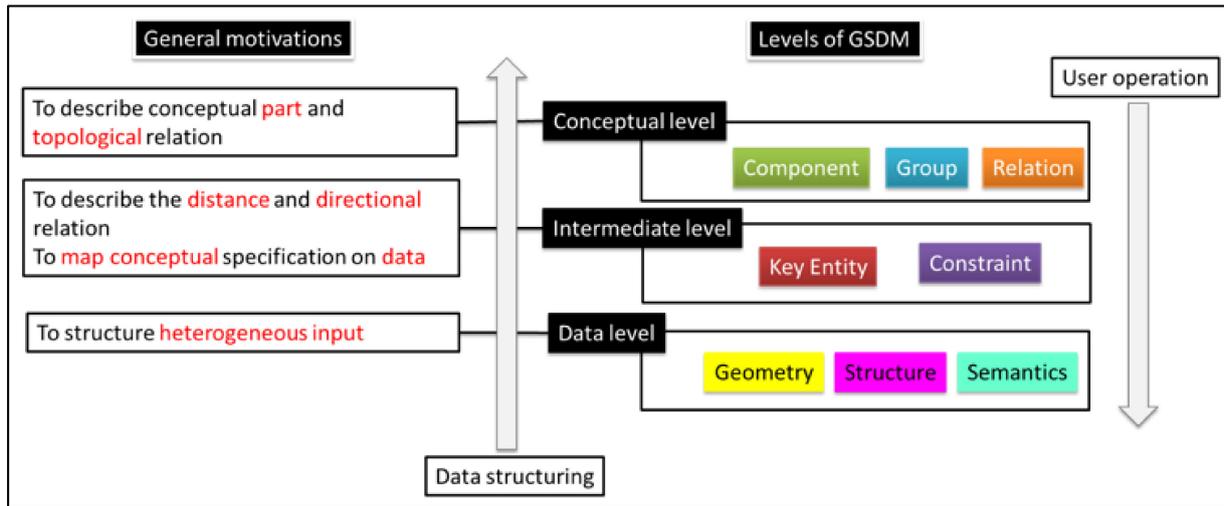


Figure 4.1 The different levels of GSDM and their motivations

The different levels and definitions are represented in **Figure 4.1**. From the user's point of view, the *conceptual level* is the easiest to understand and know where he can directly work. The user can also work on the *intermediate level* to include some more information on the object's sub-parts arrangement. The *data level* is used for representing the heterogeneous information, specifying *key entities*, visualizing *components* and modifying the shape of *components* if needed.

From the information structuring point of view, *geometry*, *structure* and *semantics* are the three basic elements without any other notion associated. Concerning *constraint* and *key entity*, these are based on the *data level* information and have their own structures. *Component*, *group* and *relation* provide the most user oriented information and are expressed in terms of the lower level information.

The following sections show the details of the different elements of the GSDM as well as the relations between them. Each one is introduced detailing four aspects: definition, motivation, properties and data structure as described in UML with examples.

4.2 DATA LEVEL (GEOMETRY, STRUCTURE AND SEMANTICS)

As presented previously in **Definition 2.6**, information on a shape can be organized in three layers: geometry, structure and semantics. This paradigm for understanding shapes offers a way to integrate shapes in application domains, to perform reasoning and comparisons independently from their representations. No matter if a shape is represented by a 2D image or a 3D mesh, the information associated can always be distributed into these three layers.

4.2.1 GEOMETRY

DEFINITION

Definition 4.1: Geometry = The spatial extent of an object.

The “*geometry*” of an object, that is the form of its bounding surface or solid body, can be represented by different “geometric representations”. Different representations of *geometry* are presented in **Chapter 2**.

PURPOSE:

Geometry is needed to describe the spatial extent of the shape represented.

As presented in **Section 3.3**, this thesis does not deal with the behaviors (or functions) of an object in a 3D scene but focuses on the shape of a single object, i.e. what an object looks like in space. To explain what an object looks like can be done in two ways. One is by describing the spatial extent of the object, showing how much space is taken by the object and in which way. The other part includes more graphic information such as its color, texture and material. This subsection focuses on the first way. In conceptual design, the spatial extent of an object is at first roughly formed in the user’s mind, and then refined while specification evolves. However, to express the ideas that come out of the user’s mind is not that easy when the user is not keen on drafting and it is even more difficult when the process is computer-mediated. Combining existing resources, independently of their type, helps represent different parts of the object in order to get closer to the imagined shape in the user’s mind. The GSDM provides the capabilities to represent the information about what the object looks like in 3D space. A 2D data, such as an image, also represents how the object looks in “3D” space but from a specific point of view. In other words, how the object looks in 3D space is projected on a 2D plane.

PROPERTIES

- Heterogeneous

No assumptions are made on the type of data that can be used to represent the shape. This means that at this level, vector and raster 2D and 3D data are all addressed. Moreover, different geometric representations can be used to describe the same object’s *geometry*.

DATA STRUCTURE IN UML

Different kinds of input data correspond to different types of geometric representations, which need different data structures for their storage. In this work, data structures for existing geometric representations are not refined. Those available in the software environment adopted are used directly. Instead, a common structure for storing the additional data needed for their combination and simultaneous manipulation is specified.

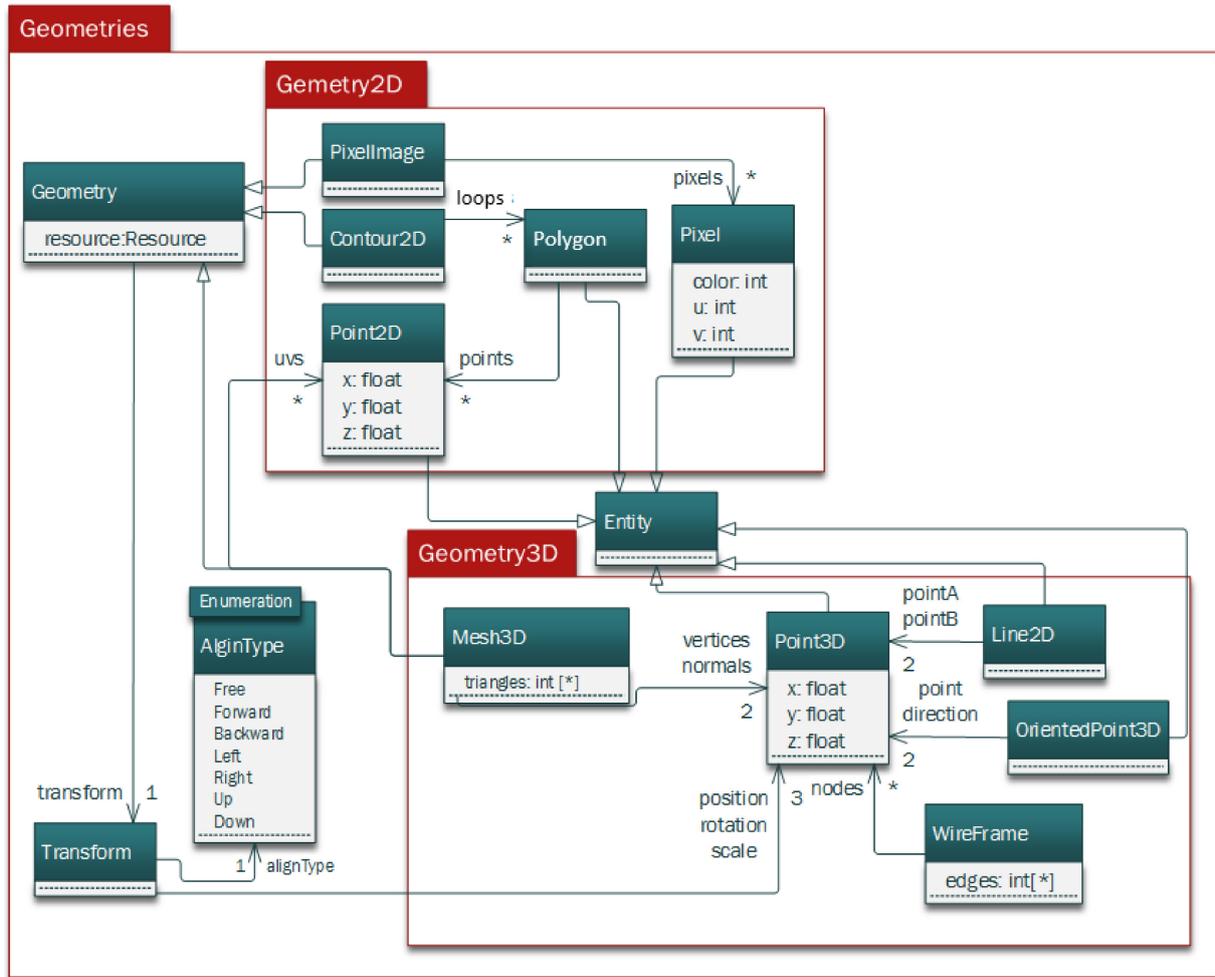


Figure 4.2 Data Structure of *geometry*

Figure 4.2 shows the data structure of *geometry* defined in UML that is specified in the package⁴⁸ (red rectangle) called “Geometries”. Three basic classes⁴⁹ (dark blue block) are defined in this package: “Entity”, “Geometry” and “Transform”. “Entity” is the basic class for low dimensional geometric primitives including point, line, oriented point, etc. The classes of those low dimensional geometric primitives are all inherited from the class “Entity”. “Geometry” is the basic class for all geometric representations associated with “Transform”. It expresses the location (including the position, the orientation and the scale) of a representation and the situation of alignment indicated by the attribute “AlignType”. If the alignment is not set to “Free” then the orientation of this transform will be fixed to a predefined direction. This is used to help users to easily move the corresponding geometric element. There are two sub packages included in “Geometries” named as “Geometry 2D” and “Geometry 3D” which define the data structures for different geometric representations and geometric primitives in two dimension and in three dimension. The attribute “resource” of “Geometry” indicates the file address of input data

⁴⁸ A package is a namespace used to group together elements that are semantically related and might change together.

⁴⁹ A class in UML describes is a classifier which describes a set of objects that share the same features, constraints and semantics (meaning).

that contains the information of the described geometric representation. The attribute “semantics” is an instance of the class “Semantics” telling the meaning of this geometric representation and will be explained later. There are other links between different classes in the GSDM, which for clarity reasons, will be illustrated when all the related classes are introduced.

A data structure called “Contour 2D” is defined in the package of “Geometry2D” to save the geometric representation of the outlines of the 2D image. The “Contour2D” has a list of “loops” for saving the interior and exterior boundary of the shape represented in an image (see the example of an image presented in **Figure 4.3.**)

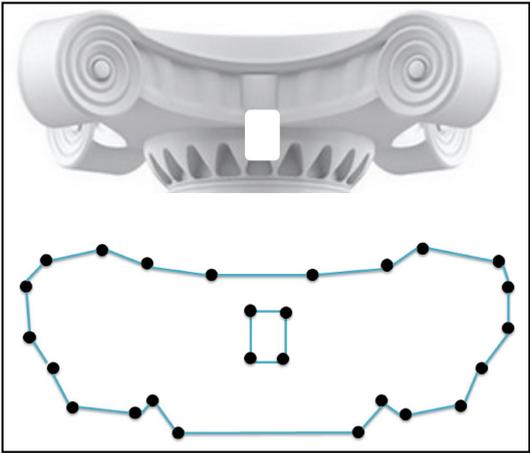


Figure 4.3 Example of a 2D image with its associated "Contour2D"

Text, as another kind of heterogeneous shape input, also has a geometric representation using a texture applied on a simple planar mesh. The manipulation of the 3D planar mesh represents the manipulation of the shape described by the text (**Figure 4.4.**)

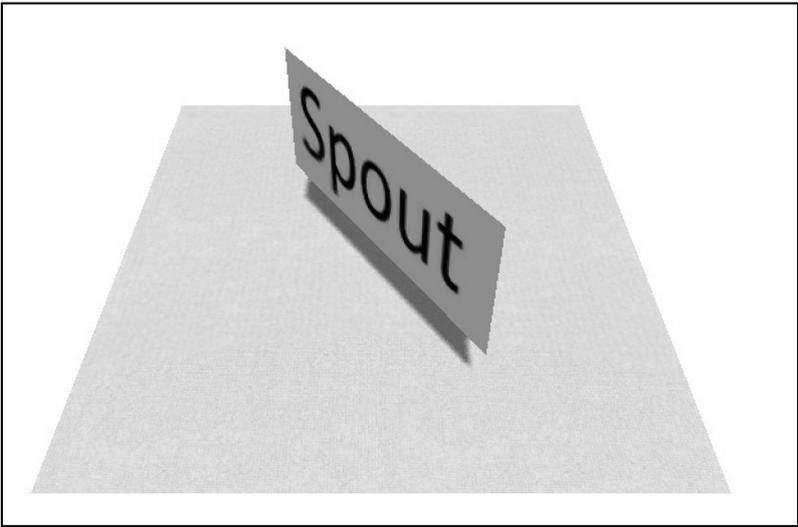


Figure 4.4 Text represented as Mesh3D

In the package “Geometry3D”, there is a class called “WireFrame” which is a 3D geometric representation described by nodes (“Point3D”) and edges (link between two nodes). This

representation can be used to represent the structure-based shape descriptors such as the skeleton, the Reeb graph, the medial axis, etc.

In this data structure, the package of “Geometry2D” and “Geometry3D” can be extended, if needed, to include other kinds of geometric representations.

4.2.2 STRUCTURE

DEFINITIONS

<p>Definition 4.2: Structure = Shape features and part-whole decomposition of a geometry represented by graph-based shape descriptors</p>
--

PURPOSE

- To help the user to position different parts composing the object.

Different kinds of structural information, such as the medial axis, the symmetry axis, the Reeb graph, the skeleton, etc., will help the user to align in position different parts represented by heterogeneous data. In traditional CAD modeling, the structure information of a CAD model such as the axis of a cylinder, the center point of a circle, etc., is used to perform the assembly of different parts. In the GSDM case, the structure information can also help support a similar assemblage such as in CAD modeling. The structure information of the CAD model used for assembly is the elements used to define the model itself (e.g. in the case of a sphere surface defined by a center point and a radius, the center point can be used for assembly). On the contrary, the structure discussed here is calculated from geometric representations of the shape to highlight the constituent shape features/parts and it is independent from the definition of this geometric representation.

- To support the future phases of Design.

As mentioned previously, the GSDM is used for the conceptual design, which is an early design phase focusing on the innovation and the creativity phases of the design process of a new object. Therefore, the GSDM is not a complete 3D model, but it has a multi-layered structure where some parts may just be a 2D model. However this low-dimensional model together with the associated structure pieces of information such as the symmetry axis, orientation axis, etc., can be used by reverse engineering approaches to build 3D models later in the next design phases.

PROPERTIES

- Heterogeneous

The *structure* at the *data level* of the GSDM can be represented by different shape descriptors obtained from either 2D or 3D data such as the medial axis, a Reeb graph, a segmentation, etc.

DATA STRUCTURE IN UML

A shape descriptor such as the Reeb graph, medial axis, etc. also has a geometric representation. In other words, the *structure* of an object can be represented differently.

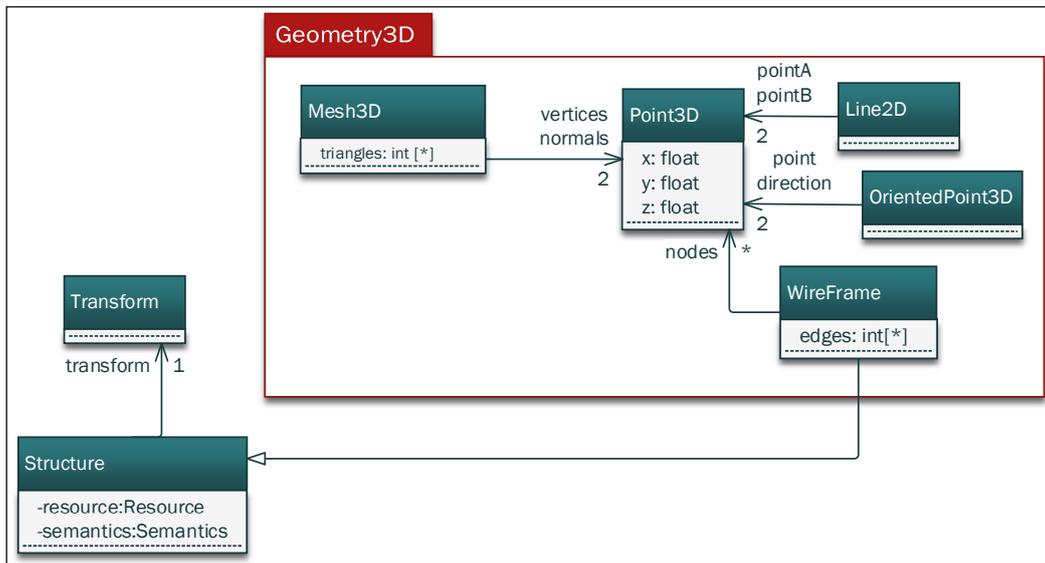


Figure 4.5 Data structure of *structure*

As presented in **Figure 4.5**, the super class “Structure” is associated with a “Transform” used to represent the location of this *structure*. Different structural representations are represented by a geometrical representation called “WireFrame” defined in the package “Geometry3D” and inherited from the class “Structure”. Examples of different structural representations are shown in **Figure 4.6**.

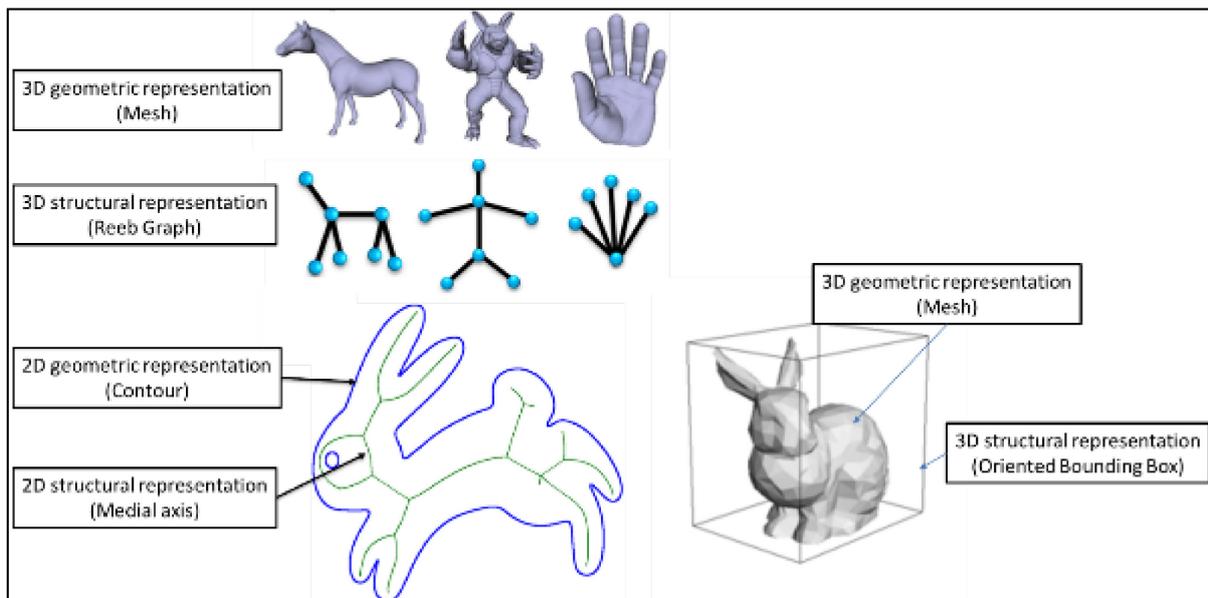


Figure 4.6 Example of geometric and structural representation

4.2.3 SEMANTICS

DEFINITION

Definition 4.3: Semantics = the purpose and meaning of an instance or an action in a specific context.

PURPOSE

- To store the name used in the conceptual design specified by user
- To specify the intention of an instance.
- To indicate the purpose of each action

With GSDM, different items of information are reorganized together following specified rules. 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 “put...together” can have a purpose of geometrically merging the two parts into one or it can have a purpose of assembling them together without merging their geometries.

- To be kept for further design phases or model retrieval

PROPERTIES

- Intrinsic or extrinsic

Intrinsic *semantics* expresses the meaning of something that can be obtained directly from it. For example, a surface can be considered as cylindrical if the distances from all the points on the surface to an axis are equal. The intrinsic *semantics* in the GSDM can be the “type” of geometry or structure. This information can be obtained by certain calculations from the original data of the referred item and this information can thus be considered as intrinsic.

Extrinsic *semantics* refers to additional information independent of the original data under a specific context. For example, the mesh of a cylinder. Some additional information such as the color, the material, the name, the function, the role in the overall object (e.g. a chair leg) can be added to this mesh depending on different contexts. This information is not contained in the mesh and therefore has to be attached/added to it.

DATA STRUCTURE IN UML

As one item or an action can have different types of *semantics* depending on the context, the class of *semantics* is defined as a super class, different types of *semantics* are inherited from this class and located in the packaged called “Semantic Data” as presented in **Figure 4.7**.

The attribute “type” of the class “Semantics” defines which *semantic* instance to be used. The types of *semantics* are listed in an enumeration⁵⁰ named “Semantic Type”. The details of

⁵⁰ An enumeration is a data type whose values are enumerated in the model as user-defined enumeration literals.

each type of semantics are going to be presented when each related notion is introduced in the following sections.

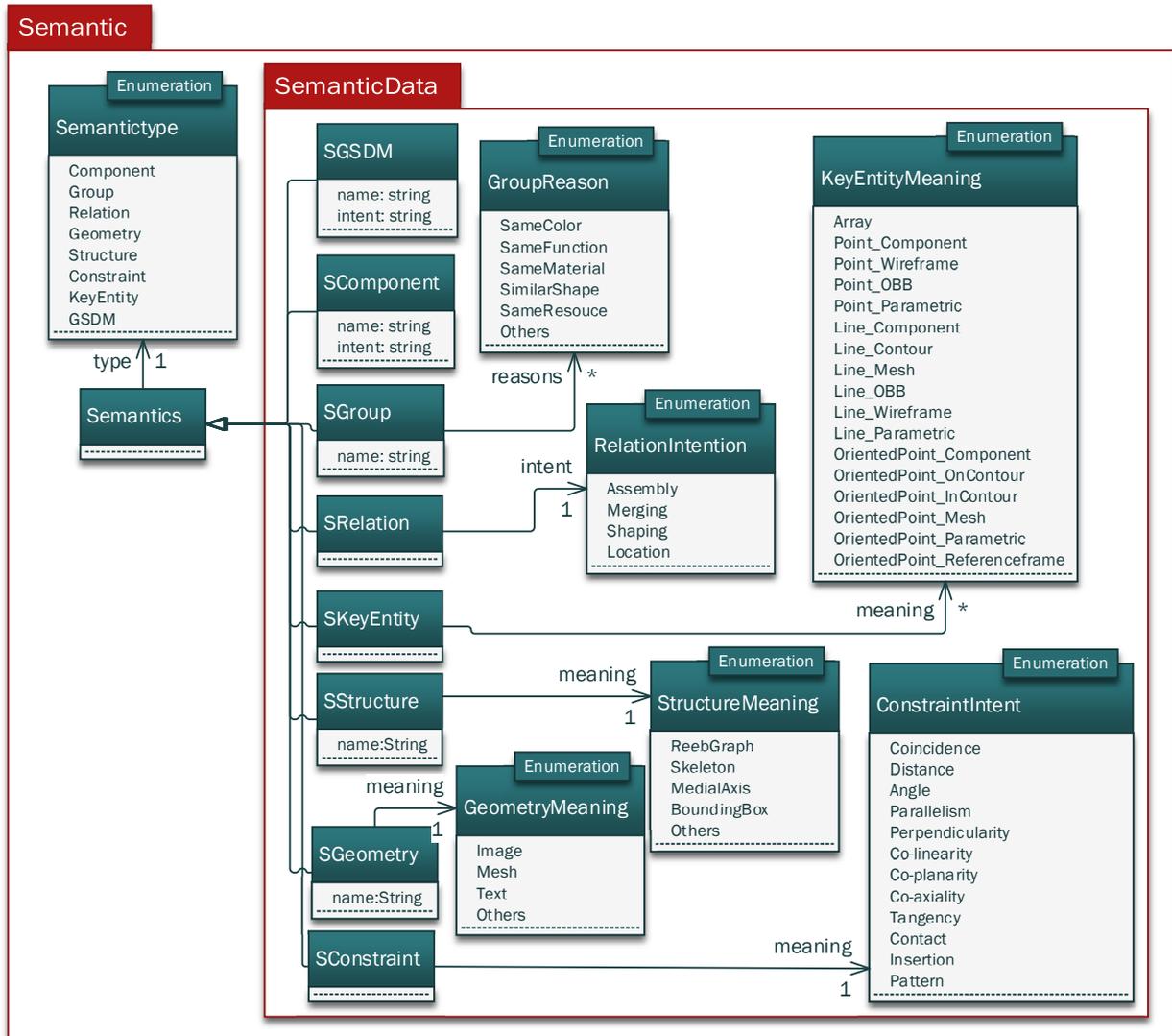


Figure 4.7 Data structure of *Semantics*

To conclude, in this section, the *data level* of the GSDM has been presented. It contains the definition of *geometry*, *structure* and *semantics*. These three notions are used to handle the information related to a shape obtained from heterogeneous data such as a segmented image with a Reeb graph, a segmented mesh with medial axis, etc.

4.3 CONCEPTUAL LEVEL (COMPONENT, GROUP AND RELATION)

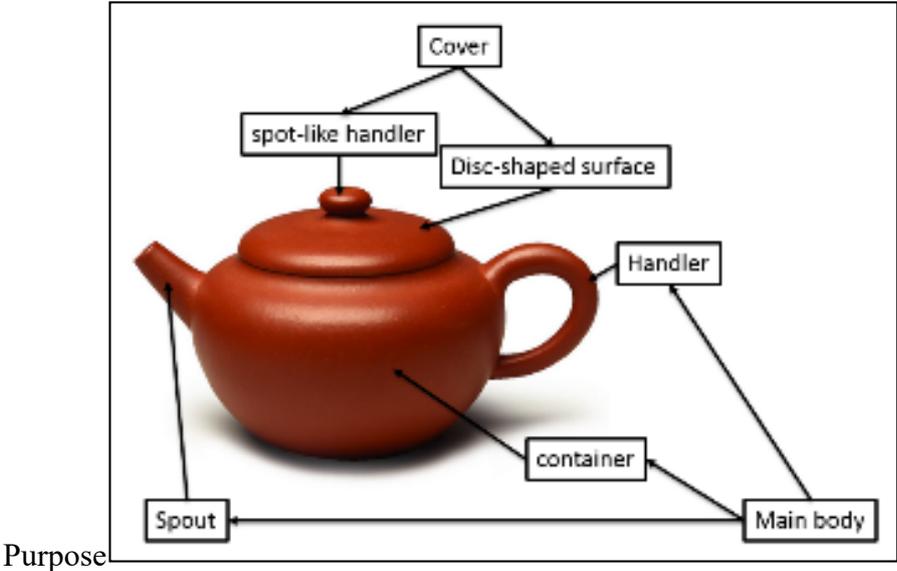
The *data level* presented in the previous section contains the basic information of a shape needed for its visualization and manipulation. However, in our system it is not directly operated by the users who focus on the overall conceptual specification of the object to be designed and not on fine-tuning its final shape. The user focuses more on the part-whole decomposition of the object to which behaviors can be directly associated. This justifies the need of a conceptual

level manipulated directly by the non-expert user to specify the decompositions into parts and the relationships between them.

4.3.1 COMPONENT

DEFINITION

Definition 4.4: Component = In a design context, a *component* is one part of an object, which re-organizes some *geometric* or *structural* representations together so as to represent a part with a basic *semantic* meaning.



Context / description	components
A teapot with two parts, the main body and a cover.	Main body, Cover
The main body is composed of a spout, a container and a handle	Spout, Container, Handle, Cover
The cover has a disc-shaped surface and a spot-like handler	Spout, Container, Handler, Disc-shaped surface, Spot-like handle

Figure 4.8 Examples of object decomposition in *components*

PURPOSE

- To define an indivisible part.

Here, “indivisible” means that the user will not decompose this part any further. The user can define a part according to its functions or any other purposes. A part can also be split into

more parts. At one moment, as the decomposition of a part offers enough information or further decomposition is meaningless, the user will not decompose it anymore. At this stage, this part can be considered as an inseparable part. For the example presented in **Figure 4.8**, each time an element composing the shape of the object is further specified, at least two new parts are added in the description (such as the spout, the container and the handle detached from the main container). The number of parts shows us both the complexity of the object and how precise a user wants to be.

- To organize data with respect to the object's functionality or re-usable shape constituents, independently of their geometric and structural representations.

One objective for using the GSDM is to use heterogeneous data and to give the possibility to represent a part using several geometric representations with different structural descriptions. However, no matter what kind of geometric or structural representation is used, in the part-whole relationship, it is still considered as the same part.

The above two points show the main reasons for introducing a new notion of *component*.

PROPERTIES

- Representation-independent.

The idea of what a specific *component* looks like can come from different ideas obtained from different heterogeneous data. In this definition, the number of geometric or structural representations of a *component* is not limited. This is to say that a *component* can have different geometric or structural representations (**Figure 4.9**).

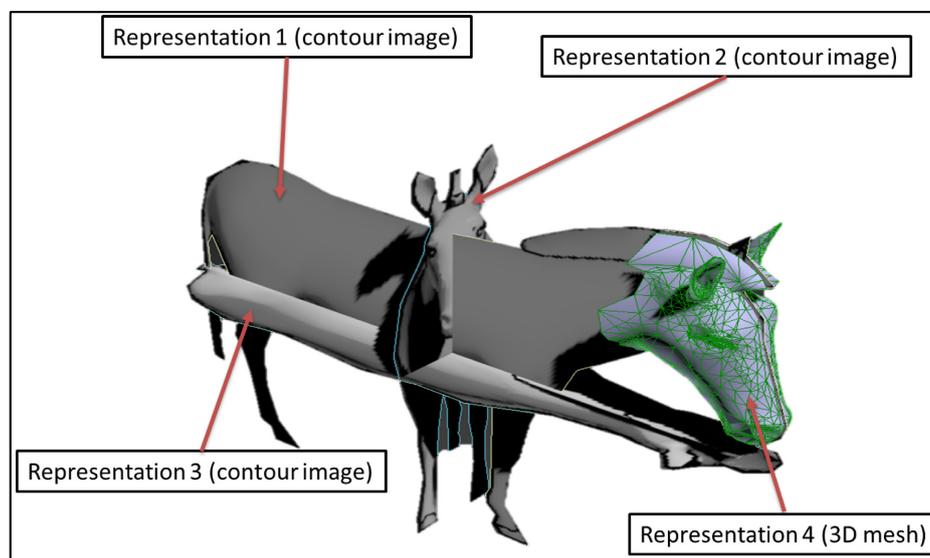


Figure 4.9 Components with multi-geometric representations

- Indivisible

There are no *components* inside a *component*. A *Component* is the basic unit in the definition of the GSDM in the sense that it cannot be further decomposed.

- Context-oriented

An object can be decomposed differently depending on the context as in the example presented in **Figure 4.8**. A further detailed decomposition may have more *components*. Therefore, which parts of a shape should be considered as *components* is to be decided by the user in a specific design context.

- Multi-layered

Every *component* could contain three layers of information: *geometry*, *structure* and *semantics*.

DATA STRUCTURE IN UML

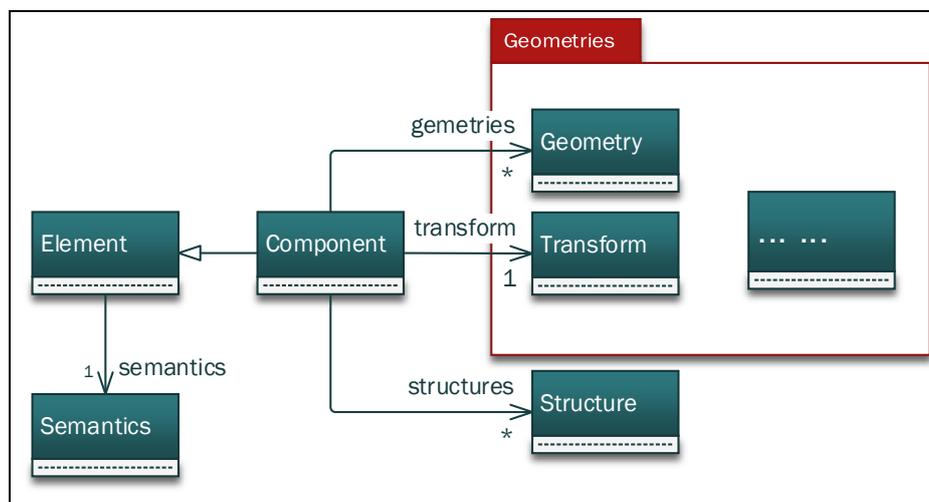


Figure 4.10 Data structure of *Component*

Figure 4.10 presents the data structure of *component*, which shows that *component* is pointing to a list of *geometry* and a list of *structure*. *Semantics* is also related to *components*. The list of *geometries* and the list of *structures* offer the possibility for a *component* to have multiple representations and multiple layered structures. The *semantics* of *component* contains the attributes of:

- name: the specific name given by the user
- intent: the meaning or intention of this component, such as “support”, “container”, etc.

In **Figure 4.11**, there is an example of two *components*. The right component has a multi-representations of a bottle, where the *geometries* of the *component* contain two 2D images (from two perpendicular views) represented by Contour2D.

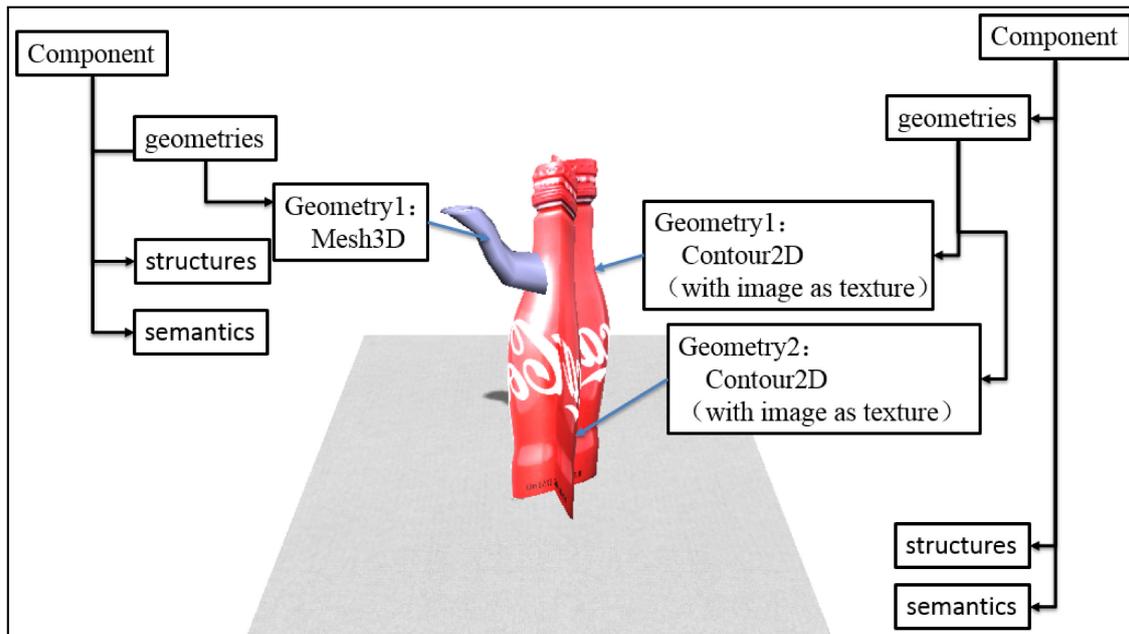


Figure 4.11 Data structure of *Component* with multi-representations

4.3.2 GROUP

DEFINITION

Definition 4.5: Group = a logical operation, which associates a specific meaning or behavior to several *components*.

Definition 4.6: Element = the general term for *component* and *group*

PURPOSE

- To associate some *components* together with specific meanings or attributes.

Sometimes, a user wants to gather some *components* together so that they can be selected, modified or searched as one. For example, the user may want to change the color of all the *components* from red to blue, or to treat as main body the spout, the container and the handle of the teapot in **Figure 4.8**.

- To associate some components to a specific behavior

In the example of **Figure 4.12**, the three arms (Arm 1, Arm 2 and Arm 3) of the robot are considered as three different *components*. However, the movement of “Arm 1” will affect the position of “Arm 2” and “Arm 3”. In other words, the three arms should be moved together when Arm 1 is moved. In this situation, they have the same behavior and should be treated all together.

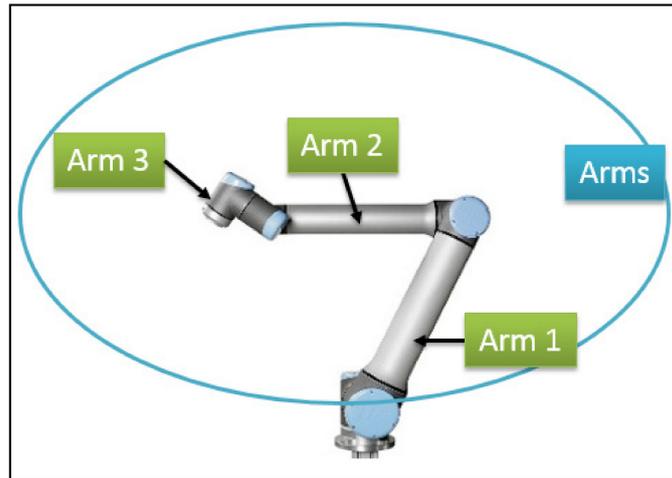


Figure 4.12 A robot with three arms

PROPERTIES

- Cardinality

A *Group* is constituted by at least two *elements*. A *component* or a *group* can be an *element* of other *groups*.

- Semantic inheritance

All the elements in a *group* should have a specified semantic meaning for indicating the purpose of being a *group*. This specific semantics tells us why different *elements* should be considered as one. For example, they have a similar function, or they have the same color.

- Non-exclusive inclusion

Non-exclusive inclusion means that a *group* can be an *element* of another *group* and an element can belong to several groups. An example of *group* is presented in **Figure 4.13**, which shows a corner of an office room. All the books on the desk can be considered as 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 are all on the desk. Another *group* called “Furniture” refers to all 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 all made by plastic.

In this example, it can be noticed that “mouse” is **shared** by four *groups*: “PC”, “plastic”, “on the table” and “Office room”. This *element* can be directly included in those *groups* as an *element* such as the “mouse” in “PC” or the “mouse” in “Plastic”. This *element* can also be an *element* of an included *group* such as the “mouse” of the *group* “On the table”, where “mouse” is an *element* of the included *group* “PC” including in the *group* “On the table”. The “mouse” is also shared by the *group* “Office room”. To locate the “mouse” of the *group* “Office room”, one has to go through minimum 1 *group* (“Office room” -> “plastic” -> ”mouse”), and maximum 2 *groups* (“Office room” -> “On the table” -> ”PC” -> ”mouse”). The number of *groups* to go through so as to locate an *element* is defined as the **depth** of the *element* in this *group*.

With different passages, there might be a minimum *depth* and a maximum depth. *Depth* shows the complexity of a *group*. For the user, the *depth* might not be very useful. However, it could be used by the algorithms for the manipulation of *groups*.

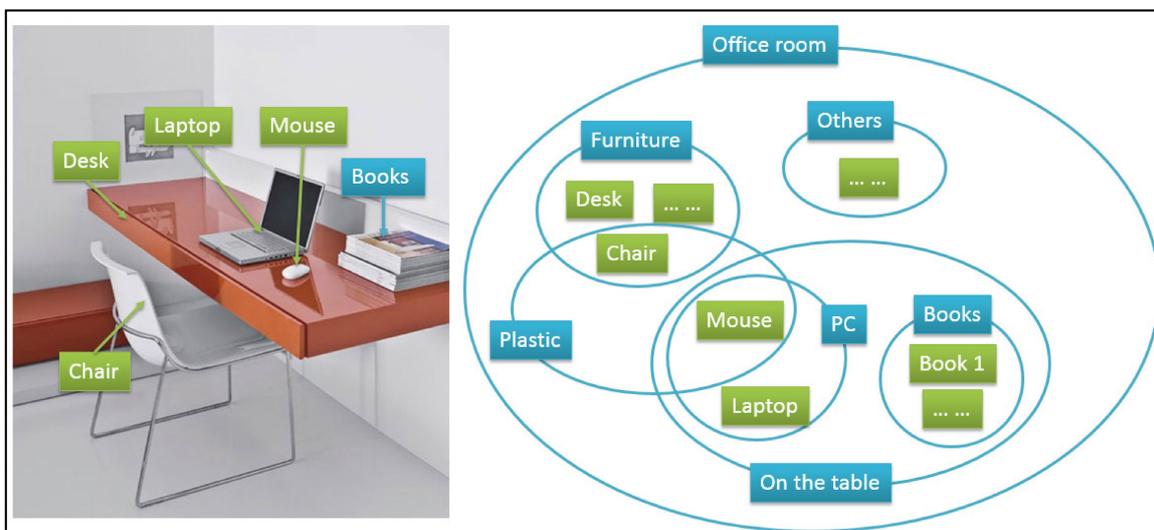


Figure 4.13 Example of groups⁵¹. (Blue circles for *groups* and green block for *components*)

DATA STRUCTURE IN UML

The data structure of a *group* is presented in **Figure 4.14** where the class “*Group*” points to a list of *components* and a list of *groups*. The class “*Component*” and “*Group*” are inherited from the super class “*Element*” which points to “*Semantics*”.

Then the *semantics* of a *group* contains:

- name: the user specified name of the *group*
- reasons: the reasons why *elements* are grouped together. Six types of grouping reasons have been proposed in the enumeration “*Group Reason*” including the similarity of shape, color, material, function and source.

⁵¹ Picture of the office room is downloaded from : <http://www.decosee.com/2014/04/07/modern-office-room-minimalist-idea-23394.html>

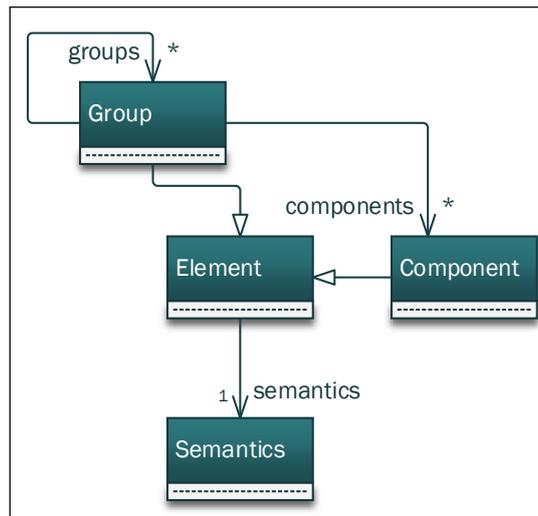


Figure 4.14 Data structure of group

4.3.3 RELATION

DEFINITION

Definition 4.7: Relation = the way two elements are connected together according to a specific meaning

PURPOSE

- To describe if and how *elements* are related from a top down perspective.

The links between different *elements* may express very complex relations and/or operations that, to be achieved, need complex and precise information. For example, to accomplish the relation expressing the geometric merge of two *components* together, the related location of the two *components* needs to be specified as well as their geometric representation and the parameters related to the algorithm used for merging. For a non-expert in Computer Graphics, describing these complex links can be very difficult. One possibility is to describe them top down, i.e. from the purpose to the specification of the *geometric* or *structural* elements and associated rules. Additionally, in the conceptual design phase, the purpose of this link is just the indication of the type of relation/operation, independently of the representations or of the associated rules. In the example presented in **Figure 4.15**, the user wants to merge the spout and the container of a teapot together as these two *components* are connected because water can flow from the container into the spout. Although the two components have different representations, the link of “merging” exists independently of their underlying representations. At the top level, different types of links indicating the purpose can be specified, and at the low level, the minimal set of details necessary to provide a visual feedback for understanding the desired outcome need to be stated. Such a minimal set normally includes the related location of the two *components*. *Relation* focuses on the top level. The detailed definition of each relation type presented in the

column “Relation” is introduced in the following subsection.

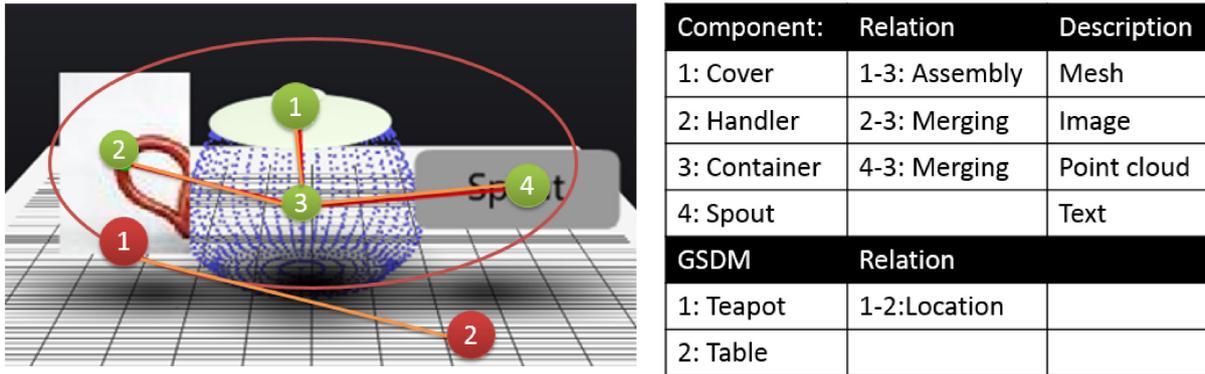


Figure 4.15 Examples of *relations*

INTENT OF RELATION

- Merging

This *relation* indicates that the two elements have to be geometrically operated to obtain a unique geometric element. It is a very common relation, which corresponds to the Boolean union operation on geometric models. For example, in **Figure 4.15**, the teapot is composed of four different *components*, each of them with a different geometrical representation. In real life, if the container and the spout are not geometrically connected together, then the water cannot come out of the spout, a merging relation between them can be required so that the spout and the container create a unique continuous volume. In the example of **Figure 4.15**, the container is represented by a point cloud, while the spout is represented by a text. On 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 it to allow its complete shape definition in the detailed design phase, without being limited and slowed down by unnecessary modeling details and operations difficult for non-expert users. As already stated, the purpose of the GSDM is to provide the representation of how an object should be created by combining sub-parts, possibly not completely defined. The relations aim at specifying the links between them. The real operation of “merging” does not take place at this stage but it can be obtained by processing the GSDM once the geometric description of each constituting *component* has been completely specified and harmonized (i.e. compatible geometric representations on which Boolean operations can be applied).

- Assembly

This is the same notion as in CAD systems. Different *elements* are connected together without fusing them into the same *geometry* but simply by linking them with different joints. In the example of **Figure 4.15**, the cover and the container are assembled together, but they could still be treated separately and change their relative positions.

- Shaping

This relation is used to indicate the intention to modify the shape of an *element*, i.e. to

reshape it. It is not simply merging the overlapping areas of two *elements* by cutting the useless areas as the merging *relation*, 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 **Figure 4.16** . A chair plus an egg becomes an egg-like chair. The algorithms used to process the *elements* and shape the resulting object have not been considered in this PhD thesis.

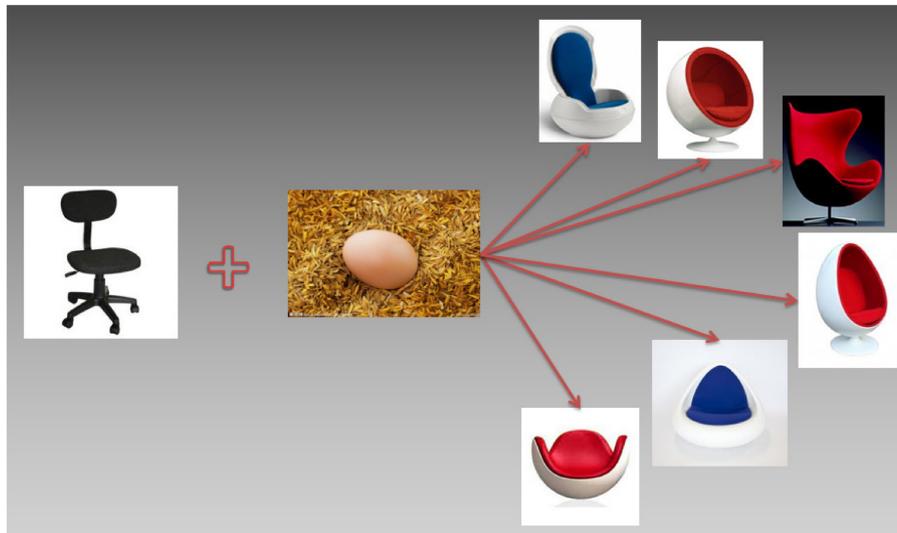


Figure 4.16 Example of the intent expressed by a relation of “shaping”

- Location

For some situations, two objects are not assembled, or merged or shaped together, but there still exist some relations in just positioning one with respect to the other. For example, a football on the floor. The floor and the ball are not assembled together or merged together. They also keep their own shapes. The relation used to only position an object with reference to another is defined as “Location”.

PROPERTIES

- Cardinality

A *relation* is only built between two *elements*. Actually, a *relation* can be built between more than two elements when using the notion of *group*. For example, if four legs of a table need to be fitted to a desktop, a *group* can be created including the four legs called “support” and then assemble the *group* “support” to the desktop. Alternatively, a *relation* between more than two elements can be obtained as a series of pairwise *relations*. For example, for the mechanical robot presented in **Figure 4.12**, if an assembly among all the arms needs to be built, then it could be separated into an assembly *relation* between Arm1 and Arm2 and another between Arm2 and Arm3.

- Independency from the representation

As already stated, being the relation aimed at specifying the purpose and rules of the link between *elements*, it is independent from the actual representation of each *element*.

- Inheritance

If there is a relation between group A and element 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). For example, a *group* of four legs is 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* legs. This inheritance property doesn't have to be specified in the data structure, it is a matter of logic.

- Uniqueness

There is only one kind of relation between two elements, including the inherited relation.

DATA STRUCTURE IN UML

The data structure of *relation* is presented in **Figure 4.17**.

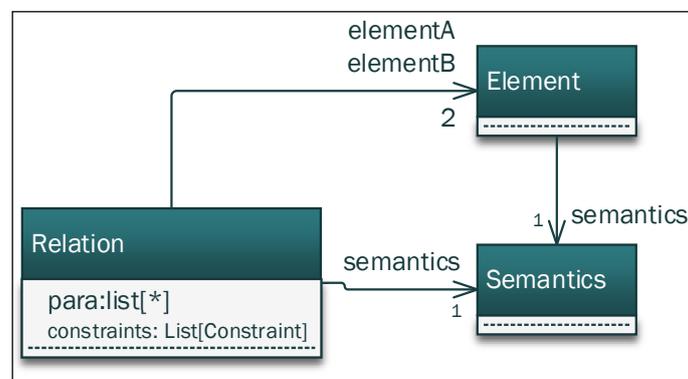


Figure 4.17 Data structure of *relation*

Relation points to two *elements* (“element A” and “element B”). The *semantics* of relation is mainly about the functional intent of this relation including *Assembly*, *Location*, *Merging* and *Shaping*. Depending on different intents, in a further design phase, operations will be applied to these two *elements* (e.g. Boolean operation to merge meshes, etc.) with parameters defined in the attribute “para”. One *relation* also contains a list of *constraints* to build the link between the *conceptual level* and the *data level*. The definition of *constraint* will be presented later.

To summarize, in this section, the three constituents of the *conceptual level* of the GSDM: *component*, *group* and *relation*, have been presented. They form a high-level description model for describing shapes that users can directly manipulate. Although it is not precise enough, a quick overview of an object can be described with the *conceptual level*. As far as *Relation* is concerned, this section has presented the types considered, even though their complete specification requires a connection between the *conceptual level* of the GSDM and the *data level* of the GSDM. This connection between the *conceptual* and *data levels* is strongly linked to the so-called “intermediate level” that is detailed in the following section.

4.4 INTERMEDIATE LEVEL (KEY ENTITY AND CONSTRAINT)

4.4.1 BASICS

PURPOSE OF SCALING

Scaling is normally not needed in the typical CAD design approach, where different parts could be assembled together by applying constraints without changing their size as they are generally designed in the same measurement system and with the same purpose. However, in GSDM, the *components* come from heterogeneous input data, which are generated in different measuring systems. Their sizes are not comparable. One solution is to give each *component* an initial size value before building relations between them. They can be considered as rigid *components* as in the CAD system. As in a conceptual design process, the *semantic* meaning of combining different *components* together is more important than the exact specification of the geometry. For example **Figure 4.18** shows some “crazy” chairs. The left-bottom chair is composed of a coffee cup- like back, a usual seat and four legs. The most interesting part in this design is the combination of the coffee cup-like back with the seat. In this case, the initial value of the size of the back cannot be specified, as it is difficult to determine which size will be appropriate to connect the back with the kind of seat the user wishes. Actually, when using the size of a real cup and the size of a real seat, the back will never satisfy this design intent. On the other hand, some *components* can be reused with a different size. Therefore, the size should be decided during the process of combining them together, not in the opposite way, i.e. before connecting them. In this sense, the *components* should be scalable.



Figure 4.18 Examples of unconventional chairs

LOCATION OF COMPONENTS

The GSDM describes shapes in 3D space. Although some *components* may be represented by 2D data, such as an image, when combined with others, these 2D data are located in a 3D space. To indicate how different *components* are located in relation to each other in a 3D space, the notion of “location of *component*” needs to be defined. A 3D space can be represented by a reference frame, which refers to a Euclidean coordinate system. In GSDM There are three types of 3D space existing in the GSDM represented by three types of reference frame: the reference frame of each geometric or structural representation in a *component* space, the reference frame of the *component* in the global space and the global reference frame. “The location of component” in a 3D space A, can be represented by the “location of the reference frame” in the 3D space A. As in the example presented in **Figure 4.19**, the global 3D space is represented by a global reference frame with an origin point “O” and three orthonormal axes “x”, “y” and “z”. The location of a component made of two geometric representations (two contour images of a coca cola bottle) is represented by a reference frame with an origin point “C”. The two reference frames of the two representations are located “R1” and “R2”. Each reference frame is structured in a class called “Transform” in the package of “Geometries” as presented in **Figure 4.2**.

To specify the location of a *component*, three types of values, the position, rotation and scale of the *component*’s reference frame in global space need to be specified. A triple (P, R, S) is used to represent the location of a *component*, where **P** is the **position** of the reference frame in a global space. **R** is the **orientation** of the *component* in a global space and **S** is the **scale** factor of the *element*.

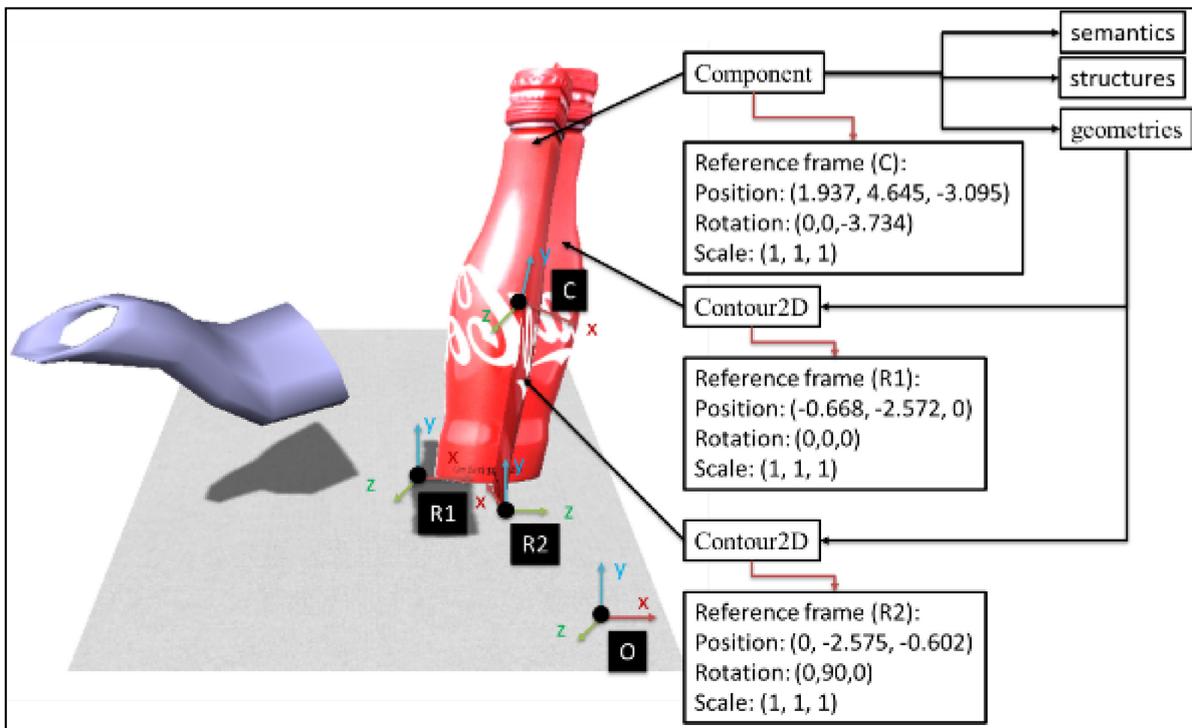


Figure 4.19 Location of a component

TRANSFORM BETWEEN LOCAL AND GLOBAL SPACES

However, positioning one element by acting on the local reference frame can be very difficult for non-expert users. A more natural and meaningful way is to link the related location to another element. Directly building links between reference frames seems meaningless. A better way is to specify their reciprocal location, depending on some characteristics of the geometric or structural representation of the *component*. For example, two *components* are assembled together by inserting a *component* into another. This assembly will align two specific axes from the two geometric representations of the *components*. The specification of this “alignment” action and the specification of the “axis” need to be defined. To be able to limit the related location of the two *components* is important for an assembly as well as for other types of *relation*.

However this “axis” is defined by a point and a direction, which is located in a local space of the representation. In fact all the vectors (points and directions) used to define the geometric or structural representation of a component, such as the position of a mesh vertex, the normal of a mesh vertex, the position of a node of a Reeb graph, etc. are defined in the local space of the representation (e.g. R1 and R2 in **Figure 4.19**). To “align” an applied vector in one local representation space to an applied vector in another local representation space, these local entities (point or direction) need first to be transformed in the same space. In GSDM, it is decided to transform all local entities (point or direction) in the global space before getting them to interact. A mathematical matrix can be used to calculate this transformation of points or directions between different spaces.

If space A is inside space B, then the reference frame of A is defined as:

$$\text{Reference frame of space A: } \text{Rf}_B^A = (\mathbf{P}_A, \mathbf{R}_A, \mathbf{S}_A) = \left(\begin{bmatrix} x_A \\ y_A \\ z_A \\ 1 \end{bmatrix}, \begin{bmatrix} \alpha_A \\ \beta_A \\ \gamma_A \\ 1 \end{bmatrix}, \begin{bmatrix} a_A \\ b_A \\ c_A \\ 1 \end{bmatrix} \right),$$

Where:

$$\mathbf{P}_A = \begin{bmatrix} x_A \\ y_A \\ z_A \\ 1 \end{bmatrix} \text{ is the origin of } \text{Rf}_B^A, \text{ represented by a column vector, } x_A, y_A, z_A \in \mathbb{R}$$

$$\mathbf{R}_A = \begin{bmatrix} \alpha_A \\ \beta_A \\ \gamma_A \\ 1 \end{bmatrix} \text{ is the rotation of } \text{Rf}_B^A, \text{ represented by a column vector, } \alpha_A, \beta_A, \gamma_A \in \mathbb{R}$$

$$\mathbf{S}_A = \begin{bmatrix} a_A \\ b_A \\ c_A \\ 1 \end{bmatrix} \text{ is the scale factor of } \text{Rf}_B^A, \text{ represented by a column vector, } a_A, b_A, c_A \in \mathbb{R}$$

\forall point $\mathbf{p}_A = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$, $x, y, z \in \mathbb{R}$ and direction $\mathbf{n}_A = \begin{bmatrix} i \\ j \\ k \\ 1 \end{bmatrix}$, $a, b, c \in \mathbb{R}$ in space A, the corre-

sponding point $\mathbf{p}'_A = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$ of p_A in space B and the corresponding direction $\mathbf{n}'_A = \begin{bmatrix} i' \\ j' \\ k' \\ 1 \end{bmatrix}$ of

\mathbf{n}_A in space B is defined as [105]:

$\mathbf{p}'_A = \mathbf{M}_B^A \mathbf{p}_A$, the multiplication of matrix \mathbf{M}_B^A and \mathbf{p}_A .

$\mathbf{n}'_A = [\mathbf{M}_B^A]^{-1T} \mathbf{n}_A$, the multiplication of matrix $[\mathbf{M}_B^A]^{-1T}$ and \mathbf{n}_A .

where \mathbf{M}_B^A is the transformation matrix from space A to space B defined as:

$$\mathbf{M}_B^A = f_M(\text{Rf}_B^A) = \begin{bmatrix} a_{Ax} \cdot \mathbf{c}[\beta_A] \cdot \mathbf{c}[\gamma_A] & -b_A \cdot \mathbf{c}[\beta_A] \cdot \mathbf{s}[\gamma_A] & c_A \cdot \mathbf{s}[\beta_A] & x_A \\ a_A(\mathbf{c}[\gamma_A] \cdot \mathbf{s}[\alpha_A] \cdot \mathbf{s}[\beta_A] + \mathbf{c}[\alpha_A] \cdot \mathbf{s}[\gamma_A]) & b_A(\mathbf{c}[\alpha_A] \cdot \mathbf{c}[\gamma_A] - \mathbf{s}[\alpha_A] \cdot \mathbf{s}[\beta_A] \cdot \mathbf{s}[\gamma_A]) & -c_A \cdot \mathbf{c}[\beta_A] \cdot \mathbf{s}[\alpha_A] & y_A \\ a_A(-\mathbf{c}[\alpha_A] \cdot \mathbf{c}[\gamma_A] \cdot \mathbf{s}[\beta_A] + \mathbf{s}[\alpha_A] \cdot \mathbf{s}[\gamma_A]) & b_A(\mathbf{c}[\gamma_A] \cdot \mathbf{s}[\alpha_A] + \mathbf{c}[\alpha_A] \cdot \mathbf{s}[\beta_A] \cdot \mathbf{s}[\gamma_A]) & c_A \cdot \mathbf{c}[\alpha_A] \cdot \mathbf{c}[\beta_A] & z_A \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where, $\mathbf{c}[x]$ refers to $\cos(x)$, $\mathbf{s}[x]$ refers to $\sin(x)$.

To transform a point or a direction in a geometric or structural representation to a global space requires first to transform it from representation space to the *component* space then from the *component* space to the global space. In this case, the transformation matrix from local to global can be defined as:

$$\mathbf{M}_{\text{Global}}^{\text{Local}} = f_M(\text{Rf}_{\text{Global}}^{\text{Component}}) f_M(\text{Rf}_{\text{Component}}^{\text{Representation}})$$

4.4.2 KEY ENTITY

DEFINITION

Definition 4.8: Key Entity = Geometric primitive (point, line or an oriented point) associated with a geometric or structural representation of a *component* or located in a component's local 3D space (represented by a local reference frame).

PURPOSE

As mentioned before, instead of describing the related location between reference frames, it is more meaningful to specify the related locations of some geometric elements of the geometric or structural representation of a *component*. Key entities are used to represent these meaningful key geometric elements.

TYPE OF KEY ENTITY

In the GSDM, two types of key entity are proposed: **Geometric Key Entity** and **Parametric Key Entity**.

Definition 4.9: Geometric Key Entity = Geometric primitive (point, line or an oriented point) situated in a *component*'s local 3D space (represented by a local reference frame).

Definition 4.10: Parametric Key Entity = Geometric primitive (point, line or an oriented point) defined by parameters to associate with the geometric or structural representation of a *component* or other key entities.

A *geometric key entity* of a *component* is not modified when the geometric or the structural representation changes. For example, a geometric key point defined for a *component* represented by a mesh corresponds to a position in the local reference frame of the mesh. Thus, if the mesh is scaled, this point will not change. Geometric key entities are useful when a *component* has no geometric or structural representations.

A *parametric key entity* can be represented by a point, a line or an oriented point (it can be used to represent a plane). These *key entities* can be associated **directly** to the geometric or structural representation of a *component* such as a vertex of a mesh with its normal. However, a *parametric key entity* can also be created by building rules between other key entities. For example, with a line defined by two points, these two points can be *geometric key entities* or *parametric key entities*. The newly created *key entity* is not directly associated with the geometric or structural representation, it is then called **indirect parametric key entity**.

A *Key entity* is represented by four types of geometric primitives: a point, a line, an oriented point and a combination of them (indicated as an array). All the proposed key entities are listed in **Table 4.1**.

Classification				Point	Line	Oriented Point	Array
Geometric key entity	Independent			E_P	E_L	E_F	\
Parametric key entity	Indirect			E_{PP}	E_{LP}	E_{FP}	E_A
	Direct	Geometric representation	Contou2D	\	E_{LC}	E_{FoC}, E_{FiC}	\
			Mesh3D	\	E_{LM}	E_{PM}	\
		Structural representation	Wireframe	E_{PW}	E_{LW}	\	\

Table 4.1 Classification of the proposed Key Entities

On **Table 4.1**, it can be noticed that not all the geometric primitives (a point, a line, an oriented point, or an array) are associated to the geometric or structural representation of a *component*. This is because some of them are meaningless or they do not exist. For example, an oriented point in a wireframe is not meaningful. The edge of the wireframe can be used to

indicate a path without specifying any additional information of “orientation”. On the contrary, a point on a mesh should be associated with an orientation, which is more useful than for just a single point, indicating the normal of the surface where this point is located.

MATHEMATICAL SPECIFICATION

In this manuscript, the letter “E” is used to represent a *key entity*. Each *key entity* is defined by a set of parameters between two breaks as below:

$$E(\text{para}_1, \text{para}_2, \dots, \text{para}_n), \text{para}_{i=0\dots n} \text{ is a parameter, } n \in \mathbb{N}^+$$

From those parameters, the four types of geometric primitives can be inferred.

Geometric primitives (point, line and oriented point)

In the following the notations used for the key entity specification.

Let $\mathbf{V} := \left\{ \begin{bmatrix} a \\ b \\ c \\ 1 \end{bmatrix}, a, b, c, d \in \mathbb{R} \right\}$ be the set of column vectors with 4 rows. 3D points and

directions used in this manuscript are represented by elements in \mathbf{V} .

$\mathbf{p} := \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \in \mathbf{V}$, is a 3D point stored by \mathbf{V} .

$l := (\mathbf{p}_{1l}, \mathbf{p}_{2l})$, is a line passing through \mathbf{p}_{1l} and \mathbf{p}_{2l} , where $\mathbf{p}_{1l}, \mathbf{p}_{2l} \in \mathbf{V}$

$f := (\mathbf{p}_f, \mathbf{n}_f)$ is an oriented point, where $\mathbf{p}_f, \mathbf{n}_f \in \mathbf{V}$

- For a point key entity: \mathbf{p}_E represents the point of key entity E
- For a line key entity : $l_E = (\mathbf{p}_{1E}, \mathbf{p}_{2E})$ represents the line of key entity E, where

\mathbf{p}_{1E} represents the first point used to define the line l_E .

\mathbf{p}_{2E} represents the second point used to define the line l_E .

- For an oriented point key entity:
- $f_E = (\mathbf{p}_E, \mathbf{n}_E)$ represents the oriented point of key entity E, where

\mathbf{p}_E represents the point used to define f_E

\mathbf{n}_E represents the direction used to define f_E

Transformation matrix

However, these primitives are located in a local reference space or on a local geometric or structural representation. As mentioned at the end of previous section (**Section 4.4.1**), these local values need to be transformed in a same measuring space (the global space) so as to be equally compared. Thus we need the transformation matrix.

- For *geometric key entities*:

The first parameter is the *component* (Com), where this *key entity* is located:

$$E(\text{Com}, \text{para}_2, \dots, \text{para}_n)$$

As introduced in Section 4.3.1, each *component* is associated with a reference frame defined in its data structure (named “Transform”, see **Figure 4.10**). It is represented here as:

$$\text{Rf}_{\text{Com}}$$

Thus, the transformation matrix of the *geometric key entity* is defined using the function $f_{\mathbf{M}}$ introduced in **Section 4.4.1**:

$$\mathbf{M}_E = f_{\mathbf{M}}(\text{Rf}_{\text{Com}})$$

- For *direct parametric key entities*:

The *direct parametric key entities* are always associated with a geometric or structural representation, which is saved in the first parameter (Rep):

$$E(\text{Com}, \text{Rep}, \text{para}_3, \dots, \text{para}_n)$$

This geometric or structural representation is also associated with a reference frame, which is located in the *component*'s space as their data structure defined in **Figure 4.10**:

$$\text{Rf}_{\text{com}}^{\text{Rep}}$$

The reference frame of this *component*:

$$\text{Rf}_{\text{com}}$$

Thus, the transformation matrix of *direct parametric key entity* is used to transform the key entity from the representation space to component space then to the global space, which can be calculated as:

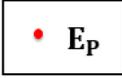
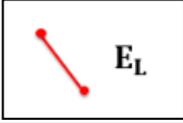
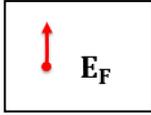
$$\mathbf{M}_E = f_{\mathbf{M}}(\text{Rf}_{\text{Com}})f_{\mathbf{M}}(\text{Rf}_{\text{Com}}^{\text{Rep}})$$

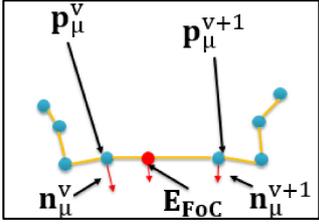
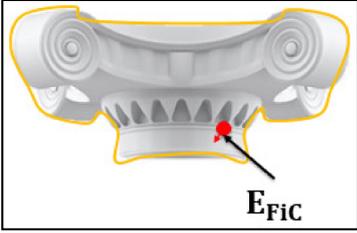
- For *indirect parameter key entities*

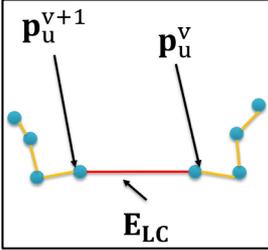
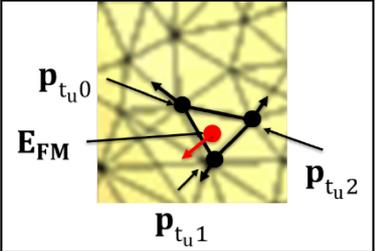
They are not related to any reference frame. However, the related key entities used for its specification should be transformed first into global space.

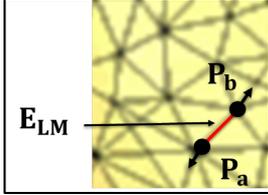
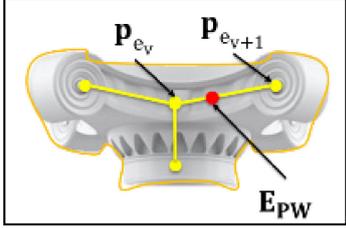
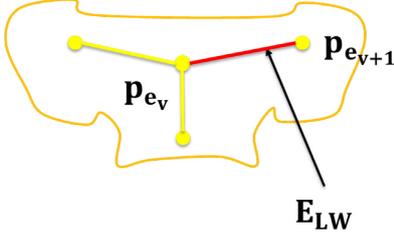
Detailed specification

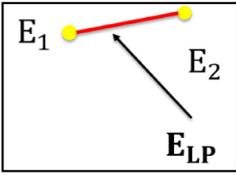
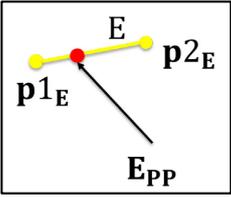
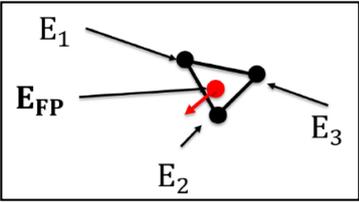
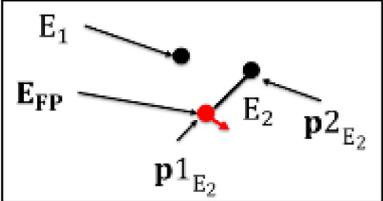
The following **Table 4.2**, presents the details of each *key entity* that is proposed in this thesis. They are described as they were internally represented to be used by the constraint-solving engine; clearly the indicated definition elements are not directly specified by the user but computed based on the direct selection through the graphical interface. In the table there is no distinction between the topological element of the geometric representation (i.e. the vertex of a mesh) and its spatial counterpart (i.e. the point in 3D space corresponding to the vertex). To clarify the meaning of the key entities, examples are presented in each definition.

Basic Definition	<p>Geometric and Structural representations:</p> <p>Con2D := (loops) is a contour 2D, where loops = $\{l_1, l_2, \dots, l_m\}$, l_i is a loop $l_i = \{\mathbf{p}_{i1}, \mathbf{p}_{i2}, \dots, \mathbf{p}_{in}\}$, $m, n, i \in \mathbb{N}$, $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n \in \mathbf{V}$, $i \in [1..m]$</p> <p>Mesh3D := (vertices, triangles, normals) is a mesh, where vertices = $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_m\}$, $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_m \in \mathbf{V}$, $m \in \mathbb{N}$ triangles = $\{t_1, t_2, \dots, t_n\}$, $t_1, t_2, \dots, t_n, n \in \mathbb{N}$ normals = $\{\mathbf{n}_1, \mathbf{n}_2, \dots, \mathbf{n}_k\}$, $\mathbf{n}_1, \mathbf{n}_2, \dots, \mathbf{n}_k \in \mathbf{V}$, $k \in \mathbb{N}$</p> <p>Wireframe := (nodes, edges) is a wireframe, where nodes = $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_k\}$, $\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_k \in \mathbf{V}$, $k \in \mathbb{N}$ edges = $\{e_1, e_2, \dots, e_n\}$, $e_1, e_2, \dots, e_n, n \in \mathbb{N}$</p> <p>$\ \mathbf{n}\$ is the Euclidean norm of the vector (or matrix) \mathbf{n} $\mathbf{a} \times \mathbf{b}$ is the cross product of vector (or matrix) \mathbf{a} and \mathbf{b}. \mathbf{ab} is the multiplication of vector (or matrix) \mathbf{a} and \mathbf{b}. $\mathbf{a} \cdot \mathbf{b}$ is the scalar multiplication of vector (or matrix) \mathbf{b} by \mathbf{a}, where $\mathbf{a} \in \mathbb{R}$.</p>	
E_P	A point in a Component's space	$E_P(\text{Com}, \mathbf{p}) = \mathbf{p}_{E_P}$, where $\mathbf{p} \in \mathbf{V}$ $\mathbf{p}_{E_P} = \mathbf{M}_{E_P} \mathbf{p}$ $\mathbf{M}_{E_P} = f_M(\text{Rf}_{\text{Com}})$ 
E_L	A line in a Component's space	$E_L(\text{Com}, l) = l_{E_L}$, where $l = (\mathbf{p}1_l, \mathbf{p}2_l)$ $l_{E_L} = (\mathbf{p}1_{E_L}, \mathbf{p}2_{E_L})$ $\mathbf{p}1_{E_L} = \mathbf{M}_{E_L} \mathbf{p}1_l$ $\mathbf{p}2_{E_L} = \mathbf{M}_{E_L} \mathbf{p}2_l$ $\mathbf{M}_{E_L} = f_M(\text{Rf}_{\text{Com}})$ 
E_F	A plane in a Component's space	$E_F(\text{Com}, f) = f_{E_L}$, where $f = (\mathbf{p}_f, \mathbf{n}_f)$ $f_{E_L} = (\mathbf{p}_{E_F}, \mathbf{n}_{E_F})$ $\mathbf{p}_{E_F} = \mathbf{M}_{E_F} \mathbf{p}_f$ $\mathbf{n}_{E_F} = [\mathbf{M}_{E_F}^{-1}]^T \mathbf{n}_f$ $\mathbf{M}_{E_F} = f_M(\text{Rf}_{\text{Com}})$ 

<p>\mathbf{E}_{FoC}</p>	<p>An oriented point on a 2D contour</p>	<p>$\mathbf{E}_{\text{FoC}}(\text{Com}, \text{Con2D}, u, v, \theta) = f_{\text{FoC}} = (\mathbf{p}_{\text{FoC}}, \mathbf{n}_{\text{FoC}})$, where</p> $\mathbf{p}_{\text{FoC}} = \mathbf{M}_{\text{FoC}}(\mathbf{p}_u^v + \theta \cdot (\mathbf{p}_u^{v+1} - \mathbf{p}_u^v))$ $\mathbf{n}_{\text{FoC}} = [\mathbf{M}_{\text{FoC}}^{-1}]^T \frac{\mathbf{n}_u^v + \mathbf{n}_u^{v+1}}{2}$ $\mathbf{M}_{\text{FoC}} = f_{\mathbf{M}}(\text{Rf}_{\text{Com}})f_{\mathbf{M}}(\text{Rf}_{\text{Com}}^{\text{Con2D}})$  <p>$\mathbf{p}_u^v, \mathbf{p}_u^{v+1}$, are the v^{th} and $v + 1^{\text{th}}$ vertex of the u^{th} loop of this Con2D, $\mathbf{n}_u^v, \mathbf{n}_u^{v+1}$, are the normal of them. θ is the distance factor of this point which equals to the distance between \mathbf{p}_{FoC} and \mathbf{p}_u^v divided by the distance between \mathbf{p}_u^v and \mathbf{p}_u^{v+1}.</p>
<p>\mathbf{E}_{FiC}</p>	<p>An oriented point inside a 2D contour</p>	<p>$\mathbf{E}_{\text{FiC}}(\text{Com}, \text{Con2D}, x, y) = f_{\text{FiC}} = (\mathbf{p}_{\text{FiC}}, \mathbf{n}_{\text{FiC}})$</p>  <p>Where $\mathbf{p}_{\text{FiC}} = \mathbf{M}_{\text{FiC}} \frac{1}{n} \sum_{i=1}^n \mathbf{p}_i + \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$, n is the total number of the vertices, i.e. considering all the loops in Con2D. $x, y \in \mathbb{R}$ is the displacement of this key point applied to the centroid of all the vertices of the Con2D.</p> $\mathbf{n}_{\text{FiC}} = [\mathbf{M}_{\text{FiC}}^{-1}]^T \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \end{bmatrix}$ <p>is the normal of the plane where is Con2D located.</p> $\mathbf{M}_{\text{FiC}} = f_{\mathbf{M}}(\text{Rf}_{\text{Com}})f_{\mathbf{M}}(\text{Rf}_{\text{Com}}^{\text{Con2D}})$

\mathbf{E}_{LC}	An edge of a 2D contour	$\mathbf{E}_{LC}(\text{Com}, \text{Con2D}, u, v) = \mathbf{l}_{E_{LC}} = (\mathbf{p}_{1_{E_{LC}}}, \mathbf{p}_{2_{E_{LC}}}), u, v \in \mathbb{N}$  <p>where:</p> $\mathbf{p}_{1_{E_{LC}}} = \mathbf{M}_{E_{LC}} \mathbf{p}_u^v$ $\mathbf{p}_{2_{E_{LC}}} = \mathbf{M}_{E_{LC}} \mathbf{p}_u^{v+1}$ $\mathbf{M}_{E_{LC}} = f_M(\text{Rf}_{\text{Com}}) f_M(\text{Rf}_{\text{Com}}^{\text{Con2D}})$ <p>where $\mathbf{p}_u^v, \mathbf{p}_u^{v+1}$ are the v^{th} and $v + 1^{\text{th}}$ vertices of the u^{th} loop of the Con2D.</p>
\mathbf{E}_{FM}	An oriented point on a surface mesh	$\mathbf{E}_{FM}(\text{Com}, \text{Mesh3D}, u, d_0, d_1, d_2) = \mathbf{f}_{E_{FM}} = (\mathbf{p}_{E_{FM}}, \mathbf{n}_{E_{FM}})$  <p>where:</p> $\mathbf{p}_{E_{FM}} = \mathbf{M}_{E_{FM}} (d_0 \cdot \mathbf{p}_{t_u,0} + d_1 \cdot \mathbf{p}_{t_u,1} + d_2 \cdot \mathbf{p}_{t_u,2})$ <p>where $\mathbf{p}_{t_u,0}, \mathbf{p}_{t_u,1}, \mathbf{p}_{t_u,2}$ are the three vertices on the triangle t_u of the Mesh3D where this point is located</p> $\mathbf{n}_{E_{FM}} = [\mathbf{M}_{E_{FM}}^{-1}]^T (d_0 \cdot \mathbf{n}_{t_u,0} + d_1 \cdot \mathbf{n}_{t_u,1} + d_2 \cdot \mathbf{n}_{t_u,2})$ $\in \mathbb{R}^3$ <p>where $\mathbf{n}_{t_u,0}, \mathbf{n}_{t_u,1}, \mathbf{n}_{t_u,2}$ are the three normals of the three vertices of the triangle t_u.</p> $\mathbf{M}_{E_{FM}} = f_M(\text{Rf}_{\text{Com}}) f_M(\text{Rf}_{\text{Com}}^{\text{Mesh3D}})$ <p>d_0, d_1, d_2 are the three factors indicating the position of this point related to the three vertices of the u^{th} triangle. $u \in \mathbb{N}, d_0, d_1, d_2 \in \mathbb{R}$</p>
\mathbf{E}_{LM}	An edge of a mesh	$\mathbf{E}_{LM}(\text{Com}, \text{Mesh3D}, a, b) = \mathbf{l}_{E_{LM}} = (\mathbf{p}_{1_{E_{LM}}}, \mathbf{p}_{2_{E_{LM}}})$

		 $\mathbf{p1}_{E_{LM}} = \mathbf{M}_{E_{LM}} \mathbf{p}_a$ $\mathbf{p2}_{E_{LM}} = \mathbf{M}_{E_{LM}} \mathbf{p}_b$ $\mathbf{M}_{E_{LM}} = f_M(\text{Rf}_{\text{Com}}) f_M(\text{Rf}_{\text{Com}}^{\text{Mesh3D}})$ <p>$\mathbf{p}_a, \mathbf{p}_b$ are the coordinates of the two vertices of the edge.</p>
\mathbf{E}_{PW}	<p>A point along the edges of a wireframe</p>	$\mathbf{E}_{PW}(\text{Com}, \text{Wireframe}, v, \theta) = \mathbf{p}_{E_{PW}}$  <p>where</p> $\mathbf{p}_{E_{PW}} = \mathbf{M}_{E_{PW}} (\mathbf{p}_{e_v} + \theta \cdot (\mathbf{p}_{e_{v+1}} - \mathbf{p}_{e_v})), \theta \in [0,1] \in \mathbb{R}^3$ $\mathbf{M}_{E_{PW}} = f_M(\text{Rf}_{\text{Com}}) f_M(\text{Rf}_{\text{Com}}^{\text{Wireframe}})$ <p>v and $v + 1$ are the indexes of values in the edge list of the wireframe. e_{v+1} and e_v are the indexes of the two nodes indicating an edge in the node list. θ is the position factor between the two nodes.</p>
\mathbf{E}_{LW}	<p>An edge of a wireframe</p>	$\mathbf{E}_{LW}(\text{Com}, \text{Wireframe}, v) = l_{E_{LW}} = (\mathbf{p1}_{E_{LW}}, \mathbf{p2}_{E_{LW}})$  <p>where</p> $\mathbf{p1}_{E_{LW}} = \mathbf{M}_{E_{LW}} \mathbf{p}_{e_v}$ $\mathbf{p2}_{E_{LW}} = \mathbf{M}_{E_{LW}} \mathbf{p}_{e_{v+1}}$ $\mathbf{M}_{E_{LW}} = f_M(\text{Rf}_{\text{Com}}) f_M(\text{Rf}_{\text{Com}}^{\text{Wireframe}})$ <p>As edges are stored as a list “e” of indices of nodes, then e_v and e_{v+1}, are the indices of the two nodes of the considered edge stored in the edge list.</p>

<p>E_{LP}</p>	<p>A parametric line (created by two key points)</p>	<p>$\mathbf{E}_{LP}(E_1, E_2) = l_{E_{LP}} = (\mathbf{p}_{1_{E_{LP}}}, \mathbf{p}_{2_{E_{LP}}})$</p>  <p>where</p> <p>$\mathbf{p}_{1_{E_{LP}}} = \mathbf{p}_{E_1}, \mathbf{p}_{2_{E_{LP}}} = \mathbf{p}_{E_2}$</p> <p>$E_0, E_1 \in \{\mathbf{E}_P, \mathbf{E}_{PW}, \mathbf{E}_{PP}, \mathbf{E}_F, \mathbf{E}_{FOC}, \mathbf{E}_{FiC}, \mathbf{E}_{FM}, \mathbf{E}_{FP}\}$</p>
<p>E_{PP}</p>	<p>A parametric point (along a parametric line)</p>	<p>$\mathbf{E}_{PP}(E, \theta) = \mathbf{p}_{E_{PP}}$</p>  <p>where</p> <p>$\mathbf{p}_{E_{PP}} = \mathbf{p}_{1_E} + \theta \cdot (\mathbf{p}_{2_E} - \mathbf{p}_{1_E}), \text{ where } \theta \in [0,1],$</p> <p>$E \in \{\mathbf{E}_{LP}\}$</p>
<p>E_{FP}</p>	<p>A parametric oriented point</p>	<p>$\mathbf{E}_{FP}(u, \text{Para}) = \mathbf{f}_{E_{FP}} = (\mathbf{p}_{E_{FP}}, \mathbf{n}_{E_{FP}})$</p> <p>1) If $u = 0$, for an oriented point specified by three points, then $\text{Para} = (E_1, E_2, E_3)$,</p>  <p>$\mathbf{p}_{E_{FP}} = \frac{1}{3}(\mathbf{p}_{E_1} + \mathbf{p}_{E_2} + \mathbf{p}_{E_3}),$</p> <p>$\mathbf{n}_{E_{FP}} = (\mathbf{p}_{E_2} - \mathbf{p}_{E_1}) \times (\mathbf{p}_{E_3} - \mathbf{p}_{E_1}), \text{ where}$</p> <p>$E_1, E_2, E_3 \in \{\mathbf{E}_P, \mathbf{E}_{PW}, \mathbf{E}_{PP}, \mathbf{E}_F, \mathbf{E}_{FOC}, \mathbf{E}_{FiC}, \mathbf{E}_{FM}, \mathbf{E}_{FP}\}$</p> <p>2) If $u = 1$, for an oriented point specified by a point and a line, then $\text{Para} = (E_1, E_2)$</p>  <p>$\mathbf{p}_{E_{FP}} = \mathbf{p}_{1_{E_1}}$</p> <p>$\mathbf{n}_{E_{FP}} = (\mathbf{p}_{1_{E_2}} - \mathbf{p}_{E_1}) \times (\mathbf{p}_{2_{E_2}} - \mathbf{p}_{E_1}), \text{ where}$</p>

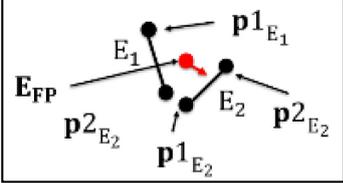
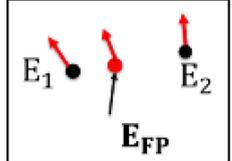
		$E_1 \in \{E_P, E_{PW}, E_{PP}, E_F, E_{FoC}, E_{FiC}, E_{FM}, E_{FP}\}$ $E_2 \in \{E_L, E_{LC}, E_{LM}, E_{LW}, E_{LP}\}$ <p>3) If $u = 2$, for an oriented point specified by two lines, then $Para = (E_1, E_2)$</p>  $p_{E_{FP}} = \frac{1}{4}(p1_{E_2} + p2_{E_2} + p1_{E_1} + p2_{E_1}),$ $n_{E_{FP}} = (p2_{E_1} - p1_{E_1}) \times (p2_{E_2} - p1_{E_2}), \text{ where}$ $E_1, E_2 \in \{E_L, E_{LC}, E_{LM}, E_{LW}, E_{LP}\}$ <p>4) If $u = 3$, for an oriented point specified by two oriented points, then $Para = (E_1, E_2, \theta)$</p>  $p_{E_{FP}} = p_{E_1} + \theta \cdot (p_{E_2} - p_{E_1}),$ $n_{E_{FP}} = n_{E_1} + \theta \cdot (n_{E_2} - n_{E_1}), \text{ where}$ $E_1, E_2 \in \{E_F, E_{FoC}, E_{FiC}, E_{FM}, E_{FP}\}$
E_A	An array of key entities	$E_A(\{E_i\}_{i \in [0, n-1]})$ E_i is one of all the above key entities

Table 4.2 Key entities' definition

PROPERTIES

- Dimension

A key entity, whether it is a geometric or a parametric one, can be finally 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 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* can also be represented by the number of elements of \mathbb{R}^3 used to specify its representation. For example, E_{LC} (an edge of the contour 2D) is represented by a line defined by two points (each of them is an instance of an element of \mathbb{R}^3). , All the key entities are represented in a 3D space. A table classifying all the key entities by their dimension is presented as below:

1D key entities	2D key entities		nD key entities
Point	Line	Oriented Point	Array
E_P, E_{PW}, E_{PP}	$E_L, E_{LC}, E_{LW}, E_{LM}, E_{LP}$	$E_F, E_{FOC}, E_{FIC}, E_{FM}, E_{FP}$	E_A

Table 4.3 Dimension of Key entities

- Independent and related (or geometric and parametric)

As already mentioned, a geometric *key entity* is independent from any representation and a parametric *key entity* is related to a geometric or structural representation or to other *key entities*.

- Direct and indirect (for *parametric key entities*)

A *parametric key entity* can be directly associated to the entities used to define a geometric or a structural representation. It can also be indirectly related to a geometric or structural representation through some existing key entities.

- Multi-modality

From the presented specification of *key entity*, it can be noticed that the parametric key entity can be associated to the geometric and/or the structural representation of the component, while in traditional CAD systems, the key entities used to specify constraints in assemblies are only located on its geometric layer.

DATA STRUCTURE IN UML

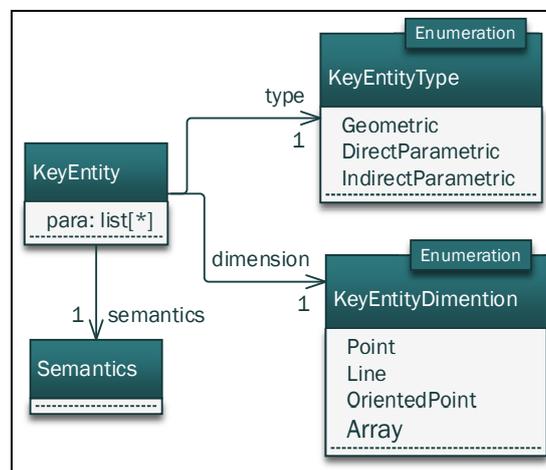


Figure 4.20 Data structure of *Key Entity*

The above **Figure 4.20** presents the data structure of *key entity*. It is structured with a list of parameters (“para”), type (“type”), dimension (“dimension”) and *semantics*. The type indicates the three types of key entity: *geometric*, *direct parametric* and *indirect parametric*. “Dimension” points to the types of geometric primitives (Point, Line, Oriented Point or an Array). The *semantics* of key entity tells its meaning such as a vertex on a mesh, an edge of a wireframe, etc.

4.4.3 CONSTRAINT

DEFINITION

Definition 4.11: Constraint = A condition limiting the related location of two *elements* by applying equations between two *key entities*

Constraints are expressed by equations or inequalities correlating different *key entities*. Therefore, to completely specify a *constraint*, we need not only to define which are the *key entities* involved, but also which equations or inequalities have to be applied.

PURPOSE

Constraint is used to limit the location of **two** *key entities*. It can occur only between two *key entities*, since, if more than two key entities are involved we can use the key entity \mathbf{E}_A , which is a list of key entities.

MATHEMATICAL SPECIFICATION

The equations and inequalities are different for different *constraints*. Even for the same *constraint*, different combinations of the two key entities may require different equations or inequalities. In this sense, the combination of the two *key entities* involved becomes very useful for defining a *constraint*. Thus, in **Table 4.4**, all the constraints are proposed taking into consideration all the possible combinations of key entities.

Sym- bol	Name	Acceptable <i>key entity</i> Combination
C_D	Distance	(Point, Point), (Point, Oriented point), (Oriented point, Oriented point)
C_A	Angle	(Line, Line), (Line, Oriented point), (Oriented point, Oriented point)
C_{Co}	Coincidence	(Point, Point), (Point, Oriented point)
C_{Pa}	Parallelism	(Line, Line), (Line, Oriented point), (Oriented point, Oriented point)
C_{Pe}	Perpendicularity	(Line, Line), (Line, Oriented point), (Oriented point, Oriented point)
C_{Cl}	Co-linearity	(Point, Line), (Oriented point, Line)
C_{Cp}	Co-planarity	(Point, Oriented point), (Oriented point, Oriented point)
C_{Ca}	Co-axiality	(Line, Line)
C_T	Tangency	(Oriented point, Line), (Oriented point, Oriented point)
C_I	Insertion	(Line, Line)
C_{Ct}	Contact	(Oriented point, Oriented point)
C_{Pt}	Pattern	(Point, Array), (Line, Array), (Oriented point, Array)

Table 4.4 Constraints with an associated combination of *key entities*

“Coincidence” is only between two points, “Co-linearity” is used to limit the position of a point along a line. “Co-planarity” is used to limit a point on a surface. “Co-axiality”, to limit two lines to be coincident. “Insertion” is used to put two lines that are “Co-axial” then limits the distance between them. “Contact” is used to put two surfaces that touch each other and “Tangent” is used to limit a line and a plane or two planes for them to be tangent. “Pattern” is used to distribute points along a line or a round a point.

The constraints that have been considered were thought to be significant for users. As a consequence, 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 calculate distance is not linear. In order to maximise the use of linear equations and linear inequalities, different equations are used to formulate “Coincidence” and “Distance”. Therefore, six basic expressions (equations or inequalities) are defined acting on vectors. All the constraints proposed are defined by using these expressions.

Thus, the equations for each constraint are defined in following **Table 4.5**.

<p>Notations:</p> <p>“==” for conditional-Equal</p> <p>“&&” for conditional -And</p> <p>“ ” for conditional-Or</p> <p>$\ \mathbf{n}\$ for the Euclidean norm of vector \mathbf{n}</p> <p>v for the absolute value of v.</p> <p>→ to give the expression of one equation or inequality defined by parameters</p>
<p>Basic expression (equations or inequalities):</p> <p>$e_0(\mathbf{v}_1, \mathbf{v}_2, a) \rightarrow \mathbf{v}_2 == a\mathbf{v}_1$, for \mathbf{v}_2 is scaled from \mathbf{v}_1 by a. (three linear equations)</p> $\rightarrow \begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} == a \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}$ $\rightarrow \begin{cases} x_2 == a \cdot x_1 \\ y_2 == a \cdot y_1 \\ z_2 == a \cdot z_1 \end{cases}$ <p>$e_1(\mathbf{v}_1, \mathbf{v}_2) \rightarrow \mathbf{v}_1 \cdot \mathbf{v}_2 == 1$ for \mathbf{v}_1 is perpendicular to \mathbf{v}_2. (one linear equation)</p> $\rightarrow \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \cdot [x_2, y_2, z_2, 1] = x_1x_2 + y_1y_2 + z_1z_2 + 1 == 1$ <p>$e_2(\mathbf{v}_1, \mathbf{v}_2) \rightarrow e_0 \left(\begin{bmatrix} x_1y_2 \\ y_1z_2 \\ x_1z_2 \\ 1 \end{bmatrix}, \begin{bmatrix} x_2y_1 \\ y_2z_1 \\ x_2z_1 \\ 1 \end{bmatrix}, 1 \right)$ for \mathbf{v}_1 is parallel to \mathbf{v}_2. (three linear equations)</p>

$$\rightarrow \begin{cases} x_1 y_2 == x_2 y_1 \\ y_1 z_2 == y_2 z_1 \\ x_1 z_2 == x_2 z_1 \end{cases}$$

$$e_3(\mathbf{v}_1, \mathbf{v}_2) \rightarrow e_0 \left(\begin{bmatrix} x_1 y_2 \\ y_1 z_2 \\ x_1 z_2 \\ 1 \end{bmatrix}, \begin{bmatrix} x_2 y_1 \\ y_2 z_1 \\ x_2 z_1 \\ 1 \end{bmatrix}, 1 \right) \&\& \begin{cases} x_1 x_2 \leq 0 \\ y_1 y_2 \leq 0 \\ z_1 z_2 \leq 0 \end{cases} \text{ for } \mathbf{v}_1 \text{ is opposite to } \mathbf{v}_2 \text{ (three linear}$$

equations and three linear inequalities)

$$\rightarrow \begin{cases} x_1 y_2 == x_2 y_1 \\ y_1 z_2 == y_2 z_1 \\ x_1 z_2 == x_2 z_1 \\ x_1 x_2 \leq 0 \\ y_1 y_2 \leq 0 \\ z_1 z_2 \leq 0 \end{cases}$$

$e_4(\mathbf{v}_1, \mathbf{v}_2, a) \rightarrow \|\mathbf{v}_2 - \mathbf{v}_1\| == a$ for the distance between \mathbf{v}_1 and \mathbf{v}_2 is a . (one non-linear equation)

$$\rightarrow \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} == a$$

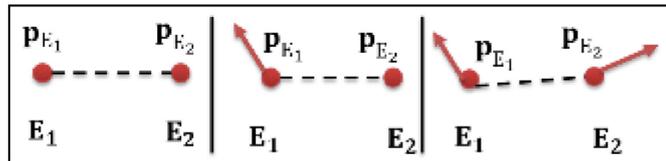
$e_5(\mathbf{v}_1, \mathbf{v}_2, a) \rightarrow \frac{\mathbf{v}_1 \cdot \mathbf{v}_2 - 1}{\|\mathbf{v}_2 - \mathbf{v}_1\|} == \cos(a)$ for the cosine of the angle between \mathbf{v}_1 and \mathbf{v}_2 is a . (one non-linear equation)

$$\rightarrow \frac{x_1 x_2 + y_1 y_2 + z_1 z_2}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}} == a$$

$\mathbf{p}_1, \mathbf{p}_2 \in \mathbf{V}$

Distance $\mathbf{C}_D(E_1, E_2, \text{distance})$

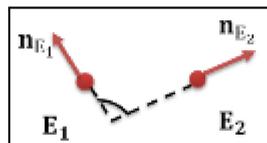
$$= e_4(\mathbf{p}_{E_1}, \mathbf{p}_{E_2}, \text{distance})$$



$(E_1, E_2) \in \{\mathbf{E}_P, \mathbf{E}_{PW}, \mathbf{E}_{PP}, \mathbf{E}_F, \mathbf{E}_{FOC}, \mathbf{E}_{FiC}, \mathbf{E}_{FM}, \mathbf{E}_{FP}\}^2$, distance $\in \mathbb{R}$, distance ≥ 0

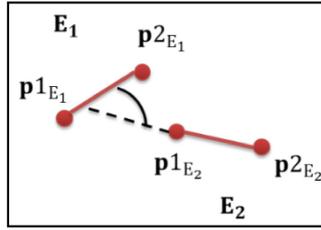
Angle $\mathbf{C}_A(E_1, E_2, \text{angle})$

$$= e_5(\mathbf{n}_{E_1}, \mathbf{n}_{E_2}, \text{angle})$$



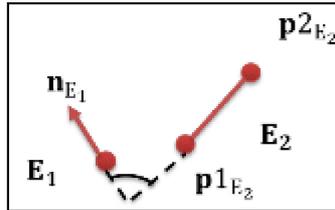
if $(E_1, E_2) \in \{\mathbf{E}_F, \mathbf{E}_{FOC}, \mathbf{E}_{FiC}, \mathbf{E}_{FM}, \mathbf{E}_{FP}\}^2$ are oriented points

$$= e_5(\mathbf{p}_{2E_1} - \mathbf{p}_{1E_1}, \mathbf{p}_{2E_2} - \mathbf{p}_{1E_1}, \text{angle})$$



if $(E_1, E_2) \in \{\mathbf{E}_L, \mathbf{E}_{LC}, \mathbf{E}_{LW}, \mathbf{E}_{LM}, \mathbf{E}_{LP}\}^2$ are lines

$$= e_5(\mathbf{p}_{2E_2} - \mathbf{p}_{1E_2}, \mathbf{n}_{E_1}, \text{angle})$$



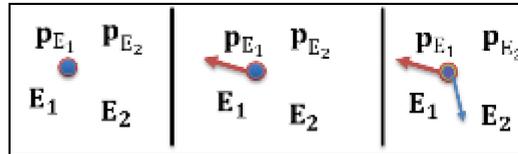
if $E_1 \in \{\mathbf{E}_F, \mathbf{E}_{FOC}, \mathbf{E}_{FiC}, \mathbf{E}_{FM}, \mathbf{E}_{FP}\}$ is an oriented point

$E_2 \in \{\mathbf{E}_L, \mathbf{E}_{LC}, \mathbf{E}_{LW}, \mathbf{E}_{LM}, \mathbf{E}_{LP}\}$ is a line,

angle $\in [0, 180]$

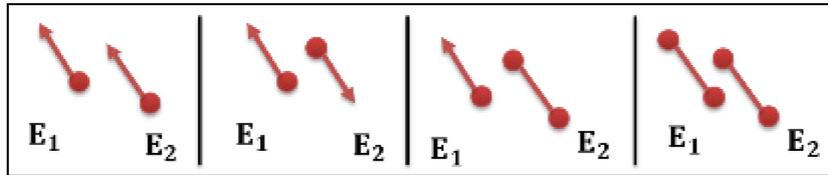
Coincidence $C_{Co}(E_1, E_2)$

$$= e_0(\mathbf{p}_{E_1}, \mathbf{p}_{E_2}, 1)$$



$(E_1, E_2) \in \{\mathbf{E}_P, \mathbf{E}_{PW}, \mathbf{E}_{PP}, \mathbf{E}_F, \mathbf{E}_{FOC}, \mathbf{E}_{FiC}, \mathbf{E}_{FM}, \mathbf{E}_{FP}\}^2$

Parallelism $C_{Pa}(E_1, E_2)$

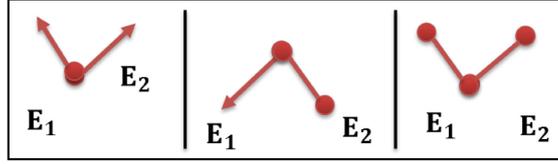


$$= e_2(\mathbf{n}_{E_1}, \mathbf{n}_{E_2}), \text{ if } (E_1, E_2) \in \{\mathbf{E}_F, \mathbf{E}_{FOC}, \mathbf{E}_{FiC}, \mathbf{E}_{FM}, \mathbf{E}_{FP}\}^2$$

$$= e_2(\mathbf{n}_{E_2}, \mathbf{p}_{2E_1} - \mathbf{p}_{1E_1}), \text{ if } E_1 \in \{\mathbf{E}_F, \mathbf{E}_{FOC}, \mathbf{E}_{FiC}, \mathbf{E}_{FM}, \mathbf{E}_{FP}\}, E_2 \in \{\mathbf{E}_L, \mathbf{E}_{LC}, \mathbf{E}_{LW}, \mathbf{E}_{LM}, \mathbf{E}_{LP}\}$$

$$= e_2(\mathbf{p}_{2E_1} - \mathbf{p}_{1E_1}, \mathbf{p}_{2E_2} - \mathbf{p}_{1E_2}), \text{ if } (E_1, E_2) \in \{\mathbf{E}_L, \mathbf{E}_{LC}, \mathbf{E}_{LW}, \mathbf{E}_{LM}, \mathbf{E}_{LP}\}^2$$

Perpendicularity $C_{Pe}(E_1, E_2)$



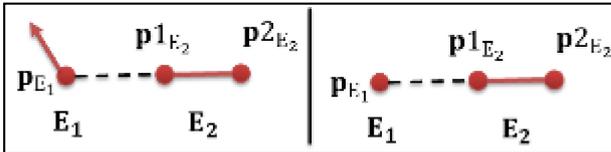
$$= e_1(\mathbf{n}_{E_1}, \mathbf{n}_{E_2}), \text{ if } (E_1, E_2) \in \{\mathbf{E}_F, \mathbf{E}_{F_0C}, \mathbf{E}_{F_{iC}}, \mathbf{E}_{F_M}, \mathbf{E}_{F_P}\}^2$$

$$= e_1(\mathbf{n}_{E_2}, \mathbf{p}2_{E_1} - \mathbf{p}1_{E_1}), \text{ if } E_1 \in \{\mathbf{E}_F, \mathbf{E}_{F_0C}, \mathbf{E}_{F_{iC}}, \mathbf{E}_{F_M}, \mathbf{E}_{F_P}\}, E_2 \in \{\mathbf{E}_L, \mathbf{E}_{L_C}, \mathbf{E}_{L_W}, \mathbf{E}_{L_M}, \mathbf{E}_{L_P}\}$$

$$= e_1(\mathbf{p}2_{E_1} - \mathbf{p}1_{E_1}, \mathbf{p}2_{E_2} - \mathbf{p}1_{E_2}), \text{ if } (E_1, E_2) \in \{\mathbf{E}_L, \mathbf{E}_{L_C}, \mathbf{E}_{L_W}, \mathbf{E}_{L_M}, \mathbf{E}_{L_P}\}^2$$

Co-linearity $C_{Cl}(E_1, E_2)$

$$= e_2(\mathbf{p}_{E_1} - \mathbf{p}1_{E_2}, \mathbf{p}2_{E_2} - \mathbf{p}1_{E_2})$$

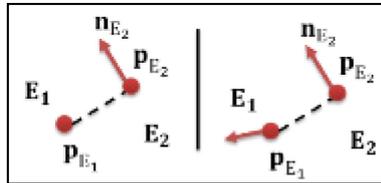


$E_1 \in \{\mathbf{E}_P, \mathbf{E}_{P_W}, \mathbf{E}_{P_P}, \mathbf{E}_F, \mathbf{E}_{F_0C}, \mathbf{E}_{F_{iC}}, \mathbf{E}_{F_M}, \mathbf{E}_{F_P}\}$ is a point or an oriented point

$E_2 \in \{\mathbf{E}_L, \mathbf{E}_{L_C}, \mathbf{E}_{L_W}, \mathbf{E}_{L_M}, \mathbf{E}_{L_P}\}$ is line

Co-planarity $C_{Cp}(E_1, E_2)$

$$= e_1(\mathbf{p}_{E_1} - \mathbf{p}_{E_2}, \mathbf{n}_{E_2})$$

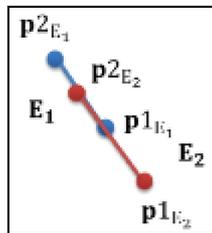


$E_1 \in \{\mathbf{E}_P, \mathbf{E}_{P_W}, \mathbf{E}_{P_P}, \mathbf{E}_F, \mathbf{E}_{F_0C}, \mathbf{E}_{F_{iC}}, \mathbf{E}_{F_M}, \mathbf{E}_{F_P}\}$ is a point or an oriented point

$E_2 \in \{\mathbf{E}_F, \mathbf{E}_{F_0C}, \mathbf{E}_{F_{iC}}, \mathbf{E}_{F_M}, \mathbf{E}_{F_P}\}$ is an oriented point

Co-axiality $C_{Ca}(E_1, E_2)$

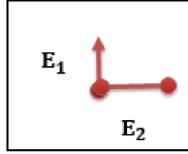
$$= e_2(\mathbf{p}1_{E_1} - \mathbf{p}1_{E_2}, \mathbf{p}2_{E_2} - \mathbf{p}1_{E_2}) \&\& e_2(\mathbf{p}2_{E_1} - \mathbf{p}1_{E_2}, \mathbf{p}2_{E_2} - \mathbf{p}1_{E_2})$$



$(E_1, E_1) \in \{\mathbf{E}_L, \mathbf{E}_{L_C}, \mathbf{E}_{L_W}, \mathbf{E}_{L_M}, \mathbf{E}_{L_P}\}^2$ are lines

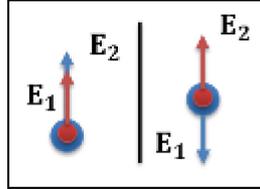
Tangency $C_T(E_1, E_2)$

$$= e_0(\mathbf{p}_{E_1}, \mathbf{p}1_{E_2}, 1) \&\& e_1(\mathbf{n}_{E_1}, \mathbf{p}2_{E_2} - \mathbf{p}1_{E_2})$$



if $E_1 \in \{E_F, E_{FoC}, E_{FiC}, E_{FM}, E_{FP}\}$ is a oriented point,
 $E_2 \in \{E_L, E_{LC}, E_{LW}, E_{LM}, E_{LP}\}$ is a line

$$= e_0(\mathbf{p}_{E_1}, \mathbf{p}_{E_2}, 1) \ \&\& \ e_2(\mathbf{n}_{E_1}, \mathbf{n}_{E_2})$$



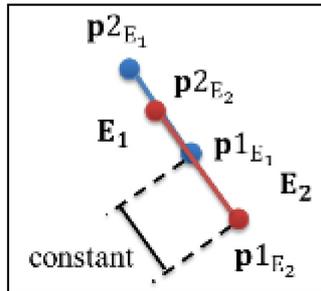
if $(E_1, E_2) \in \{E_F, E_{FoC}, E_{FiC}, E_{FM}, E_{FP}\}^2$ are oriented points

Insertion $C_I(E_1, E_2, \text{distance})$

$$= e_2(\mathbf{p}_{1E_1} - \mathbf{p}_{1E_2}, \mathbf{p}_{2E_2} - \mathbf{p}_{1E_2})$$

$$\ \&\& \ e_2(\mathbf{p}_{2E_1} - \mathbf{p}_{1E_2}, \mathbf{p}_{2E_2} - \mathbf{p}_{1E_2})$$

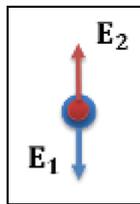
$$\ \&\& \ e_4(\mathbf{p}_{1E_1}, \mathbf{p}_{1E_2}, \text{distance}),$$



$(E_1, E_2) \in \{E_L, E_{LC}, E_{LW}, E_{LM}, E_{LP}\}^2$, $\text{distance} \in \mathbb{R}$, $\text{distance} \geq 0$

Contact $C_{Ct}(E_1, E_2)$

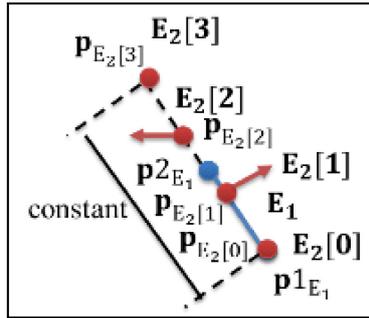
$$= e_0(\mathbf{p}_{E_1}, \mathbf{p}_{E_2}, 1) \ \&\& \ e_3(\mathbf{n}_{E_1}, \mathbf{n}_{E_2}), (E_1, E_2) \in \{E_F, E_{FoC}, E_{FiC}, E_{FM}, E_{FP}\}^2$$



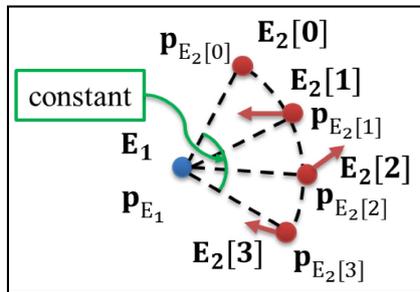
Pattern $C_{Pt}(E_1, E_2, \text{distance}, \text{radius})$, $\text{distance}, \text{radius} \in \mathbb{R}$, $\text{distance}, \text{radius} \geq 0$

$$= C_{CI}(E_2[i], E_1) \ \&\& \ e_0\left(\mathbf{p}_{1E_2}, \mathbf{p}_{E_2[i]}, i * \frac{\text{distance}}{n-1}\right)$$

for $\forall E_2[i]$ in E_2 , $i \in [0, n-1] \subset \mathbb{N}$, n is the number of key entities in E_2



if $E_1 \in \{E_L, E_{LC}, E_{LW}, E_{LM}, E_{LP}\}$ is a line,
 $E_2 \in E_A, E_2[k]_{k=0,1\dots n-1} \in \{E_P, E_{PW}, E_{PP}, E_F, E_{FoC}, E_{FiC}, E_{FM}, E_{FP}\}$
 $= C_D(E_1, E_2[i], \text{radius}) \& \& e_5 \left(p_{E_2[i-1]} - p_{E_1}, p_{E_2[i]} - p_{E_1}, \frac{\text{distance}}{n-1} \right)$
 for $\forall E_2[i], \text{ in } E_2, i \in [1, n-1] \subset \mathbb{N}, n$ is the number of key entities in E_2



if $E_1 \in \{E_P, E_{PW}, E_{PP}, E_F, E_{FoC}, E_{FiC}, E_{FM}, E_{FP}\}$ is point,
 $E_2 \in E_A, E_2[k]_{k=0,1\dots n} \in \{E_P, E_{PW}, E_{PP}, E_F, E_{FoC}, E_{FiC}, E_{FM}, E_{FP}\}$

Table 4.5 Equations for constraints

Figure 4.21 presents the links between each constraint and the six basic expressions.

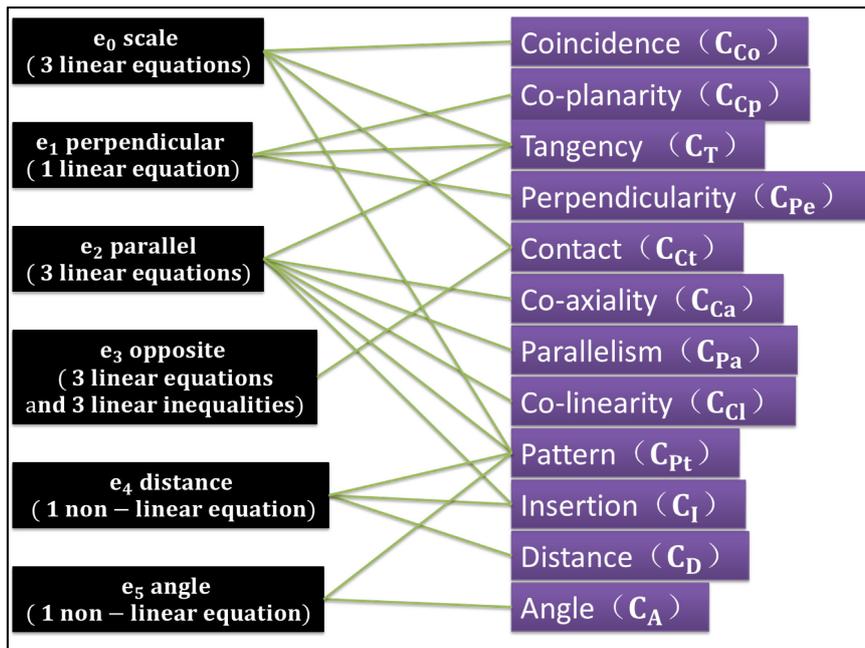


Figure 4.21 Dependencies among the Constraints and the basic functions

PROPERTIES

- Cardinality

As mentioned previously, constraints defined in this thesis are built only between two *key entities*.

- Boolean-valued formula

Constraint is a condition, not a function or an equation. 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.

- Multi-modality

Because of the specification of *key entity*, the constraints are built at both the structural and the geometric level.

DATA STRUCTURE IN UML

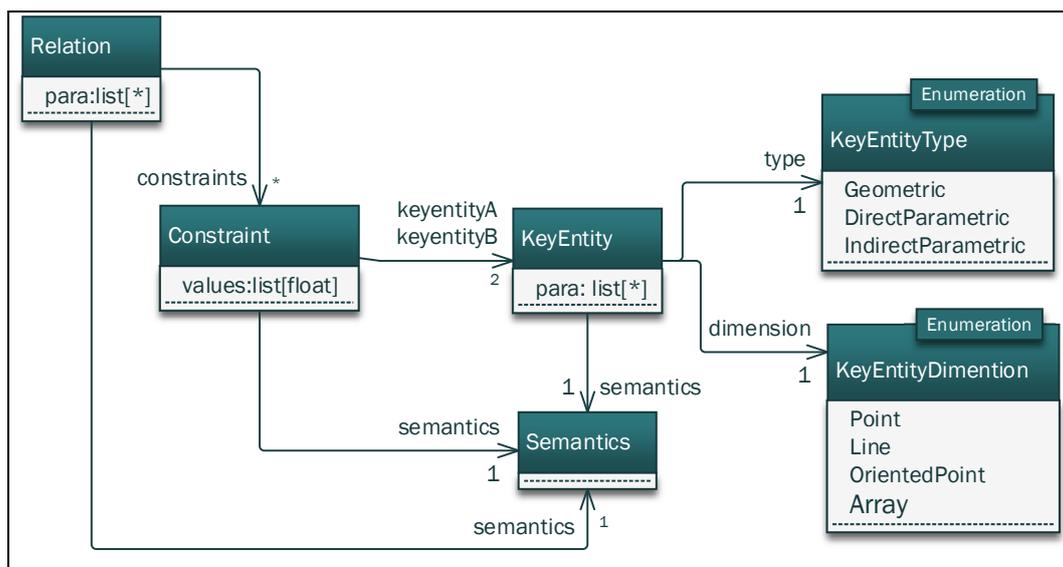


Figure 4.22 Data Structure of a *constraint* and links with the other classes

The data structure of a *constraint* is presented in **Figure 4.22** where *constraint* points to two *key entities* (“keyentityA” and “keyentityB”), to *semantics* (“semantics”) and to an attribute “values” to define the constant value used in the *constraint* if needed. *Relation* indicates a list of constraints, which bridges the *intermediate level* with the *conceptual level*. The *semantics* of constraint tells its meaning.

At the *intermediate level*, we have proposed 14 different kinds of key entities and 12 types of *constraints*. *Key entities*, which are not always explicitly specified by the user, but may be automatically computed by processing his/her input. In the previous example, the user wants to put a ball on a floor. He/she would probably specify a “contact” *constraint* between the ball and the floor. However, on which exact points of the ball and the floor they are touching might be

meaningless for the user. In this case, the user only needs to specify the type of *constraint* without specifying the *key entities*. As in the data structure presented in **Figure 4.22**, *constraints* are saved in a list derived from *relation*, therefore even if the *key entities* are not specified, the *constraint* can still automatically find two suitable temporary *key entities* on the geometric or structural representations of the two *elements* saved in the *relation*. More details about the smart constraining mechanism are given in **Section 5.3**.

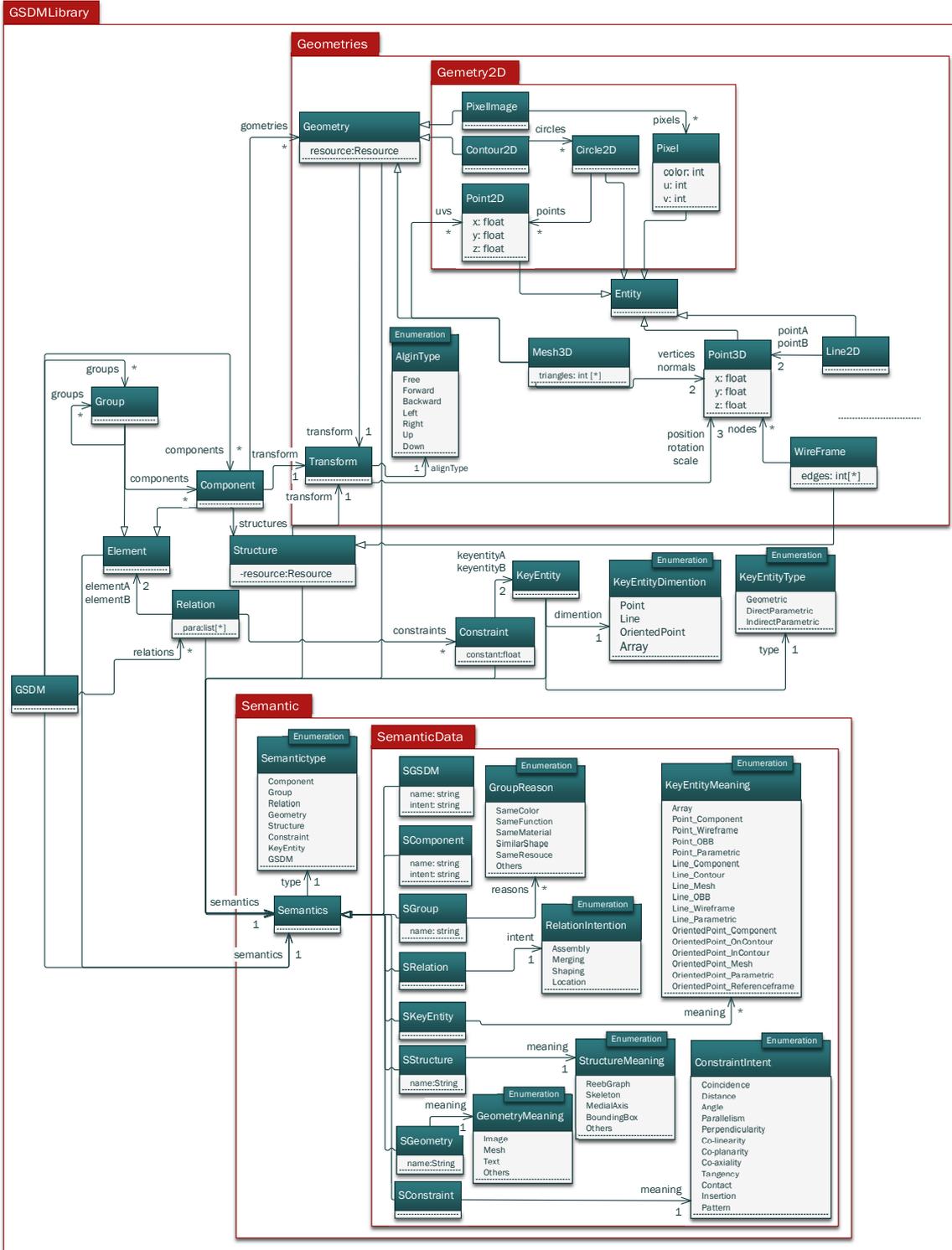


Figure 4.23 Whole data structure of GSDM

4.5 GENERAL OVERVIEW

The complete organization of the GSDM is presented in **Figure 4.23**. In future, other kinds of *geometry*, *structure*, *constraints* or *key entities* or even other types of *relations* can also be developed if they are more useful and convenient for the user. So, the proposed data structure is modular and can be extended.

4.6 CONCLUSION AND REMARKS

In the previous sections, all the constituents of GSDM have been introduced together with the data structure presented in UML. The data structure of GSDM can be implemented in any programming language. The GSDM presented in this chapter is not a fixed and closed model.

The class of “GSDM” comprises a list of *components* a list of *groups*, a list of *relations* and *semantics*. Through the different definitions and data structures, this chapter shows the GSDM’s capability to combine heterogeneous data and constrain them so as to define a structured object whose final geometric representation will be processed at a later stage. At that stage, heterogeneous data are mixed up in a common 3D viewer where 2D images and meshes coexist. However, these data structures cannot be instantiated automatically without user interaction. The way an object can be modeled with the GSDM needs to be clarified. Besides, the information stored in the GSDM needs to be checked. This is especially true for the user-specified *constraints* that can be inconsistent at a given stage. Therefore, various steps in the process of modeling with GSDM are needed to check the rationality of the information stored during the creation of *groups*, the building of *relations* and the specification of *constraints*. These issues are going to be introduced in **Chapter 5**. **Chapter 6** will present an implementation with user-friendly interface.

The good life is a process, not a state of being. It is a direction not a destination.

Carl Rogers

5.

MODELING WITH GSDM

CHAPTER OVERVIEW

THIS CHAPTER PRESENTS the way to create a conceptual model by using the GSDM and solve the related issues presented at the end of the previous **Chapter 4**. It is organized in five sections: workflow of the modeling process with GSDM (**Section 5.1**), working at the conceptual level (**Section 5.2**), working at the intermediate level (**Section 5.3**), solving Constraint Satisfaction Problems (CSP) (**Section 5.4**) and finally some remarks in the conclusion (**Section 5.5**).

5.1 WORKFLOW OF THE MODELING PROCESS WITH GSDM

The objective of using the GSDM, as presented in **Section 3.3**, is to reduce the gap between non-expert users and specification tools in the conceptual design phase, possibly in a collaborative working environment, allowing the re-use of existing heterogeneous data resources (e.g. images, 3D models). In this section, the workflow of the modeling process with GSDM is introduced.

This process mimics the natural way users follow, i.e. focusing on his/her own objectives without bothering about the underlying model organization. Thus, the adopted approach can be classified as a top-down one.

The user usually knows what he/she wants to create but does not necessarily know how to create it step by step. To be more user-friendly, the modeling process of GSDM is defined as presented in **Figure 5.1**.

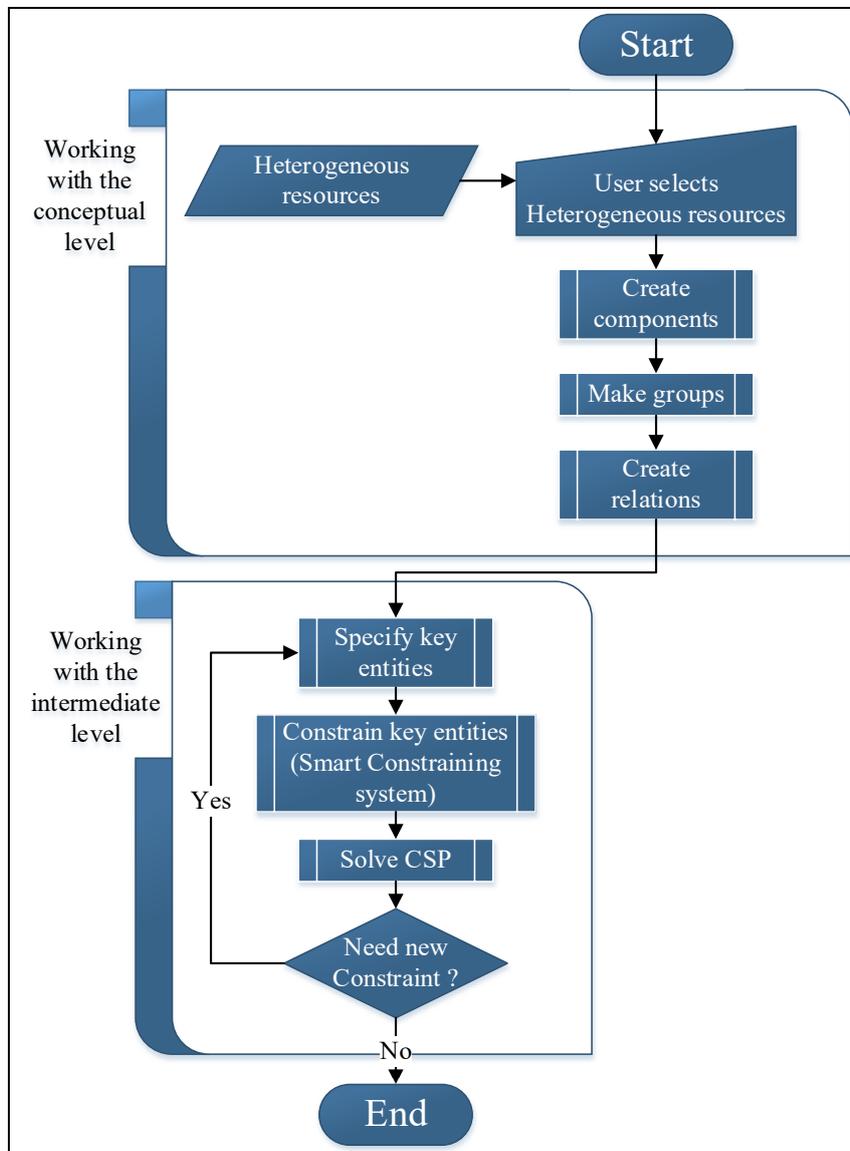


Figure 5.1 The GSDM user's modeling process

The modeling process starts from the *conceptual level*, where the user is actually acting. The first thing the user needs to do is to create *components* from the input heterogeneous resources. The *geometric*, *structural* or *semantic* information stored in the input data is automatically extracted and inserted into the corresponding three layers of the *component*'s description. Then, the user can regroup the *components* or existing *groups*, and specify *relations* between them. At this stage, the *conceptual level* elements are all initialized. The user has to give a general view of the different parts and how they are combined together to form the object that he/she wants to create.

The next step is to define the *relations* more precisely. This step is carried out by specifying the *constraints* between *key entities*. There might be some *key entities*, which can be specified directly from the geometric or structural representations (*direct parametric key entity*) obtained from the heterogeneous data, but the user can also create his/her own *key entities* (*indirect parametric key entity*). With the specified key entities, the user can link two of them and build the related *constraints*. After their specification, the system starts to check the *constraints* and try to find the best solution to satisfy all the user-specified *constraints*. This step is called Constraint Satisfaction Problem (CSP) solving. Each time a new *constraint* is specified, the CSP solver is automatically launched. The way the CSP is solved is discussed in **Section 5.4**.

These steps can be repeated until the complete specification of the object is achieved. **Figure 5.1** shows only one feedback loop. However, the process is iterative and the user can return to the previous steps to modify the specification at any time. For example, after specifying *relations*, the user can still go back to group or to add new *components*. It should be noted that those changes might affect the following steps, e.g. requiring a new constraint solving process.

The following sections focuses on the details of each step specifying the corresponding actions on the GSDM. The graphical user-interface will be presented in the next chapter.

5.2 WORKING AT THE *CONCEPTUAL LEVEL*

5.2.1 CREATION OF *COMPONENTS*

The first step in creating a GSDM is to instantiate *components* from heterogeneous data. The user's interaction will play a very important role in this step. The whole process of instantiation of a *component* is represented in **Figure 5.2**.

As the input heterogeneous data may contain the information related to segmentations, the user needs to specify how to use the segmented parts in the input data. First, the user has to select the part(s) to be used and specify a name for the *component* to be created. Subsequently, three options are proposed to the user to declare how selected parts should be considered. This possibility is given because it is not guaranteed that the available segmentation corresponds to the actual needs of the user, e.g. over segmented.

Option one is the default option and is to consider all selected parts as a single *component*. If this option is selected, an empty *component* will be created with the specified name associated to its *semantics*. Option two indicates that the decomposition of the segmentation should be preserved, since each of them is meaningful by itself, so separate *components* should be created

in the GSDM, each of them associated to a name corresponding to the part name in the input data. Option three indicates that selected parts should behave as a unique *group*. If the user chooses it, a *group* with a list of *components* will be created. The *group* built in the last option will be associated to the specified name in its *semantics* layer, and each *component* of the *group* will have the same part name in the input data. The details of creating a group will be introduced in the next subsection. There will also be a *semantic* information as “same source” associated to the “reason” for the *semantics* of the *group*.

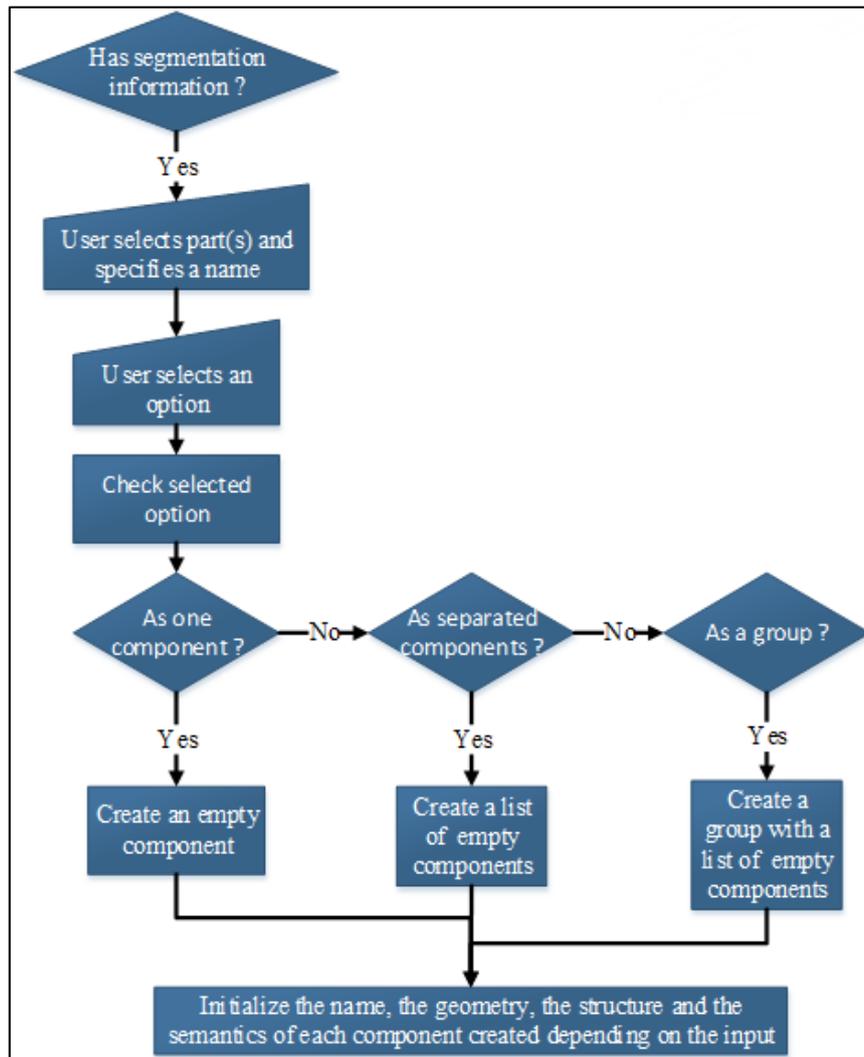


Figure 5.2 Creation of a *component*

Then, for each empty *component* created, the *geometry*, *structure* and *semantics* will be added to the corresponding data structure defined by the GSDM, derived from the information of the related segmented part of the input data. A minimum oriented bounding box is also added as the *structure* representation of this *component*. **Figure 5.3** shows an example of a *component* instantiated with point clouds and a graph-based *structure*.

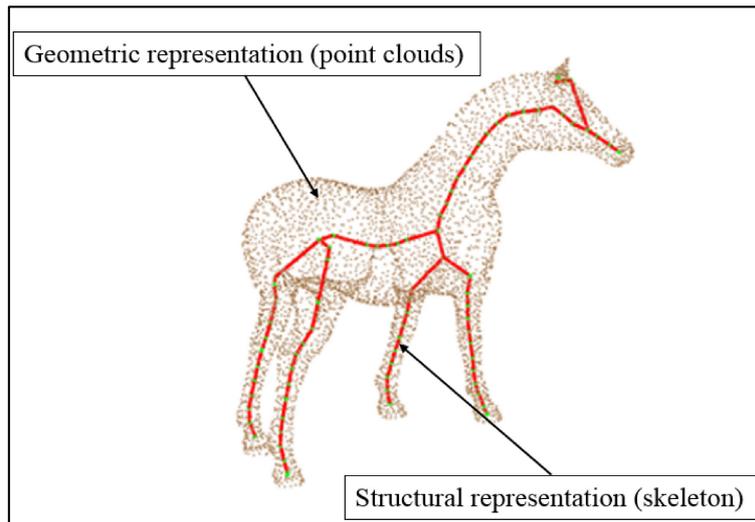


Figure 5.3 Example of a *component*

5.2.2 MAKE GROUPS AND RELATIONS

Once the *components* have been created, the user can associate them in different *groups* and/or specify *relations* between them. The process for making a group is presented in **Figure 5.4**.

To fully instantiate a *group*, we need four kinds of information: the name of the *group*, the reason for grouping, the type of the *group*, the *components* to be grouped and possibly the *groups* to be grouped. In the process of specifying a *component* (see **Subsection 5.2.1**), some information specified by the user for the creation of *component* will be used to automatically instantiate the corresponding *group*. For other situations, the user needs to specify the four kinds of information as presented in **Figure 5.4**. After choosing the *elements* to be grouped, all the existing *groups* are checked to detect if another *group* with the same selected *elements* already exists. If such a *group* already exists, the user will be warned to go back and to reselect the *elements*. Several motives for grouping are proposed as presented previously in the *semantics* of *group* (**Figure 4.7**): “similar shape”, “same color”, “same material”, “same function”, “from the same resource” and one additional choice: “other”. The user can choose one or more reasons, and if there is no suitable reason, then the user can choose “other” and enter a new reason.

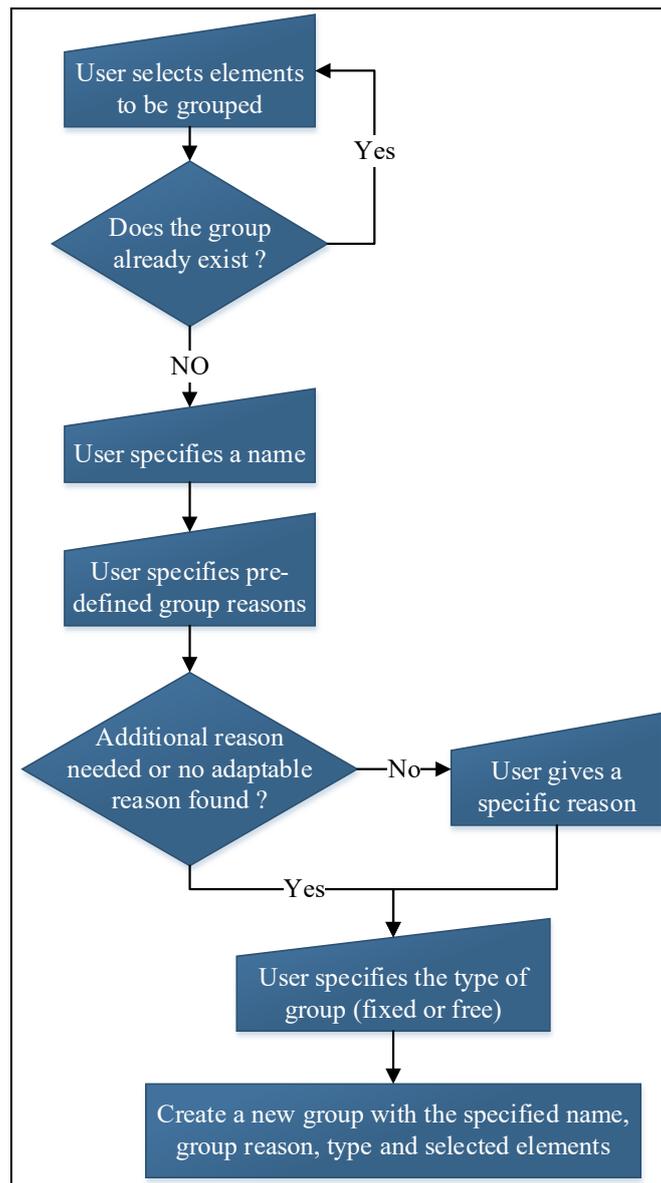


Figure 5.4 Create groups

Similar to the creation of *group*, to instantiate a *relation*, the user also needs to provide some information including the type of *relation* and the associated two *elements* on which the relation is built. However, it is necessary to verify if the selected *two elements* can be associated by a *relation*. There can be four situations occurring when asking for the creation of a new *relation* between two *elements* as shown in **Figure 5.5**.

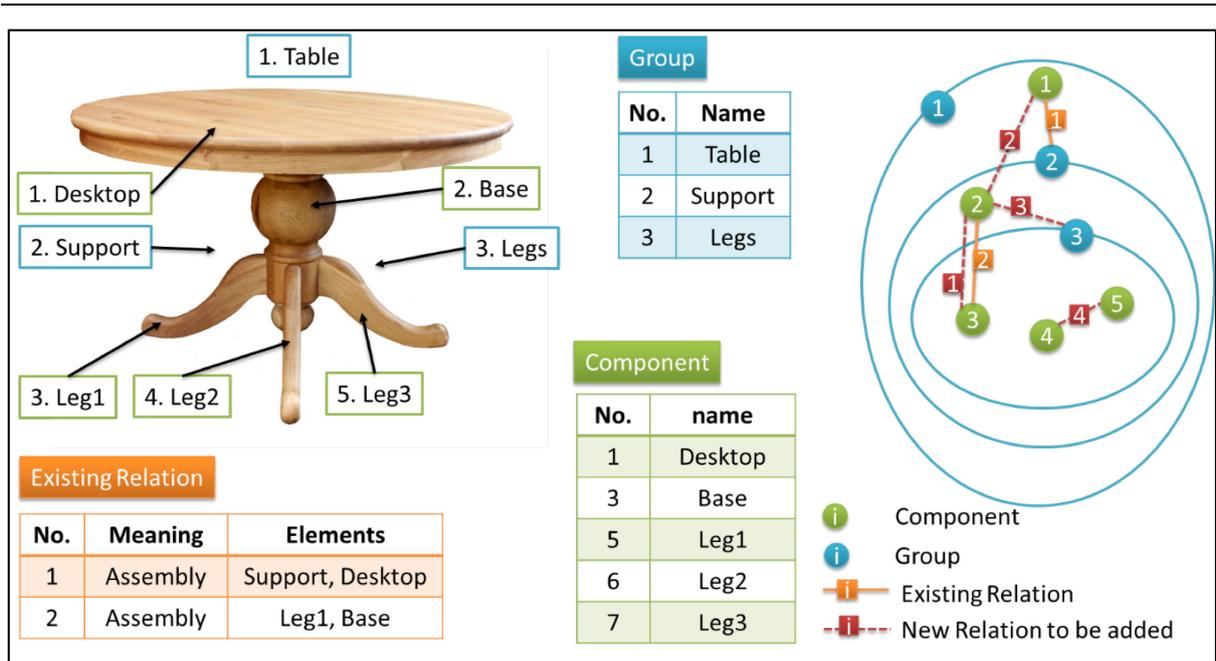


Figure 5.5 Situations for adding new *relation*

1. The selected two *elements* are already connected by an existing relation. In this case, the relation cannot be created and so the user will be asked to reselect the *elements*.
2. There exists a higher level *relation*, i.e. a relation between the groups containing the selected elements. Also in this case the *relation* cannot be created. In the example presented in **Figure 5.5**, there is an existing *relation* of assembly (relation 1) between the “Desktop” (component 1) and the “Support” (group 2). With this *relation*, it is clear that all the elements belonging to the “Desktop” and the “Support” *groups* are assembled together. Therefore there is no need to indicate that “Desktop” and “Base” (component 2) are assembled as “Base” is a part of “Support”. The fact that relation 1 is carried out through the link between “Desktop” and “Base” will be specified by *constraint* not at the *relation* level.
3. There exists a lower level *relation*. It is the opposite case of situation 2. If there is a *relation* between *elements* of the selected *groups*, then adding a new *relation* between the two *groups* means to upgrade the lower *relation* between the included elements to a more general one. For an example in **Figure 5.5**, there is a *relation* of assembly (relation 2) between the “Base” (component 2) and “Leg1” (component 3). If a new *relation* of assembly is to be created between the *group* of “Legs” (group 3) and the “Base”, then the lower level *relation* 2 will disappear while a new *relation* between “Base” and “Legs” will be created. As in the data structure of *constraint* defined in **Subsection 4.4.3**, each *relation* has a list of *constraints*, when a lower *relation* disappears, the related *constraints* will be restructured into the list of *constraints* in a higher *relation*.
4. None of the three above situations. In this case, a new *relation* will be created.

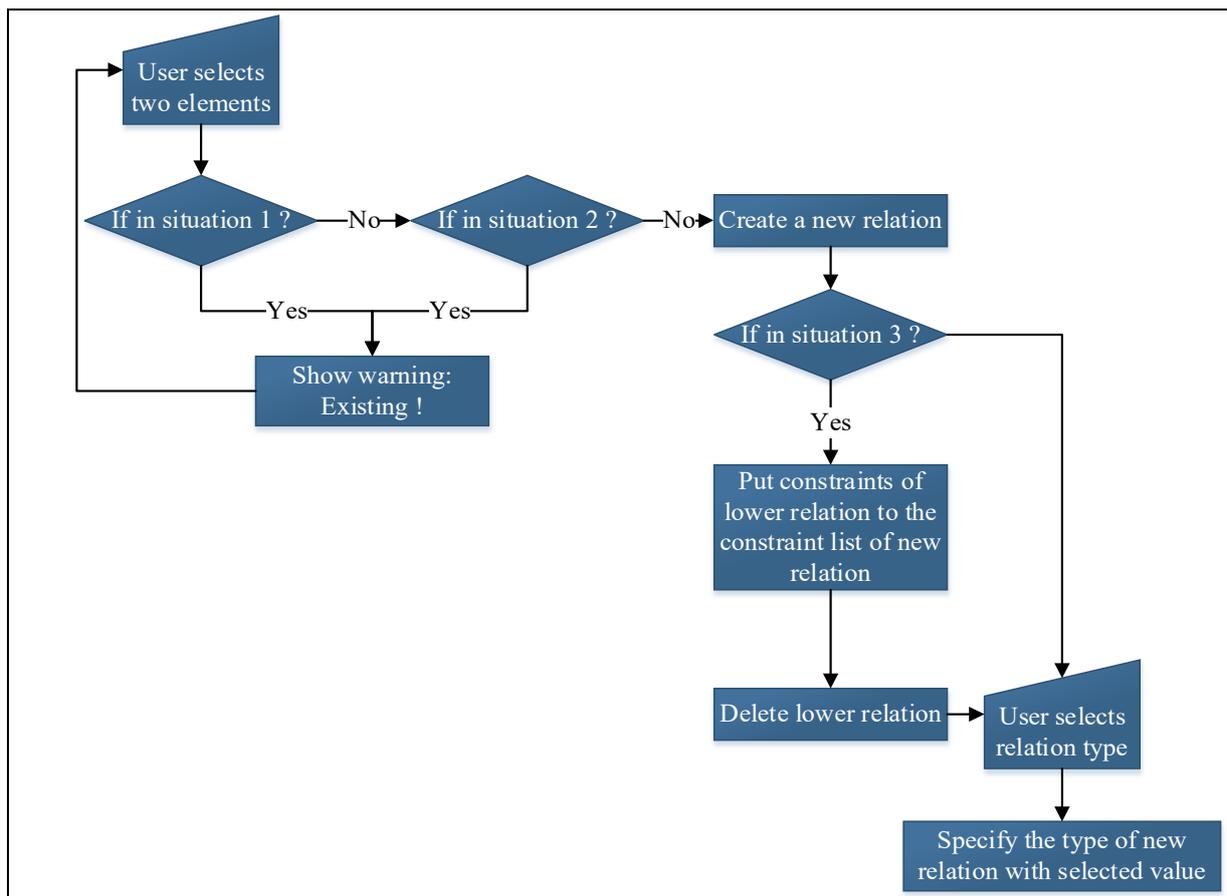


Figure 5.6 Create of a *relation*

The whole process is represented in **Figure 5.6**. In this process, situation 2 and 3 actually explain the “Inheritance” and “uniqueness” properties of relation, which means that there should be only one kind of relation between two *components*, including the inherited *relation* (For details see **Subsection 4.3.3**).

5.3 WORKING WITH THE *INTERMEDIATE LEVEL*

Most of the time a *relation* is first instantiated without specifying the associated *constraints*. The specification of constraints requires first the concerned *key entities* to be specified, then the *constraints* establishing the *relation*. However, to specify the *key entities*, one needs to access the *core level* information, which might be complicated for a non-expert user. Therefore, a “smart” system has been defined to help the user. The system’s “smartness” consists of three capabilities, described in the following:

- automatic identification of *key entities*;
- automatic selection of the type of *constraint* and assign it to the right *relation*;

5.3.1 AUTOMATIC IDENTIFICATION OF KEY ENTITIES

As mentioned at the end of **Section 4.4**, sometimes the user is not really interested in defining the *key entities* between which a *constraint* has to be specified. For instance, in the example of the ball lying on the floor it is more important to consider those two *components* in contact, than to exactly know the points where they are in contact. The ball and the floor are linked by a *relation* of “Location”. For this situation, the user does not need to specify the two *key entities* while instantiating a *constraint*. The system will automatically choose two temporary *key entities* for the *constraint*. They are “temporary” because they do not correspond to specific geometric points in space. Thus, with the specification of other *constraints*, or through modifications performed by the user, they could change. For different *constraints*, the system will apply different approaches to specify the temporary *key entities*. The different strategies are summarized in **Table 5.1**. Concerning the example in **Figure 5.5**, the user can instantiate a “Contact” between “Support” and “Base” without specifying the *key entities*. The system will automatically find the closest points between the “Desktop” and the “Base” and then constrain them together. For the *constraint* “Pattern”, it is necessary to specify the *key entity*. If the specified *relation* is between two *elements*, which are all inside of a *fixed group*, then no *constraints* can be built between them as their reciprocal positions are already constrained. In this case, a warning will alert the user who will be required to reselect *key entities*. The steps to activate this “smart” function are: first select two *components*, then add a *constraint* related to them. If only one *key entity* is specified, then the approaches presented in **Table 5.1** will be used to find the other temporary one. If none of the two *key entities* are specified, then two temporary *key entities* will be instantiated.

<i>Constraints</i>	Approach to find temporary <i>key entities</i>
Coincidence, Distance, Coplanar, Tangent, Contact	E_{FiC} , E_{FoC} or E_{FM} Find the closest points on the two geometric representations of the selected two <i>components</i>
Angle, Parallel, Perpendicular, Coaxial, Insert	E_{LW} Find the longest edge of the structure of <i>component</i> if it has any.
Collinear	E_{LW} Find the longest edge of the structure of <i>component</i> if it has any.

Table 5.1 Automatic computation of default temporary *key entities*

5.3.2 AUTOMATIC SELECTION OF THE TYPE OF *CONSTRAINT* AND ASSIGN IT TO THE RIGHT *RELATION*

A user can also specify a constraint by selecting two *key entities*. Depending on the situations of the specified *key entities*, a new *constraint* will be created through a predefined *constraint* type by default. They are summarized in the following **Table 5.2**. If the predefined *constraint* type is not the desired one, the user can modify it later.

Key entity 1 \ Key entity 2	Point	Line	Oriented point
Point	Coincidence		
Line	Co-linearity	The angle of these two lines is: $\in [45,135]$: Perpendicularity Others: Parallelism	
Oriented point	Co-planarity	The angle between the oriented point and the line: $\in [45,135]$: Tangency Others: Parallelism	Contact

Table 5.2 Predefined constraints type depending on the type of key entities

For the above two kinds of “smartness”, the user directly built a *constraint* by selecting *components* or by selecting *key entities*. Then, the new specified *constraint* needs to be saved in a *constraint* list of the *relation*. How to associate the new generated *constraint* to a *relation* is introduced in the following subsection.

From the two key entities of the new created *constraint*, two related *components* where these *key entities* located should be identified first, then from these *components*, the *relation* to associate should be found. If such a *relation* does not already exist, then the system will not add this *constraint*. The approach of finding the associated *relation* can be detailed as follows:

- Step 1: Identification of the *component* list for each *key entity*: list 1 and list 2. Most of the time one *key entity* is related to only one *component* (*direct parametric key entities*). However, *parametric key entities* can be created by building *relations* between other *key entities* which could possibly belong to different *components*, therefore more than one *component* might be related to an *indirect parametric key entity*. A *geometric key entity* is also related to a *component*, while there might be no geometric or structural representation of this *component* (For e.g. a text).
- Step 2: For each existing *relation* linking two *element* A and B, find in A and B if they separately contain all the *components* in list 1 and list 2 (e.g. *components* in list 1 are all found in A and *components* in list 2 are all found in B, or vice versa).

- Step 3: If such a *relation* is found, then associate this *constraint* with this *relation*.

For example in **Figure 5.5**, if a *constraint* is built between a *key entity* on “Leg 1” and a *key entity* on “Desktop”, then relation 1 will be found and be associated with this new *constraint*. To summarize, the “smartness” is outlined in the following **Figure 5.7**.

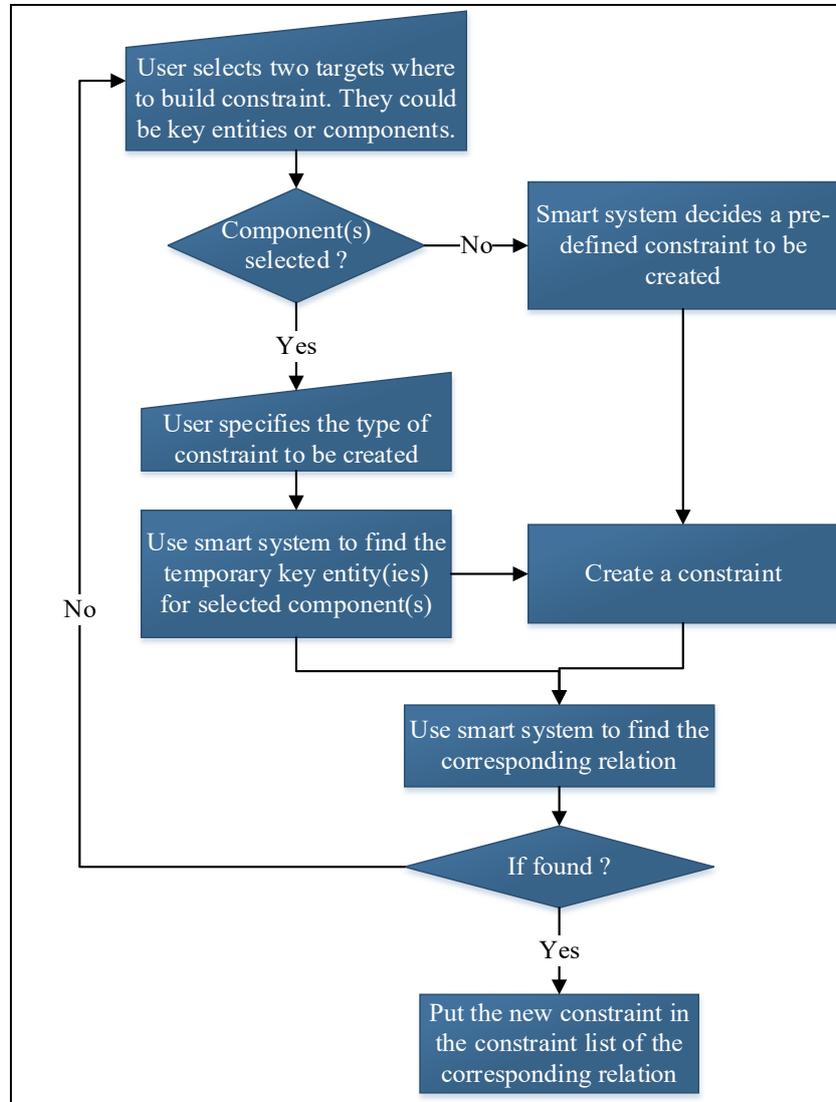


Figure 5.7 Process for instantiating a *constraint*

5.3.3 SPECIFICATION OF *KEY ENTITIES*

The “smartness” of the system consists in the capability of specifying the temporary *key entity* for the user. The user is free anyhow to explicitly specify a *key entity* by following the process presented in **Figure 5.8**. The user can use existing geometric or structural representations to directly specify a *direct parametric key entity*, or create a new one, which is indirectly associated to the geometric or structural representation (*indirect parametric key entity*). The process is quite simple for creating new *indirect parametric key entities* (**Figure 5.8**). First, the user selects the existing *key entities*. When finishing the selection, different types of indirect

parametric key entities will be created. Then the user can indicate other parameters of the new *key entity* if needed.

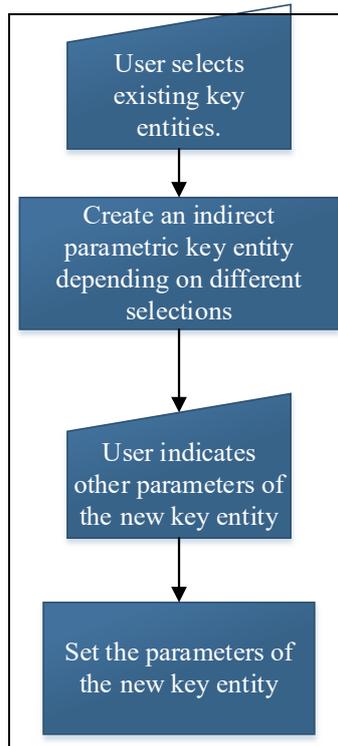


Figure 5.8 Process for create indirect parametric key entity

In this section, some predefined *constraints* and *key entities* have been proposed to help users to quickly specify *constraints* between different *elements*. When the *key entities* and *constraints* have been specified, the crucial issue is to verify if all the constraints can be satisfied. The solution adopted is described in the next section.

5.4 CONSTRAINT VERIFICATION

As presented in **Subsection 4.4.3**, *constraints* are used to specify the related location between key entities. To verify if all the constraints will be satisfied corresponds to solving a Constraint Satisfaction Problem (CSP). During the modeling process with GSDM, the CSP solving starts automatically as the *constraints* is fully specified.

5.4.1 INTRODUCTION TO NUMERICAL OPTIMIZATION

Numerical optimization is a way to solve the CSP, which consists of:

- A set of n variables $X = \{x_1, \dots, x_n\}$ whose values are to be found;
- For each variable x_i , a finite set D_i of possible values (its domain);
- A set of constraints $C = \{c_1, \dots, c_n\}$ limiting the values that variables can take.

Numerical optimization is used to solve the problem of minimizing or maximizing a function $f(x)$ subject to constraints $\Phi(x)$. Here $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is called the objective function and

$\Phi(x)$ is a Boolean-valued formula defined on top of the set of constraints \mathcal{C} .

Global Optimization

A point $\mu \in \mathbb{R}^n$ is said to be a global minimum of f subject to constraints Φ if μ satisfies the constraints and for any point v that satisfies the constraints, $f(\mu) \leq f(v)$.

A value $a \in \mathbb{R}$ is said to be the global minimum value of f subject to constraints Φ if for any point v satisfies the constraints, $a \leq f(v)$.

The global minimum value a exists for any f and Φ . The global minimum value a is attained if there is a point μ such that $\Phi(x)$ is true and $f(\mu) = a$. Such a point μ is necessarily a global minimum.

If f is a continuous function and the set of points satisfying the constraints Φ is compact (closed and bounded) and nonempty, then a global minimum exists. Otherwise, a global minimum may or may not exist.

Local Optimization

A point $\mu \in \mathbb{R}^n$ is said to be a local minimum of f subject to constraints Φ if μ satisfies the constraints and there exists another point v that satisfies the constraints and such that $f(v) \leq f(\mu)$.

A global minimum is always also a local minimum, while the opposite is not guaranteed.

The methods used to solve local and global optimization problems depend on specific problem types. Optimization problems can be categorized according to several criteria. Depending on the type of functions involved there are **linear** and **nonlinear** (polynomial, algebraic, transcendental, etc.) optimization problems. Additionally, optimization algorithms can be divided into numeric and symbolic (exact) algorithms.

Several numeric algorithms can be used to solve the optimization problem. For the linear problem, simplex algorithms, revised simplex algorithms, interior point algorithms etc. can be used. For nonlinear local optimization, the interior point algorithm can also be used. For nonlinear global optimization, Nelder–Mead, differential evolution, simulated annealing and random search can be used. Therefore, to solve the CSP, first we need to check to which type our CSP belongs.

In our case, the variables are the position, orientation and scaling of each *component's* local reference frame in 3D space. Therefore, for each *component* there are nine different variables: the 3D positions, 3D orientations and 3D scales. The variables of each *component* could be described mathematically as below.

$$\text{Rf}_i = (\mathbf{P}_i, \mathbf{R}_i, \mathbf{S}_i) = \left(\begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}, \begin{bmatrix} \alpha_i \\ \beta_i \\ \gamma_i \\ 1 \end{bmatrix}, \begin{bmatrix} a_i \\ b_i \\ c_i \\ 1 \end{bmatrix} \right)$$

The domain for each variable is different. The positions and scales are real values and the orientations should be from $-\pi$ to π degrees:

$$D_i = (\mathbb{R}^4, [-\pi, \pi]^4, \mathbb{R}^4)$$

The *constraints*, as presented in **Subsection 4.4.3**, are the *equations* built on different variables. There are three types of basic functions used to specify the equations including the difference function f_0 , the distance function f_1 and the angles f_2 .

The goal of numerical optimization is to find the best location for each *component* after constraining. To use the *constraint* optimization algorithms for solving our CSP, these three issues need to be considered first:

- All variables of equations should be in the same measure space.
As presented in **Subsection 4.4.1**, all key entities are transformed into the global space, which makes it possible to apply the numerical optimization algorithms.
- Undefined objective function

To find an optimized solution, a meaningful objective function needs to be defined.

5.4.2 SPECIFICATION OF THE OBJECTIVE FUNCTION

In general, there are three types of solutions for a CSP:

- A single solution;
- An infinite set of solutions;
- An optimal solution according to an objective function defined in terms of some or all of the variables.

The numerical optimization will give the third type of solution, which needs to specify an objective function. This choice has been made to be able to access more meaningful solutions compared to traditional CAD systems. As in CAD system if multiple solutions are found, there is only one that will be returned, while this one is meaningless compared with other solutions, possibly just the first solution the system has found.

For choosing the objective function to be used, we considered real life phenomena behaviors. In real life, energy input is required to relocate an object. For example, to move an object from point A to point B, it is always desirable to spend less energy to realize the desired action. In physics, the energy or work (w) spent to move, rotate or deform an object can be described as below:

POSITIONING

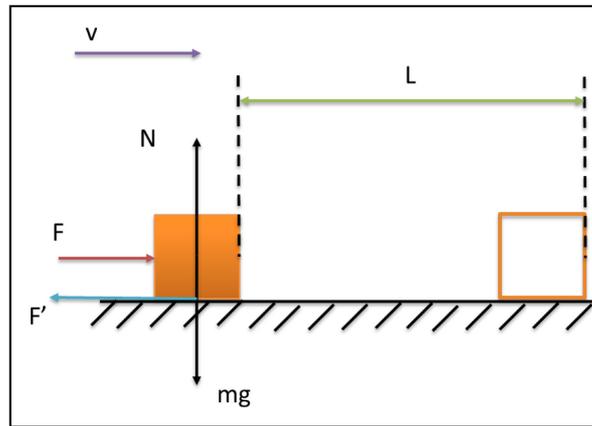


Figure 5.9 Positioning energy

For the simplest situation, as presented in **Figure 5.9**, an object is moved on a floor for a distance of “L” along the same direction as a pushing force “F” with a constant speed “v”. Considering “f” is the coefficient of the resistance between the floor and this object, “N” is the supporting force from the floor (where, $N = mg$). “F'” is the friction from the floor, the work spent for this positioning can be expressed as:

$$w_p = F \cdot L = F' \cdot L = f \cdot m \cdot g \cdot L = f \cdot g \cdot \rho \cdot V \cdot L = \mu_p(V) \cdot L$$

$$\mu_p(V) = f \cdot g \cdot \rho \cdot V$$

Where g is the acceleration of gravity, m is the mass of this object, ρ is the density of the object, V is the volume of the object.

If we suppose that each *component* has the same material and in the same environment (on the same floor), then $\mu_p(V)$ can be considered as a factor of positioning energy which only depends on the volume of this *component*.

ROTATION

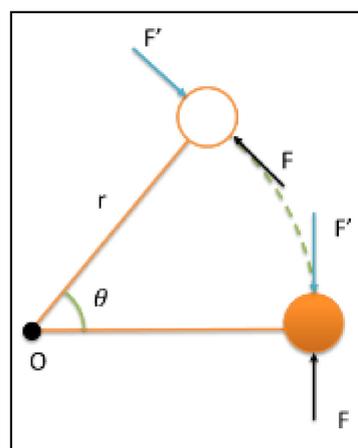


Figure 5.10 Rotation energy (top view)

Similarly, the rotation of an object, as presented in **Figure 5.10**, can be described as rotating an object around a point “O” with a force “F” and a constant speed. “F'” is the friction from

the floor, the work used for this rotation can be expressed then as:

$$w_r = F' \cdot r \cdot \theta = f \cdot m \cdot g \cdot r \cdot \theta = f \cdot g \cdot \rho \cdot V \cdot r \cdot \theta = \mu_r(V) \cdot \theta$$

Where, r is the radius for the rotated arc (the distance between the mass center and the rotation center) and θ is the rotated angle, g is the acceleration of gravity, m is the mass of this object, ρ is the density of the object, V is the volume of the object, “ f ” is the coefficient of the resistance between the floor and this object.

If we consider that each *component* has the same material, in the same environment - a rotation with the same rotation radius- then $\mu_r(V)$ can be considered as a factor of rotation energy which only depends on the volume of this *component*.

SCALING

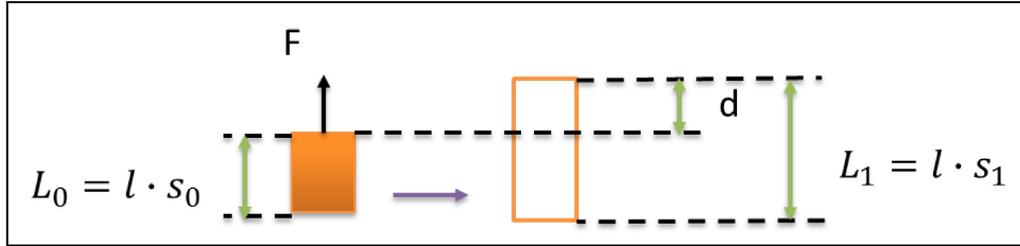


Figure 5.11 Scaling energy

As the scale property of a component is also considered as a variable for CSP solving, therefore each component can be considered as a spring (**Figure 5.11**). According to Hooke’s law, the scaling energy can be expressed as:

$$\begin{aligned} w_s &= \frac{1}{2} \cdot k \cdot d^2 = \frac{1}{2} k(L_1 - L_0)^2 = \frac{1}{2} k(l \cdot s_1 - l \cdot s_0)^2 = \frac{1}{2} k \cdot l^2 \cdot \Delta s^2 \\ &\approx \frac{1}{2} k \cdot V \cdot \Delta s^2 = \mu_s(V) \cdot \Delta s^2 \end{aligned}$$

Where “ F ” is the force used to scale the spring, “ k ” is the coefficient (Young’s modules) of the elastic deformation. “ d ” is the scaled distance along the scaling direction. “ l ” is the initial length of this object along the direction of “ F ”. “ s_0 ” and “ s_1 ” are the scale factors before and after scaling.

If we consider l^2 is approximately equal to the volume (V) of the object, then $\mu_s(V)$ can be considered as a factor of scaling energy, which only depends on the volume of this *component*.

As a result of these energies, an objective function has been suggested, which reports the energy used to realize the relocation of *components* after constraint satisfaction.

The objective function is defined as below:

$$F = W_p + W_r + W_s$$

Where,

$$\begin{aligned} W_p &= \sum_{i=1}^{i=n} \mu p_i \| \mathbf{P}_i^{k+1} - \mathbf{P}_i^k \| \\ W_r &= \sum_{i=1}^{i=n} \mu r_i \| \mathbf{R}_i^{k+1} - \mathbf{R}_i^k \| \end{aligned}$$

$$W_s = \sum_{i=1}^{i=n} \mu s_i \|\mathbf{S}_i^{k+1} - \mathbf{S}_i^k\|^2$$

$n \in \mathbb{N}$ is the number of *components*

W_p is the positioning energy of all the *components* from the previous position \mathbf{P}_i^k to the final position \mathbf{P}_i^{k+1} after constraining. The longer the distance between the previous and the final positions of this *component*, the larger the positioning energy. μp_i is the positioning energy factor for each *component*. Inspired from the physical energy of movement presented previously, by default it is equal to the volume of the OBB of the *component* (for an image, instead of using volume, surface is considered), which means that if the *component* is bigger, then more energy is needed to reposition it. The user can specify a desired value of this factor for each *component*.

W_r is the orientation energy of all the *components* from the previous orientation \mathbf{R}_i^k to the final one \mathbf{R}_i^{k+1} . Similar to the positioning energy, the larger the angle between the two orientations is, the more energy is needed. μr_i is the orientation factor for each *component* which also equals the volume of the OBB of the *component* (surface if it is an image), and can also be modified.

W_s is the scaling energy of all the *components* from the previous scale \mathbf{S}_i^k to the final scale \mathbf{S}_i^{k+1} of the *component*. μs_i is the scale factor which is related to the the volume of the OBB of the *component* (surface if it is an image).

Our objective is to minimize the sum of these three energies. If we do not want one specified *component* i to change its position too much, we can set its μp_i as a very large value so that this component will move a small distance to limit the required positioning energy. In this sense, we build a link between the relocations of each *component* with a semantic meaning of the intention for the relocation. In other words, our CSP is more meaningful compared with typical CAD CSP. We can use different factors for different *components*; we can also use a global factor for all *components*. The energy factors actually limit the flexibilities of positioning, rotating and scaling each *component*.

Different algorithms can be used to solve this CSP as presented briefly in **Subsection 5.4.1**. In our case, the constraint equations are mostly nonlinear (except for coincidence) and the objective function is also nonlinear. As the development of the algorithm for CSP solving is not the main objective of this thesis, it was decided to use existing tools to solve the CSP. The chosen tool and some related applications will be presented in the next chapter.

To summarize, the whole process of CSP specification can be represented as in **Figure 5.12**. If the alignment of a component's transform is not set to "Free" as introduced in Chapter 4, then the rotation of this component will be fixed to a predefined value. They are not going to be considered as unknowns for the CSP solving process. It saves a lot of time.

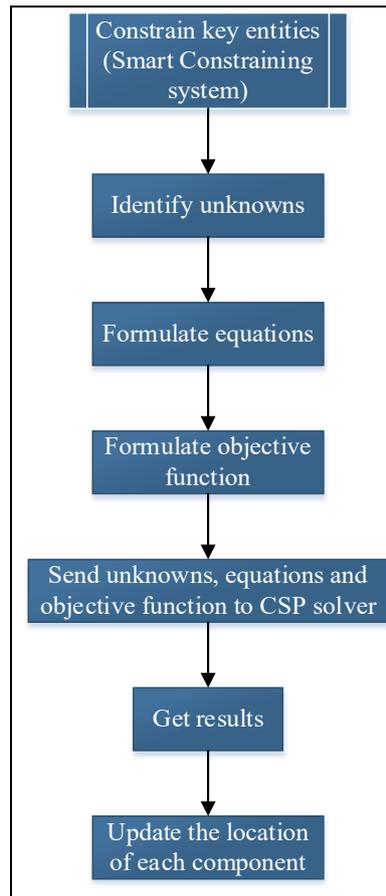


Figure 5.12 Process for CSP solving

5.5 CONCLUSION AND REMARKS

In this chapter, the top-down user-oriented and GSDM-based modeling process have been presented. All the processes designed during this PhD thesis and presented in this chapter aim at helping a non-expert user to quickly describe a conceptual model. Starting from the instantiation of the *conceptual level*, the user can assign heterogeneous data to the indicated constituents. With the *conceptual level*, the user can get a fast feedback of what to use and how heterogeneous input components are linked to each other to form an object. To complete the arrangement specification of conceptual model constituents, some semi-automatically defined *key entities* and *constraints* are proposed to help the user. Once the *constraints* have been specified, the CSP solving process starts to find an optimized solution to satisfy all the *constraints*. This process not only helps the user to set the related locations of each *component*, it also ensures that the information stored in the GSDM is correct, so that it can be used for future design phases.

Compared to traditional CAD modeling processes, the proposed approach is more meaningful and the scale factors are also considered as unknown values to be constrained, thus allowing an automatic adaptation of the *component* size according to the specified *constraints*.

The process of modeling with GSDM presented in this chapter has been implemented in a demonstrator, with a user-friendly interface, which will be described in the next chapter.

I've always believed that if you put in the work, the results will come.

Michael Jordan

6.

IMPLEMENTATION AND RESULTS

CHAPTER OVERVIEW

THIS CHAPTER IS organized in four sections which aim at introducing the adopted implementation environment, some tests related to CSP solving, the implemented modelling framework and the validation of the proposed approach through different examples.

First of all, in **Section 6.1**, the implementation needs and requirements are presented together with the adopted software and hardware environments. Then, an overview of the whole implemented framework is proposed in **Section 6.2**. **Section 6.3** depicts three examples created using the proposed implemented approach. Finally, some remarks and conclusions about the implementation are discussed in **Section 6.4**.

6.1 THE ADOPTED DEVELOPMENT ENVIRONMENT

To better demonstrate the capabilities and features of the modelling approach based on the GSDM, it is important to select the most appropriate development environment, i.e. both software and hardware. To take into account all the features and capabilities presented in **Chapters 4 and 5**, the implementation should enable the visualization, some user interactions as well as the creation and manipulation of the GSDM. The following **Table 6.1** summarizes the requirements and facilities that the development environment has to possess.

Operational requirements		Functional requirements of the environment
Visualization of GSDM	Visualization of the eight notions of GSDM	3D/2D rendering
	Visualization of the Graph view	2D rendering
	Visualization of the Constraint tree	2D rendering
Modeling of GSDM	Import external heterogeneous data	IO (Input and Output) operation
	Manipulation of <i>Components</i> and <i>Groups</i>	3D rendering 2D rendering
	Building <i>Relation</i>	Mouse and Keyboard interface
	Manipulation of <i>Key Entities</i> and <i>Constraints</i>	Touchable Interface
	CSP solving	Mathematical calculation engine
	<i>Operation</i> related to <i>Relation</i>	Capacity of accessing/modifying the <i>core level</i> information
Controllers (graphic user interface)	Menus, buttons and other controllers	2D Rendering Mouse and Keyboard interface Touchable Interface

Table 6.1 The operational and functional requirements for the development environment
Based on the MVC architectural pattern⁵², the implementation requirements are structured

⁵² Model, View and Control.

About the implementation architecture: <http://martinfowler.com/eaDev/uiArchs.html>

into three 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 CPS 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*, etc.). GUI controllers are mainly for developing easy and friendly interfaces for non-expert users. The three modules work together to create an object GSDM representation from heterogeneous data gathered together by a non-expert user. With these requirements, the environment should provide Input and Output facilities, 2D/3D rendering, mouse/keyboard/touchable interaction, mathematical calculation and access to/modification of the *core level* information.

In **Section 2.5**, the existing environments for the development of VE have been listed. As a result, the Windows 7 operating system has been chosen, with Unity 3D installed to realize IO operations, 2D/3D rendering, mouse/keyboard/touchable interactions and access/modification of the *core level* information. A plug-in from Mathematica 10.0.1 has also been chosen to perform the CSP solving. Regarding hardware, a PC (with mouse and keyboard) with a multi-touch screen has been used. The selected development environment is outlined in **Table 6.2**.

Capacities of the environment	Chosen software environment	Chosen hardware environment
IO operation	Unity 3D v4 (C#) Windows 7	PC (with mouse and key- board) Multi-touch screen
2D Rendering		
3D Rendering		
Mouse/keyboard/touchable interaction		
Accessing/Modifying the core level information		
Mathematical calculation	Mathematica 9.0.1.0 .NET/Link	

Table 6.2 Adopted development environment

As shown in Chapter 3, Unity 3D is a powerful tool to develop games or VR applications. The reasons why it has been chosen for this work are summarized below:

- Unity 3D has its own high level APIs (Application Programming Interfaces). These APIs have a set of high-level classes and functions that allow a developer to create 2D and 3D graphics without considering the low-level rendering process (such as the structure of a mesh or dealing with the graphic card).

-
- If desired, the developer can also access the low level information such as the “*core level*” information of the GSDM through these APIs. For example, he/she can get the coordinates of each vertex of a mesh and modify them.
 - Unity 3D also has a GUI system helping the developer to create 2D and 3D menus, buttons and other controllers. This GUI system works both for the mouse/keyboard mode and the multi-touch mode.
 - Moreover, other libraries can also be plugged into these APIs to call external functions such as the Mathematica .NET/Link that is used to solve the CSP.

For the mathematical calculation engine, the Mathematica .NET/Link has been chosen. Mathematica is a computational software which includes a whole package for constraint optimization. It works for both local and global optimizations, as well as for linear and nonlinear optimizations. Mathematica .NET/Link is a product that integrates the Mathematica engine and Microsoft .NET platform. It lets us call .NET from the Mathematica language in a completely transparent way, it also allows us to use and control the Mathematica kernel from a .NET program. In our case, it is used to call Mathematica kernel for solving the CSP from the scripts developed in Unity 3D. With this external library, it is only necessary to formalize the CSP in the Mathematica language then send it to the Mathematica kernel. Then the CSP will be solved without entering the details of each algorithm. This external library is not only interesting for solving CSP, but it also includes many other interesting packages. For sure, future developments of GSDM will require solving other kinds of mathematical problems that could make use of other packages included in this library (for example some operations related to the *relation*).

6.2 OVERVIEW OF THE IMPLEMENTATION

In this section, we provide an overview of the whole implemented framework for the GSDM modeling. The input of this implemented framework is a set of 2D images or 3D meshes together with their structural information. The output is a conceptual model defined by the GSDM.

6.2.1 OVERVIEW OF THE USER INTERFACE

The user interface of the implemented tool has mainly two areas consisting of a 3D viewer and a control panel as shown in **Figure 6.1**. It is designed to have as few buttons as possible and be “easy-to-use”, with the objective of being a tool for non-expert users.

The 3D viewer is the main workspace to select, manipulate and modify the different notions of the GSDM such as *component*, *group*, *relation*, etc. In the 3D viewer, there is a gray 3D plane which is fixed with the global reference frame that cannot be moved, rotated nor scaled.

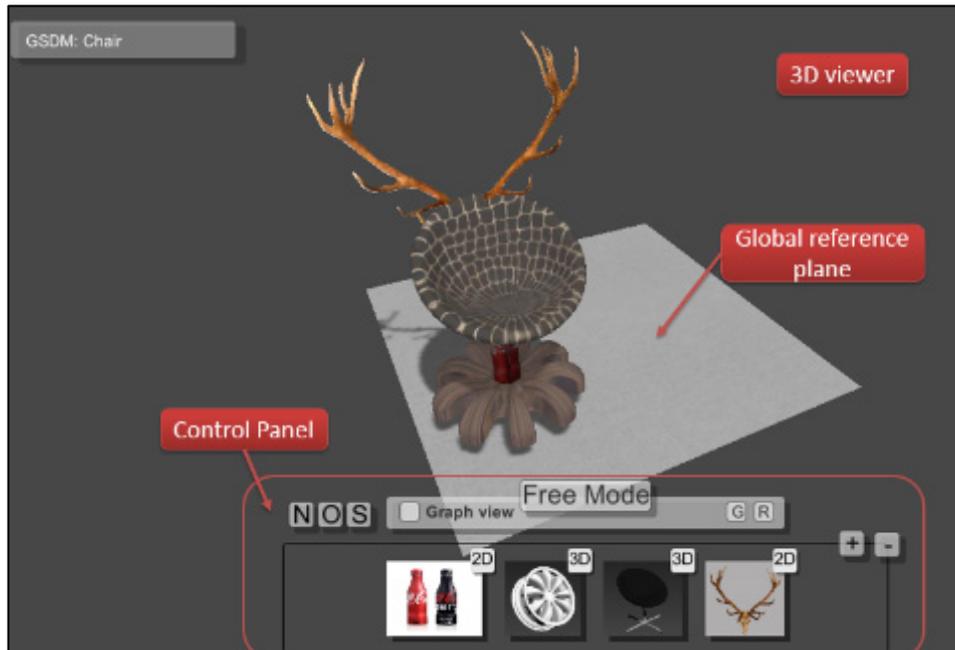


Figure 6.1 Overview of the user interface

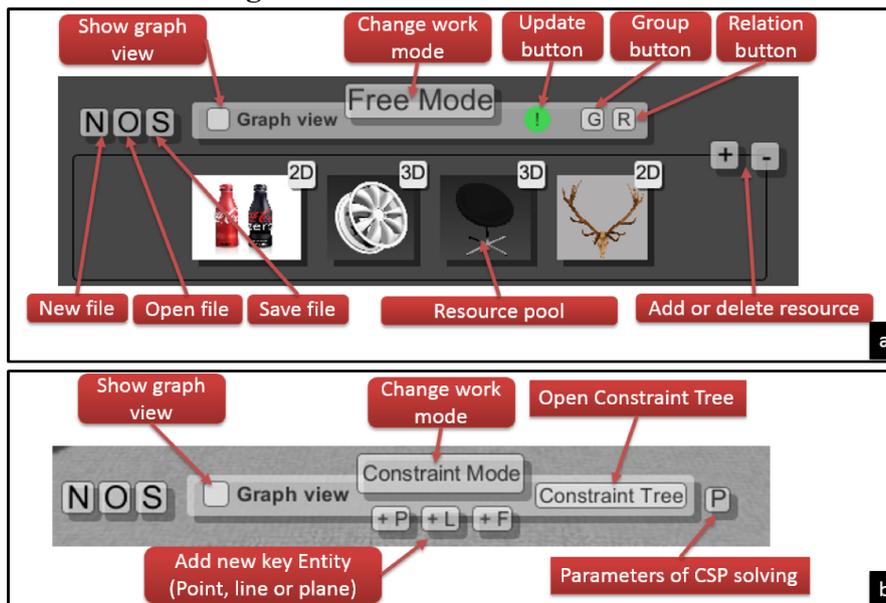


Figure 6.2 Control panel and two work modes. (a) Free mode and (b) Constraint mode

The control panel includes the main controllers which execute the complex functions of the GSDM. The user can choose between two work modes. One is called “Free mode”, which is conceived for the manipulation of the *conceptual level* of the GSDM (as presented in **Figure 6.2.a**). The other is called “Constraint mode”, which is designed for working on the *intermediate level* (as presented in **Figure 6.2.b**). There is a mode switch button to change from one mode to another. In the bottom left hand corner, there are three buttons for creating a new file (N), opening a file (O) or saving a current file (S). The show graph view button is a check box for showing a graph view of the GSDM.

In the “Free mode” (see **Figure 6.2.a**), there is a resource pool below the mode switch

button represented by a scroll area showing all the heterogeneous inputs that have been imported into the workspace. The user can select and use “drag and drop” to add an input from the resource pool to the 3D scene. On the top right of the resource pool, there are two buttons for adding or deleting a resource. There is also an update button (see **Figure 6.2**) for manually calling the CSP solver. When clicking on this green dot, the mathematical engine is called to solve the CSP of the GSDM. The “Group button” and “Relation button” are for creating *groups* and building *relations*.

For the “Constraint mode” (see **Figure 6.2.b**), there are three buttons for adding new indirect parametric *key entities*. There is another button called “Constraint tree” to visualize a 2D tree structure of the *relations* and *constraints*. The last button on the right with the label “P” is for configuring the global energy parameters of the CSP solving.

6.2.2 VISUALIZATION OF HETEROGENEOUS DATA

One important objective in using the GSDM is to reuse existing heterogeneous data. Therefore, it is important to be able to visualize those heterogeneous data. This thesis mainly considers three types of these data: 3D meshes, 2D images and texts. It was decided to visualize them all together in a 3D space so as to be able to specify directly the *relations* between them.

- Solution for 3D Meshes: Unity 3D native 3D rendering system.

For 3D mesh data, we are using the Unity 3D native rendering system. Unity 3D can import a 3D mesh from the most popular applications such as Maya, 3ds Max, Modo, Cinema 4D, Blender or Autodesk FBX. A list of supported file formats is shown on the web site of Unity 3D⁵³. The 3D mesh is visualized in a 3D scene (e.g. in **Figure 6.3**). As mentioned in Chapter 6, if there is the segmentation information of the input data, then it can help the user to choose the different parts that he wants to use. This information also needs to be visualized. This implementation presents the different segmented parts in different colors.

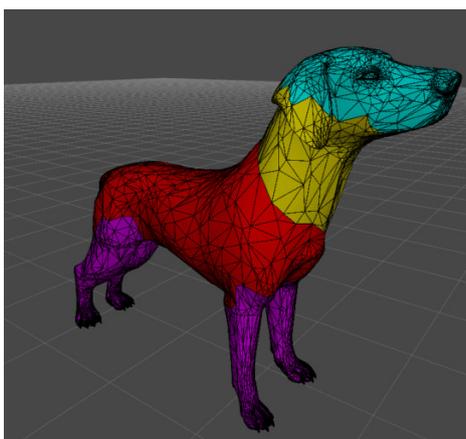


Figure 6.3 Visualization of a segmented 3D mesh

⁵³ <http://unity3d.com/unity/workflow/asset-workflow>

-
- Solution for 2D images: Contour representation in a mesh structure and texture mapping.

For a 2D image, it has been decided to use the contour information and then generate a mesh from it (using a triangulation algorithm). Here it is supposed that the input data already contain this information. This can be implemented as a plugin in future version. Therefore, a 2D image is represented by a planar mesh that can be visualized by the Unity 3D native rendering system. This planar mesh is located in a 3D scene. It can be moved, rotated or scaled in all three dimensions (e.g. in **Figure 6.4**). If there is the segmentation information of the image, the different parts will be represented by different meshes easy to choose for the user. The planar mesh is then textured using information included in the 2D image.



Figure 6.4 Visualization of a 2D image: the right picture (b) shows the head bones of a deer skull from the left picture (a) represented by a planar mesh with texture in a 3D space

- Solution for texts: Texture with alpha channel of a rectangular mesh.

For textual data, we decided to render them with a texture applied on a rectangular mesh. Each letter of the text is rendered as one rectangle. All letters are laid on the same plane as the word “Spout” represented in **Figure 6.5**. Text is also represented in a 3D space so that 3D transformations can be applied.

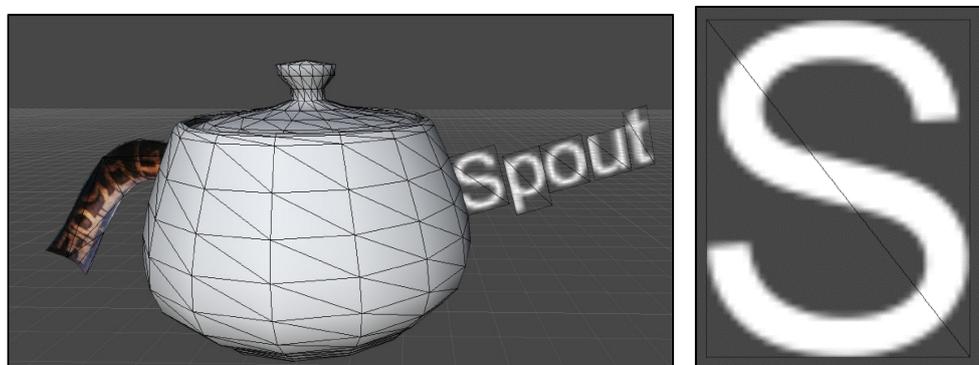


Figure 6.5 Visualization of textual information

Finally, although heterogeneous data have different data structures, they are all restructured into a mesh in the same 3D scene of Unity3D. Those transformed structures are only used for the visualization of the GSDM. Of course, their native structures do not change when transforming them into a visualized mesh. But, this makes it easier for the user to visualize and then integrate different data in the same environment. In our implementation, the link between the representation of the heterogeneous data and the original data is also kept. Modifications of representation in Unity 3D do not affect the original data but are saved in the GSDM data structure.

- Solution for the graph-based structure: wireframe.

A wireframe is used to represent the graph-based structure of a *component* such as a Reeb graph, a skeleton, etc. The following **Figure 6.6** shows an example of a graph-based structure of a *component*. In this example, the pink dots represent the nodes of this structure and the pink segments between two dots represent the edges of this structure. At a later stage, when specifying the *parametric key entities*, the user can choose the nodes and the edges of the structure.

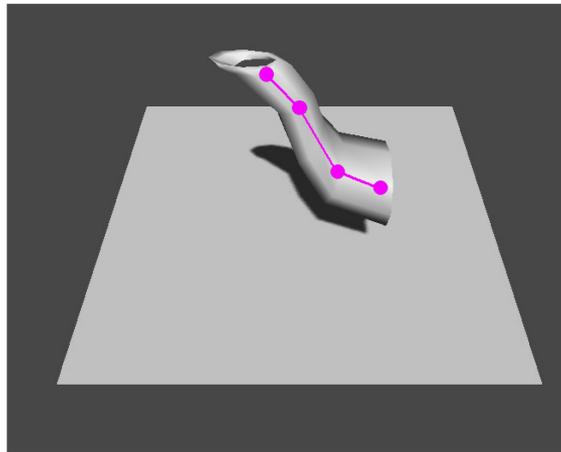


Figure 6.6 Visualization of the structure of a component

6.2.3 MANIPULATION OF A COMPONENT (OR GROUP)

Today, due to the development of VE technologies, it is not a problem anymore to manipulate a 3D object in a 3D scene. There are several operations that can be classified in three types: positioning, rotating and scaling. Besides manipulating a 3D object, the viewer allows one to visualize an object or a scene from different points of view. In today's professional 3D design software, positioning, rotation and scaling are usually realized by different ways of interaction.

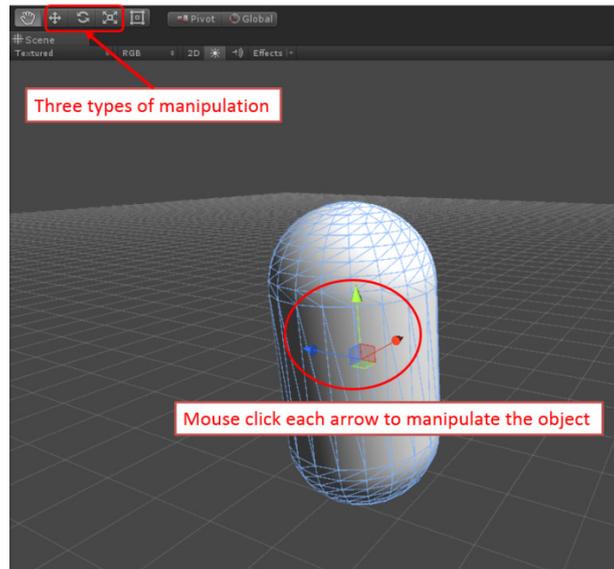


Figure 6.7 Manipulation modes in Unity 3D

One solution is to set one or multiple buttons to change between the three types of manipulation. As an example in Unity 3D⁵⁴ (see **Figure 6.7**), on top left, there are three buttons to select the type of manipulation the user wants to do and at the center of the selected object there is a handler with three arrows for the user to choose. When selecting one arrow and dragging the mouse, the user manipulates the object in the direction specified by the chosen arrow. Another solution is to specify the three different types by different ways of interaction, without using buttons to change between the three types. This is usually used to handle the camera of the viewer. For example in 3ds Max⁵⁵, to reposition the camera, the user needs to press the wheel of the mouse and drag it. To rotate the camera, the user needs to hold the “Alt” key on the keyboard and press the wheel of the mouse and drag it. To zoom the camera, the user has to scroll the wheel of the mouse. These two solutions show that handling an object or the viewer in a 3D scene is not very friendly in professional 3D design software in the sense that the user needs to change to different manipulation types or interaction modes to realize a sequence of actions so as to finally relocate an object in a desired location. This is because each type of manipulation is in 3D, while the mouse is a 2D input. Besides, these two solutions are designed for a professional user, so the manipulation needs to be accurate. The manipulation methods used in professional software do not seem suitable for a non-expert user. For example, the three arrows shown in **Figure 6.7** may confuse a non-expert user.

Therefore, we defined a new way to manipulate objects and the viewer in a 3D scene, using only drag and drop. This “drag and drop” interaction mode can be realized by both a mouse and a touch screen so as to be suitable for a non-expert user. The example of **Figure 6.8** explains this new method. First, when an *element* is selected, a round spot appears in the center

⁵⁴ Manipulation of object in Unity 3D: <http://docs.unity3d.com/Manual/PositioningGameObjects.html>

⁵⁵ Manipulation of Scene in 3ds Max: <http://help.autodesk.com/view/3DSMAX/2015/ENU/?guid=GUID-D65DE5FD-F859-4F66-9E14-F9A5C1016411>

of this selected *element* with an index number in the center. This round spot is called the selection handler. It is divided into three zones and each of them is a sector of 120 degrees. When the mouse moves over one of the three zones, it is lit with a specified color and a letter in the middle of this spot: orange and the letter “M” is for moving/positioning; blue and the letter “R” is for rotating and purple and “S” is for scaling. At that moment, if the user presses the selection handler (or touches the screen) to realize a drag action, then different types of operations will be carried out, depending on which zone is pressed. The dragged direction and distance will be used to calculate the values of how far it (the handler) has to be moved, rotated or scaled. When the user drops the handler, this action will be finished. A window (called “window of Selected Object”) in the top right hand corner of the 3D viewer shows the exact values of the position, rotation, scale and the semantic information about the selected *element*.

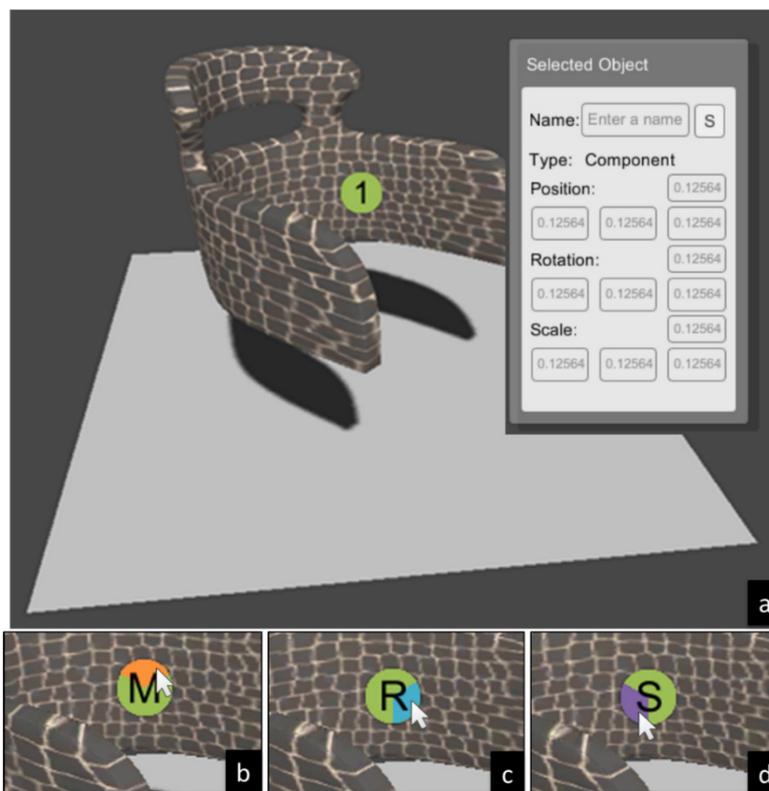


Figure 6.8 Manipulation of a *Component* (or *Group*): (a): when a *Component* is selected. (b), (c), (d): when the mouse moves over different zones of the selection handle.

- Positioning: Press move zone then drag mouse

For positioning an element, a smart positioning system has been designed. As mentioned at the beginning of this section with the example of **Figure 6.7**, repositioning a 3D object in professional software requires the user to specify in which direction or on which plane the positioning is applied. This is because the dimension of the action that the user can apply to a 3D object (3D as position in three directions) and the dimension of the interaction (2D as “drag and drop” on a screen is 2D) differ. In other words, no matter what kind of 3D manipulation the user is applying, they are all in fact 2D manipulations by the handler on the 2D screen with a 2D input such as a mouse or a touch screen. This 2D input can only offer two variables (the

vertical and horizontal position of the handler), whereas the manipulation of a 3D object in a 3D scene requires three variables. Therefore, in Unity 3D or in other 3D modeling software, the user needs to specify in which direction (1D) or on which plane (2D) this repositioning is applied. The repositioning along one direction requires only one variable and the positioning on a plane requires two variables, which it is possible to get by means of 2D input. It is the same situation for our implementation as we decided to use mouse and touch screen as input devices. Therefore, it is also essential to specify in which direction or on which plane the repositioning has been applied.

In our implementation, this decision is automatically made by the system. From the users' point of view, if they want to reposition an object on a plane, naturally, they would prefer to turn the viewer to face this plane so that the movement of the object can be clearly seen. Based on this consideration, we exploit the direction corresponding to the one of the camera view to apply for the repositioning. For example, if the user wants to reposition an object on a plane (the orange plane in **Figure 6.9.a**), which is parallel to the global reference plane (the gray plane as presented in **Figure 6.9**), then it is better to have a top view. Thus, our implementation will automatically specify a plane for the user to position the selected object.

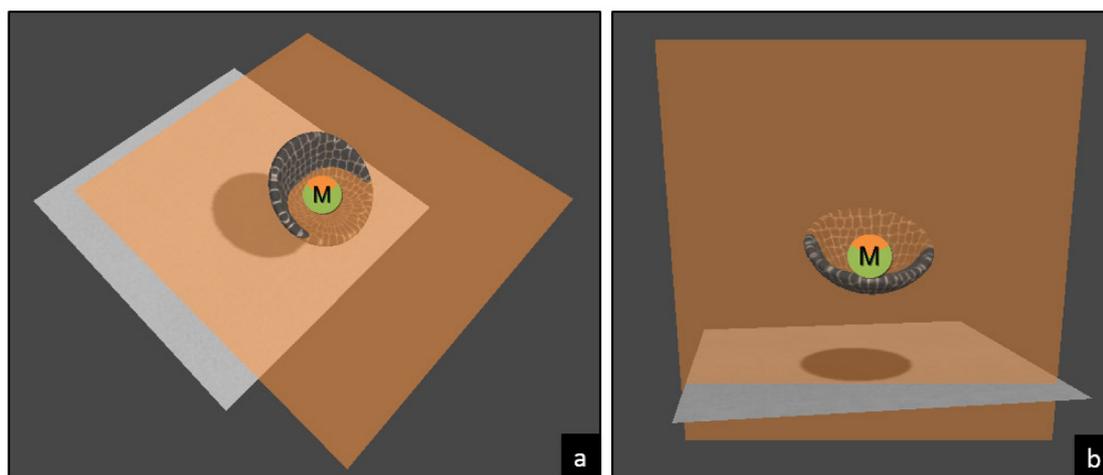


Figure 6.9 Positioning of a component. (a): positioning on a plane parallel to the camera, in a top view. (b): positioning on a perpendicular plane in a side view

There are two possibilities for automatically specifying planes. One is parallel to the global reference plane crossing the pivot point of the selected element. The other one is perpendicular to the global reference plane crossing the pivot point of the selected *element* and facing the user. The specified plane will be highlighted by an orange and transparent color as in **Figure 6.9**. When the user rotates the 3D viewer close to a top view of the global reference plane (**Figure 6.9.a**) and he/she moves the mouse pointer over the “position” zone of the selection handler, the specified positioning plane will be the one which is parallel to the global reference plane. When the viewer comes close to the side view of the global reference plane (**Figure 6.9.b**), the specified positioning plane will be perpendicular to the global reference plane facing the user. Deciding whether the camera will give a top view or a side view depends on the angle between the normal position of the global reference plane and the direction of the viewer's camera. When

the angle is between 60 and 120 degrees it is considered as a side view, otherwise it is a top view. The scheme in **Figure 6.10** shows this angle. The blue plane represents the 3D viewer of the camera which is also what the user sees on the screen. The two red arrows represent the direction of the viewer's camera and the direction of the global reference plane.

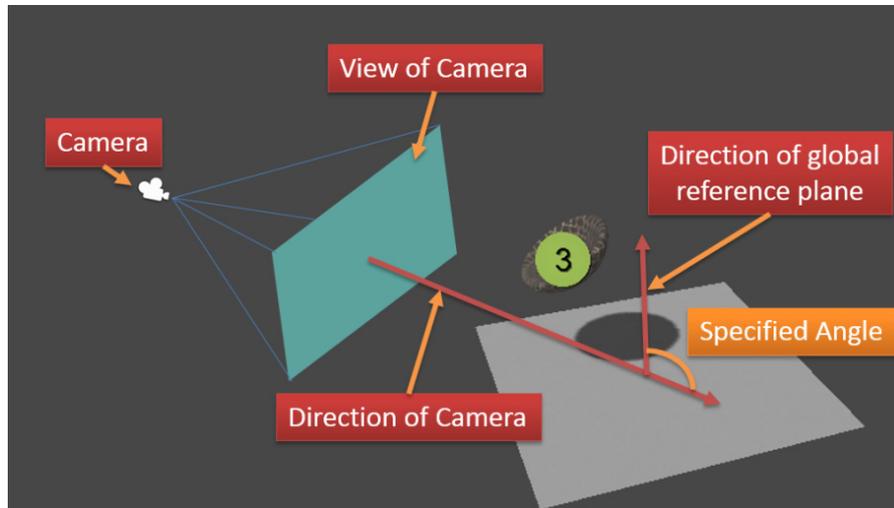


Figure 6.10 The angle between the direction of the viewer's camera and the direction of the global reference plane

- Rotation: Press rotate zone then drag mouse

The rotations are designed to be always along a vertical axis and a horizontal axis around the pivot point of the selected *element* depending on the dragging movement. A horizontal dragging movement will affect the rotation along the vertical axis and a vertical dragging movement will affect the rotation along the horizontal axis. An example of the two rotation axes is presented in **Figure 6.11**. The “horizontal” and “vertical” directions are defined in the viewer's space.

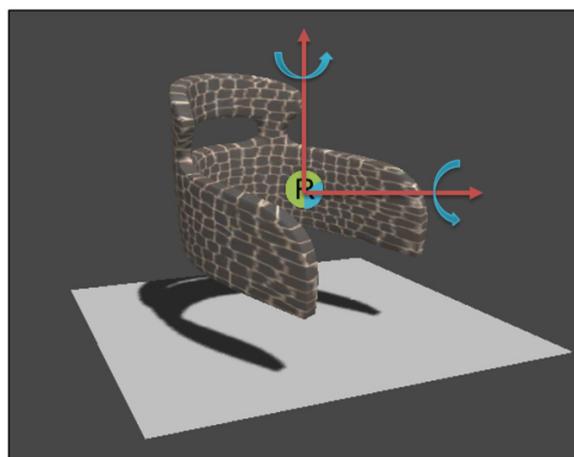


Figure 6.11 Rotation axes

- Scaling: Press scale zone then drag mouse

The scaling action will scale the selected *element* from its pivot in all three directions or

only in selected directions so as to generate heterogeneous scaling effects. The scaling value depends on the distance of the dragging movement from the dragging start point.

All the three types of manipulation are carried out by dragging the selection handler. In this way, the user does not need to change interaction mode or click other buttons. However, all these manipulations are applied from a specific point of view of the 3D scene. To change the point of view, we also need a manipulation of the viewer (in other words, the viewer's camera) to implement a manipulation in the third dimension. The viewer is always focusing on the center of the 3D scene. There is no scaling action for the camera, but only positioning and rotating. When the user starts a drag and drop action without pressing the selection handler, this will move the camera. The dragging action rotates the camera along the vertical and horizontal axes depending on the dragging movement, similar to the rotation of a *component*. The scroll of the mouse wheel (zoom gesture for the touch mode, e.g. two open fingers) will reposition the camera forward or backward to create the zoom effect of the viewer.

To simplify the user's interaction, the user can also align a component to face eight global directions including (forward, backward, left, right, top and down). When a component is selected in "Free" mode then, a click on the corresponding button in the alignment tool bar (**Figure 6.12**) will set the selected component in the corresponding direction. If a component is aligned in a global direction, then its rotation is fixed. They (its rotation) are not going to be considered as unknown values for the CSP solving processing, which would increase the computation time for CSP solving.

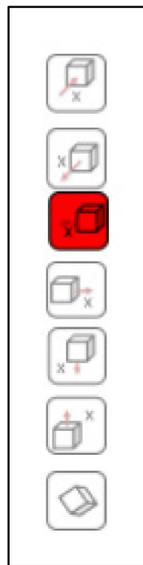


Figure 6.12 Alignment tool bar

To conclude, the manipulations of an *element* and the viewer are realized by drag and drop interaction thus inaugurating a new way to interact with the 3D object in a 3D viewer. This way of doing is much more adapted to non-expert users than the traditional combinations of keyboard buttons and mouse buttons. If the dragged target is the selection handler, then the manipulation will be applied to the selected *element*, otherwise the manipulation is left to the viewer. To zoom the camera, the user needs to scroll the wheel of the mouse or apply a zoom gesture

for the touch mode.

6.2.4 VISUALIZATION OF RELATION AND CONSTRAINT

Relations and *constraints* have to be visualized in a way that allows the user to select and modify them. That's the reason why a specific tree has been implemented to display the relevant hierarchy, modify and delete a *constraint* or a *relation*. When the constraint tree button in the constraint mode is clicked, the window of **Figure 6.13** is being displayed.

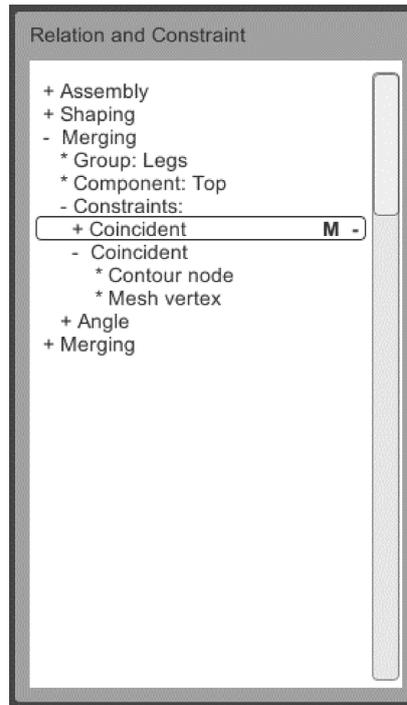


Figure 6.13 Relation and Constraint list

This list contains all the *relations* of the GSDM. Under each Relation we can see the two associated elements (*component* or *group*) to which this Relation is applied and a list of *constraints*. Under each *constraint*, there are two *key entities* if they are specified by the user, otherwise they are marked as “unspecified” and will automatically be determined by the system as described in **Subsection 5.3.1**. When choosing a *constraint* or *relation* in this list, two buttons (“M” and “-”) appear on the right side of the selected row, to modify or delete the chosen *constraint* or *relation*.

6.2.5 SYMBOLIC REPRESENTATION AND GRAPH VIEW OF GSDM

As presented in **Section 6.2.2**, heterogeneous data can be represented in the implemented 3D scene. The geometry, structure, component and group can also be presented in the 3D scene. However, some other notions of the GSDM do not have an implicit geometric representation, such as the *relation* and *constraint*. Basically, the types of connection between *components* or

groups cannot be directly displayed in the 3D viewer. Although **Chapter 4** presents a list showing the existing *relations* and *constraints* of the GSDM, it is more comfortable to also have a representation of them in the 3D viewer. Therefore, we have implemented a dynamic symbolic graph visualization, which overlaps the related 3D objects in the 3D scene to show the different notions of the GSDM and the links between them.

This symbolic graph view is a 2D representation of the GSDM overlapping the 3D viewer. Before showing an example of the symbolic view, a list of symbolic representations of the different notions of the GSDM is presented in the following **Table 6.3**. The symbolic representations can also be used as a selection handler for *group* and *component* as in the examples presented in **Subsection 6.2.3**.

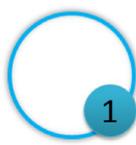
Notion of GSDM	Symbolic representation	Description
Component		A green dot with a black index number at the center
Group		A blue circle/ellipse attached to a blue dot. A black index number is set at the center of the spot. The grouped elements are inside the domain defined.
Relation		An orange segment with an orange rectangle attached to the middle and a black index number located at the center of the rectangle. The two elements in this Relation are located at the two ends of this segment.
Constraint		A purple line with one or multiple purple rectangles attached to it. Each rectangle represents a Constraint and a white index number is set at the center of the rectangle.
Key Entity		Dark blue dot for key point or oriented point. Dark blue line segment for key line.
Semantics		It is used to represent the color of the button to open Semantics and to present different classes in a Class map.
Structure		Pink dot circled in black for a node of the structure and pink line for an edge of the structure
Geometry		This color is used to represent a wireframe if it is used to show a contour

Table 6.3 Specification of symbolic representations for displaying notions of the GSDM

With these symbols, a GSDM could be represented by a symbolic graph overlapping the 3D viewer, as in the example presented in **Figure 6.14**. They are displayed together with the 3D objects and will be updated when the location of the related object changes.

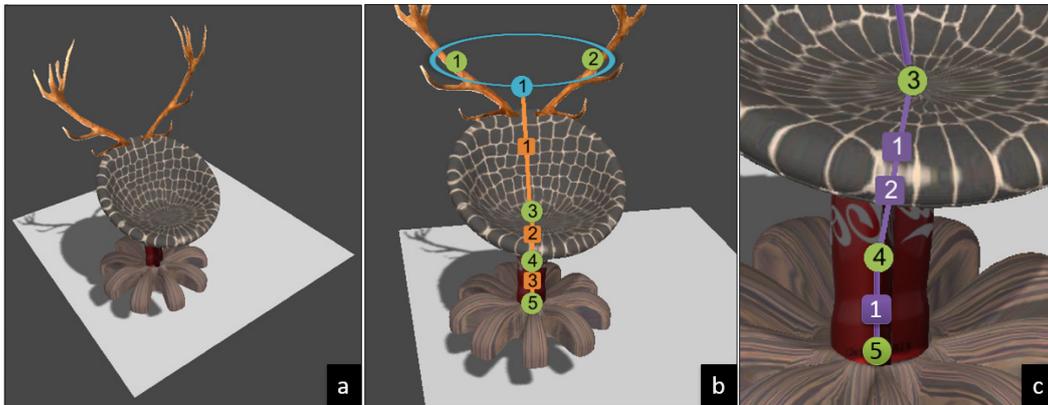


Figure 6.14 Symbolic Graph view of a GSDM: (a) for the 3D viewer without the symbolic graph, (b) for the 3D viewer with a symbolic graph at a Conceptual level, (c) for the 3D viewer with a symbolic graph of Constraints

In this example, if the user turns on the graph view check button in the “Free mode”, then he/she sees a symbolic graph of the *conceptual level* of this GSDM (**Figure 6.14.b**). If it is in a “Constraint mode”, symbols of *relation* (orange line with number in the middle) are replaced by symbols of *constraint* (purple line with numbers in the middle) as presented in **Figure 6.14.c**. When a symbol is double clicked, a property window is being displayed to show its content.

6.2.6 CSP SOLVING

Section 5.4 introduced the theory of CSP solving and the related algorithms. This section explains how the selected mathematical tool works to solve the CSP during the GSDM modeling process.

There are three functions in Mathematica which can be used for solving the CSP, as this CSP contains nonlinear equations and nonlinear objective functions. To use these three functions, we need to formalize the CSP as below⁵⁶:

$$S[\{f, \text{cons}\}, \{x, y, \dots\}]$$

Where f is the function to be minimized, cons is the list of constraints and $\{x, y, \dots\}$ the list of variables. S is the name of the different functions as describe below.

With the formalization of *constraint*, unknown variables, and the objective function introduced previously in **Chapter 5**, the remaining tasks for the implementation is to transform this formalization into a set of expressions in the Mathematica language, and subsequently send it to the solver, get the results and interpret them, and update the GSDM. **Section 5.4** presented a workflow for specifying constraints. The following **Figure 6.15** presents a workflow of CSP

⁵⁶ This expression is using the language of Mathematica 9.0.1.0

solving after the specification of *constraints*.

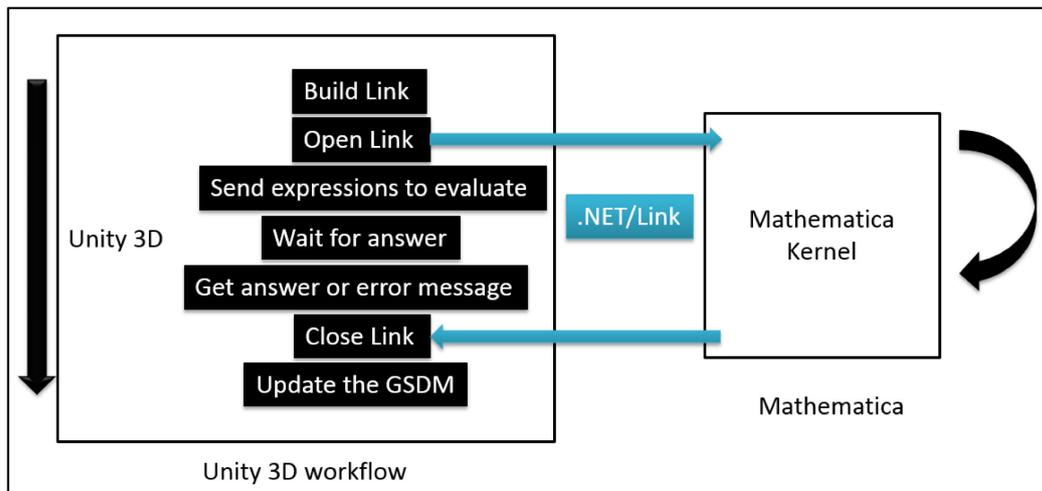


Figure 6.15 Workflow for CSP solving

In this workflow it can be noticed that the library `.Net/Link` is a bridge between Unity 3D and Mathematica kernel. The Mathematica kernel runs when the link is opened and stops when it is closed. CSP solving will be automatically applied when adding a new constraint, modifying a constraint or when the user clicks the update button as described in **Subsection 6.2.1**.

In Mathematica, there are three functions for solving the CSP (**Table 6.4**), depending on the type of inputs, different algorithms can be used in different functions. It is decided automatically by Mathematica.

Function	Use
FindMinimum	numeric local optimization
NMinimize	numeric global optimization
Minimize	exact global optimization

Table 6.4 Functions used in Mathematica to solve the CSP

As a global optimization is addressed in the specified CSP, the function “FindMinimum” is chosen to be used in this implementation.

6.3 EXAMPLES CREATED USING THE IMPLEMENTED APPROACH

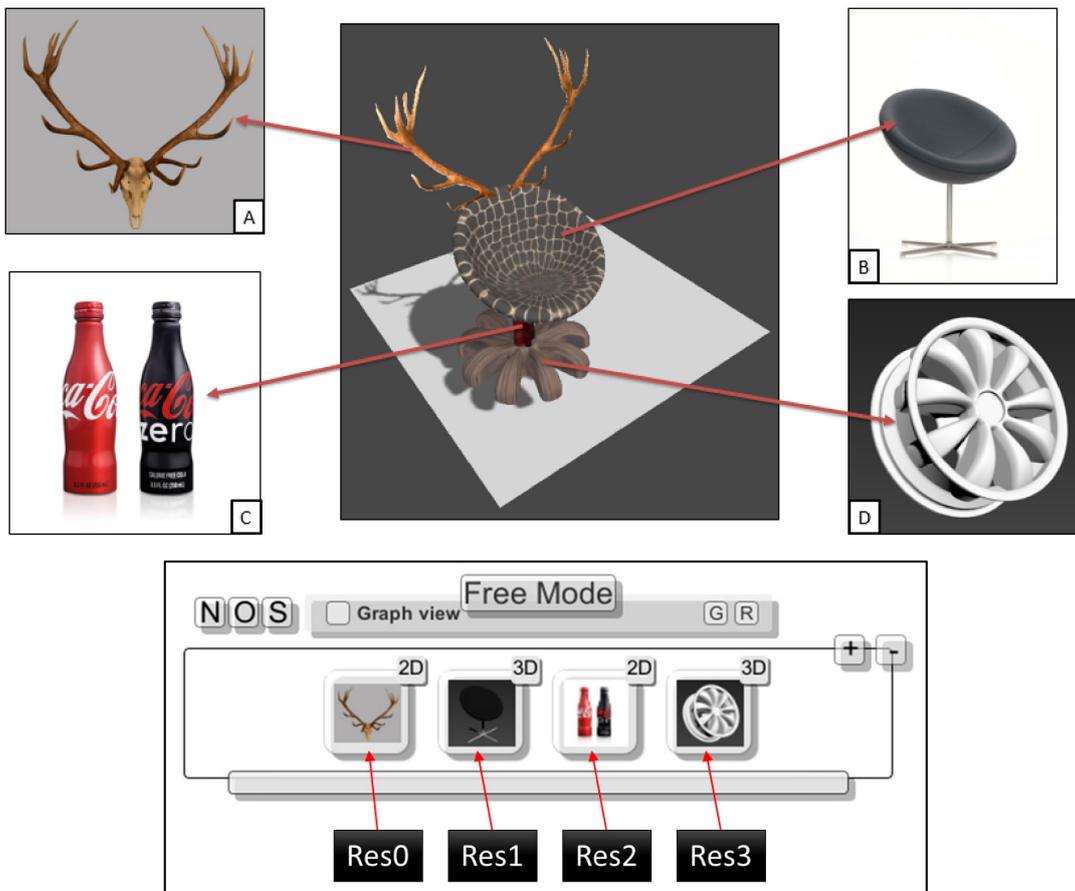
In this section, three examples are used to illustrate the proposed conceptual design approach using GSDM and the implemented tools. The first example is more detailed than the others since it aims at showing the data structure of each instance.

6.3.1 EXAMPLE ONE: CRAZY CHAIR

Example description:

- Object to be described by GSDM: A chair.
- The user of this object: Museum of arts.
- Features of this object:
 - Comfortable for sitting with a back rest,
 - Possibility to hang up clothes (such as a hat and jacket) on the back,
 - Possibility of pivoting around a central support (a swivel chair)

Resources: 4 resources (Res0, Res1, Res2, Res3)



Res0: Picture of antlers of a deer with contour and skeleton information (picture A)

Res1: 3D mesh of a chair with structure information (Picture B)

Res2: Picture of two bottles of Coca Cola with contour and structure information (Picture C)

Res3: Scanned mechanical part of a car wheel hub represented in a mesh (Picture D)

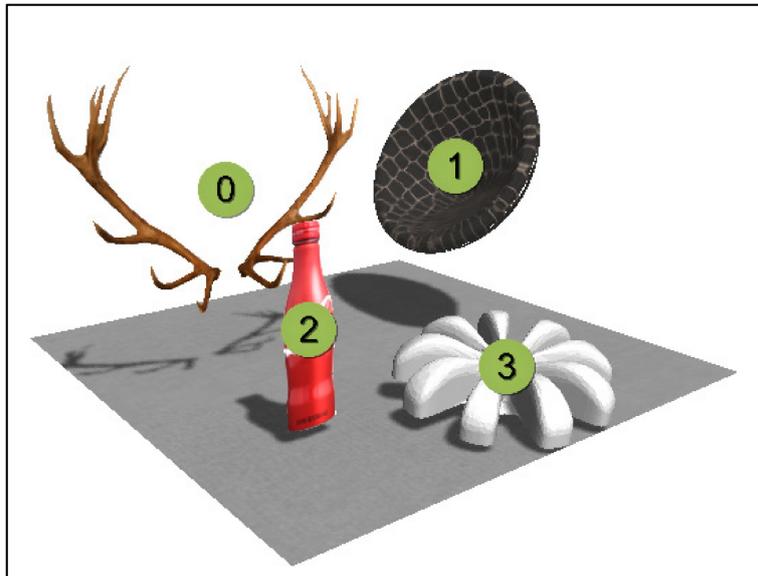
Example of resource data structure (Res0)

Res0: Resource

```
Id = "caca4ad1-fa64-4118-a0a5-ae80a3210cdb"  
Type = ResourceType.Image  
File address = "c:\GSDM\Resources\anlters.jpg"  
Structure address = "c:\GSDM\Resources\anlters_Str.xml"  
Icon address = ""
```

(for image input, if there is no icon file then the origin file address will be used as the icon file address)

Components: 4 Components (Com0, Com1, Com2, Com3)



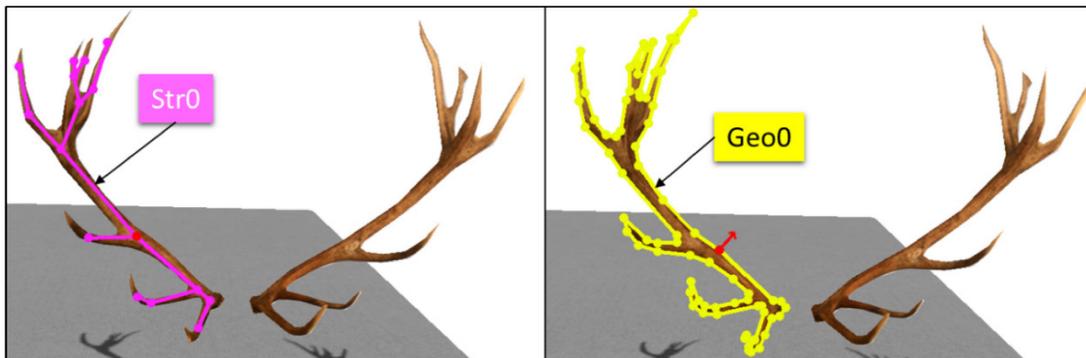
Com0: two antlers of the input image from Res0, aligned to face right direction

Com1: the seat of in input chair mesh from Res1

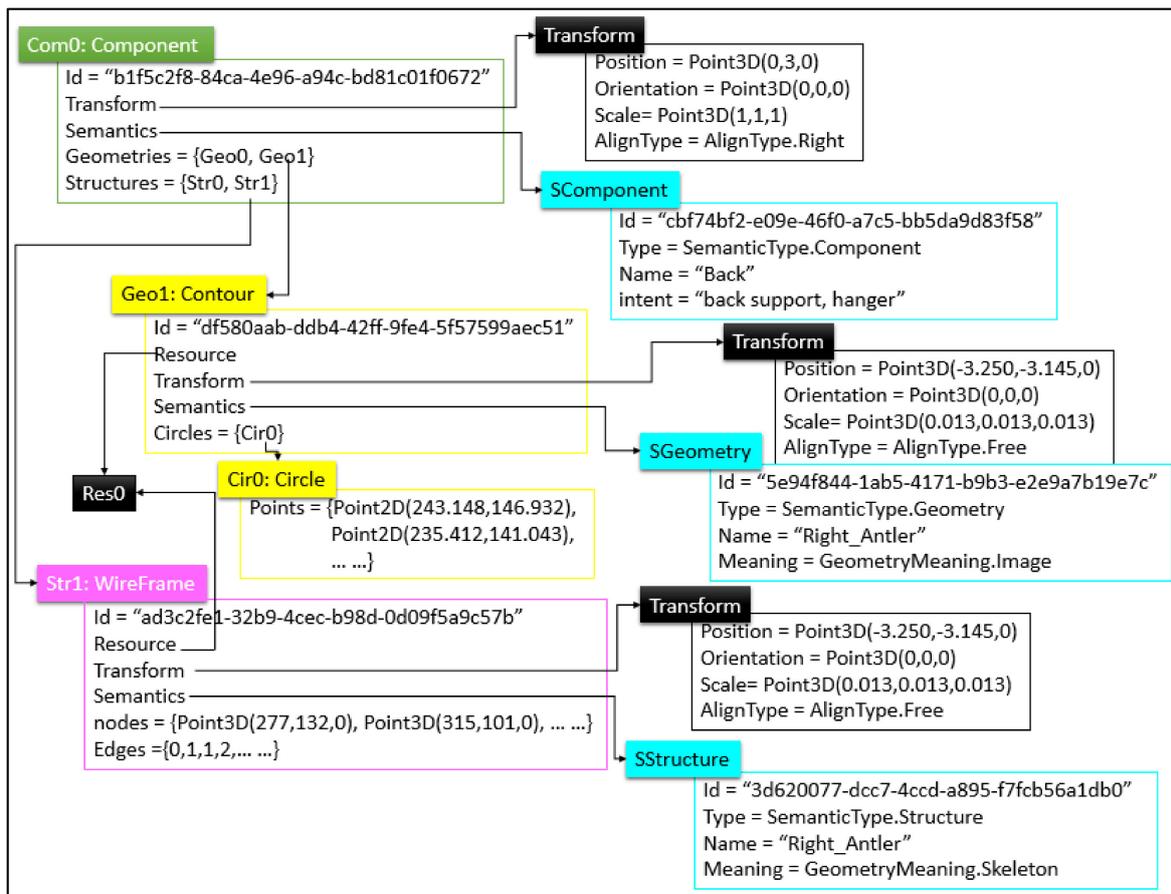
Com2: one bottle of the input image from Res2, aligned to face right direction

Com3: the input mesh from Res3, aligned to face up direction

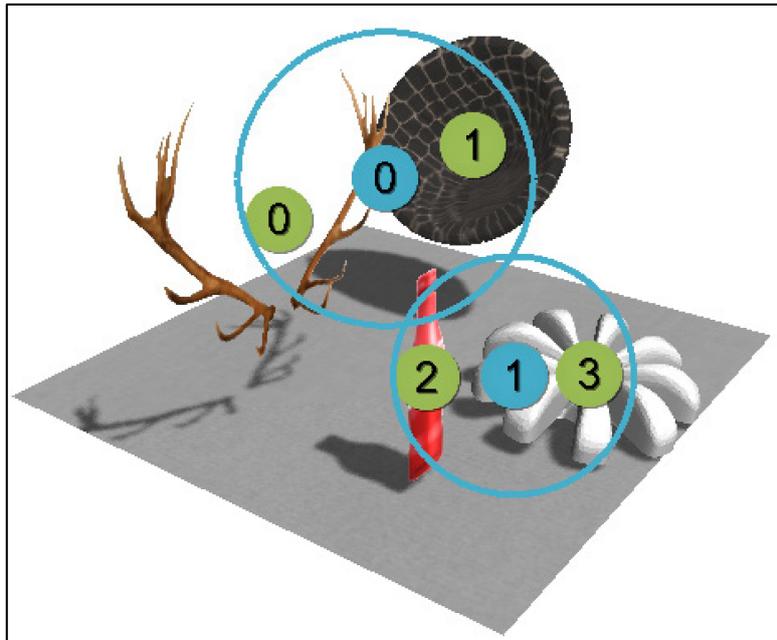
Example of geometry of component (Geo0 of Com0) and structure of component (Str0 of Com0)



Example of a component data structure (Com0)



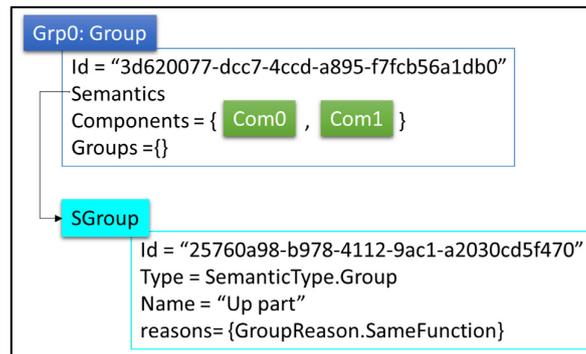
Groups: two groups (Grp0 and Grp1)



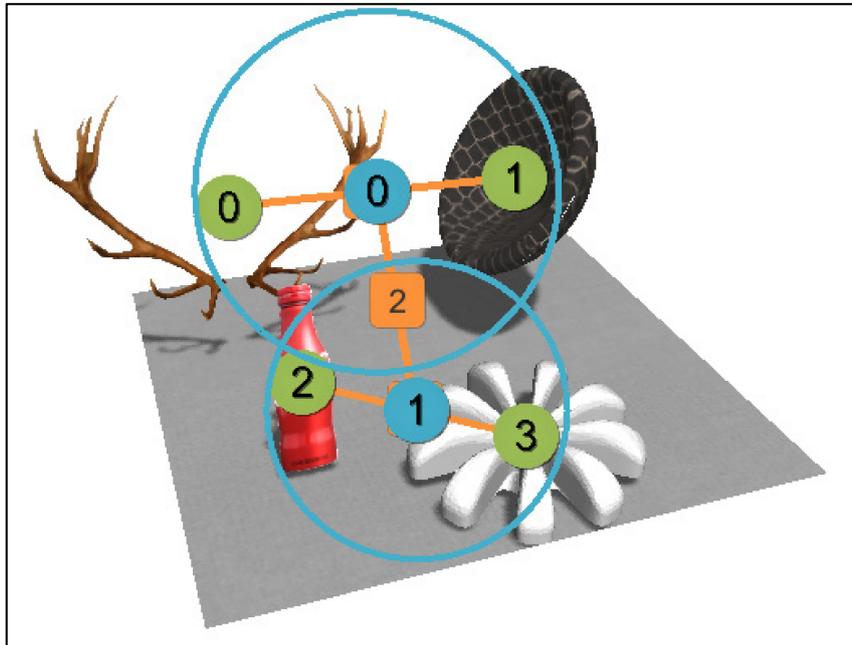
Grp0: contains Com0 and Com1, named as “Up part”

Grp1: contains Com2 and Com3, named as “Support”

Example of a group data structure (Grp0):



Relations: 3 Relations (Rel0, Rel1, Rel2), noticing that Rel0 and Rel1 are not visible in the following figure as it is covered by the group symbol (Grp0 and Grp1).

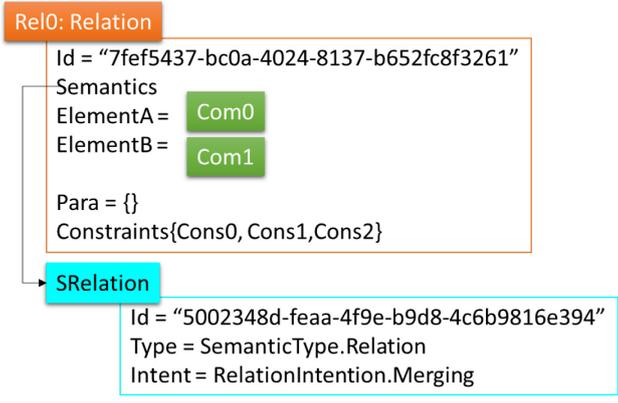


Rel0 : links Com0 and Com1 as a merging

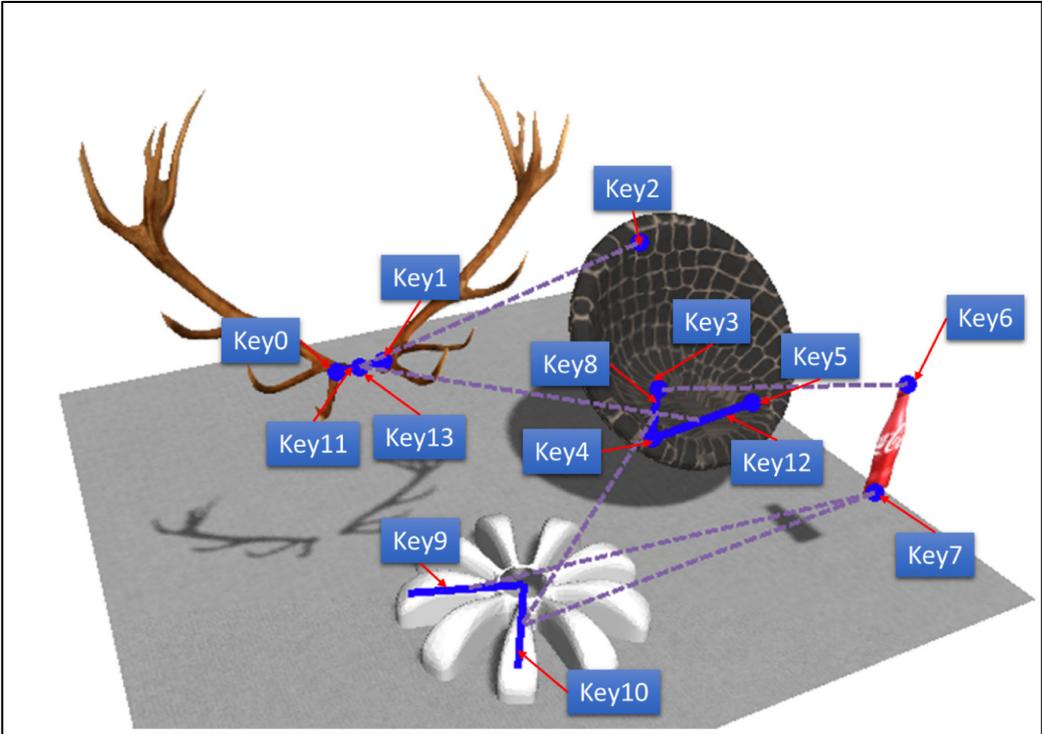
Rel1: links Com2 and Com3 as an assembly

Rel2: links Grp0 and Grp1 as an assembly

Example of Relation data structure (Rel0):



Key entities : 11 direct parametric key entities (Key0 to Key 10)
 And 3 indirect parametric key entities (Key 11 to Key 13)



Direct node of a wireframe (on structure representation):

Key0, Key1, Key2, Key3, Key4, Key5, Key6 and Key7

Direct edge of a wireframe (on structure representation):

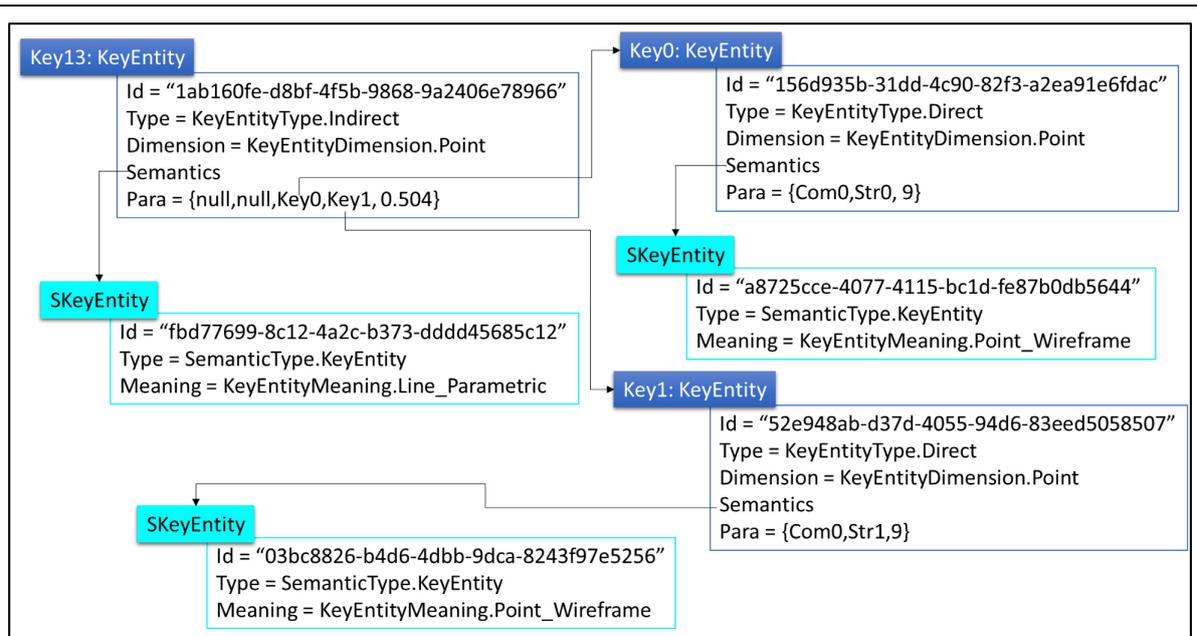
Key8, Key9 and Key10

Indirect parametric line:

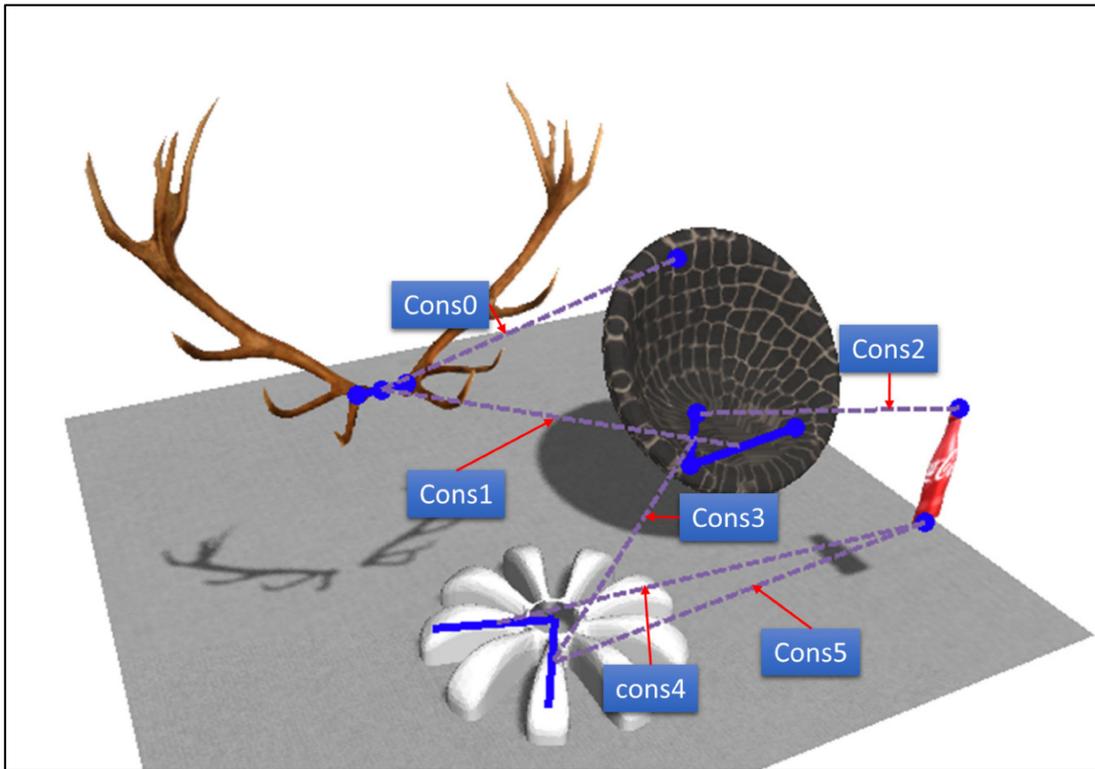
Key11 and Key12

Indirect parametric point: Key13

Example of Key Entity data structure (Key13): In the “Para” of key13, the last value “0.504” represents the factor of Key13 between Key0 and Key1.



Constraint: 6 constraints defined by 18 equations. The type of each constraint is automatically decided by the smart constraining system (Section 5.3).



- Cons0: Coincident between Key2 and Key13 defined by 3 linear equations
- Cons1: Parallelism between Key13 and Key13 defined by 3 linear equations
- Cons2: Coincident between Key3 and Key6 defined by 3 linear equations
- Cons3: Parallelism between Key8 and Key10 defined by 3 linear equations
- Cons4: Co-linearity between Key7 and Key9 defined by 3 linear equations
- Cons5: Co-linearity between Key7 and Key10 defined by 3 linear equations

Result:

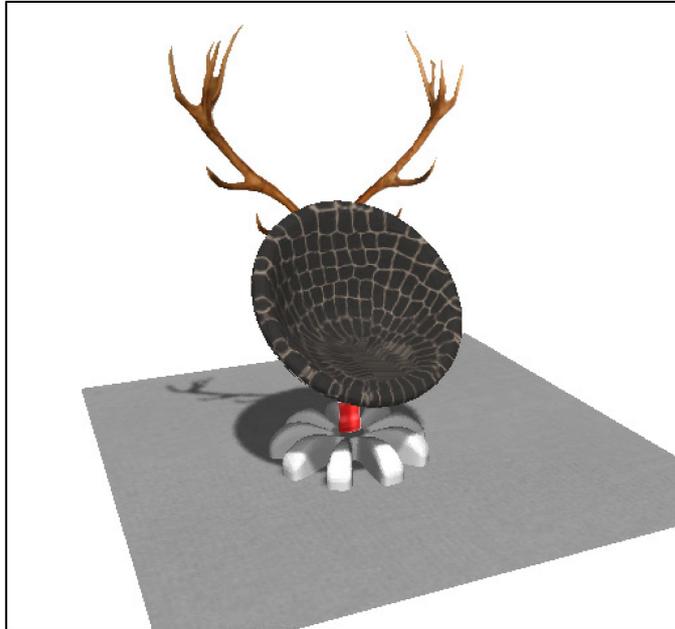
First result with the configuration of CSP parameters as bellow:

Global positioning energy factor: 500

Global rotation energy factor: 5

Global scale energy factor: 10000

Time using for CSP solving: 11.5 s, constraints are all satisfied



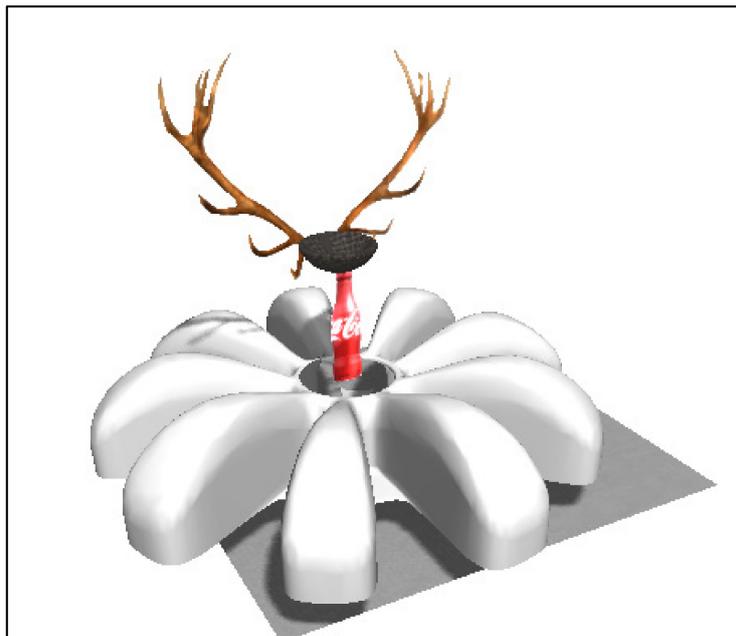
Section result with the configuration of CSP parameters as bellow:

Global positioning energy factor: 500

Global rotation energy factor: 5

Global scale energy factor: 1

Time using for CSP solving: 10.5 s, constraints are all satisfied



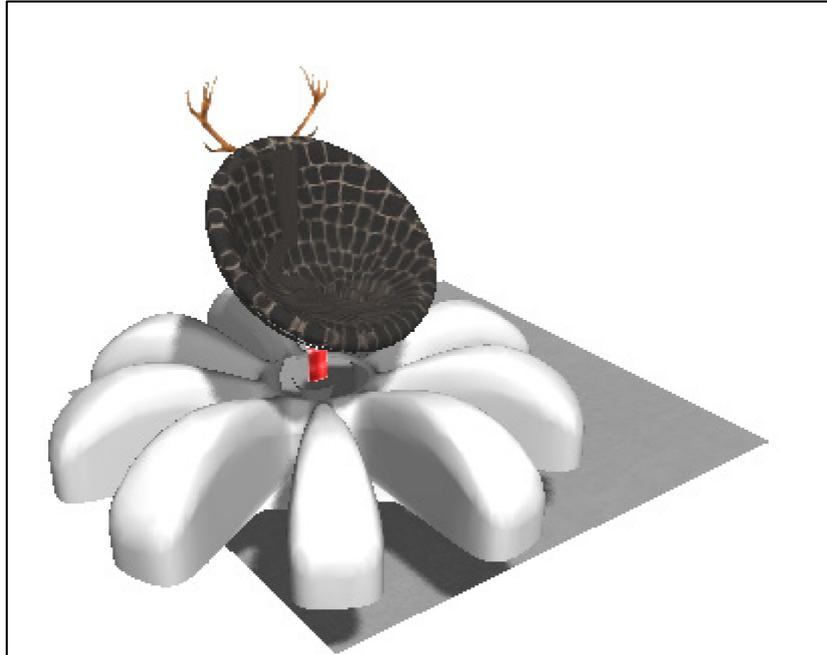
Third result with the configuration of CSP parameters as bellow:

Global positioning energy factor: 500

Global rotation energy factor: 5

Global scale energy factor: 100

Time using for CSP solving: 10.2 s, constraints are all satisfied



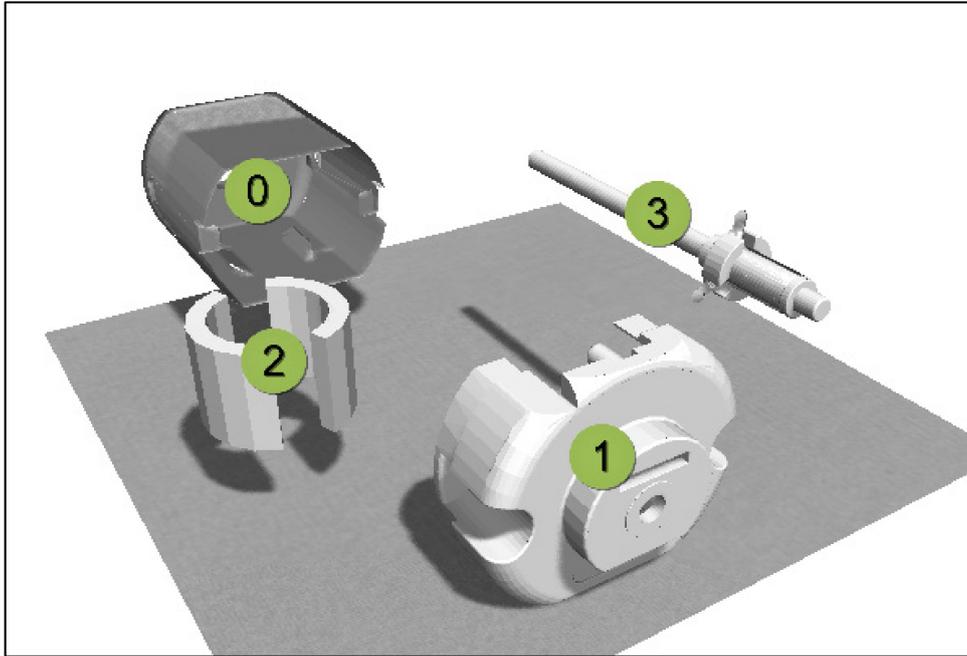
6.3.2 EXAMPLE TWO: ASSEMBLY SCANNED PIECES

Example description:

This example aims at showing how the proposed approach can be used to assemble 3D scanned pieces of a real object. As they have not been generated in CAD system, the considered components do not contain the CAD information but just surface information stored as meshes. Therefore, a traditional assembly approach, available in most CAD software, cannot be applied to assemble them together. In contrary, our GSDM gives the possibility to assemble meshes.

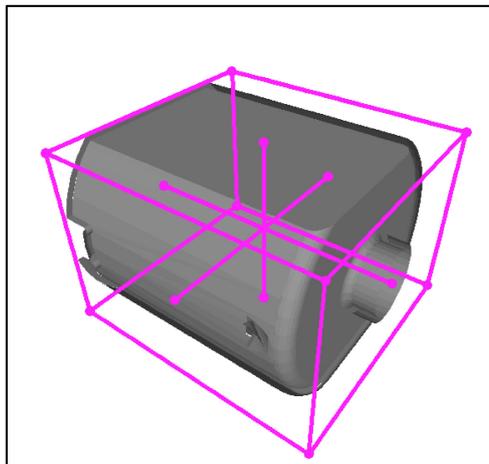
This example is not illustrated with as many details as in the example one, but only with basic GSDM description.

Component: 4 components (Com0, Com1, Com2, Com3)



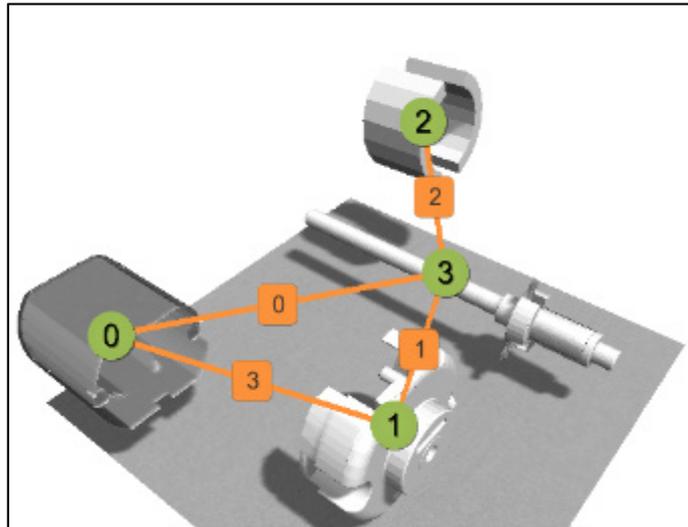
As there is no structure information of the input data, the system adds the minimum oriented bounding box as the structure representation of each component. Com0 and Com1 are aligned to global right direction.

Example of structure representation for Com0:



Group: no group in this description

Relation: 4 relations (Rel0, Rel1, Rel2, Rel3)



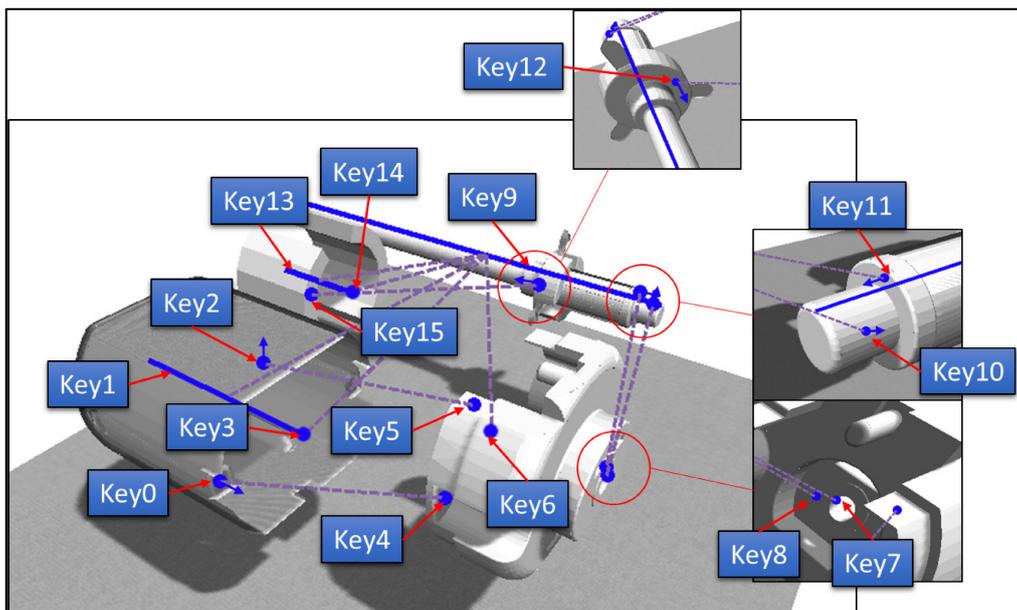
Rel0: Assembly between Com0 and Com3

Rel1: Assembly between Com3 and Com1

Rel2: Assembly between Com3 and Com2

Rel3: Assembly between Com0 and Com1

Key entity : 16 direct parametric key entities



Direct edge of wireframe (oriented bounding box of each component):

Key1, Key9, Key13

Direct node of wireframe (oriented bounding box of each component)::

Key3, Key6, Key14

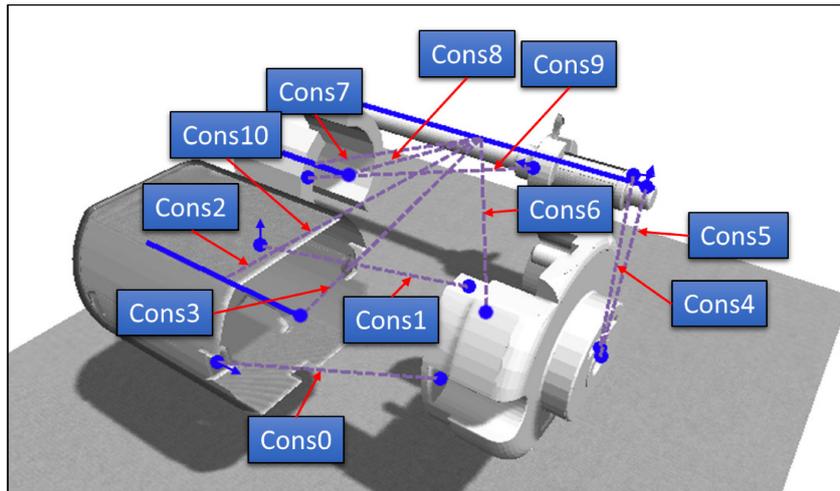
Direct oriented point on mesh:

With dimension type: point: Key4, Key5, Key7, Key8, Key15

With dimension type: oriented point: Key0, Key2, Key10, Key11, Key12

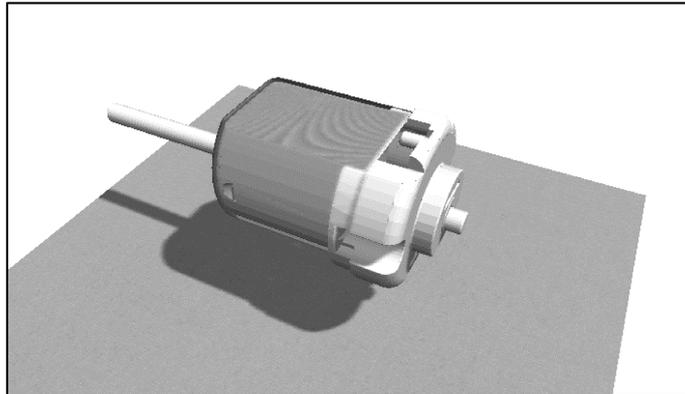
Constraint: 11 constraints, including 20 linear equations. The type of each constraint is

automatically decided by the smart constraining system.



- Cons0: Co-planarity between Key0 and Key4 defined by 1 linear equation.
- Cons1: Co-planarity between Key2 and Key5 defined by 1 linear equation.
- Cons3: Co-linearity between Key3 and Key9 defined by 3 linear equations.
- Cons4: Co-planarity between Key8 and Key11 defined by 1 linear equation.
- Cons5: Co-planarity between Key7 and Key10 defined by 1 linear equation.
- Cons6: Co-linearity between Key6 and Key9 defined by 3 linear equations.
- Cons7: Parallelism between Key9 and Key13 defined by 3 linear equations.
- Cons8: Co-linearity between Key9 and Key14 defined by 3 linear equations.
- Cons9: Co-planarity between Key12 and Key15 defined by 1 linear equation.
- Cons10: Parallelism between Key1 and Key9 defined by 3 linear equations.

Result:



Global positioning energy factor: 500

Global rotation energy factor: 5

Global scale energy factor: 10000

Time using for CSP solving: 39.7 s, constraints are all satisfied.

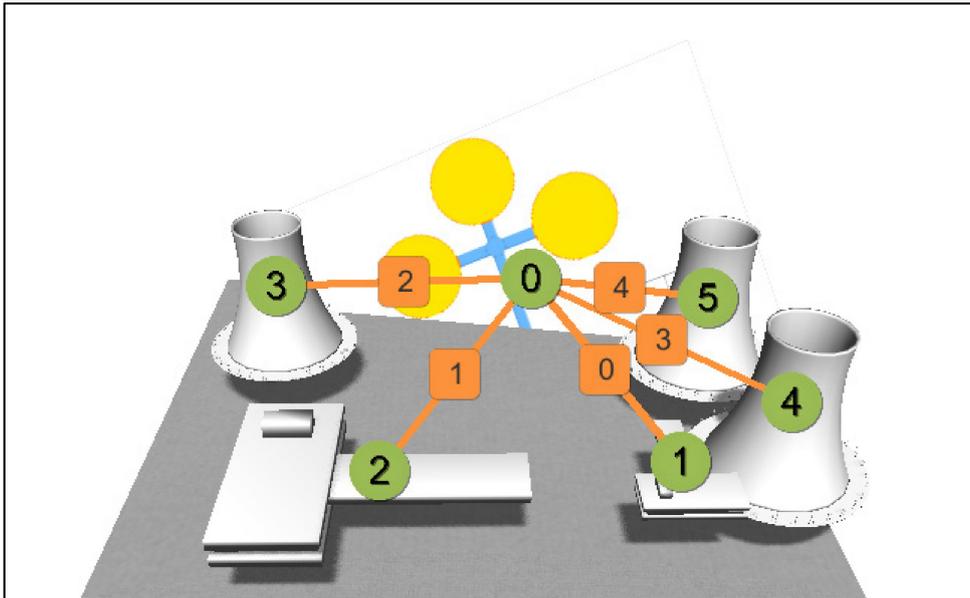
6.3.3 EXAMPLE THREE: POWER PLANT CONFIGURATION

Example description:

For architectural design, there is usually a case for manipulating 3D buildings according to a 2D plan. This example shows how to use GSDM to realize the manipulation of the 3D buildings of a nuclear power plant according to a configuration defined in a 2D image.

This example is illustrated with less detail than the example one, but only with the GSDM description.

Component: 6 components (Com0, Com1, Com2, Com3, Com4, Com5)



Com0 is the 2D plan

Com1 and Com2 are 3D models of two office buildings

Com3, Com4 and Com5 are 3D models of three cooling towers

As there is no structure information associated with those input files, the system creates a minimum oriented bounding box as the structure representation of each component.

Group: no group specified

Relation: 5 relations (Rel0, Rel1, Rel2, Rel3, Rel4)

Rel0: Location between Com0 and Com1

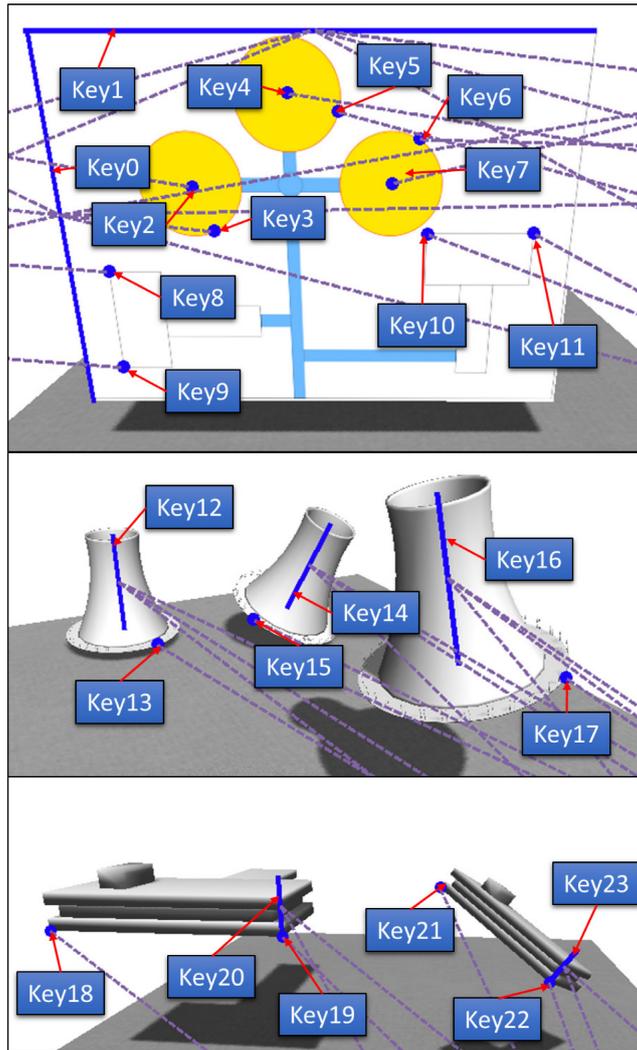
Rel1: Location between Com0 and Com2

Rel2: Location between Com0 and Com3

Rel3: Location between Com0 and Com4

Rel4: Location between Com0 and Com5

Key entity: 24 direct parametric key entities



Direct parametric line of a wireframe's edge:

Key0, Key1, Key12, Key14, Key16, Key20, Key23

Direct parametric point of a wireframe's node:

Key18, Key19, Key21, Key22

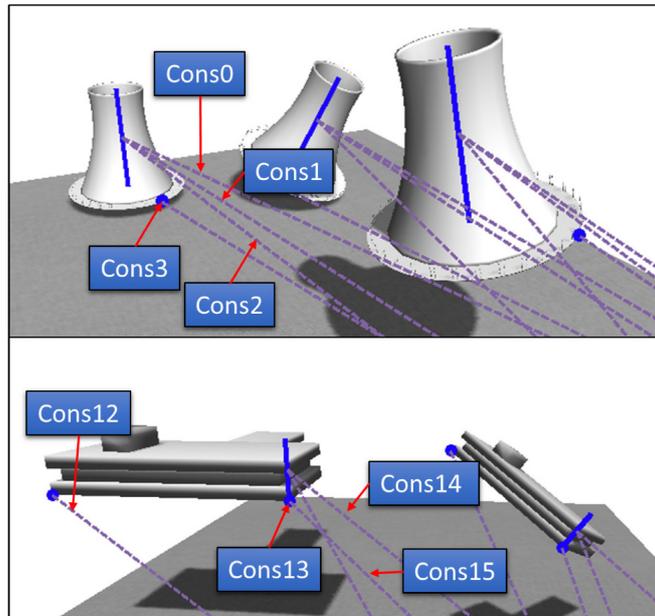
Direct parametric point on an image:

Key2, Key3, Key4, Key5, Key6, Key7, Key8, Key9, Key10, Key11

Direct parametric point on a mesh:

Key13, Key15, Key17

Constraint: 20 constraints including 40 linear equations. The type of each constraint is automatically decided by the smart constraining system. As the constraining of each office building and each cooling tower is the same, the following picture shows an example of constraining of one office building and one cooling tower.



Example of constraining one cooling tower (Com3)

Cons0: perpendicularity between Key0 and Key12 defined by 1 linear equation.

Cons1: perpendicularity between Key1 and Key12 defined by 1 linear equation.

Cons2: Co-linearity between Key12 and Key2 defined by 3 linear equations.

Cons3: Coincidence between Key3 and Key13 defined by 3 linear equations.

Example of constraining one office building (Com1)

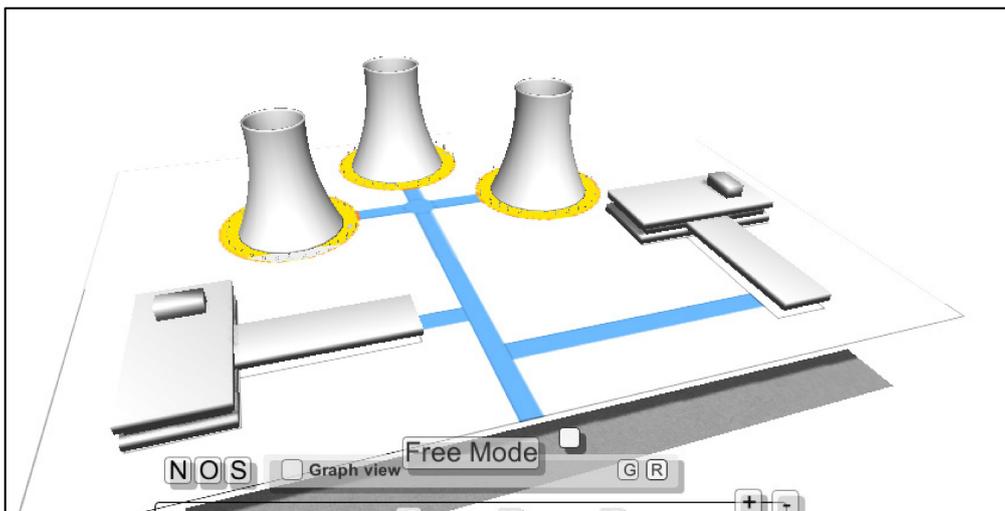
Cons12: Coincidence between Key18 and Key8 defined by 3 linear equations.

Cons13: Coincidence between Key19 and Key9 defined by 3 linear equations.

Cons14: perpendicularity between Key0 and Key20 defined by 1 linear equation.

Cons15: perpendicularity between Key1 and Key20 defined by 1 linear equation.

Result:



Global positioning energy factor: 500

Global rotation energy factor: 5

Global scale energy factor: 10000

Time using for CSP solving: 57.9 s, all constraints are satisfied

6.4 CONCLUSION AND REMARKS

This chapter has presented the implemented tool for conceptual object modeling with GSDM. It proves that:

- The GSDM has the capacity of working with heterogeneous data
- Compared with the traditional 3D design tools, the proposed tool is easier to use by a non-expert user and allows touch screen interaction by exploiting metaphor well-known from smartphone use. A smart manipulation system can help prepositioning the objects. A smart constraint system can automatically build constraint and solve them. The non-expert user only needs to use click or drag and drop to realize a conceptual design. No specific and complex combinations of mouse clicks are required.
- The three examples show different application domains that the implemented tool can be applied to. This is not an exhaustive list and other applications could be imagined.
- The related location of each component when building constraints is important, as well as telling the smart constraining system to choose a suitable type for this generated constraint. Therefore, it is better to manipulate each component to be approximately in the final expected location, and then add constraints, so that the system can smartly understand the intention of this constraint.

However, there are still some improvements to be brought in the future:

- It is better to have a segmentation tool plugged in to deal with non-segmented resources. We can also imagine having a selection tool to use for cutting an area of the imported image or mesh to instantiate a *component*.
- Other high-level *constraints* can be developed to make the smart constraining system even smarter and more complete.
- The effective execution of the “operation” of a *relation* on the geometry is not implemented yet (e.g. the merging of two meshes, properly saying).
- Some reverse engineering algorithms can also be implemented and plugged into the implemented tool to turn the image into a 3D mesh.
- Search engines can also be plugged in, to find potentially usable 3D meshes in a specific database if a *component* is only instantiated by a text.
- Other different CSP solving algorithms can be plugged in. A smart CSP solving system can be developed to automatically choose the most efficient algorithm for a specified CSP.

For complete VR environment creation, the system can be merged into a scene design tool, and behavioral information should also be included.

SYNTHESIS, CONCLUSIONS AND PERSPECTIVES

Today, creativity and innovation in product design and communication are more important than ever in the past. New technologies and the available data and information can provide a good source of inspiration and support for creativity. However, due to the fact that design tools are still expert-oriented weakly supporting fast idea mock-up and communication, the design process is long and tedious. To overcome these issues, this thesis proposes a new approach to create conceptual shapes by re-using heterogeneous digital shape data resources. The approach requires the specification of a new shape representation model capable to handle such a combination of multidimensional data and the development of the required modeling and manipulation capabilities. In this perspective, this thesis develops a Generic Shape Description Model (GSDM) together with a user-friendly modelling system prototype aimed at helping non-expert users to describe shapes.

GENERIC SHAPE DESCRIPTION MODEL...

To overcome the limits of current methods and tools, the proposed GSDM is based on three information levels: data, intermediate and conceptual levels (**Section 4.1**). The data level (**Section 4.2**) reduces the differences between heterogeneous data inputs. Different data are all considered as described by **Geometry, Structure and Semantics**. Thus, the data level is less sensible to the type of data that can be manipulated in the same way for different heterogeneous data. For example, a search engine can look for similar data based on the structure layer (e.g. graph-based representation) independently of the underlying geometries (e.g. meshes, images). The conceptual level (**Section 4.3**) bridges the gap between the non-expert user and the GSDM and immediately provides an overview of the elements constituting the object. The different

parts of the shape are represented as **Components** coming from heterogeneous inputs. **Components** sharing a common characteristic (e.g. color, meaning, behavior) can be clustered in a **Group**. Besides grouping different components together, **Relations** can be built between components or groups to describe how they are combined. Four types of relations can be applied: **Assembly**, **Merging**, **Shaping** and **Location**. Assembly is used to put together different parts that still exist by their own; merging is applied to create a single elements from different parts (component or group); shaping indicates the use shape features of one element to modify another one; and location is used to position different parts in an object or in the scene. The intermediate level (**Section 4.4**) links the conceptual description (conceptual level) and the underlying data (data level) through a set of constraints acting on key entities which lie on the components. All the user-specified constraints are finally sent to a solver system. This system finds an optimal solution to satisfy all the constraints by considering an energy function to be minimized. This function is based on physical energies used to translate, rotate and scale the components. Compared to the traditional CAD modelers, our approach provides a more meaningful assembled solution.

MODELING TOOL...

A user-centered workflow for using the GSDM to describe shapes has been presented in **Chapter 5**. It is implemented into a design tool based on Unity3D development platform. The whole process starts from importing external heterogeneous data resources. The information related to each input is restructured in the three layers of geometry, structure and semantics. If available segmentation information is imported allowing the user to select sub-parts as Components. Groups and Relations can be added later. User can create new key entities or take existing ones to build constraints. Finally, the CSP solving system is automatically activated to achieve an optimized solution.

More user-oriented capabilities have been developed in this tool than those available in other 3D modeling tools. First, a smart 3D object manipulation system automatically decides in which plane to place the imported component. This positioning is based on the user's view direction. A single drag-and-drop action is also defined to simultaneously apply the component position, rotation and scaling. Second, a smart constraint system is designed to automatically assign a constraint between two elements depending on the current location of the two user-specified key entities.

CURRENT ISSUES...

This manuscript defines 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 using a unified approach. 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 to the current version of the GSDM has a very limited usage. It is mainly used to store information for initializing different constituents of the GSDM, such as the “type” or “reason”. For some specific applications, other “types” or “reasons” need to be 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 visual resources, additional representations should be considered and manipulated. This could be done through the development of new plugins. 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. Therefore, currently most of the resources require some human intervention to be used in our system for the component selection. 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
 - A set of key entities and a set of constraints have been proposed. Some of them have been tested in the examples presented in **Chapter 6**. However, for some specific applications, the ones here proposed might not be the most suitable. Therefore, the general concepts of key entity and constraint are well defined, while their exact nature might need to be extended for different situations. This could be addressed by simply extending the constraints toolbox. One could also imagine the possibility to have user-specified constraints.
 - The relation type of “Shaping” is not fully expressed in this manuscript, while just a general concept has been proposed. However, such a relation can be of real interest for design and creativity issues.
 - In the current prototype, the resolution of the optimization problem is implemented as a plugin using “Mathematica v9.0”. The produced results are appropriate but the execution is a bit slow for interactive modelling of complex configurations. Having the resolution fully integrated in the developed prototype software would drastically speed up the process.
 - The current modeler cannot generate shapes starting from scratch but only by combining existing ones. However, this is not a real limitation since the idea was not to redevelop existing modeling tools but rather to develop a new approach capable of mixing existing heterogeneous representations.

From the above discussions, it can be noticed that, what presented here is mainly a proof of concepts, while to achieve an efficient and operative system further activity is needed.

PERSPECTIVES...

For future versions, internal modeling capabilities could be integrated to generate shapes starting from scratch but also to effectively modify the imported ones. More user-oriented functions need to be implemented or optimized for the developed tool. This is the case for the optimization problem solving that still requires the user to specify weights between the terms of the energy function.

The GSDM can also be further used to describe a whole scene in a VR environment. In this case, additional specifications of the GSDM concepts needs to be added, which include for example new types of grouping and new types of constraints.

A full exploitation of semantics for easing objects and scene creation, e.g., for specific types of objects, can be defined and then exploited to automatically set relations among components.

As a long-term perspective, mechanisms for the pre-processing and post-processing phases should be included. Some automatic segmentation and structure analysis system should be integrated, so that even raw geometry input data can be associated with segmentation and structure information. This requires the solution to research problems related to the choice and combination of the segmentation and structure analysis algorithms to obtain the most meaningful object decompositions and structure descriptors.

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 reverse engineering. New research on structure and semantics merging can also be imagined for their correct updating according to the achieved 3D object model. With both the pre-processing and post-processing phases, the GSDM can be used in the whole 3D object design process.

From the application point of view, the last Chapter (Chapter 6) validates the approach and demonstrates the possibilities of using GSDM in several domains such as conceptual design, reverse engineering of assemblies and 3D objects manipulation. It can be also imagined to use GSDM in medical analysis domain, such as representing different medical data and results (CT images, type-B ultrasonic images, etc.) in a unified 3D environment, probably aligned to a 3D model of a human body. GSDM could also be used as a plugin for a 3D presentation tool such as Microsoft's PowerPoint but in 3D. In this case, text and animation abilities should be further developed.

These considerations indicate that there are still many encouraging open issues and applications of the proposed method.

REFERENCES

- [1] D. Backlund, "Product cost analysis in early stages of a product development process," Arvika, 2013.
- [2] E. Nasr and A. Kamrani, "Chapter 2 Product Life Cycle Cost Model," in *computer Based Design and Manufacturing*, Springer, 2007, pp. 31-58.
- [3] Y. ASIEDU and P. GU, "Product life cycle cost analysis: state of the art review," *int. j. prod. res.*, vol. 36, no. 4, pp. 883-908, 1998.
- [4] F. Bianca., G. Franca, L. Jean-Claude and P. Jean-Philippe, "Processing free form objects within a Product Development Process framework," in *Advances in Computers and Information in Engineering Research*, J. G. Michopoulos, C. J. Paredis, D. W. Rosen and J. M. Vance, Eds., ASME, 2014, pp. 317-344.
- [5] H. Istvan T., G. U. W. Martin, S. Olga, D. Loris, D. A. Raffaele and R. Graphitech, "Shape Semantics from Shape Context," 2004.
- [6] G. P. S. Brar, V. Jain and A. Singh, "Research Methodology," *International Journal of Humanities Social Sciences and Education*, vol. 1, no. 8, pp. 63-67, 2014.
- [7] N. Brikci and J. Green, *A Guide to Using Qualitative Research Methodology*, MSF UK: Michael Quinn Patton and Michael Cochran, 2007.

-
- [8] C. R. Kothari, *Research Methodology : Methods and Techniques*, Second Edition ed., New Age International, 2004.
- [9] Y. Singh, *Fundamentals of Research Methodology and Statistics*, New Age Interantional, 2008.
- [10] WiseGeek, "What Are the Different Types of Research?," 2014. [Online]. Available: <http://www.wisegeek.org/what-are-the-different-types-of-research.htm>. [Accessed 2014].
- [11] K. Best, *Design Management: Managing Design Strategy, Process and Implementation*, Switzerland: AVA Publishing SA, 2006.
- [12] K. Goodwin, *Designing for the Digital Age: How to Create Human-Centered Products and Services*, John Wiley & Sons, 2009.
- [13] S. Cox, "Cox Review of Creativity in Business: building on the UK's strengths," 2005.
- [14] C. Mok, *Designing Business: Multiple Media, Multiple Disciplines*, Pap/Cdr edition ed., Adobe Pr, 1996.
- [15] R. House, *Random House Webster's Unabridged Dictionary*, 2 Revised edition ed., Random House Reference, 2005.
- [16] Dictionary.com, "Dictionary.com," 2014. [Online]. Available: <http://dictionary.reference.com/>.
- [17] C. Abras, "User-Centered Design," in *Encyclopedia of Human-Computer Interaction*, Bainbridge, W., Thousand Oaks: Sage Publications, 2004, pp. 763-768.
- [18] D. A. Norman and S. W. Draper, *User Centered System Design: New Perspectives on Human-Computer Interaction*, Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1986.
- [19] K. Eason, *Information technology and organizational change*, London: Taylor and Francis, 1987.
- [20] J. Preece, Y. Rogers and H. Sharp, *Interaction design: Beyond human-computer interaction*, New York: John Wiley & Sons, Inc., 2002.
- [21] P. Gabriel-Petit, "Design Is a Process, Not a Methodology," *UXmatters*, 2010.
- [22] J. Jonson, *GUI Bloopers 2.0: Common User Interface Design dont'ts and Dos*, Burlington, MA, USA: Dense E.M. Penrose, 2008.
- [23] G. Smith, "Idea-generation techniques: A formulary of active ingredients," *Journal of Creative Behavior*, vol. 32, no. 2, pp. 107-133, 1998.

-
- [24] K. Sawyer, *Zig Zag: The Surprising Path to Greater Creativity*, Jossey-Bass, 2013.
- [25] O. Shai, Y. Reich and D. Rubin, "Creative conceptual design: Extending the scope by infused design," *Computer-Aided Design*, vol. 41, pp. 117-135, 2009.
- [26] S. Ellis, "Nature and origin of virtual environments: a bibliographic essay," *Computer Systems in Engineering*, p. 325, 1991.
- [27] J. Palfreman and D. Swade, *The Dream Machine-Exploring the Computer Age*, London: BBC Books, 1991.
- [28] I. Sutherland, "The ultimate display," *Proceedings of International federation of Information Processing Congress*, pp. 506-508, 1965.
- [29] T. P. Hughes, *American Genesis - A Century of Invention and Technological Enthusiasm*, New York: Penguin Books, 1989, pp. chapter 1-3.
- [30] A. Sevalnikov, "Virtual reality and problems of its description," 1999.
- [31] V. Bychkov and N. Mankovskaya, "Virtual reality in the space of aesthetic experience," 2006.
- [32] W. R. Sherman and A. B. Craig, *Understanding Virtual Reality: Interface, Application, and Design (The Morgan Kaufmann Series in Computer Graphics)*, San Francisco, CA: Morgan Kaufmann; , 2002.
- [33] J. Vince, *Introduction to Virtual Reality*, Springer, 2004.
- [34] M. A. M. Lumbreras, L. F. Pulido, A. P. Flores and F. J. A. Velasco, "Incorporating three dimensional sound in Virtual Environments," *Procedia Technology*, vol. 3, p. 259–266, 2012.
- [35] J. Frederick P. Brooks, "What's Real About Virtual Reality," *IEEE Computer Graphics and Applications*, vol. 19, no. 6, pp. 16-27, 1999.
- [36] P. Chen, "The entity-relationship model: Towards a unified view of data.," *ACM Transactions on Database system*, vol. 1, no. 1, pp. 471-522, 1976.
- [37] T. Halpin, *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*, Morgan Kaufmann, 2001.
- [38] M. Fowler and K. Scott, *UML Distilled: a brief introduction to the standard object modeling language.*, Addison-Wesley Professional, 1999.

-
- [39] W. Bille, O. De Troyer, F. Kleinermann, B. Pellens and R. Romero, "Using Ontologies to Build Virtual Worlds for the Web.," in *Proc. of the IADIS International Conference WWW/Internet 2004 (ICWI2004)*, Madrid, Soain, 2004.
- [40] O. De Troyer, W. Bille, R. Romero and P. Stuer, "On Generating Virtual Worlds from Domain Ontologies.," in *Proc. of the 9th International Conference on Multi-Media Modeling*, Taipei, Taiwan, 2003.
- [41] F. K. B. P. W. B. Olga De Troyer, "Conceptual Modeling for Virtual Reality," in *Tutorials, Posters, Panels and Industrial Contributions at the 26th International Conference on Conceptual Modeling*, Auckland, New Zealand, Australian Computer Society, Inc., 2007, pp. 3-18.
- [42] S. GC and S. S, "Latent Semantic Engineering – A New Conceptual User-Centered Design Approach," *Advanced Engineering Informatics*, vol. 26, no. 2, pp. 456-473, 2012.
- [43] J.-P. Pernot, "Fully Free From Deformation Features for Aesthetic and Engineering Designs," 2004.
- [44] M. Mantyla, "A modeling system for top-down design of assembled products," *IBM Journal of Research and Development*, vol. 34, no. 5, pp. 636-659, 2010.
- [45] J. J. Shah and M. Mantyla, *Parametric and Feature-Based CAD/CAM: Concepts, Techniques, and Applications*, Wiley-Interscience, 19995.
- [46] T. D. Martino, B. Falcidieno, F. Giannini, S. Hassunger and J. Ovtcharova, "Feature-based modeling by integrating design and recognition approaches," *Computer-Aided Design*, vol. 26, no. 8, pp. 646-653, 1994.
- [47] CAM-I, *CAM-I's Illustrated Glossary of Workpiece Form Features*, CAM-I, 1981.
- [48] O. Salomons, F. v. Houten and H. Kals, "Review of research in feature-based design," *Journal of Manufacturing Systems*, vol. 12, no. 2, pp. 113-132, 1993.
- [49] S. Coquillart, "Extended Free-Form Deformation: A Sculpturing Tool for 3D Geometric Modeling Computer Graphics," *SIGGRAPH '90*, vol. 24, no. 4, pp. 187-196, 1990.
- [50] R. Smelik, T. Tutenel, K. d. Kraker and R. Bidarra, "A declarative approach to procedural modeling of virtual worlds," *Computers & Graphics*, vol. 35, no. 2, pp. 352-363, 2011.
- [51] J.-D. Boissonnat, "Geometric structures for three-dimensional shape representation," *ACM Transactions on Graphics*, vol. 3, no. 4, pp. 266-286, 1984.
- [52] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, "Surface reconstruction from unorganized points," *Proceeding of SIGGRAPH*, pp. 71-78, 1992.

-
- [53] N. Amenta, M. Bern and M. Kamvyselis, "A new Voronoi-based surface reconstruction algorithm," *Proceedings of SIGGRAPH*, pp. 425-421, 1998.
- [54] R. Kolluri, J. Shewchuk and J. O'Brien, "Spectral surface reconstruction from noisy point clouds," *Proc. Symposium on Geometry Processing*, pp. 11-21, 2004.
- [55] J. Carr, R. Beatson, J. Cherrie, T. Mitchell, W. Fright, B. McCallum and T. Evans, "Reconstruction and representation of 3D objects with radial basis functions," *Proceedings of SIGGRAPH*, pp. 67-76, 2001.
- [56] N. Jiang, P. Tan and L.-F. Cheong, "Symmetric Architecture Modeling with a Single Image," *ACM Transactions on Graphics*, 2009.
- [57] El-Hakim and S. F., "Semi-automatic 3D reconstruction of occluded and unmarked surfaces from widely separated views," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 34, no. 5, pp. 143-145, 2002.
- [58] L. D. Luca, P. Veron and M. Florenzano, "Reverse engineering of architectural buildings based on a hybrid modeling approach," *Computers & Graphics*, vol. 30, no. 2, pp. 160-176, 2006.
- [59] P. M., P. JP. and V. P., "Towards recovery of complex shapes in meshes using digital images for reverse engineering applications," *Computer-Aided Design*, vol. 42, no. 8, pp. 693-707, 2010.
- [60] P. M., P. J-P. and V. P., "Polybedral simplifications preserving character lines extracted from images," in *Proceedings of the IEEE Shape Modeling International Conference (SMI'07)*, Lyon, 2007.
- [61] T. Matsuyama, X. Wu, T. Takai and S. Nobuhara, "Real-time 3D shape reconstruction, dynamic 3D mesh deformation, and high fidelity visualization for 3D video," *Computer Vision and Image Understanding*, vol. 96, no. 3, pp. 394-434, 2004.
- [62] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops and R. Koch, "Visual modelling with a hand-held camera," *International Journal of Computer Vision*, vol. 59, no. 3, pp. 207-232, 2004.
- [63] M. M. Oliveira, "Image-Based Modeling and Rendering Techniques: A Survey," *RITA*, vol. 9, no. 2, pp. 37-66, 2002.
- [64] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," *SIGGRAPH '96 Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 303-312, 1996.
- [65] J. S. Dan Lv, Q. Li and Q. Wang, "Point-based integration for 3D object reconstruction from

-
- Ladar range images," *Optik - International Journal for Light and Electron Optics*, vol. 124, no. 23, pp. 6318-6326, 2013.
- [66] H. M. Nguyen, B. Wunsche and P. Delmas, "A hybrid image-based modeling algorithm," *Proceedings of the thirty-sixth Australasian Computer Science Conference*, pp. 115-123, 2013.
- [67] P. E. Debevec, C. J. Taylor and J. Malik, "Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach," *Proceedings of SIGGRAPH 96*, 1996.
- [68] E. E. Catmull, "A subdivision algorithm for computer display of curved surfaces," 1974.
- [69] R. L. Cook, L. Carpenter and E. Catmull, "The Reyes image rendering architecture Reyes image rendering architecture," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 95-102, 1987.
- [70] I. K. Kazmi, L. You and J. J. Zhang, "A Survey of 2D and 3D Shape Descriptors," *Computer Graphics, Imaging and Visualization (CGIV), 2013 10th International Conference*, pp. 1-10, 2013.
- [71] T. Pavlidis, "A review of algorithms for shape analysis," *Computer Graphics and Image Processing*, vol. 7, pp. 243-258, 1978.
- [72] J. Mantas, "METHODOLOGIES IN PATTERN RECOGNITION AND IMAGE ANALYSIS--A BRIEF SURVEY," *Pattern Recognition*, pp. 1-620, 1987.
- [73] A. Rosenfeld, "Image analysis: Problems, Progress and Prospects," in *Computer vision*, m. A. Fischler and O. Firschein, Eds., California, Morgan kaufmann, 1987, pp. 3-12.
- [74] L. d. F. Costa and R. M. C. Jr., *Shape Analysis and Classification: Theory and Practice*, FL, USA: CRC Press, 2000.
- [75] D. Zhang and G. Lu, "Review of shape representation and description techniques," *Pattern Recognition*, vol. 37, pp. 1-19, 2004.
- [76] L. Zhang, M. João da Fonseca and A. Ferreira, "Survey on 3D Shape Descriptors," 2004.
- [77] Y. Yang, H. Lin and Y. Zhang, "content-Based 3-D model Retrieval: A Survey," *IEEE Transactions on System, Man and Cybernetics - Part C : Applications and Reviews*, vol. 37, no. 6, pp. 1081-1098, 2007.
- [78] R. Jain, R. Kasturi and B. G. Schunck, *Machine Vision*, McGraw-Hill, 1995.

-
- [79] F. van der Heijden, *Image Based Measurement Systems*, New York: John Wiley and Sons, 1994.
- [80] K. R. Castleman, *Digital Image Processing*, Englewood Cliffs, New Jersey: Prentice Hall, 1996.
- [81] J. M. Chassery and A. Montanvert, *Géométrie Discrète*, In French ed., Paris: HERNES, 1991.
- [82] X. Y. Jiang and H. Bunke, "Simple and fast computation of moments," *Pattern Recognition*, vol. 24, no. 8, pp. 801-806, 1991.
- [83] N. Kiryati and D. Maydan, "Calculating geometric properties from Fourier representation," *Pattern Recognition*, vol. 22, no. 5, pp. 469-475, 1989.
- [84] O. D. Trier, A. K. Jain and T. Taxt, "Feature extraction methods for character recognition - a survey," *Pattern Recognition*, vol. 29, no. 4, pp. 641-662, 1996.
- [85] J. Wood, "Invariant pattern recognition: A review," *Pattern Recognition*, vol. 29, no. 1, pp. 1-17, 1996.
- [86] R. C. Gonzalez and P. Wintz, *Digital Image Processing*, Addison-Wesley, 1987.
- [87] G. & T. A. Sapiro, "Affine invariant scale-space," *International Journal of computer Vision*, vol. 11, no. 1, pp. 25-44, 1993.
- [88] P. J. van Otterloo, *A Contour-Oriented Approach to Shape Analysis*, Englewood Cliffs, NJ: Prentice-Hall, 1991.
- [89] D. Sarkar, "A simple algorithm for detection of significant vertices for polygonal-approximation of chain-coded curves," *Pattern Recognition Letters*, vol. 14, no. 12, pp. 959-964, 1993.
- [90] P. O'Higgins, "Methodological Issues in the Description of Forms.," in *Fourier Descriptors and Their Applications in Biology*, P. E. Lestrel, Ed., 1997, p. Cambridge University Press.
- [91] S. L. a. C. S. L. Lam, "Thinning methodologies - a comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 9, pp. 869-885, 1992.
- [92] A. Carlo, S. d. B. Gabriella and S. Luca, "A parallel algorithm to skeletonize the distance transform of 3D objects," *Image and Vision Computing*, vol. 27, no. 6, pp. 666-672, 2009.
- [93] C. Niblack, P. Gibbons and D. Capson, "Generating skeletons and centerlines from the distance transform," *CVGIP: Graphical Models and Image Processing*, vol. 54, no. 5, p. 420-437, 1992.

-
- [94] J. A. Sethian, "Level set methods and fast marching methods," *Cambridge University Press*, 1999.
- [95] M. Näf, G. Székely, R. Kikinis, M. Shenton and O. Kübler, "3D Voronoi Skeletons and Their Usage for the Characterization and Recognition of 3D Organ Shape," *Computer Vision and Image Understanding*, vol. 66, no. 2, p. 147–161, 1997.
- [96] G. Reeb., " Sur les points singuliers d'une forme de Pfaff compl`etement int`egrable ou d'une fonction num`erique," *Comptes-rendus de l'Acad`emie des Sciences*, pp. 848-849, 1946.
- [97] S. Biasotti, D. Giorgi, M. Spagnuolo and B. Falcidieno, "Reeb graphs for shape analysis and applications," *Theoretical Computer Science*, vol. 392, no. 1-3, p. 5–22, 2008.
- [98] A. W. Smeulders, M. Worring, S. Santini, A. Gupta and Ramesh Jain, "Content-Based Image Retrieval at the End of the Early Years," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 22, no. 12, pp. 1349-1380, 2000.
- [99] H. H. Wang, D. Mohamad and N. Ismail, "Approaches, Challenges and Future Direction of Image Retrieval," *JOURNAL OF COMPUTING*, vol. 2, no. 6, pp. 193-199, 2010.
- [100] G. Stiny, J. Gips, G. Stiny and J. Gips, "Shape Grammars and the Generative Specification of Painting and Sculpture," in *Segmentation of Buildings for 3D Generalisation. In: Proceedings of the Workshop on generalisation and multiple representation* , 1971.
- [101] I. Bloch, "Fuzzy relative position between objects in image processing: A morphological approach.," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 7, pp. 657-664, 1999.
- [102] C. Hudelot, J. Atif and I. Bloch, "Fuzzy spatial relation ontology for image interpretation," *Fuzzy Sets Syst*, vol. 159, no. 15, pp. 1929-1951, 2008.
- [103] C. M. Takemura, "Modelagem de posições relativas de formas complexas para análise de configuração espacial," 2008.
- [104] AIM@SHAPE, "AIM@SHAPE," 2004. [Online]. Available: <http://www.aimatshape.net/>.
- [105] A. Saxena and B. Sahay, *Computer Aided Engineering Design*, Springer, 2005.



Conceptual design of shapes by reusing existing heterogeneous shape data through a multi-layered shape description model and for VR applications

ABSTRACT

Due to the great advances in acquisition devices and modeling tools, a huge amount of digital data (e.g. images, videos, 3D models) is becoming now available in various application domains. In particular, virtual environments 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 innovations, the design of application-oriented virtual environment still results from a long and tedious iterative modeling and modification process that involves several actors (e.g. experts of the domain, 3D modelers and VR programmers, designers or communications/marketing experts). Depending of the targeted application, the number and the profiles of the involved actors may change. Today's limitations and difficulties are mainly due to the fact there exists no strong relationships between the expert of the domain with creative ideas, the digitally skilled actors, the tools and the shape models taking part to the virtual environment development process. Actually, existing tools mainly focus on the detailed geometric definition of the shapes and are not suitable to effectively support creativity and innovation, which are considered as key elements for successful products and applications. 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. Therefore, the availability of software tools allowing the re-use and combination of such digital data would be an effective support for the conceptual design phase of both single shapes and VR environments. To answer those needs, this thesis proposes a new approach and system for the conceptual design of VRs and associated digital assets by taking existing shape resources, integrating and combining them together while keeping their semantic meanings. To support this, a Generic Shape Description Model (GSDM) is introduced. This model allows the combination of multimodal data (e.g. images and 3D meshes) according to three levels: conceptual, intermediate and data levels. The conceptual level expresses what the different parts of a shape are, and how they are combined together. Each part of a shape is defined by an Element that can either be a Component or a Group of Components when they share common characteristics (e.g. behavior, meaning). Elements are linked with Relations defined at the Conceptual level where the experts in the domain are acting and exchanging. Each Component is then further described at the data level with its associated Geometry, Structure and potentially attached Semantics. In the proposed approach, a Component is a part of an image or a part of a 3D mesh. Four types of Relation are proposed (merging, assembly, shaping and location) and decomposed in a set of Constraints which control the relative position, orientation and scaling of the Components within the 3D viewer. Constraints are stored at the intermediate level and are acting on Key Entities (such as points, a lines, etc.) laying on the Geometry or Structure of the Components. All these constraints are finally solved while minimizing an additional physically-based energy function. At the end, most of the concepts of GSDM have been implemented and integrated into a user-oriented conceptual design tool totally developed by the author. Different examples have been created using this tool demonstrating the potential of the approach proposed in this document.

KEYWORDS

Virtual Environment, conceptual design, shape description, hybrid object, heterogeneous data, optimization problem.

Design conceptuel de formes par exploitation de données hétérogènes au sein d'un modèle de description de forme multi-niveaux et pour des applications de RV

RESUME

Les récentes avancées en matière de systèmes d'acquisition et de modélisation ont permis la mise à disposition d'une très grande quantité de données numériques (e.g. images, vidéos, modèles 3D) dans différents domaines d'application. En particulier, la création d'Environnements Virtuels (EVs) nécessite l'exploitation de données numériques pour permettre des simulations et des effets proches de la réalité. Malgré ces avancées, la conception d'EVs dédiés à certaines applications requiert encore de nombreuses et parfois laborieuses étapes de modélisation et de traitement qui impliquent plusieurs experts (e.g. experts du domaine de l'application, experts en modélisation 3D et programmeur d'environnements virtuels, designers et experts communication/marketing). En fonction de l'application visée, le nombre et le profil des experts impliqués peuvent varier. Les limitations et difficultés d'aujourd'hui sont principalement dues au fait qu'il n'existe aucune relation forte entre les experts du domaine qui ont des besoins, les experts du numérique ainsi que les outils et les modèles qui prennent part au processus de développement de l'EV. En fait, les outils existants focalisent sur des définitions souvent très détaillées des formes et ne sont pas capables de supporter les processus de créativité et d'innovation pourtant garants du succès d'un produit ou d'une application. De plus, la grande quantité de données numériques aujourd'hui accessible n'est pas réellement exploitée. Clairement, les idées innovantes viennent souvent de la combinaison d'éléments et les données numériques disponibles pourraient être mieux utilisées. Aussi, l'existence de nouveaux outils permettant la réutilisation et la combinaison de ces données serait d'une grande aide lors de la phase de conception conceptuelle de formes et d'EVs. Pour répondre à ces besoins, cette thèse propose une nouvelle approche et un nouvel outil pour la conception conceptuelle d'EVs exploitant au maximum des ressources existantes, en les intégrant et en les combinant tout en conservant leurs propriétés sémantiques. C'est ainsi que le Modèle de Description Générique de Formes (MDGF) est introduit. Ce modèle permet la combinaison de données multimodales (e.g. images et maillages 3D) selon trois niveaux : Conceptuel, Intermédiaire et Données. Le niveau Conceptuel exprime quelles sont les différentes parties de la forme ainsi que la façon dont elles sont combinées. Chaque partie est définie par un Élément qui peut être soit un Composant soit un Groupe de Composants lorsque ceux-ci possèdent des caractéristiques communes (e.g. comportement, sens). Les Éléments sont liés par des Relations définies au niveau Conceptuel là où les experts du domaine interagissent. Chaque Composant est ensuite décrit au niveau Données par sa Géométrie, sa Structure et ses informations Sémantiques potentiellement attachées. Dans l'approche proposée, un Composant est une partie d'image ou une partie d'un maillage triangulaire 3D. Quatre Relations sont proposées (fusion, assemblage, shaping et localisation) et décomposées en un ensemble de Contraintes qui contrôlent la position relative, l'orientation et le facteur d'échelle des Composants au sein de la scène graphique. Les Contraintes sont stockées au niveau Intermédiaire et agissent sur des Entités Clés (e.g. points, des lignes) attachées à la Géométrie ou à la Structure des Composants. Toutes ces contraintes sont résolues en minimisant une fonction énergie basée sur des grandeurs physiques. Les concepts du MDGF ont été implémentés et intégrés au sein d'un outil de design conceptuel développé par l'auteur. Différents exemples illustrent le potentiel de l'approche appliquée à différents domaines d'application.

MOTS CLES

Environnement Virtuel, design conceptuel, description de formes, objet hybride, données hétérogènes, problème d'optimisation numérique.

Progettazione concettuale creativa di forme: un approccio centrato sull'utente basato su un modello di descrizione delle forme a più livelli e sul riutilizzo di dati eterogenei

SINTESI

Grazie ai grandi progressi raggiunti dai dispositivi di acquisizione e dagli strumenti di modellazione, una quantità enorme di dati digitali (immagini, video, modelli 3D,..) sta diventando ora disponibile in vari domini applicativi. Tali dati possono essere quindi sfruttati in ambienti virtuali per una comunicazione e simulazione efficace di ambienti e oggetti reali o possibili. Nonostante i notevoli sviluppi nel settore, la progettazione di ambienti virtuali richiede ancora un lungo processo iterativo di modellazione e modifica che coinvolge diversi attori. A seconda dell'applicazione, il numero e i profili degli attori coinvolti possono variare, includendo, ad esempio, esperti del dominio per il quale l'applicazione è sviluppata, esperti di modellazione 3D, programmatori di ambienti virtuali, progettisti o esperti di comunicazione o marketing. Gli attuali limiti e ostacoli sono principalmente dovuti alla difficoltà di comunicazione tra l'esperto del dominio con idee creative, gli esperti di modellazione CAD e i tecnologi competenti negli strumenti coinvolti nel processo di sviluppo dell'ambiente virtuale. Gli strumenti esistenti si concentrano principalmente sulla definizione geometrica dettagliata delle forme e non sono adatti a supportare efficacemente la creatività e l'innovazione, elementi chiave per la realizzazione di prodotti e applicazioni di successo. Inoltre, questa caratteristica non permette di sfruttare al meglio la grande quantità di dati digitali disponibili. Questi dati possono infatti costituire un'importante fonte di ispirazione per nuove soluzioni, essendo le idee innovative spesso provenienti dalla (inusuale) combinazione di elementi esistenti. Pertanto, la disponibilità di strumenti software che consentano il riutilizzo e la combinazione di tali dati digitali costituirebbe un supporto efficace per la fase di progettazione concettuale di singole forme e ambienti virtuali. Per rispondere a queste esigenze, questa tesi propone un nuovo approccio e un sistema per la progettazione concettuale che permette la creazione di forme a partire da elementi esistenti combinando insieme loro sotto parti e mantenendone il loro significato semantico. Il tutto è supportato dalla definizione di un nuovo modello per la rappresentazione delle forme denominato Generic Shape Description Model (GSDM). Questo modello consente la combinazione di dati multimodali (ad esempio immagini e mesh 3D) organizzata su tre livelli: concettuale, intermedio e dati. Il livello concettuale indica la scomposizione di una forma nelle sue parti elementari e relazioni tra di esse. Ogni parte è rappresentata da un elemento, che può corrispondere a una Componente o a un Gruppo di Componenti che condividono caratteristiche comuni (ad esempio comportamento, significato). Gli elementi sono collegati da Relazioni. Ogni Componente è ulteriormente descritta a livello dati tramite le corrispondenti informazioni relative a Geometria, Struttura e Semantica. Nell'approccio proposto, una Componente è una parte di un'immagine o di una mesh 3D. Le Relazioni possono essere di quattro tipi (fusione, assemblaggio, modifica di forma e posizionamento) e sono espresse tramite un insieme di vincoli che controllano la posizione relativa, l'orientamento e il ridimensionamento delle componenti all'interno della nuova forma. I vincoli sono memorizzati a livello intermedio e agiscono su Entità Chiave (come punti, a linee, ecc.), appartenenti alla Geometria o alla Struttura delle Componenti. La maggior parte dei concetti del GSDM sono stati implementati e integrati in uno strumento di supporto alla progettazione concettuale user-oriented sviluppato dall'autore. Vari esempi di utilizzo del sistema sono riportati nella tesi a dimostrazione delle potenzialità del metodo proposto.

PAROLE CHIAVE

Ambienti virtuali, progettazione concettuale, descrizione delle forme, modelli ibridi, dati eterogenei, ottimizzazione.