A Declarative System to Design Preliminary Surfaces

Raphaël La Greca
raphael.la.greca@esil.univ-mrs.fr

Marc Daniel
marc.daniel@esil.univ-mrs.fr

LSIS (UMR CNRS 6168)
Case 925 - 163 Av. de Luminy
13288 Marseille cedex 09 - (France)

ABSTRACT

B-Spline and NURBS surfaces are most often considered to model objects. The object shape is designed by manipulating several control points, which is often very complex and tedious. The declarative approach of surface modelling is a fast and easy way to obtain sketches of parametric surfaces. The designer provides a description of the shape he/she wants to obtain. The semantic extracted from this description is structured through XML language. As a result, a set of parametric surfaces corresponding to the given constraints and features is proposed to the user. This approach is specially devoted to speed up the preliminary design process. This paper introduces our system as a high level tool of surface modelling. Details dealing with the different models and processing involved in our system are proposed. The document is illustrated by the first results of our research study.

Keywords
Geometric modelling, Declarative, Surface, NURBS, semantics, CAD, Geometric constraints.

1. INTRODUCTION

Present days, Computer Aided Geometric Design is a key computing field in industrial activities. This technology enables to visualise objects and to simulate their behaviours before being manufactured. In most instances, these objects are designed by a set of NURBS surfaces. Their shape can be manipulated by several control points which can often be very complex and tedious. In order to help the designers and to provide them a high level tool of design, we suggest a declarative approach during the modelling process. The purpose of this system is to produce easily and quickly sketches of NURBS surfaces [Dan96]. The designer has only to give to the declarative system a description of the shape he/she wants to obtain, and the process suggests him/her one or several solution models which satisfy the specified constraints and features [CDMM97]. This relieves the user of any knowledge about the underlying mathematical model and the designer can really express his/her creative feeling. He/she can see all the surfaces which correspond to the description. If they match the requirements the process can be stopped. Otherwise, he/she can change or improve the description and start again the declarative process to receive new solutions. This loop can be iterated until an expected shape is produced. This paper presents the framework of the declarative system of surface modelling we developed (see Figure 1).

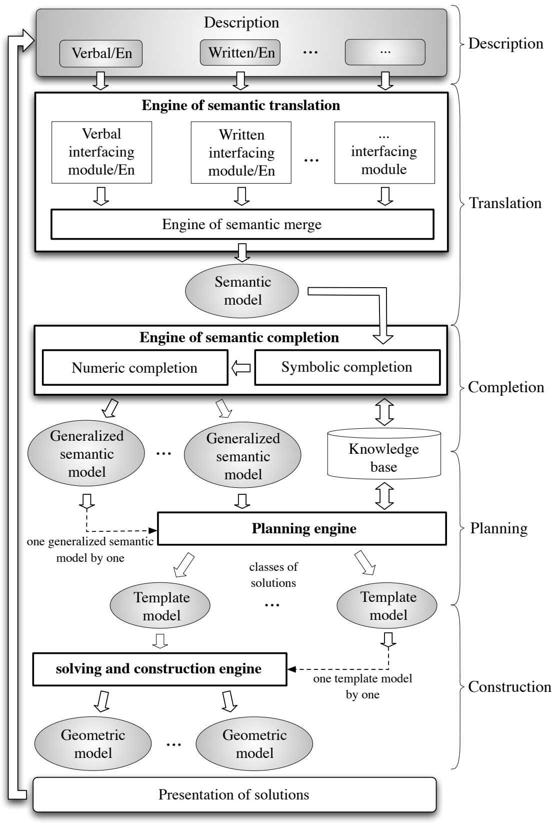

Description
Verbal/En
Written/En
...
Engine of semantic translation
Verbal interfacing module/En
Written interfacing module/En
... interfacing module
Engine of semantic merge
Semantic model
Engine of semantic completion
Numeric completion
Symbolic completion
Generalized semantic model
Knowledge base
Planning engine
one generalized semantic model by one
classes of solutions
Template model
solving and construction engine
one template model by one
Geometric model
Presentation of solutions
Description
Translation
Completion
Planning
Construction

Figure 1: The framework of the declarative system.

The first part introduces the description concept and its translation. The three next sections give details about the three other steps, Completion, Planning and Construction, used to obtain one or several solution surfaces. Some practical results and future works conclude this document.

2. THE DESCRIPTION AND ITS SEMANTIC TRANSLATION

The input of the declarative system we developed is a description of the surface expected by the designer. The purpose of the translation process is to extract the semantic from the description, and to structure it into a single final model called *semantic model*.

2.1 From the Description To the Semantic

A declarative system must be able to adapt itself to the user. In order to follow this rule, we introduce the concept of interfacing modules. A description can be given by the designer using several media like written English, verbal French or dialog box and button interface. In every instance, the description depends of the situation and the knowledge of the designer. Thus, an object can be described differently by many people. We choose to focus our work on the mechanical domain to obtain trade-oriented descriptions. A specific vocabulary is identified and can be classified into eight categories: comparing, junction, constraint, topology, morphology, quantification, localisation and geometry. Each term is a key word with a signification. One interfacing module is dedicated to translate one specific media into a structured semantic language by key words extraction and knowledge organisation. All the knowledge extracted from the description is stored in the semantic model using a XML[1] tree. The XML format has been chosen because it allows us to easily structure the description and because it is well-adapted to introduce specific fields which model the semantic of the final surfaces. Let us propose a very simple example based on a written English description:

“ *The surface has a rectangular shape and its right part is a little bulged.* ”

After the semantic translation step, the corresponding semantic model could be:

```
<Surface Id="Z0">
   <Shape Id="RECTANGULAR"/>
   <Zone Id="Z1">
      <Deformation Id="BULGED">
         <Quantifier Id="LITTLE"/>
      </Deformation>
      <Localisation Id="RIGHT"/>
   </Zone>
</Surface>
```

This XML representation is a direct translation of the designer description. In this case of written text, the data processing of the corresponding interfacing module consists of parsing the description sentence to identify key words [EH04] and to fill out the matching XML fields. An interfacing module is considered like a plugin which can be switched on or off according to the requirements of the situation.

If several media are used to describe the same object or surface, all the structured semantics is merged by the *engine of semantic merge* in order to obtain a single semantic model. This last model is a semantic representation of the surface expected by the designer.

2.2 Semantic Model Structure

The XML structure of a semantic model is organised around the notion of zones also called parametric zones. A surface is viewed as a set of zones. Each of them can be a child of another one and is defined by a localisation, a shape, a set of deformations, a set of constraints and a closed polygon inferred from all these data. This framework is illustrated by Figure 2 using the UML notation.

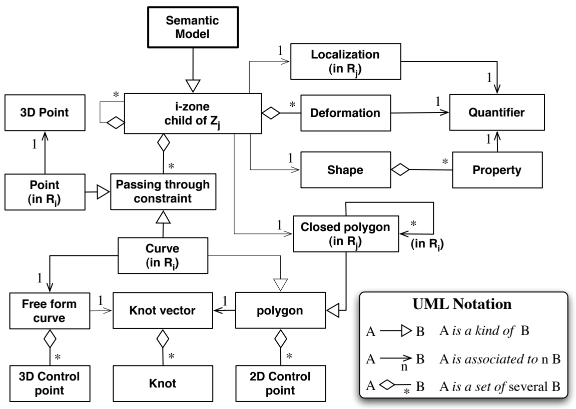


Figure 2: Framework of a semantic model using the UML notation.

Thus, a declarative surface is defined as the root of the XML tree of which each node is a zone. We can formalise this framework with the following definition:

Definition 2.1 - Let Z_i, child of Z_j, be the *i-zone* defined by the set $\{Z,S,L,P,D,C\}$ where:

- Z is the set of all local zones defined on Z_i. If Z is empty, Z_i has no child, otherwise a child-zone is valid only if its shape is fully included in the shape of Z_i. The bounding box of Z_i is written R_i and defines the new coordinate system of all its child-zones (see Figure 3). We choose to fit the area of each new coordinate system to $[0,100] \times [0,100]$. It is very important to remark that R_0 which is the coordinate system defined by the root-zone, will be matched with the parametric plane of each solution surface during the construction process. A child-zone is defined with the `<Zone>`...`</Zone>` markups. The interweaving of all these sets Z leads to the tree framework of the XML structure. This tree is called structuring tree and is denoted by T_s,

[1] e**X**tended **M**arkup **L**anguage.

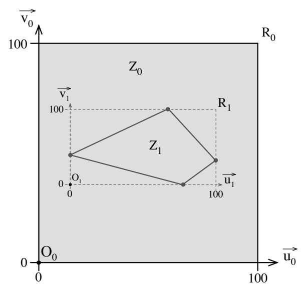


Figure 3: Reference frame of the declarative surface.

- *S* is the symbolic data which describes the shape of Z_i. This data must be in the knowledge base and could be, for example, “QUADRATE” or “ROUND”. The `<Shape>...</Shape>` markups are used to store this data on the XML structure,
- *L* is the symbolic data which describes the localisation of Z_i. Two kind of localisation can be distinguished, the relative one and the absolute one. The first can locate Z_i according to other objects or zones, the second can directly situate Z_i on R_j. This data must be in the knowledge base. For example, “ON THE RIGHT OF” or “NEAR TO” could be used in the relative localisation case and “RIGHT PART” or “TOP LEFT CORNER” in the absolute localisation case. The `<Localisation>...</Localisation>` markups are used to store this data on the XML structure. The difference between the two localisations is made using the `<Relative>...</Relative>` and `<Absolute>...</Absolute>` markups,
- *P* is the closed control polygon of the NURBS curve C_i defining the borderline of Z_i on R_j. If the degree of C_i is equal to 1 the borderline is exactly *P*. If *P* is not directly given by the designer, *P* is inferred during the numeric completion process, using the knowledge base and the symbolic data of Z_i which describe its shape and its localisation on R_j. It is possible to define some areas on Z_i as holes in order to obtain zones of different topology (see Figure 4),

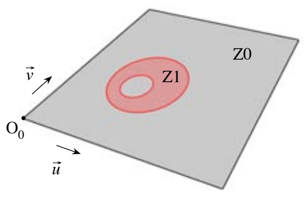


Figure 4: $Z1$, a zone with a hole.

  The `<Polygon>...</Polygon>` markups are used to store all the polygon data. The markups `<Minus>...</Minus>` add the holes definition of *P* on the XML structure,

- *D* is a set of deformations to apply on Z_i. A deformation is also called a *soft constraint* which is the final aspect the designer expects to find on the solution surfaces. Each deformation can be linked to a quantifier which is able to control its magnitude. The available deformations and their relationships are stored on the knowledge base. The `<Deformation>...</Deformation>` markups are used on the XML structure to list the deformations to be applied on Z_i,
- *C* is a set of constraints or *strict constraints* which have to exactly be satisfied on the solution surfaces. One of these constraints is typically to pass through one or many specific points of the space. These constraints are stored on the XML structure by the `<Constraints>...</Constraints>` markups. To add a new constraint, we use specific markups: `<Point>...</Point>`.

This XML structure must be complete to be exploited during the planning process. That is why the completion process has to add the missing data according to the knowledge base of our system.

3. SEMANTIC COMPLETION

The input of this process is a semantic model. Its purpose is to check if the XML structure of this model is complete. In such a case, the semantic model is called *generalised semantic model*. Otherwise the completion process has to add the missing data using the *knowledge base* to generate one or several generalised semantic models. This processing is divided into two parts: the *symbolic completion* and the *numeric completion*.

3.1 The Knowledge Base

The knowledge base stores all the available data used over all the processing steps. The information is organised using the XML language. We can find the definition of all the available shapes of zone. Their borderlines are defined by NURBS curves to be able to obtain several different rounded shapes.

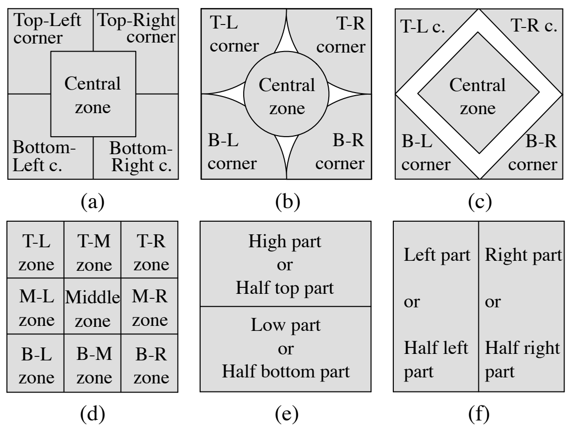
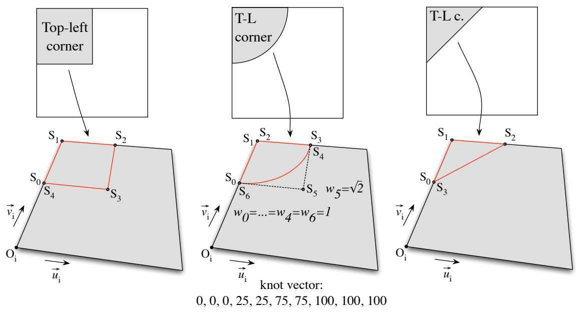


Figure 5: Some examples of absolute localisation.

All the available absolute localisations (see Figure 5) are also stored on the knowledge base. For one localisation, it is possible to find several different *settings* like the `Top-Left corner` illustrated in Figures 5.a, 5.b and 5.c. Each available setting is defined in the knowledge base by a specific closed NURBS curve (see Figure 6).



(a) Square setting (1-degree)
(b) Round setting (2-degree)
(c) Triangular setting (1-degree)

Figure 6: NURBS definitions of different settings of the `Top-Left corner` localisation.

The knowledge base contains the *graph of dependences between deformations* written G_{dbd}. This graph stores the order of deformations to apply if many of them have to be performed on a same area of a surface. The nodes of G_{dbd} are the available deformations of our system. A single oriented link from a deformation *A* to a deformation *B* means that *A* must be applied before *B*. A double oriented link means that two cases must be considerated, the case when *A* is applied before *B* and the case when *B* is applied before *A*. A part of the G_{dbd} graph is shown in Figure 7. The deformation abbreviations used are:

Ar:	Arched	*Bl*:	Bulged	*Lu*:	Lumpy
Ho:	Hollowed	*Hl*:	Hole		

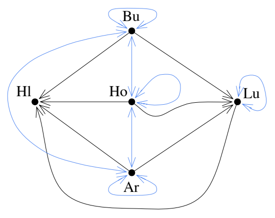


Figure 7: Part of a *graph of dependences between deformations*.

The lumpy deformation must be done at the end. We made this choice because this deformation is viewed as a finishing touch.
The knowledge base also stores all *quantifiers* the designer can use in the description like "LITTLE" or "VERY". The goal of a quantifier is to shade properties, localisations and deformations according to a value from 0 to 100. By definition, a declarative approach must allow the user to give a fuzzy description [Des00]. In order to keep this very important aspect of the declarative design, each quantifier is assigned to a fuzzy set (see Figure 8).

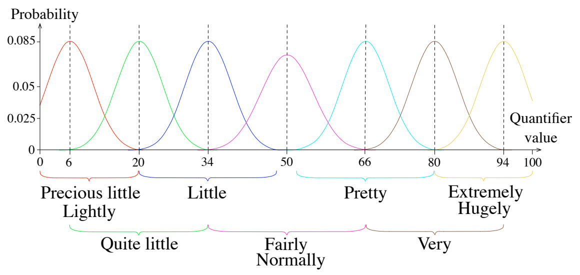


Figure 8: Fuzzy sets assigned to each quantifier.

Each fuzzy set is here defined by a Gaussian probability density function. Using all these data stored in the knowledge base, the completion process begins by the *symbolic completion*.

3.2 Symbolic Completion

In most cases, the designer gives to the system an incomplete description. This can happen because he/she forgets an information or because the data seem to be obvious for him/her. Nevertheless, it is not so obvious for the system. Thus, the latter has to find and complete all the missing data of the current semantic model. The technique is iterative. Each iteration consists in applying several simple rules on existing data to infer the missing ones. This is for example two rules to enforce:

- *If a zone has no deformation, the deformation "NONE" is added to this zone,*
- *If a deformation has no quantifier, the quantifier "FAIRLY" is added to this deformation.*

Some other rules entail the generation of several more precise semantic models. For example, if the designer does not give the shape of a zone or does not specify the localisation setting to consider, the system must generate a new semantic model for each possible shape or setting. The symbolic completion is carried on using all these new models. The process is stopped when no rule can anymore be applied. At the end of the symbolic completion, several semantic models are available and the numeric completion is applied on each of them.

3.3 Numeric Completion

This step adds to each semantic model the numeric data required to be processed. The method is not iterative. Each model is processed one by one using several simple rules, like for example:

- *If a quantifier has no specified values, the corresponding data are added from the knowledge base,*

- *If the control polygon of a zone is not present, all its control points are placed on the corresponding local reference frame according to the localisation and the shape of this zone.*

The second rule is one of the most important because the control polygons define the zone borderlines which are principally used during the *planning process* and the *construction process*. After the numeric completion, each semantic model is complete and represents the knowledge of the final shapes the system will produce. These full models are called *generalised semantic models* and each of them will be taken into consideration by the two last steps.

4. CONSTRUCTION PLANNING

A generalised semantic model is made of zones which are linked to one or many deformations. If many zones overlap each other, their deformations have to be applied on the shared area. The deformations are usually not commutative as illustrated in Figure 9.

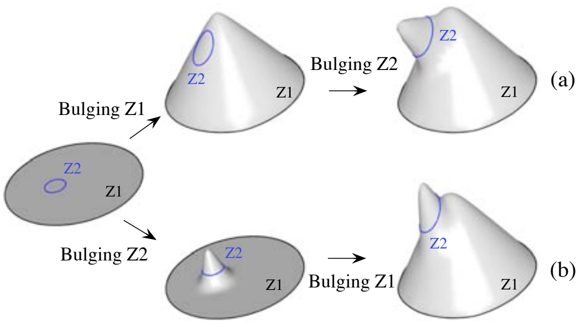
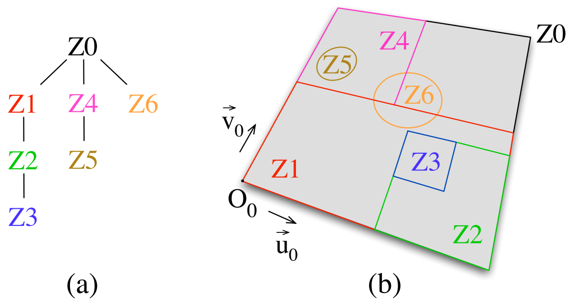


Figure 9: Non-commutativity of the "bulged" deformation.

That is why, it is possible to obtain several kinds of solution according to the overlaps and the deformations of each zone. The purpose of the planning process is to generate all possible sequences of deformations in order to obtain distinct solution surfaces. This is done through the analysis of the *graph of dependences between zones* written G_{dbz} which stores the ordering constraints of application of deformations to all the zones of the surface. Each node of this graph is an operation $(zone, deformation)$ which can be read "*Apply the deformation called $deformation$ to the zone called zone*". If a zone has two deformations, the graph has two nodes which involve the same zone but with a different deformation to each of them. The edges of this graph are symbolised by two kinds of links, le single oriented link and the double oriented link which meanings are similar to the ones of the graph of *dependences between deformation* stored in the knowledge base (see section 3.1). Figure 10.c illustrates the *graph of dependences between zones* obtained from the structured tree of Figure 10.a and the position in R_0 of the corresponding zones which is shown by Figure 10.b. This example uses these data: *Z0 is lumpy, Z1 is arched, Z2 and Z5 are hollowed, Z3, Z4 and Z6 are bulged.*



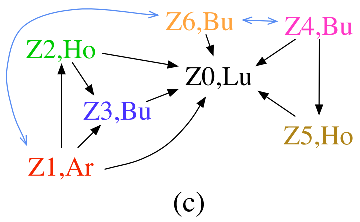


Figure 10: (a) Structuring tree, (b) possible zone layout and (c) the corresponding graph of dependences between zones.

The aim of the analysis of G_{dbz} is to find all the construction sequences of $(zone, deformation)$ to apply in order to obtain all the different solution surfaces. A single oriented link in G_{dbz} is an ordering constraint. Thus if we have (*Z2*,*Ho*)→(*Z3*,*Bu*), the resulting sequence is (*Z2*,*Ho*)(*Z3*,*Bu*) which can be read: "First *Z2* is hollowed, next *Z3* is bulged". A double link in G_{dbz} means that two cases are possible and gives different results. Thus if we have (*Z1*,*Ar*)↔(*Z6*,*Bu*), a first class of solutions is the sequence (*Z1*,*Ar*)(*Z6*,*Bu*) and a second class of solutions is (*Z6*,*Bu*)(*Z1*,*Ar*). This last step analyses all these combinations and infers all the different classes of solutions corresponding to G_{dbz}. One class of solutions can contain several construction sequences which are equivalent. That is, if all the deformations of each construction sequence are applied, the resulting surfaces are strictly the same. One construction sequence is randomly chosen in each class of solutions to represent it. This chosen sequence comes in addition to the generalised semantic model to become a *template model*. The graph analysis of Figure 10.c generates four classes of solutions represented by these four template models:

- → `Template model 1: (Z6,Bu) (Z4,Bu) (Z5,Ho) (Z1,Ar) (Z2,Ho) (Z3,Bu) (Z0,Lu)`
- → `Template model 2: (Z4,Bu) (Z5,Ho) (Z1,Ar) (Z6,Bu) (Z2,Ho) (Z3,Bu) (Z0,Lu)`
- → `Template model 3: (Z4,Bu) (Z6,Bu) (Z1,Ar) (Z5,Ho) (Z2,Ho) (Z3,Bu) (Z0,Lu)`
- → `Template model 4: (Z1,Ar) (Z2,Ho) (Z6,Bu) (Z4,Bu) (Z5,Ho) (Z3,Bu) (Z0,Lu)`

At the end of the planning process, several template models can be generated. To present one possible solution surface per template model, the system makes an instance of each of them and shows them to the designer. The surface modelling is done by the construction process.

5. CONSTRUCTION OF SURFACES

The input of the *solving and construction engine* is one template model. This process consists in solving the soft and the strict constraints given by the designer. At the first time, our method consists in creating a surface of degree 3×3 defined by 4×4 control points which are placed on a plane and two uniform clamped knot vectors. The parametric plane of this surface and the reference frame of the root zone R_0 are merged to link the zone definitions with the surface. This is the initial surface which will be deformed to obtain one instance of the current template model. Each deformation needs control points to be performed. That is why the net of control is refined by adding lines and columns of control points inside the zones which have to be deformed. This operation is done using the knot insertion algorithm developed by Boehm [BP85]. The control points which are inside the borderlines of a zone are assigned to it in order to be moved during the deformation of this zone.

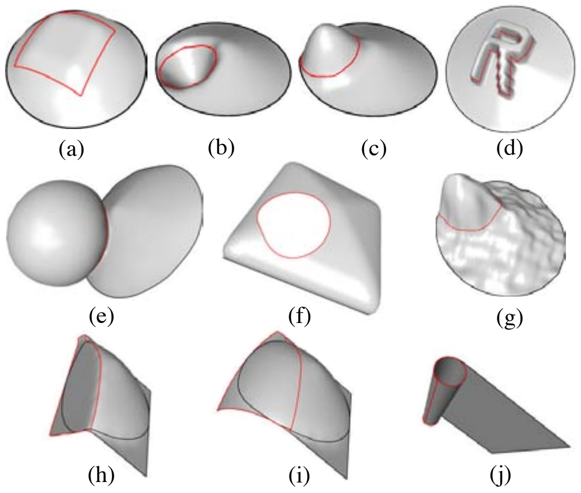


Figure 11: Available deformations: (a) $flattened$, (b) $hollowed$, (c) $bulged$, (d) $extruded$, (e) $inflated$, (f) $hole$, (g) $lumpy$, (h) $folded$, (i) $arced$ and (j) $rolledup$.

In a second time, our method solves the soft constraints sculpting the global shape of the surface. This step consists in deforming the net of control of the surface defined by refinement. Each operation ($zone$,$deformation$) is done one by one applying the $deformation$ to the corresponding $zone$ following the construction sequence of the current template model. Many deformation techniques [PT97, Per04] can be used. However, we choose to develop our geometric tools to be closer to our needs. The current available deformations of our declarative modeller are illustrated in Figure 11. Each deformation is set by its qualifier stored in its semantic model. Because of the fuzzy approach of the qualifiers, each launching of the current engine can generate several different[2] instances of the same template model. At the end of this construction step, the surface is a sketch of the designer expected surface. Figure 12 illustrates one instance of template 4 of the last example with its net of control.

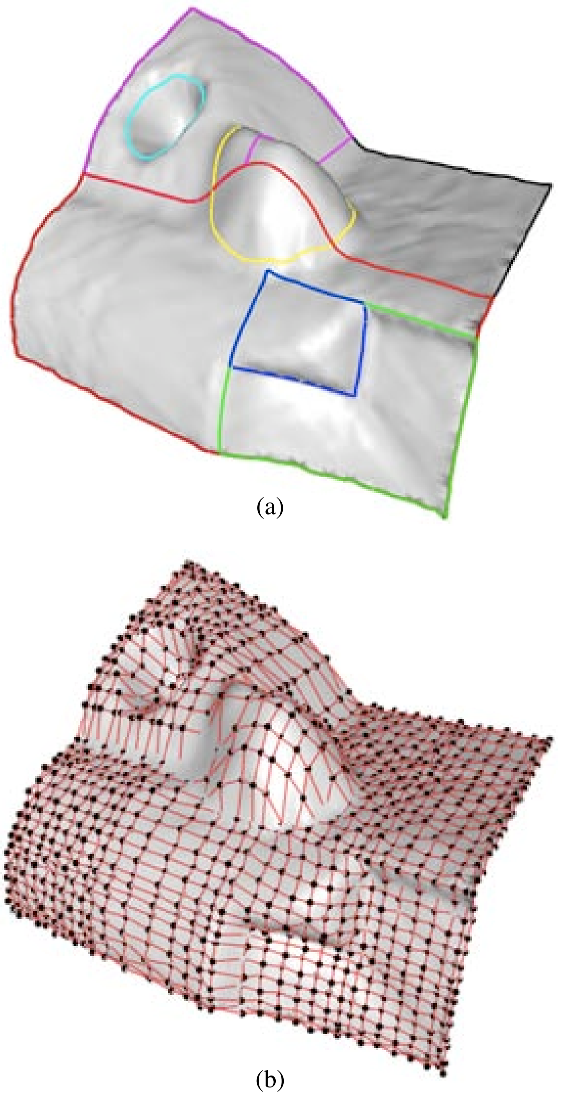


Figure 12: Template model 4 of the previous example with its zones (a) and with its net of control (b).

In a third time, the engine solves the strict constraints[3] using a geometrical solving method we developed [LDB05]. This method is able to set the range of each punctual constraint in order to control the

[2] but pretty close.

[3] passing through specific points in space.

local deformations of the surface. The method can also be applied to satisfy several constraints. This case is illustrated in Figure 13 where a constraint is symbolised by a cross.

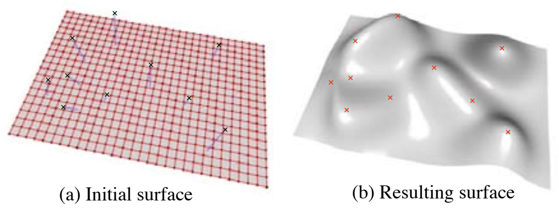


Figure 13: Deformation of a B-spline surface of degree 3×3 defined by 30×20 control points to satisfy ten constraints with a medium range of influence.

At the end of the construction process, one or many instances of resulting template models are presented as solution surfaces to the designer. He/she can choose the one which is closest to his/her requests. If no surface satisfies the designer, he/she can modify the description and start again the declarative system. In order to illustrate the current capabilities of our system, the next section presents two practical results.

6. FIRST RESULTS

The two practical results introduced in this section were obtained using our first prototype of declarative modeller of surfaces. We give it a possible generalised semantic model and it proposes us one or several template models showing one instance of each of them.

6.1 Hood of car

The first example consists in modelling a surface close to the car's hood of Figure 14.

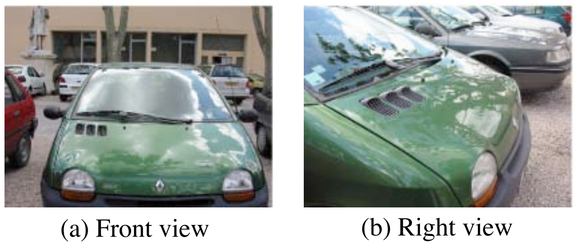


Figure 14: The expected surface.

A piece of a possible description of this surface could be:

"The surface is not very wide. Its main low part, called Z1, is a little arched. The very low part of Z1, called Z2, is very arched ...".

Figure 15 presents one possible framework answering to the complete description. It shows a structuring tree (a) and one possible location in R_0 of the zones (b) involved in the description where *"Z1 and Z2 are arched, Z3, Z4 and Z5 are bulged and Z6, Z7 and Z8 are three holes"*.

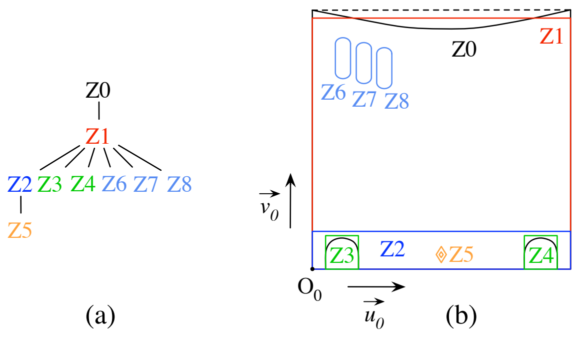


Figure 15: A possible framework.

These data could be stored in a generalised semantic model after the translation process and the completion process. Using this information, the planning process and the construction process give us four template models. These models are very close to each other. The main differences between all of them are around Z3 and Z4. Figure 16 presents one possible instance of the first resulting template model.

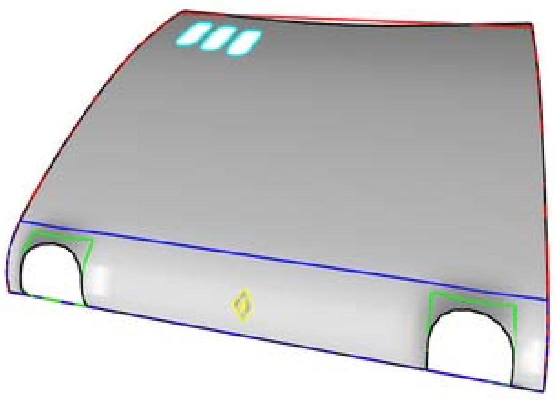


Figure 16: Instance of the first resulting template model with its zones.

6.2 Streamlined Motorcycle

We try to design a part of the streamlined of a motorcycle (see Figure 17.a). This part is illustrated in Figure 17.b.

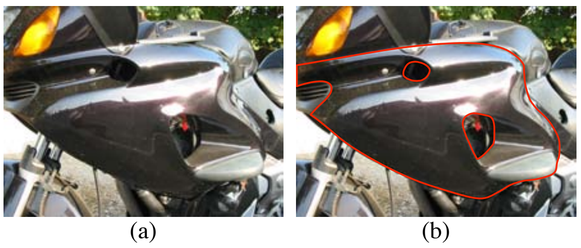


Figure 17: (a) Streamlined of the motorcycle and (b) borderlines of the chosen part.

We give our modeller a possible generalised semantic model storing all the requested deformations and zones like it is shown in Figure 18.a. All the resulting solutions are close to the expected surface and can be considered as fine sketches. Figure 18.b presents one possible instance of the first template model proposed by our system and shows its zones. Figure 18.c illustrates the net of control which is generated to obtain this result.

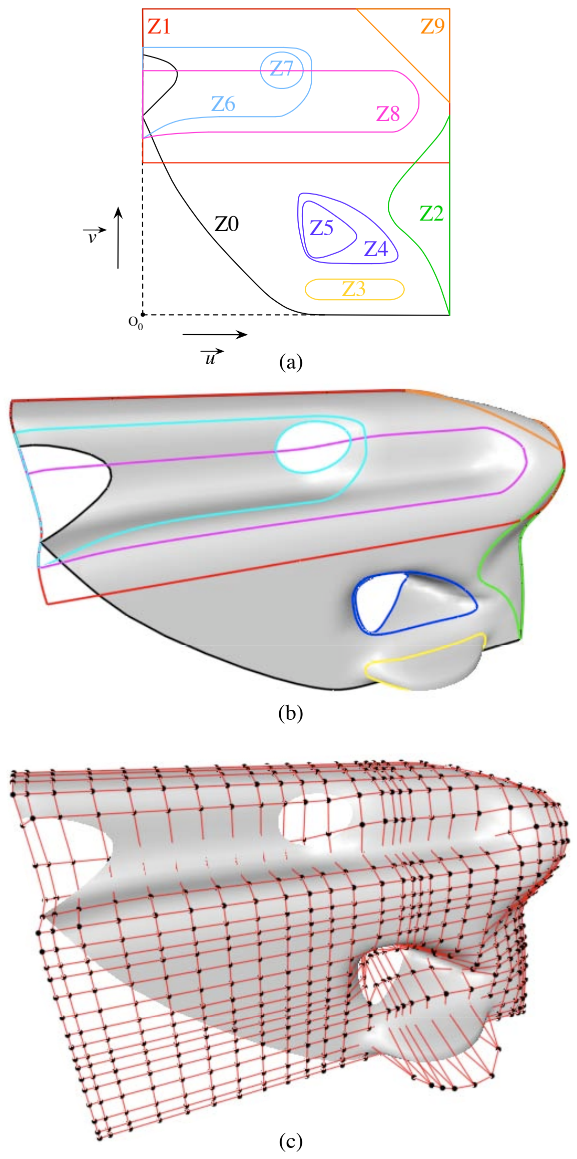


Figure 18: (a) A possible framework corresponding to the expected surface, (b) Instance of the first resulting template model with its zones and (c) its net of control.

7. FUTURE WORKS

We outlined a declarative system able to design parametric surfaces from one or many given descriptions. The implementation of this project is currently in progress and can already give some very interesting results.
The final purpose of our study is to have a complete declarative modeller to design objects using many patches of NURBS surfaces. In order to reach this goal, it is fundamental to develop interfacing modules giving the designer the uttermost means to describe the surface or the object he/she expects. A second very important point is closely related to the declarative approach. Using this method, the user does not need to have any knowledge about the modeller or about the underlying mathematical models. The software has to adapt itself to the designer: a learning process integrated to our system could be a worth feature. A third perspective consists in improving the number of soft and strict constraints currently available in order to be closer and closer to the designer requirements.
We are thinking this tool is a new way of considering modelling process which could be very helpful to designers in a near future.

8. REFERENCES

[BP85] W. Boehm and H. Prautzsch. The insertion algorithm. *Computer Aided Design*, 17(2):58–59, 1985.

[CDMM97] C. Colin, E. Desmontils, J.Y. Martin, and J.P. Mounier. Working modes with a declarative modeler. *Compugraphics*, 1997.

[Dan96] Marc Daniel. Declarative Modeling of fair shapes An additional approach to curves and surfaces computations. In J. Hoschek and P. Kaklis, editors, *Advanced Course on FAIRSHAPE*, pages 77–85. B. G. Teuber Stuttgart, 1996.

[Des00] Emmanuel Desmontils. Expressing constraint satisfaction problems in declarative modeling using natural language and fuzzy sets. *Computers & Graphics*, 24(4):555–568, 2000.

[EH04] Mathieu Estratat and Laurent Henocque. Parsing languages with a configurator. In *European Conference on Artificial Intelligence*, pages 591–595, Valencia, 2004.

[LDB05] R. La Greca, M. Daniel, and A. Bac. Local Deformation of NURBS Curves. In M. Daehlen, K. Morken, and L. L. Schumaker, editors, *Mathematical Methods for Curves and Surfaces Tromso 2004*, pages 243–252. Nashboro Press, Brentwood, TN, 2005.

[Per04] Jean-Philippe Pernot. *Fully Free Form Deformation Freatures for Aesthetic and Engineering Designs*. Phd, Institut National Polytechnique de Grenoble, 2004.

[PT97] L. Piegl and W. Tiller. *The NURBS Book 2nd Edition*. Springer, Berlin, 1997.