Responsive Interaction Based on Sketch in Concept Styling

Li Han[1], Giuseppe Conti[2], and Raffaele De Amicis[3]

[1,2,3] Graphitech, Via Dei Molini, 2, 38000, Trento (TN), Italy
{li.han, giuseppe.conti, raffaele.de.amicis}@graphitech.it

Abstract. In the CAS/CAD field, the increasing adoption of Spline-based free-form methods to generate surfaces, has introduced a higher degree of freedom into the design process. However, on the other hand, this evolution has made the process of creating and manipulating surfaces more complex. For this reason a much more intuitive and intelligent task-centered and user-centered interaction paradigm is required. This paper presents a responsive interaction technique which adopts a sketch-based interface capable of exploiting the stepwise refinement process typical of conceptual designing. Further it makes use of adaptive user modeling techniques by introducing an innovative adaptive decision-tree structure for top-down designing. We illustrate the implementation of the algorithm and we highlight its efficiency and feasibility for its adoption within Sketch-Based Modeling Systems (SBMS).

1 Introduction

The capability to design innovative products is a key factor to foster the competitiveness of industrial products. In particular, the conceptual design phase plays a strategic role since the creativity and synthesis, which characterize it, are of great importance for the design of an industrial product. Furthermore it has been estimated that up to three quarters of the designing costs are generated during the initial phases. Therefore, it appears clear that, in order to boost efficiency and enhance creativity innovative and adequate tools have to be developed.

In the CAS/CAD field the adoption of Spline-based free-form surfaces has provided designers with greater freedom. However on the other hand, this has introduced a higher complexity in the process of creation and manipulation of shapes since the full exploitation of Splines' advantages it requires knowledge of their mathematical representation.

Recent years' experience suggests that a great improvement to design such tools can be introduced by improving their level of "intelligence", i.e. the capability to discern the commands expressed by the user. This new tendency will yield a new generation of systems capable to dynamically adapt to designers' needs, in opposition to current systems that requires designers to adjust to the technology adopted. The new generation of design system in fact should be able to understand the designer's behavior. That is, they should be able to comprehend the users' actions, the way they identify a shape, the way they interact with the drawing tools during the design process. Finally such a system should present the information consequently provided to the user in a flexible, efficient and supportive manner.

M.F. Costabile and F. Paternò (Eds.): INTERACT 2005, LNCS 3585, pp. 94 – 105, 2005.

The aforementioned requirements have fuelled recent years' research on sketch-based modeling systems. These allow the user to quickly create 3D models by simple freehand strokes rather than by typing in parameters. The systems developed range from those pursuing a constraint-based or feature-based approach [1-3, 12, 13], to those providing different forms of suggestive feedback and VR rendering [4-5, 14-15]. However, the majority of these systems have been designed as further extension of classical CAD tools. Therefore these cannot be used when information at a higher, semantic level is to be decoded and manipulated. Furthermore these systems propose interaction metaphors far from the designer's traditional approach, typically featuring a top down and stepwise refinement process.

This paper tries to bridge this gap by proposing a formal theory which models the process of comprehension of the user's sketches. Such methodology, applied to conceptual designing, is able to support the stepwise process of sketching by continuously interpreting the flow of data and constraints generated by the designer's actions. Further the approach developed is capable to dynamically adapt to the different users' styles by constantly modeling their personal behaviors. The resulting methodology has being implemented in an architecture described throughout the paper.

The paper is organized as follows: in section 2 we present the related works whilst section 3 introduces a general architecture for sketch-based modeling systems (SMBS) that integrates the *interface module*, the *user adaptation module* and the *rendering module*. In section 4 we will detail the developed interaction algorithm. This is based on sketching and features stroke recognition, dynamic shape modeling and to multi-user adaptation. Finally in section 5 we conclude with the summary and we describe the directions of future work.

2 Related Work

In the last two decades, computer-aided design/styling (CAD/CAS) systems have developed powerful 2D and 3D modeling functionalities. However conventional interfaces based upon the WIMP (Windows Icon Menus Pointer) paradigm have proved to be inadequate being cumbersome, tedious and time-consuming. Sketch-based free-drawing interfaces, combining the ease of freehand drawing with the advantages of computer processing, are becoming prevailing. In particular these allow designers to freely sketch shapes as they would on ordinary paper. This allows designers to fully concentrate on the design process.

Earlier sketch-based systems allowed sketching of geometric models by using a set of pre-defined gestures. Gesture recognition algorithms converted sketches into a series of specific commands to create and manipulate geometric primitives.

The system described in [5] defines geometric features of objects through the drawing of a set of auxiliary lines. It supports over-sketching of lines and both snapping and adjustment are preformed in real time. However the system is not capable to process sophisticated input techniques necessary to generate curve and surface. The work in [4] introduces a new type of so-called "suggestive" interface for 3D drawings: it extends the gesture interface by offering multiple candidates. The system is easy to use although inefficient to understand the user's intentions.

Recently, research has adopted constraint-based methods to resolve unclear sketching input. The work in [3] presents a simple touch-and-replace technique to edit 2D and 3D curves. The authors introduce auxiliary surfaces that allow a reliable interpretation of the users' pen strokes in 3D. The main limitation of this approach is the lack of degrees of freedom, which are restricted by the auxiliary surfaces.

In the literature [6] Teddy proposes an easy way to allow the users to draw 2D silhouettes and the system automatically proposes a 3D surface using polygonal mesh whose projection matches the object contour. However it cannot create multiple objects and it is the impossible to some editing operations.

In [7-8] the authors suggest a Variational Implicit Surfaces (VIS) representation. Here surfaces are defined by a set of constraints that specify the points on the surface boundary. The modeling and editing operations are easily controlled further the method is fast when designing 3D approximate models. However this method is not sufficient for full-featured free-form shape modeling. Furthermore, as the number of constraints increases, the time the algorithm takes to compute the coefficients for the variational implicit surfaces grows considerably.

The main limit of existing modeling tools lies in their lack of the interaction between refinement cycles which take place during the process of designing. Further they show limited intelligence during the decision-making process which takes place during the free-form drawing.

Recently another approach, the so-called declarative designing method, has received increasing interest across the CAD research community [9-10]. As illustrated in Fig. 1, the method provides designers with a more progressive and dynamic model specification based on declarative (rather than imperative) methods. This permits to describe models by means of rules, properties or constraints that are computed by successive refinements, to adopt the logical rules to be dealt with, and finally to provide a solution among the potential ones. To a certain extent this method provides a formalism, which integrates architectural rules and attributes to provide the designer with an interactive graphic language. However, these methods are incomplete, i.e. they do not present specific knowledge representation and, in particular, they do not provide details of the interaction mechanism and of the way the system deals with complex objects.

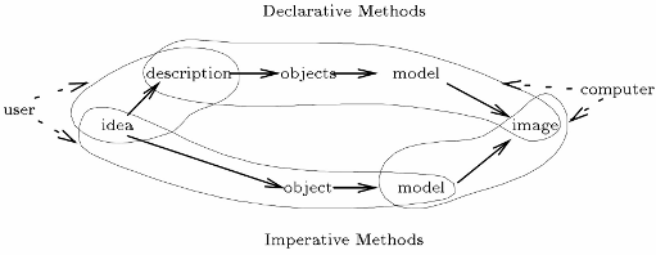


Fig. 1. The comparison between declarative and imperative model [10]

A distinctive feature of current sketch-based interfaces is their continuous visual feedback. This usually involves successive transformations of graphic objects and some form of suggestive rendering between the user and the tools.

3 A General Architecture for Sketch-Based Modeling Systems (SBMS)

We consider the process of sketching as the information flow from/to the designer's brain. For this reason the sketch-based system developed had to be able to show the evolution of the corresponding designing behavior by providing intelligent reasoning of the user's input. As illustrated in Fig. 2, the proposed architecture for such sketch-based modeling system consists of an *interface module*, a *user adaptation module* and *rendering module*.

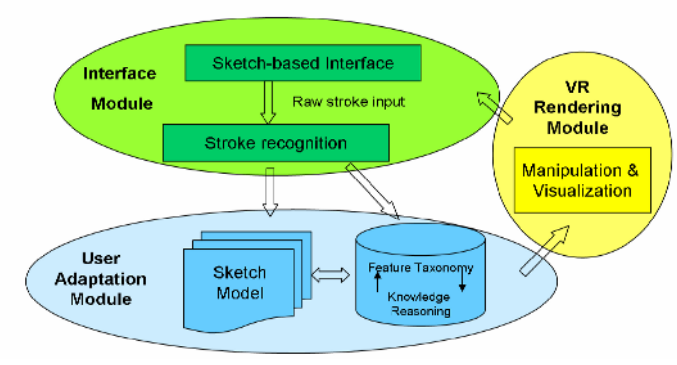


Fig. 2. The proposed architecture for a sketch-based modeling system

The *interface module* provides the interactive behavior and it supports stroke recognition. Since usually the raw stroke input information is unclear, the module plays a key role for the effective extraction of geometric features and constraints. At this stage we analyze the input and we obtain the speed, length, vertex sets, other object attributes so on. Then, based on the evaluation of these features and corresponding fuzzy sets, we assess the shape and save it within a *shapeList*. As illustrated in the class diagram of Figure 3, which shows the main components' features, as a result the system is capable to generate a number of geometric primitives, which are sent, together with other data such as topological constraints, time sequences etc., to the *user adaptation module* for further processing.

The *user adaptation module* accepts such basic information (e.g. geometric and topological constraints, basic geometric entities and time sequences) and it processes it through a stepwise refinement process. To do so the module constantly extracts attributes from the latest sketches arriving from *interface module* through a series of real-time operations. This data is then transformed according to the constraint solving model adopted. This is based on the characteristic features and behaviors typical of the conceptual stage. The *user adaptation module*, which represents the center of the whole system, combines adaptive reasoning with dynamic user modeling in order to deal with the uncertainty typical of hand drawn sketches. As a result, the module, which takes into account the differences between users' drawing styles, decodes the corresponding design behavior and it inserts its different possible interpretations into a ranked probability list. The user then can decide, among those suggested by the system, the most appropriate action to be taken.

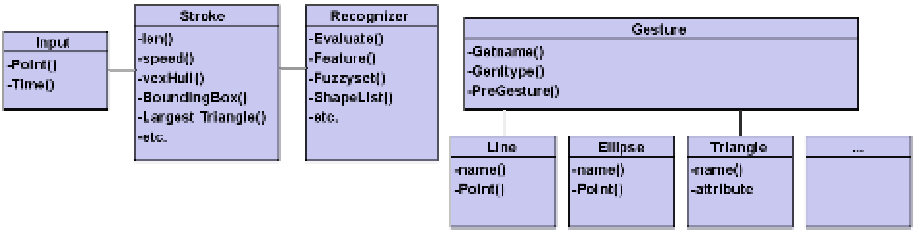


Fig. 3. Class Diagram for the Stroke Recognition

Finally, the *rendering model* performs the post-processing and visualization process allowing the use of tailored representation such as non-photorealistic rendering or alike.

Throughout the following sections we will illustrate the details the interaction mechanism proposed, we will describe the algorithms developed and we will discuss their strengths and weaknesses.

4 Responsive Interaction Based on Sketching

In order to provide to be able to retain the difference between users' drawing styles and to enable user to generate graphic objects of higher complexity, we have introduced a "responsive" interaction mechanism which can dynamically adapt to the constraints defined by the user's input. As described in the following pages, its implementation is based on an innovative decision-tree structure.

4.1 Responsive Interaction Based on Automatic Sketches Identification

In order to deal with the unclarity typical of the sketched input in the initial stages of the designing, we have modeled the data flow which characterizes the early design process according to a mechanism where on-line recognition, user adaptation and decision making are performed at the key stages as illustrated in Fig. 4.

The process of recognition, which is performed at the *interface module* level, is made of a stroke and shape recognition sub-processes. When each stroke is processed by this first stage, the system automatically activates the *user adaptation module* which performs the constraint reasoning and matching process according to the different characteristics typical of each user's drawing style. Finally the system, after suggesting a list of possible solutions, asks the user for feedback to confirm the command.

The adaptive constraint reasoning process takes places at two levels: either by the *stoke recognition process* (for single-stroke identification) or by the combination of *stoke recognition process* and *user adaptation module* (for more complex shape understanding). The latter is responsible to compare the information contained in the relevant model of the user with the information contained in a database of geometric templates organized by parameters and constraints. During this process the stroke represents the base unit of the response mechanism. When the pen is lifted the system automatically reacts by extracting the relevant features and, if required, by performing

the adaptive reasoning. As a result of this first process a graphic object is produced. Then the interaction module asks the designer for feedback, i.e. the user has to confirm whether he/she wants to continue renewing the previous stroke or he/she wishes to begin a new object.

The details of this process are illustrated in Table 1 where each raw stroke input information is represented by a 4-tuple $G\ (L_i, S_i, P_i, V_i)$ where L is the length of the stroke, S is the speed, P is the point set extracted, V is the vertex set extracted. We also assume that T_i represents a graphic object while C_i is a constraint. From Table 1 it is possible to see how, during STEP 2, the basic geometric object T_i which satisfies a series of specific constraints is retrieved through the stroke recognition process. If the user's sketch is not a single stroke (see STEP 4), the constraint solving module is called to analyze the history of the previous strokes and to perform the constraint matching. As a result the T_m, which represents the most suitable graphic object, is selected and used for higher level analysis.

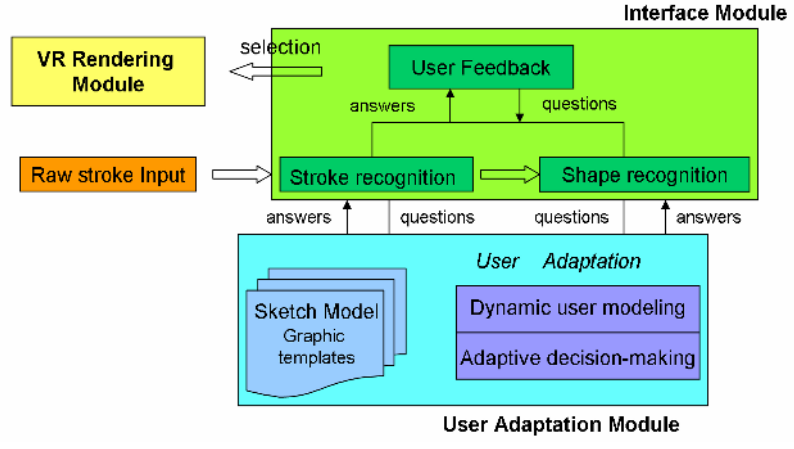


Fig. 4. The framework for the responsive interaction mechanism

Table 1. The responsive interaction algorithm based on adaptive reasoning

STEP1: Raw stroke information $G\ (L_i, S_i, P_i, V_i)$,
STEP2: Stroke recognition: $G\ (L_i, S_i, P_i, V_i) \rightarrow T_i\ (C_j)$
STEP3: If (Single-stroke) then GOTO END.
Else GOTO STEP4
STEP4: Composite constraint reasoning: $C_m\ (T_i \wedge \Lambda T_{i-1} \wedge \ldots T_1) \rightarrow T_m\ (T_j\ (C_k), \ldots T_n(\ C_l), C_m)$
STEP5: Check T_m whether it exists in the pre-defined model database:
If (!existed) then INSERT (T_m)
Else if (Continue) then GOTO STEP1
Else GOTO END.
END: Results confirmation by user's feedback and visualization.

4.2 Intelligent Stroke Interpretation System

It is known that designers tend to draw shapes through several primitive sub-shapes. Likewise they tend define a primitive shape either by a single stroke or by several consecutive strokes. In our recognition algorithm, we first discover latent primitive shapes among user strokes. Then we recognize and regularize them and, finally, we show the regularized drawing on the screen. After being recognized and regularized, the primitive shapes which belong to the same graphic object are grouped together. Eventually they are segmented and combined to form an object skeleton.

In our system the geometric entities are positioned using parameters and constraints rather than using a specific coordinates and spatial orientation. Each time we extract the features' parameters from the free-form drawing and then we match them on the basis of a probability rank.

Specifically, in order to recognize the users' stokes, we employ an SVM-based (Support Vector Machine) incremental active learning method [16]. Partial structural similarities are calculated between the graphic object being drawn and the candidate ones in the database. The most similar graphic objects are then suggested to the users in a ranked list in order for them to choose from and confirm the command. The responsive feedback strategy makes the user interaction smoother and more natural. Further it has the extra advantage to reduce inner-stroke and inter-stroke noise, which are generated when the user is not a professional or proficient in the task. As a result of this process we then get regularized stroke segments or objects, which are regarded as parts of a composite object.

The presence of a high number of shapes increases difficulty of sketch understanding since it can be difficult for the system even to assess correctly whether or not a shape is completely drawn. Likewise it is essential to make the system automatically re-organize the relevant features when a synchronous editing operation takes place. The answer to these issues is provided by the adaptive decision-making system which is described in the following section.

4.3 The Process of Adaptive Decision-Making for Shape Understanding

The process of shape understanding is grounded upon an "ad hoc" internal model which adopts a decision tree to arrange and organize parameters and constraints. As illustrated in Fig. 5 the tree contains basic constraints (geometry primitives, geometric relationships, algebraic relationships) as well as high-level features which are composed of basic constraints used for describing the operation behavior. All the features and parameters are arranged starting from the low-level constraint layer to a high-level feature layer. Sandwiched between these two levels, the constraints are cross-checked and shared, until eventually the semantic descriptions are attached.

This is a reversed tree structure where the top level contains the leaf nodes whilst the bottom level holds the root. Leaf nodes represent a series of basic geometric primitives and constraints. Furthermore, in order to better capture the user's sketching commands, we have adopted an adaptive method which adapts the decision tree according to dynamically upgraded user models. Specifically, we set the attribute of being a "possible shape" to each branch node whilst the root represents the final geometric object which is reached only when the constraint branches are satisfied. The

user tree is created to match with the pre-defined decision tree of database. As a result if we find that the model is not present in the pre-defined decision tree within the database, we place the object into the relevant tree as a new node. Then the decision tree within the database is synchronously updated.

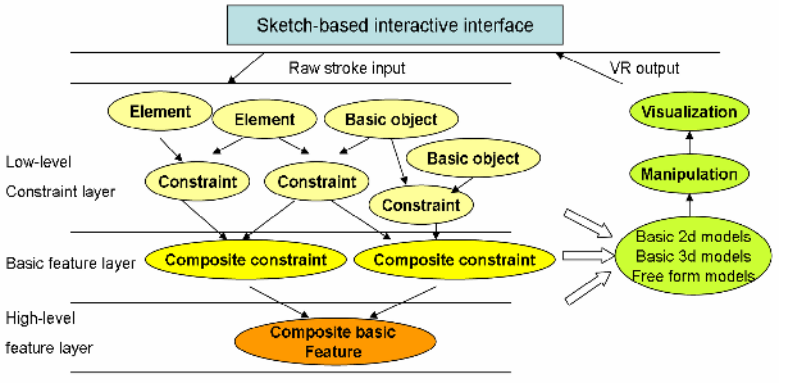


Fig. 5. The structure of the user adaptation module

An example of this process, illustrated in Fig. 6, will help better illustrate the process. In the illustration the indexes from 1 to 11 refer to different constraints. Specifically we suppose that constrains C_7, C_8, C_9, C_{10} are used to obtain general geometrical relationships between graphical objects, whilst C_1, C_2, C_3, C_4, C_5, C_6 are used to recognize the geometric primitives. Let T be a geometric object set, which includes all the nodes in the graph. The raw stroke input information is represented as a 4-tuple $G(L_i, S_i, P_i, V_i)$, where L is the length, S is the speed, P the point set and V the vertex set. As defined in [16], for each single stroke a number of specific features are extracted and the relevant constraints are analyzed. As a result of this process a number of regularized geometric primitives are produced whose relative logic representations are listed below:

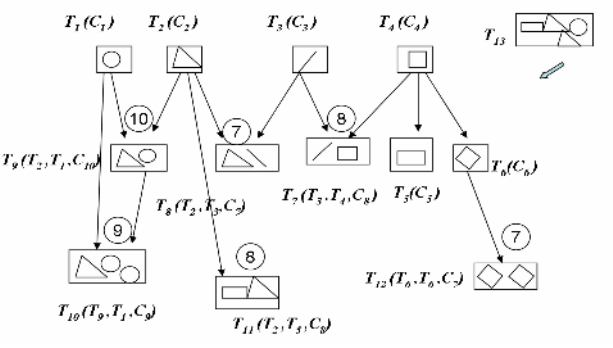
- $C_1(L_1, S_1, P_1, V_1) \rightarrow$ *circle;* $C_2(L_2, S_2, P_2, V_2) \rightarrow$ *triangle;*
- $C_3(L_3, S_3, P_3, V_3) \rightarrow$ *line;* $C_4(L_4, S_4, P_4, V_4) \rightarrow$ *square;*
- $C_5(L_5, S_5, P_5, V_5) \rightarrow$ *rectangle;* $C_6(L_6, S_6, P_6, V_6) \rightarrow$ *diamond;*
- $C_7(X, Y) =$ *(X is overlap to Y)* $\wedge$ *(X is precede Y)* $\wedge$ $(X \in T; Y \in T)$;
- $C_8(X, Y) =$ *(X is parallel to Y)* $\wedge$ *(X is precede Y)* $\wedge$ $(X \in T; Y \in T)$;
- $C_9(X, Y) =$ *(X is Vertical to Y)* $\wedge$ *(X is precede Y)* $\wedge$ $(X \in T; Y \in T)$;
- $C_{10}(X, Y) =$ *(X is touched to Y)* $\wedge$ *(X is precede Y)* $\wedge$ $(X \in T; Y \in T)$;
- $C_{11}(C_8, C_9) = (C_8 \rightarrow T_m) \wedge (C_9 \rightarrow T_n) \wedge (T_m$ *is inside* $T_n)$;

We now assume that a decision tree (see Fig. 6-a) is already defined in the database. Each node represents an object and in particular each leaf node (top-level) is a geometric primitive, whilst each branch node is a possible shape (where the index refers to the relevant constraint). For instance in the scene T_9 (Fig. 6-a, second level on the left), where a triangle is adjacent to a circle, the constraints such as C_1, C_2, C_{10}

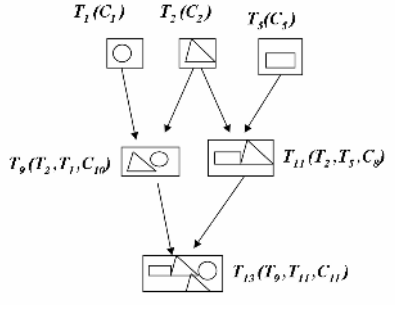
must be satisfied. This condition is represented with the notation $T_9\ (T_2\ (C_2),\ T_1\ (C_1),\ C_{10})$. This way each node can be represented as a multi-tuple $T_i\ (T_p\ (C_j),\ T_q\ (C_k),\ C_n,$ and i, p, q, j k, n, $\in$ *integer*).

When the new object T_{13} is added, a dynamic user model is created. First, based on sketching input, we extract a series of features in time sequence and then, according to the pre-defined constraints, we create the dynamic tree (see Fig. 6-b) and we produce new object information T_{13}, which can be represented as $T_{13}\ (T_9,\ T_{11},\ C_{11})$. Eventually, the new object features are used to update the original tree structure (see Fig. 6-c).

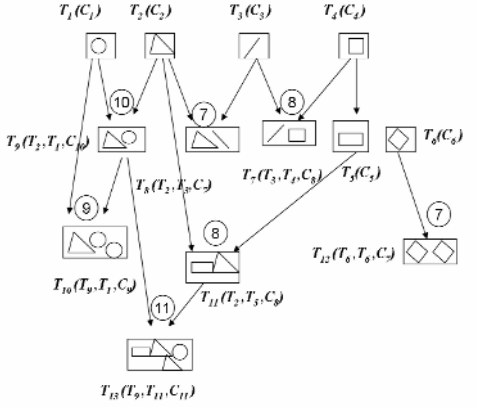
In order to optimize the whole reasoning process, we define the nodes in a pre-defined database as chain structures (see). Each node is represented with two link fields (in **Table 2** referred to as *Pre-Node* and *Curr-Node*) where the first points to the previous stroke node while the second points to the current node. Since the value of any link refers to a node's index, it is possible to find the information of a certain node by following the corresponding index. Finally the *Constraint* field in **Table 2** defines the new node's condition.



(a) Before the new object is inserted (pre-defined decision-tree structure)



(b) Dynamic user tree for new object T_{13}



(c) the new object T_{13} is inserted in database

Fig. 6. The process of adaptive reasoning

Table 2. The structure of the predefined decision-tree storage structure

Index	Name	Pre-Node	Curr- Node	Constraint
1	T_1	*NULL*	*NULL*	C_1
2	T_2	*NULL*	*NULL*	C_2
3	T_3	*NULL*	*NULL*	C_3
4	T_4	*NULL*	*NULL*	C_4
5	T_5	*NULL*	*NULL*	C_5
6	T_6	*NULL*	*NULL*	C_6
7	T_7	3	4	C_8
8	T_8	2	3	C_7
9	T_9	2	1	C_{10}
10	T_{10}	9	1	C_9
11	T_{11}	2	5	C_8
12	T_{12}	6	6	C_7

The specific user modeling and updating process can be described according to the following pseudo-code excerpt:

```
PROCEDURE: USER MODELING
  Begin
    P=NULL, /*Initialize a node pointer*/
    Stop=0, FindFlag=0
    While (! Stop)
          Q= head   /* head of pre-defined link table*/
          Accept (T)  /* T: the current node pointer*/
          Constraint (P, T) • Cn
          If (P•Pre-Node == NULL) then P• Pre-Node = T,
          P• Constraint= Cn
          Else If (P• Curr-Node == NULL) then P• Curr-
          Node = T
          While (Q! =NULL)
            If (Q= = P) then
              P• Pre-Node = Q /*set node to prev node*/
              P• Curr-Node = NULL
              FindFlag=1
          End while
    If (!FindFlag) then GOTO INSERT(P)/*update table*/
    If (Stop) then END
    End while
END PROCEDURE

PROCEDURE: INSERT (P)
  Begin
    Q=End /* the end of the pre-defined link table*/
    Q=ALLOC (new node);
    Q=P
  End
END PROCEDURE
```

If we assume that the number of nodes in pre-defined database is equal to n, then the modeling algorithm and insert procedure are bound by a complexity of $O(n)$. From **Table 2** it is also possible to easily assess the partial structural similarities between users' drawing. This is done by tracking all the pre-node links. If two nodes have the same value then the corresponding nodes will be candidates. For instance, assuming that T_2 is the first stroke, if the system can not extract from the second stroke the exact features then, by tracing the Table 2, it is possible to find that T_8, T_9, T_{11} have the same pre-node link value '2'. As a result of this, T_8, T_9, T_{11} would be offered to the user for selection. The whole process would be still bound to $O(n)$.

In general, this adaptive reasoning method effectively solves the stepwise refinement designing process. In fact the approach proposed is truly incremental since each stroke links to the information about the previous and it depends to the constraints defined up to that point. Furthermore the database can be extended through the insertion of new constraints and new nodes.

The dynamic user tree model, which tracks the information on the strokes, focuses on the extraction of features and matching of constraints, and then it efficiently hides the diversity between different users' input style. When an object is finished the tree model is dynamically stored and the tree is replaced by the one corresponding to next free-form drawing.

5 Summary and Future Work

In this paper, we introduce a sketch-based conceptual design system, which aims at achieving the stepwise refinement process which is typical of the early stages of the design process. The system presented features an adaptive decision-making mechanism which is capable of understanding complex objects thanks to an efficient reasoning method. This approach naturally fits within the normal workflow since it seamlessly integrates within the natural process of sketching. The user in fact is not asked to interact with menus or button. Instead the method, based on the analysis of partial structure similarity, is at the same time efficient and user friendly since it provides designer with a series of possible alternatives to choose from while he/she is sketching. Moreover by improving the representation of the graphical object and the logic behind the reasoning technique, the method presented can easily adapt to the designers' graphical styles making it very robust and reliable to use.

However, although the user modeling and the decision making techniques proposed can support stroke recognition through a progressively evolving process they have, in the need for post-processing, their greatest limit. As a result when an object is changed a great computational effort must be spent to update the constraints.

Future developments will try to solve these issues and will try to further improve the constraint reasoning technique. In particular we will look for a better representation of high-level features through semantic description and we will try to provide the required advanced reasoning technique.

References

1. Zeleznik, R. C., Herndon, K. P. and Hughes, J. F: SKETCH: An Interface for Sketching 3D Scenes. In Computer Graphics (Proc. SIGGRAPH 1996) 163-170.
2. Bloomenthal, K., Zeleznik, R. C.: SKETCH-N-MAKE: Automated machining of CAD sketches. In Proceedings of ASME DETC' (1998) 1-11.

3. Michalik, P., Kim, D., Bruderlin, B.: Sketch-and Constraint-based Design of B-spline Surfaces. In Proceedings of the seventh ACM symposium on Solid modeling and applications. (2002) 297-304.
4. Igarashi, T., Hughes, J. F.: A Suggestive Interface for 3D drawing. In Proceedings of the 14th annual ACM symposium on User interface software and technology (2001) 173 - 181.
5. Contero, M.: Smart Sketch System for 3D Reconstruction-Based Modeling. In Smart Graph-ics Proceedings, Springer Lecture Notes in Computer Science (LNCS) vol. 2733 (2003) 58-68.
6. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: A Sketching Interface for 3D Freeform Design. In Computer Graphics (Proc. SIGGRAPH 1999) 409-416.
7. Karpenko, O., Hughes, J. F., Raskar, R.: Free-Form Sketching with Variational implicit Sur-faces. In Computer Graphics Forum Volume 21, Issue 3 (2002).
8. Rodrigues de Araújo B., Pires Jorge, J. A.: BlobMaker: Free form modeling with Variational Implicit Surfaces. In Proceedings of 12° Encontro Português de Computação Gráfica, Porto, (2003), 17-26.
9. Hagen, P. J. W. and Tomiyama, T. (Eds.): Intelligent CAD Systems. In Theoretical and methodological aspects. Springer-Verlag, (1987).
10. Flemming, U., Coyne, R. F., Glavin, T., Hsi, H., and Rychener, M.: A Generative Expert System for the Design of Building Layouts. In Technical Report EDRC 48-15-89, Carnegie Mellon University (1989).
11. Lucas, M.: Equivalence classes in object shape modeling. In Proceedings of working conference on Modeling in Computer Graphics (1991), 17-34.
12. Catalano, C. E.: Feature-Based Methods for Free-Form Surface Manipulation in Aesthetic Engineering. Ph.D. thesis - Genoa University (2004).
13. Cheutet, V., Catalano, C. E., Pernot, J. P.: 3D Sketching with Fully Free Form Deformation Features (δ-F4) for Aesthetic Design. In EUROGRAPHICS Workshop on Sketch-Based Inter-faces and Modeling (2004), 9-18.
14. Fiorentino M., Monno G., Renzulli, P. A., Uva, A. E.: 3D Sketch Stroke Segmentation and Fitting in Virtual Reality. In International conference on the Computer Graphics and Vi-sion, GRAPHICON, Moscow, Russia (2003).
15. Wesche, G., Droske, M.: Conceptual Free-Form Styling on the Responsive Work-bench. In ERCIM News No.44 (2001).
16. Peng B. B., Sun, Z. X. and Xu, X. G.: SVM-based Incremental Active Learning for User Ad-aptation For Online Graphics Recognition. In The First International Conference on Machine Learning and Cybernetics, Beijing , Nov. (2002) 73-84.
17. Donikian, S. and Hegron, G.: The kernel of a declarative method for 3D scene sketch mod-eling. In Graphicon '92, Programming and Computer Science, Moscow, Russia, September (1992).