Computer-Aided Design 42 (2010) 545–557

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Knowledge-guided inference for voice-enabled CAD

X.Y. Kou, S.K. Xue, S.T. Tan *

Department of Mechanical Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong

ARTICLE INFO

Article history:
Received 3 April 2009
Accepted 15 February 2010

Keywords:
Voice
Speech
Voice-enabled CAD
Knowledge
Synonym expansion
Template matching
Semantic inference

ABSTRACT

Voice based human–computer interactions have raised much interest and found various applications. Some extant voice based interactions only support voice commands with *fixed* vocabularies or *preset* expressions. This paper is motivated to investigate an approach to implement a flexible voice-enabled CAD system, where users are no longer constrained by predefined commands. Designers can, to a much more flexible degree, communicate with CAD modelers using natural language conversations. To accomplish this, a knowledge-guided approach is proposed to infer the semantics of voice input. The semantic inference is formulated as a *template matching* problem, where the semantic units parsed from voice input are the "*samples*" to be inspected and the semantic units in the predefined library are the feature *templates*. The proposed behavioral glosses, together with CAD-specific *synonyms*, *hyponyms* and *hypernyms* are extensively used in the parsing of semantic units and the subsequent template matching. Using such sources of knowledge, all the semantically equivalent expressions can be mapped to the same command set, and the Voice-enabled Computer Aided Design (VeCAD) system is then capable of processing new expressions it has never encountered and inferring/understanding the semantics at runtime. Experiments show that this knowledge-guided approach is helpful to enhance the robustness of semantic inference and can effectively eliminate the chance of *overestimations* and *underestimations* in design intent interpretation.

1. Introduction

Contemporary Computer-Aided Design (CAD) systems are facing increasing pressure to get more work done in less time, so that users can shorten the product development cycle and maintain a lead. To facilitate the realization of high productivity and design efficiency, modern commercial CAD systems are getting more powerful and complex to cater for users' versatile requirements. Even in a simple CAD system, it is not surprising to see hundreds of menu items and toolbar buttons squeezed into limited screen regions. Users are compelled to frequently toggle on/off certain modules, traverse across a number of menu hierarchies or conjecture among many "look-alike" icons, just to get a desired command executed. As the model complexity increases, such seemingly affordable overheads, however, may have significant impacts on the design efficiency and productivity [1]. In addition, users spend much time on the graphical user interface instead of in the design itself, and the design intents are frequently interrupted since considerable attention must be paid to finding the right commands via a sequence of menu/button clicks. In the face of such complex CAD systems, a designer who remembers a large number of shortcuts and GUI (Graphic User Interface) tricks, in many cases, is deemed as an experienced user, irrespective of his actual knowledge and skills in the design of interest. Therefore, to get the full functionalities of a CAD system, targeted training and ample familiarization of the software are usually required, resulting in a longer time for user adaptation.

The complexities of current CAD systems can degrade their ease of use and the design productivity. Part of the reason for this can be attributed to the current cumbersome human–computer interactions [2,3], which are mostly implemented by means of direct manipulations [2,3] (i.e. mouse or keyboard based). These well established interactions, though mature enough, yet have considerable limitations.

Recently, a number of investigations have been proposed to integrate other human–computer interactions into the CAD modeling environments. For instance, Liu et al. [4,5] proposed to use haptic devices to aid surface and solid modeling in their virtual DesignWorks system; designers can use a haptic interface to touch a B-rep CAD model for direct surface manipulations [5]. Diniz and Branco [6] proposed a direct freehand drawing system, which tracks the movement of two LED lights attached to the designer's hands. The 3D positions and paths of each light/hand are taken as the truly 3D input in surface construction.

These novel human–computer interaction schemes, to certain degrees, either enhanced the ease of use or improved the design efficiency of CAD/CAM systems. Speech, which is by far the most

* Corresponding author. Tel.: +852 2859 7909; fax: +852 2858 5415.
E-mail addresses: kouxy@hku.hk (X.Y. Kou), micxue@hku.hk (S.K. Xue), sttan@hku.hk (S.T. Tan).

0010-4485/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2010.02.002

natural and intuitive communication medium between people [3], has also been proposed and incorporated into CAD systems, mostly in Virtual Reality based CAD environments [1,7–9]. In these applications, voice commands, combined with other interaction mechanisms, are used to construct and manipulate solid models. However, all of these systems include voice commands with *fixed* vocabularies [10], and only when a user utters *exactly* the same words/expressions can the corresponding CAD operations be activated. Not surprisingly, users are required to familiarize themselves with the predefined voice commands as well. This significantly constrains the freedom and ease of use of Voice-enabled CAD (VeCAD) systems.

This paper is motivated to investigate effective approaches to implement a flexible voice-enabled CAD system, where users are no longer constrained by predefined voice commands. Designers can communicate with the CAD modeler using natural language conversations to a much more flexible degree. Instead of using fixed, one-to-one mapping from *word expressions* to *modeling actions*, the proposed paradigm allows the same action to be activated by a variety of meaningful inputs. For instance, users can utter "Circle", "Draw a circle", "Create a circle", "Give me a circle" and other variations to add a circle to a CAD model, thus the flexibilities of the voice-enabled CAD system can be greatly enhanced.

The rest of the paper is logically structured as follows. Section 2 reviews related work and discusses the problems and challenges. Section 3 presents the detailed approaches to endow the VeCAD system with the ability of smart semantic inference. Section 4 provides the implementation details, and case studies are offered to show the efficacy of the proposed approaches. Finally conclusions and future work are presented in Section 5.

2. Related work and motivations

With the recent development in speech synthesis and recognition technologies [3,11,12], voice based interaction has demonstrated excellent flexibility and wide applicability in a variety of areas, such as word processing [13], medical [14], biomedical [2] and telephony [15] applications.

In the CAD field, voice-enabled CAD systems have also attracted considerable attention in the past few decades. Dani and Gadh [16, 17] proposed a "bimodal voice and tracking based interface" for creating "concept shape designs" in their COVIRDS system; voice commands were used to set the "context" for an action, while the parameters (e.g. the length of a block) were determined from the hand positions. Gao et al. [8] combined direct 3D manipulations (e.g. 3D mouse and 3D menu) with 80+ voice commands in a semi-immersive environment. Voice commands were primarily used to activate different operations, for instance, the utterance of "Attachment operation" can attach "translational sweep object" onto an existing 3D block, and "It is OK" was used to confirm an object's dimensions. Chu et al. [7] further extended the multi-sensory user interface in their VR-CAD system, where more than two (visual, auditory as well as tactile) types of interactions were simultaneously used in CAD environments. They compared different interactions and their pros and cons in entity generation, query and resizing. Their results show that voice commands are highly advocated by CAD users and industrial designers. Werich and Drews [9] integrated speech recognition, synthesis, 6D pointing pens and data gloves in their virtual manufacturing environment, and in their system a 50-word vocabulary was used. Bolt and Herranz [18] utilized voice input, two-handed gesture and eye tracking approaches to manipulate solids such as blocks and prisms: as an example, the expression "Turn the block" activated the object rotation command, and the block at which the eyes was looking was chosen as the object to rotate.

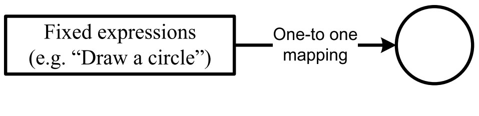


Fig. 1. The one-to-one mapping from fixed expressions to CAD modeling actions.

In these applications, voice inputs are mostly used in virtual reality based CAD systems (VR-CAD), and special devices such as data gloves [9,16–18] and eye trackers are required. In addition, the CAD models under design remained simple and primitive [16–18]. Kou and Tan [10] implemented a voice-enabled CAD modeler without the aids of such expensive devices. In conjunction with auxiliary mouse clicks, the VeCAD system demonstrated the practicability of using voice to construct complex 3D models [10] in a quasi hand-free fashion.

Despite the recent advancement, current voice-enabled CAD systems, either at research [7–10,16–18] or commercial [19,20] level, still lack sufficient flexibility, intelligence and user friendliness. Most of them use *fixed* vocabularies or *handcrafted* grammatical rules, and a modeling action can be activated if, and only if exactly the same expression is present in the predefined command repositories. This type of approach can be abstracted by the model shown in Fig. 1, in which direct mapping from a voice input to a CAD operation is overwhelmingly used. For instance, Speak4CAD [19] directly maps the spoken words to the menu items or keyboard shortcuts, and uttering predefined words such as "File New" will trigger the command whose shortcut is "Ctrl+N". To lessen the critical constraints, Kou and Tan [10] used grammatical rules to allow more variable inputs to be properly handled. Using the grammar rule shown in Fig. 2, users can use "*create circle*" or "*append a circle*" etc. to issue the circle drawing command, and the article "*a*" or "*an*" can be optionally provided or omitted. Such a model is relatively more flexible, however, still far from satisfactory in practical applications, as the allowable voice commands are still hardcoded: if a user says "*give me a circle*" (which is semantically equivalent to "*create a circle*"), the desired operation may not be successfully activated using current available approaches.

Studies show that users may, in many cases, use significantly different expressions to activate the same actions [18], rather than using pre-formed vocabularies the developer provides. A voice-enabled CAD system capable of processing new expressions it has never encountered, and inferring/understanding the semantics at runtime, is therefore, highly desirable. However, to the best of our knowledge, there seems to be no existing systems that have such features and functionalities.

To implement such intelligent Voice-enabled CAD systems, the key challenge is to properly understand the semantics of the voice input, and the inferred semantics can be then mapped to corresponding CAD actions. In literature, investigations on Natural Language Processing (NLP), Machine Translation (MT) and Computational Linguistics (CL) etc. [21–23] have offered many valuable insights into the mechanism of semantic understanding.

Lesk [24] proposed to get the meaning of a word from the definitions/interpretations of nearby words. For instance, the word *finish* has many different senses as a noun: it may refer to a *stopping point*, an *ending* or the *appearance of a surface*. To resolve the specific senses in the expression "*surface finish*", the adjacent word "surface" can be helpful. By counting the overlaps (i.e. shared words) of the dictionary definition of *surface* with all the nine definitions of *finish* in the WordNet [25] lexical database, it can be found that the sense "*appearance of a surface*" wins out. The sense with this maximum overlap is then regarded as the semantic of the word "finish" in this context. Banerjee and Pedersen [26] introduced the Adapted Lesk algorithm to determine the semantic relatedness of two words, where not only the definitions of the *words* but also the definitions of *related word set* (i.e. the synonym

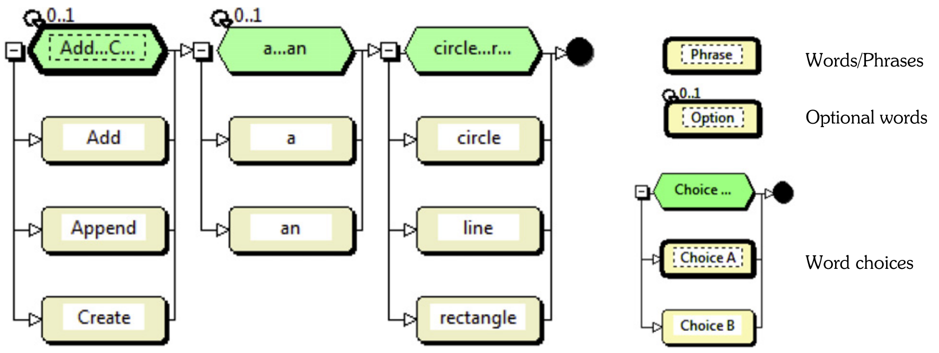


Fig. 2. A relatively flexible grammatical rule used in a voice-enabled CAD system. The elements with thick borders indicate the currently selected nodes in the grammar rule editor.

set or synset) were involved in the overlap calculations. For the example above, all the glosses/senses of "surface", "finish" and their synonyms such as "external"/"exterior", "close"/"terminate" etc are also included in the calculation of the *extended gloss overlaps*. Although the computational cost of the *Adapted Lesk* algorithm is much higher than that of the *Original Lesk* algorithm, the former proved to be more accurate and robust. Naskar and Bandyopadhyay [27] further proposed to incorporate the words in the entire sentence in the process of word sense disambiguation, and they even extended such strategies to take into account the words in the previous and next sentences. These approaches took advantage of contextual (phrasal, sentential or inter-sentential) knowledge in inferring the word semantics and have been widely employed in many versatile applications.

Even though the *Lesk-family* algorithms and the WordNet based information retrieval are very effective in judging the word semantics, directly applying them in Voice-enabled CAD systems seems less plausible. This is because they are based on *general knowledge* across most disciplines, and can hardly account for the CAD-specific knowledge and relations, which are of crucial importance in understanding the CAD jargon and its semantics. For instance, the verb "*sweep*" in WordNet has nine senses (see Appendix), but none of them shows a relation with such words as "*profile*" and "*path*". In the context of CAD design, however, these words are apparently highly related. Such relatedness is apparently out of the scope of currently available lexical databases.

To remedy the aforementioned limitations, this paper is motivated to introduce CAD specific knowledge to achieve smart semantic inferences in the VeCAD system. A novel type of behavioral gloss, together with CAD-specific synonyms, hyponyms (*a hyponym is a word or phrase whose semantic range is included within that of another word*) and hypernyms (a *hypernym is a word that is more generic than a given word*) are proposed to take advantage of CAD-targeted knowledge in reasoning a user's intent. For instance, the fact that "a *swept* cut is obtained by sweeping a closed *profile* along an open or closed *path*" can be used to relate "profile" and "path" to the sweeping operation. By leveraging such CAD targeted knowledge, a user can activate the "*swept cut*" command not only using the literal command "swept cut", he can also use more causal and verbal phrases such as "*cut it along this curve*". The detailed methodologies are presented as follows.

3. Knowledge-guided semantic inference for voice-enabled CAD

3.1. Problem statement

Given an expression E recognized from a user's voice input, the task of semantic inference is to get the most pertinent CAD

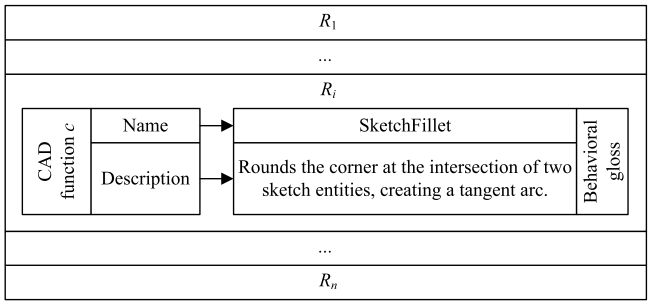

R_1
...
R_i
CAD function c
Name
Description
SketchFillet
Rounds the corner at the intersection of two sketch entities, creating a tangent arc.
Behavioral gloss
...
R_n

Fig. 3. The logical representation of a command and its behavioral gloss.

modeling command set $\hat{C}$ from the available command repository $C = \{c_1, c_2, \dots, c_n\}$, such that:

$$\hat{C} = \{c_i | F(E, c_i,) > 0,\ \forall c_i \in C,\ 1 \le i \le n\} \tag{1}$$

where n is the number of available commands, and $F(E, c)$ is a likelihood measure that models the matching of a command c with the expression E. To distinguish traditional approaches which use predefined vocabularies $\{E'\}$ and one-to-one mapping from E to c, it is explicitly stipulated that $E \cap \{E'\} = \varnothing$, i.e. the expression E is not a predefined, handcrafted expression.

In Eq. (1), it is possible that multiple commands match the same input E. This occurs if the input word expression is ambiguous or semantically non-unique, for instance, depending on the user's design intents, the expression "*fillet*" may refer to "Sketch Fillet", "Feature Fillet" or "Bead Fillet", as shown in Table 1. For such ambiguous inputs, the approach in this study is to output all the related voice commands and it is at the user's absolute discretion to decide which one is desirable.

3.2. Knowledge guided semantic inference

In order to endow a voice-enabled CAD system with the ability to understand the CAD jargon and semantics at runtime, there needs to be a proper representation to describe CAD-specific knowledge. Inspired by the successful adoption of the WordNet lexical database in Natural Language Processing (NLP) applications, this paper introduces a CAD-targeted knowledge database. The idea is to store the behavioral interpretations of well-established CAD operations (for example their types of parameters, general procedures and execution prerequisites etc.) and consider these factors as related glosses.

For a CAD modeling command/function c, a function name and a function description are associated as its behavioral glosses. A logical representation of such a structure is illustrated in Fig. 3. The "Name" field provides a concise description showing what the function is for, and the "Description" field contains

Table 1
Ambiguous semantics of "Fillet", extracted from the SolidWorks [28] help manual.

Commands	Descriptions
FeatureFillet	Creates a rounded internal or external face along one or more edges in solid or surface feature.
SketchFillet	Rounds the corner at the intersection of two sketch entities, creating a tangent arc.
BeadFillet	Adds a fillet weld bead feature between two disjoint bodies.

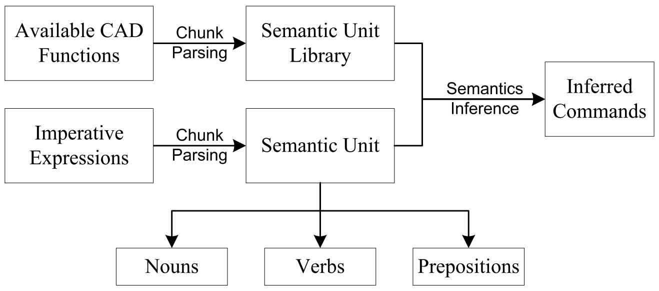


Fig. 4. A two-step approach to semantics inference.

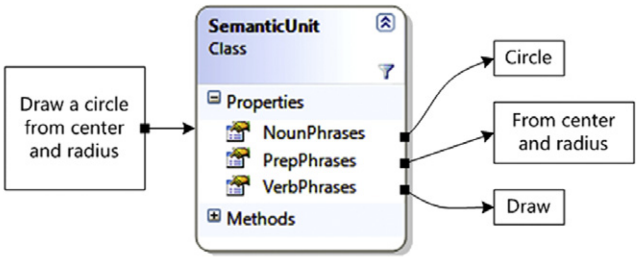


Fig. 5. Semantic parsing of the expression "Draw a circle from center and radius".

natural language sentences that depict in more details what this function does, how it is done, and possibly what is generated. The commands and their behavioral glosses are maintained in a table $T = \{R_1, R_2, \ldots, R_n\}$, as shown in Fig. 3, and here n is the number of records in table T.

As discussed earlier, significantly different expressions could be used to activate the same CAD operation, for example, "draw a circle", "insert a circle", "give me a circle", or simply just "circle" are all valid expressions to start circle drawing. Despite the diversity of the voice inputs, it is believed that commonly used voice commands follow certain patterns. Hauptmann [29] showed that over 90% of users' utterance were in the form of *imperatives*, as opposed to *declarative* ones. This indicates that "verb + noun" (e.g. *draw* a *circle*, where "draw" is the verb and "circle" is the noun) is the most natural pattern to activate a CAD command via voice. As such, the user's intent is not directly inferred from the freeform input; instead a two-step approach is proposed, as shown in Fig. 4.

In the first step, an imperative expression is analyzed and converted into a semantic unit (composed of semantic chunks) via chunk parsing. The chunk parsing formulates the user's freeform input into a structured representation, which filters out less important details (e.g articles, pronouns etc.) and keeps only the functional parts of interest (primarily the verb and noun phrases that contribute to the imperative patterns).

In the second step, the derived semantic unit is compared against a list of semantic units in a library, which are pre-parsed from the behavioral glosses (see Fig. 3) of available voice commands. A CAD modeling function c is identified as the candidate output if its semantic unit "matches" the input unit. The structure of the semantic unit and the definition of "matching" are described in more detail below.

3.2.1. Semantic unit and chunk parsing

A semantic unit refers to a structural representation of a *meaningful* expression in this study. An expression is said to be *meaningful* if the following conditions are satisfied:

1. The expression is CAD targeted: general words such as "Good morning" etc. are not meaningful to our voice-enabled CAD system;
2. The expression is syntactically correct or human understandable: "Create do radius" is therefore not a meaningful expression;
3. The expression is imperative, as opposed to descriptive: "A circle is a closed curve" is not meaningful in this sense, whereas "Draw a circle" is.

A typical semantic unit $U = \{ VP, NP, PP \}$ is composed of three semantic chunks, namely the Verb Phrases (VP), Noun Phrases (NP) and Preposition Phrases (PP). The verb/noun phrases contain a sequence of verbs/nouns, and each verb/noun is a simple string type. The Preposition Phrase (PP) is a string, which provides additional details such as how a command is executed.

Take the expression $E =$ "*Draw a circle from center and radius*" as an example. The verb phrases contain a single verb "Draw", the noun phrase is the word "Circle", and the preposition phrase is "from center and radius", as shown in Fig. 5.

The conversion from a freeform expression E to a semantic unit U is accomplished through a chunk parsing process. In what follows, the semantic unit parsed from E is denote as $U(E)$, and the VP, NP and PP of $U(E)$ are denoted as $U(E).V$, $U(E).N$and $U(E).P$ respectively.

For the example above, the expression is first divided into separate tokens ["draw", "a", "circle", "from", "center", "and", "radius"] as illustrated in Fig. 6 and on line 3 of the pseudo code shown in Fig. 7. Each token is then associated with a Part Of Speech (POS) tag using the Brill [30] algorithm (line 4, Fig. 7). Here the tags "NN" and "VB" etc. are well established tags borrowed from the Penn Treebank [31], standing for "noun" and "verb" respectively. The tagged tokens are then populated into different chunks according to the tag attributes, as shown in Fig. 6.

The aforementioned chunk parsing approach is conceptually straightforward; however it is applicable to simple cases only. Consider another case where the input expression is "Create a chamfer feature". If the above algorithm is used, then the parsed semantic unit would be $U_1 = \{ VP = \{\text{"}Create\text{"}\}, NP = \{\text{"}chamfer\text{"}, \text{"}feature\text{"}\}, PP = null \}$. One of the problems is that the word "feature" in NP is semantically redundant, since in the CAD field it is known that a chamfer *is a kind of* feature. Therefore an equivalent but more specific semantic unit should be $U_2 = \{ VP = \{\text{"}Create\text{"}\}, NP = \{\text{"}chamfer\text{"}\}, PP = null \}$. As will be demonstrated shortly in Section 3.2.3, using U_1 in the subsequent semantic inference may also result in severe robustness problems.

To troubleshoot this problem, the computer should have the ability to "understand" "a chamfer *is a kind of* feature" and then by applying this knowledge, the unnecessary word "feature" can be pruned from the NP chunk. To accomplish this, CAD specific knowledge is utilized to make the computer "smarter". In linguistics, it is known that the "*is-a*" relation is described with hyponym–hypernym pair. For instance the hyponym–hypernym pair (train-automobile) indicates that a train is *a specific kind of* automobile. Using a similar notion, CAD-specific hyponym–hypernym pairs are applied to tackle the redundancy removal problem. If more than two candidate nouns are available in the tokens (line

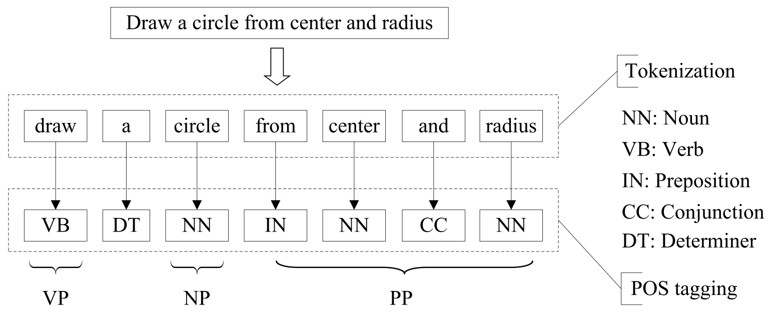


Fig. 6. Semantic parsing of expression "Draw a circle from center and radius".

```
1   SemanticParsing(in string Expression, out SemanticUnit unit)
2   {
3       Tokenize ( in Expression, out Tokens );
4       GetPosTags ( in Tokens, out Tags );
5       GetPrepPhraseIndexRange ( in Tags, out PrepIndexRange );
6       PopulatePrepChunk  ( in Tokens, in PrepIndexRange, out unit.PrepPhrases );
7       GetVerbIndices ( in Tags, out VerbIndices );
8       PopulateVerbChunk ( in Tokens, in VerbIndices, out unit.VerbPhrases );
9       GetNounIndices ( in Tags, out NounIndices );
10      GetCandidateNouns ( in Tokens, in NounIndices, out CandidateNouns );
11      if ( CandidateNouns.Count >=2 ) {
12          foreach ( noun1 in CandidateNouns ) {
13              foreach ( noun2 in CandidateNouns )
14                  if ( HasHypoHyperRelation ( noun1, noun2 ) )
15                      ExcludeHypernym ( in noun2, out NounIndices );
16          }
17      }
18      PopulateNounChunk ( in Tokens, NounIndices, out unit.NounPhrases );
19  }
```

Fig. 7. Pseudo code of chunk parsing algorithm, "in" and "out" refer to input and output respectively. Italic words represent pseudo subroutines.

Table 2
Some CAD specific Hyponym–Hypernym pairs.

Hyponym	Hypernym
Assembly	Model
Chamfer	Feature
Circle	Ellipse
Dimension	Annotation
Feature	Entity
Part	Document
Plane	Feature
Spline	Curve

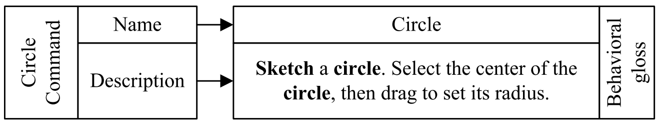

Circle Command
Name
Description
Circle
Sketch a **circle**. Select the center of the **circle**, then drag to set its radius.
Behavioral gloss

Fig. 8. Behavioral gloss of "Circle" command.

11, Fig. 7), a further check is then conducted to see if there are hyponym–hypernym relations in between (line 14, Fig. 7). This can be accomplished by traversing the records in the proposed CAD-specific hyponym–hypernym data table (some examples are listed in Table 2 and Fig. 13(b)), and check the availability of the word pairs therein. If a word pair is found in the table, then the hypernym is semantically redundant and will be removed (line 15, Fig. 7), while the hyponym is kept in the noun phrase. For the example above, since the (chamfer-feature) pair matches such conditions, therefore the hypernym "feature" is excluded, and only the hyponym "chamfer" remains in the noun chunk. The pseudo code of the chunk parsing algorithm is schematically shown in Fig. 7.

It should be particularly noted that there are no existing lexical databases which can solve the above redundancy removal problem, as they fail to account for the CAD specific knowledge as exemplified above.

3.2.2. Template matching based semantic inference

With the adoption of the proposed semantic unit structure and chunk parsing algorithm, the user's imperative inputs can be formulated into a structured representation. Less important words such as pronouns and articles are pruned, leaving behind the most pertinent parts such as verbs (what to do) and nouns (what/which entity to operate on). If the verb and noun phrases are used as keywords and searched within the behavior glosses shown in Fig. 3, the commands whose description fields contain the verbs and nouns can be identified. Intuitively, these commands can be considered as the intended functions a user refers to.

Using this keywords based searching method, however, has certain limitations. First, the verbs and nouns in the semantic unit may differ from the words used in the behavioral glosses. For instance, the behavioral gloss of the "Circle" command is shown in Fig. 8, in which the verb used is not "Draw", but "Sketch". This problem, known as word/query mismatch problem, has been well studied in the area of information retrieval, and synonym expansion [32–35] proves to be an effective approach to resolve this. The idea is to utilize not only the input words, but also their synonyms as keywords such that the allowable words are no longer constrained to be the same as the one in the behavioral glosses. As long as the inputs are semantically equivalent, the desired functions can be identified without much difficulty.

Using "machine readable dictionaries" [24] such as WordNet [25], it is trivial to get the synonyms of a word, however, it is possible that two words are synonymous in the CAD context but the general dictionaries are ignorant of this. For instance, the word "sketch" and "add" are synonyms in the sense that "sketch

Table 3
Some CAD-specific synonyms.

Word	Synonym
Add	Sketch
Combine	Join
Heal	Merge
Join	Union
Mesh	Tessellation
Rebuild	Update
Suppress	Remove
Undo	Revert

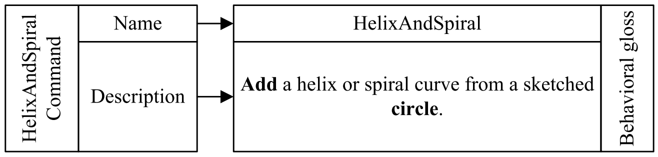


Fig. 9. Behavioral gloss of "HelixAndSpiral" command.

a circle" and "add a circle" are semantically identical. Unfortunately, none of the existing lexical databases incorporates such CAD-specific knowledge and they fail to regard "sketch" and "add" as synonyms. To cope with the deficiencies of existing synonym expansion approaches, CAD-specific synonyms are organized in a data table (some examples are listed in Table 3 and Fig. 13(a)), and in the case that an exact match of a verb/noun is not found in the behavioral gloss, a further check is applied to see if their *CAD-synonyms* are available in the behavioral gloss of a CAD function. If the answer is affirmative, then the function is output as the inferred result. Using this method, the voice input "draw a circle" and "insert a circle" can be equally used to activate the circle drawing command, as "sketch", "draw" and "insert" are synonyms in the proposed CAD-synonym databases.

Another limitation of the keyword based searching is related to the robustness of this approach. For instance if "Draw" and "Circle" are used as the keywords, since "Draw" and "Add" are CAD synonyms, using direct searching may also return the command "HelixAndSpiral", since its description field also contains "add" and "circle", as shown in Fig. 9. In this scenario, if the user utters "Draw a circle", the semantics might be mistakenly interpreted as "HelixAndSpiral" as well.

It is seen that in Fig. 9, even the verb "add" and noun "circle" appear in the description field of the gloss, however, "circle" is not the entity to be created; instead it is related to how a "helix" is created. In other words, the same word "circle" plays different roles in the input expression and in the function description shown in Fig. 9. Using the notion of semantic unit, it can be seen that the word "circle" in Fig. 9 is in fact in the preposition phrase "from a sketched circle", rather than in the noun phrase. If the searching of a word is limited within the same functional counterpart, i.e. verbs/nouns are compared against verbs/nouns only, the above problem can be resolved. In this paper, two semantic chunks A and B are said to be *comparable*, if they satisfy such constraints.

In order to enable the word searching to be performed within *comparable* chunks, the chunk parsing approach (Section 3.2.1) is proposed to be applied on the behavioral glosses as well. The description field of a CAD function c is similarly transformed from natural language sentences into structural semantic units. The parsed results are called c's semantic units and denoted as $U(c)$. Table 4 lists several example CAD functions and their parsed semantic units. For efficiency reasons, all the descriptions of available CAD functions are pre-parsed in advance and these semantic units $\{U_i(c)\},\ i = 1, 2, \ldots, n$, collectively form a library that can be accessed at runtime.

Armed with this pre-parsed semantic unit library, the semantic inference problem then amounts to a *template matching* problem, where the semantic units parsed from voice input are the "*samples*" to be inspected and the semantic units in the library are the feature *templates*.

To be precise, the term "match" needs to be rigorously defined. Let the function $Syn\ (S)$ be the function which returns the synonyms of a word s or a word collection $S = \{s_1, s_2, \ldots, s_m\}$, here m is the number of words in the collection S:

$$Syn(S) = \{Syn(s_1)\} \cup \{Syn(s_2)\} \cup \cdots \cup \{Syn(s_n)\}. \quad (2)$$

For any two comparable semantic chunks A and B, their matching scenarios may fall into one of the following cases:

- A is an *exact match* of B, if A and B are identical, i.e. $A = B$;
- A is a *strong match* of B, if the words in chunk A have partial overlaps with the words in B, i.e. $A \cap B \neq \varnothing$;
- A is an *equivalent match* of B, if the words in chunk A have no overlaps with the elements in B, while the synonyms of A have overlaps with B, i.e. $Syn(A) \cap B \neq \varnothing$;
- A is an *unmatch* of B, if both the words in A and their synonyms have no overlaps with B, i.e. $\{A \cup Syn(A)\} \cap B = \varnothing$.

The following quantitative measure is proposed to model the Degree Of Matching (DOM) of two semantic chunks:

$$m(A, B) = \begin{cases} 1, & A = B \\ w_1 \dfrac{|A \cap B|}{|B|}, & A \cap B \neq \varnothing,\ A \neq B \\ w_2 \dfrac{Syn(A) \cap B}{|B|}, & Syn(A) \cap B \neq \varnothing,\ A \cap B = \varnothing, \\ 0, & \{A \cup Syn(A)\} \cap B = \varnothing, \end{cases}$$
$$0 \leq w_2 \leq w_1 \leq 1 \quad (3)$$

where the symbol $|X|$ denotes the cardinality of set X, and w_1 and w_2 are custom defined weights. This DOM is essentially based on the weighted overlaps of A and B. If an exact match occurs, a full score of 1 is assigned; and in case of a strong match, the measure is proportional to the degree of word overlaps between A and B, with a custom weight w_1 applied. Similarly for an equivalent match, the DOM is determined from the overlaps of $Syn(A)$ with B, but with a lower or equal weight w_2 applied; otherwise it is considered that A does not match B and a score 0 is assigned.

For two semantic units $U_1 = \{V_1, N_1, P_1\}$ and $U_2 = \{V_2, N_2, P_2\}$, their degree of match $M(U_1, U_2)$, is then defined as a function of a two-tuple (m_V, m_N), here $m_V = m(V_1, V_2)$ and $m_N = m(N_1, N_2)$ are the DOM of the verb phrases and noun phrases respectively. The $M(U_1, U_2)$ is defined as:

$$M(U_1, U_2) = \begin{cases} m_V, & N_1 = null, V_1 \neq null, V_2 \neq null \\ m_N, & V_1 = null, N_1 \neq null, N_2 \neq null \\ f(m_V, m_N) & V_1 \neq null, V_2 \neq null, N_1 \neq null, N_2 \neq null \\ 0 & \text{otherwise}. \end{cases} \quad (4)$$

Eq. (4) states that if the noun phrase of the input N_1 is empty, then the DOM of the semantic unit is evaluated as the DOM of the verb phrases, if the verb phrases V_1 and V_2 are nonempty; and this principle also applies to the case where verb phrase V_1 is empty while N_1 and N_2 are nonempty. If V_1, V_2, N_1 and N_2 are all nonempty, then the DOM of the semantic unit is a function of m_V and m_N. In our current implementation, the function $f(m_V, m_N)$ is a function in the form of Eq. (5):

$$f(m_V, m_N) = \begin{cases} \dfrac{w_V m_V + w_N m_N}{w_V + w_N} & m_V > 0,\ m_N > 0 \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where w_V, w_N are custom defined weights, which reflect the influences/contributions of the DOM of verb and noun phrases in the DOM evaluation of two semantic units. It is seen that the

Table 4
CAD command examples and their parsed semantic units.

Command c	Semantic units $U(c)$			
		Verb Phrase (VP)	Noun Phrase (NP)	Preposition Phrase (PP)
$c_1 =$ "*Circle*"	U_1	"Sketch"	"Circle"	Null
	U_2	"Select"	"Center "	"Of the circle"
	U_3	"Drag, "set "	"Radius"	Null
$c_2 =$ "*HelixAndSpiral*"	U	"Add"	"Helix", "spiral"	"From a sketched circle"
$c_3 =$ "*PerimeterCircle*"	U_1	"Sketch"	"Circle"	"By its perimeter"
	U_2	"Select"	"Point"	"On the perimeter"

verb phrases in use may have more variations than nouns, for instance, the verbs "give", "insert" and "draw" refer to the same meaning in this case. In contrast, the noun phrases (e.g. circle) have relatively less metamorphoses, as they are usually well-established geometric entities such as ellipse, axis or extrusion etc. In this regard, the noun phrases are relatively more stable matching indicators than the verb phrases. As such, it is explicitly stipulated that $w_N \geq w_V$ to account for such discriminative strategies.

In Eq. (5), if both the verb and noun phrases' DOM are larger than 0 (i.e. both the verb and noun phrases match, see also Eq. (3)), it indicates that entire pattern (semantic unit) has found a match. Otherwise, if either the verb or noun phrases mismatch, or both fail to find matches (i.e. the DOM m_V, m_N or both equal to 0), it is regarded that the two semantic units mismatch.

The CAD commands/functions that match the voice input are then defined as the set which follows Eq. (6):

$$\hat{C} = \{c_i | M(U(E), U(c_i)) > 0,\ c_i \in C\} \tag{6}$$

where $C = \{c_1, c_2, \ldots, c_n\}$ refers to the available command set.

Note that the matching of preposition phrases is not taken into account in Eqs. (4) and (5), this is because verbs and nouns are the dominant information to be captured in the semantic inference. It might be too restrictive to require the complementary details to also match in order to claim a match of two semantic units. However, the preposition phrases do have their places in semantics clarification, and it can be used for ranking the commands in $\hat{C}$. The above semantic matching measure in Eq. (5) can then be extended by incorporating the preposition phrases, and a ranking function is proposed:

$$r(E, c_i) = \frac{\sum\limits_K w_K m_K}{\sum\limits_K w_K}, \quad K = V, N, P \tag{7}$$

where w_K are custom weights and K refers to V, N or P, and $m(\ ,\ )$ is the DOM function as formulated in Eq. (3). For the CAD functions $\hat{C}$ derived from the template matching step, all the elements in $\hat{C}$ are ranked and sorted in a descending order, and the reordered results are saved in another set $\hat{C}'$. Depending on the user's preferences, either the very first N (which is a user-defined constant) elements of $\hat{C}'$ or the complete set is then regarded as the final output.

Let us take several aforementioned voice inputs as examples to better understand the rational of the proposed approach. Table 5 lists the input expressions and their corresponding semantic units. For simplicity, it is assumed that the weights are $w_1 = 1$, $w_2 = 0.8$ in Eq. (3) and $w_V = w_N = 1$ in Eq. (5).

- For E_1 and E_2 in Table 5, according to Eqs. (4) and (5), the DOM of semantic units is equal to the weighted sum of the DOM of verb and noun chunks. In the evaluation of the DOM of verbs m_V, as "Draw" and "Insert" are CAD synonyms of "Sketch", therefore the verbs of E_1 and E_2 have *equivalent* matches in the commands c_1 and c_3 shown in Table 4, so $m_V = 0.8$. For the noun phrases, *exact* matches can be found in command c_1 and c_3, so the DOM of nouns $m_N = 1.0$. According to Eqs. (5) and (6), $M(E_1|E_2, c_1|c_3) = (1 + 0.8)/(1 + 1) = 0.9 > 0$, therefore both the command c_1 = "*Circle*" and c_3 = "*PerimeterCircle*" are the inferred CAD functions that E_1 and E_2 refer to. For command c_2 = "*HelixAndSpiral*", although *equivalent* matches (Add-Draw, Add-Insert) can be found for the verb phrases of E_1 and E_2, however, there is no matching in the noun phrases ("Helix"/"Spiral" vs. "Circle"), therefore $M(E_1|E_2, c_2) = 0$, and the command "*HelixAndSpiral*" is then not interpreted as the desired CAD command.
- For E_3 in Table 5, since only noun phrase exists, therefore the DOM of semantic units is calculated from the m_N only. It is easy to derive that $M(E_3, c_1|c_3) = 1 > 0$ and $M(E_3, c_2) = 0$, so the expression E_3 = "Circle" is actually meant to activate the commands c_1 = "*Circle*" or c_3 = "*PerimeterCircle*".

3.2.3. *Robustness considerations*

The robustness problems usually take in two forms: *underestimation* and *overestimation*. The underestimation occurs when a user inputs reasonably acceptable expressions while the expected results are not returned; and the overestimation takes place where irrelevant results are also returned along with the desired CAD commands.

The overestimation case has been implicitly discussed in Section 3.2.1, where CAD specific hyponym–hypernym relations are used to remove redundant words and maintain specificity. As discussed in Section 3.2.1, if the hypernyms are not properly pruned as proposed, then the parsed semantic unit for the expression E = "Create a chamfer feature" will be:

$$U(E) = \{\ VP = \{\text{"}Create\text{"}\}, \\ NP = \{\text{"}chamfer\text{"}, \text{"}feature\text{"}\},\ PP = null\ \}. \tag{8}$$

Since both the verb "create" and noun "feature" are frequently used words in the CAD field, it is possible that there exist many commands whose comparable chunks have *exact* or *equivalent* matches. As an example, the description field of the command "Extrude" reads "*Create* an extrude *feature* from the selected sketch entities", and its corresponding semantic unit is:

$$U(c) = \{\ VP = \{\text{"}Create\text{"}\}, NP = \{\text{"}extrude\text{"}, \text{"}feature\text{"}\}, \\ PP = \text{"}from\ the\ selected\ sketch\ entities\text{"}\ \}. \tag{9}$$

Now that the verb phrases of $U(E)$ in Eq. (8) and $U(c)$ in Eq. (9) have an exact match (i.e. "Create"–"Create"), and the DOM of verbs $m_V = 1$. The noun phrases also have an strong match (i.e. "feature"–"feature"), and by Eqs. (3), (8) and (9), $m_N = 0.5$. This leads to the DOM of the semantic units $M(U(E), U(c)) = 0.75$. From Eq. (6) it can be found that the command "Extrude" will be *mistakenly* regarded as the referred commands. Obviously, the expression "Create a chamfer feature" is irrelevant with the command "Extrude", resulting in an overestimation in this case. Clearly the proposed redundancy removal approach based on CAD-specific hyponyms and hypernyms (Section 3.2.1) can properly avoid this robustness problem.

Table 5
Voice input expressions and their parsed semantic units.

Imperative expression E	Semantic units $U(E)$		
	Verb phrase (VP)	Noun phrase (NP)	Preposition phrase (PP)
E_1 = "Draw a circle"	"Draw"	"Circle"	Null
E_2 = "Insert a circle"	"Insert"	"Circle"	Null
E_3 = "Circle"	Null	"Circle"	Null

For the underestimation cases, they are usually originated from excessive usage of harsh constraints on the matching conditions of semantic units. In the approach presented earlier, the most critical condition prescribed is the one shown in Eq. (5), which stipulates that a command is a match of the input if, and only if, both the verb and noun phrases match. This is a reasonable condition in many cases. For instance if only the verb phrases are required to match in order to claim a command match, then for the input E_1 and E_2 in Table 5, all the commands with such verbs as "add", "insert", "make" etc. will all be regarded as the output, and there are in fact dozens of such commands. On the other hand, this condition may become an over-strong constraint in certain scenarios. Consider the semantic inference problem for another input E_4 = "try a circle", for which the parsed semantic unit is:

$$U(E_4) = \{VP = \{\text{"}try\text{"}\},\ NP = \{\text{"}circle''\},\ PP = null\}. \qquad (10)$$

It is easy to see that for the commands listed in Table 4, there are no exact matching verbs. Using the approach presented in Section 3.2.2, the synonyms of the verb "try" are searched in the CAD synonym database, and no results are found either. The general synonyms are then searched within the WordNet lexical databases to try to get some equivalent matches. Among the 17 synonyms of the word "try" (such as "seek", "attempt", "test" etc.), none of them matches the verb phrases in Table 4. Using the criteria shown in Eq. (5), it is legitimate to say there are no matching commands for the expression "try a circle". However, to human beings, "try a circle" is a fairly acceptable expression to activate the circle drawing commands.

It is found that when both verb and noun phrases are present in a semantic unit, the matching criteria should not be fixed as shown in Eq. (5), instead a more *adaptive* strategy should be used. Clearly, for verbs which relate closely to concrete CAD operations such as "Create", "Delete" and "Rotate" etc., it is reasonable to use the DOM evaluation criteria described in Eq. (5) to avoid overestimations. Conversely for verbs with less explicit CAD meanings such as "have" and "try", the incomprehensible (to the computer, and in the CAD sense) verbs can simply be neglected and the noun phrases be used as the judging criteria, as if the verb phrase is "null". For instance, if only the noun chunk (i.e. "circle") is used as effective information in the previous example, then the input "try a circle" actually degenerates to the case of E_3 in Table 5, therefore this expression can be correctly interpreted as the commands such as "Circle" or "PerimeterCircle", as discussed in Section 3.2.2.

Here the challenge turns to *judiciously* distinguish whether the verb chunk should be abandoned or not in the DOM evaluation of semantic units, and this can be surmounted based on the following observations:

- If a verb has concrete unambiguous CAD meanings, it has a higher probability to appear in the CAD command glosses; the stronger CAD meanings it carries, the higher its occurrence frequency will be in the glosses;
- Words without strong CAD meaning or ambiguous in the CAD sense are usually those that are highly polysemous, such as "try" and "have".

Table 6
Some CAD-specific synonyms.

Word	*VRI*
Add	13.83
Create	10.16
Insert	4.75
Combine	0.42
Suppress	0.4
Cut	0.24
Give	0
Try	0

As such a Verb Reliability Index (*VRI*) is proposed to indicate if it is safe to incorporate the verb chunks in the DOM evaluation of the semantic unit:

$$VRI(v) = \frac{f_{CAD}(v)}{f_{poly}(v)} \qquad (11)$$

where $f_{CAD}(v)$ refers to the occurrence frequency of a verb v in the CAD glosses (see Fig. 3) and $f_{poly}(v)$ is the polysemy of a verb in computer-readable dictionaries such as WordNet. The lower $f_{CAD}(v)$ and the higher $f_{poly}(v)$ are, the lower the reliability (for the participation in the DOM evaluation) of the verb is. If the *VRI* is lower than a threshold, then it will not be used in the DOM evaluations, then Eq. (5) can be modified as:

$$f(m_V, m_N) = \begin{cases} \dfrac{w_V m_V + w_N m_N}{w_V + w_N} & VRI(V) > t,\ m_N > 0, m_V > 0 \\ m_N & VRI(V) \le t,\ m_N > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (12)$$

here the $VRI(V)$ of a verb phrase $V = \{v_1, v_2, \dots, v_k\}$ is defined as the maximum *VRI* value of its constituent verbs, and k is the count of verbs in V:

$$VRI(V) = \max\left(VRI(v_1), VRI(v_2), \dots, VRI(v_k)\right). \qquad (13)$$

Table 6 shows some verb examples and their Verb Reliability Index (*VRI*). In our current implementation, the threshold t is set to $t = 0.1$. For the previously discussed examples such as "Give me a circle" or "Try a circle", since their VRI are below the preset threshold, therefore only the DOM of the noun chunk contributes to the DOM of the semantic units, and the underestimation problem is then resolved.

3.2.4. *Pseudo codes of the proposed algorithm*

After taking all the aforementioned algorithms and robustness issues into account, Fig. 10 shows the complete pseudo code of the proposed algorithm. The algorithm takes a string expression as input (which is obtained from the speech recognition engine), and outputs the most pertinent N voice commands according to the ranks evaluated from Eq. (7). Here it is assumed that all the parameters such as w_1, w_2 in Eq. (3), w_V, w_N, w_P, t in Eq. (7), Eq. (12) and N are predefined constants.

```
InferSemantics(in string Expression, out List ActionsToPerform)
{
    SemanticParsing( in string Expression, out SemanticUnit $U_i$);          //Fig. 7
    foreach(SemanticUnit $U_c$ of a CADFunction $c_1$ in UnitLibrary)
    {
        double $m_V$=$m(U_i.V, U_{c1}.V)$;                                    // Eq. (3)
        double $m_N$=$m(U_i.N, U_{c1}.N)$;                                    // Eq. (3)
        double $M$=$M(U_i, U_{c1})$;                                          // Eq. (4), (12)
        if($M>0$)
            CandidateActionsToPerform.Add($c_1$);
    }
    foreach(CADFunction $c_2$ in CandidateActionsToPerform)
    {
        $r_i$=$r$(Expression, $c_2$);                                         // Eq. (7)
    }
    DescendentSort(CandidateActionsToPerform, $r$ );
    ActionsToPerform= DescendentSort.HeadElements ($N$);
}
```

Fig. 10. Pseudo codes of the proposed algorithm.

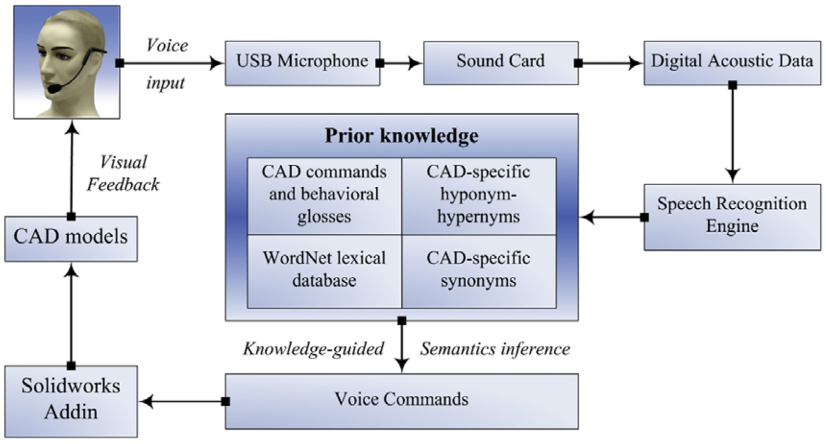


Fig. 11. System architecture of the proposed Voice-enabled CAD system.

4. Implementations and case studies

The system architecture of the proposed Voice-enabled CAD system is shown in Fig. 11. A microphone is used to capture the input voice signal, which is then digitized by the sound card and further transferred to the computer memory. The digital acoustic data is then passed to the Speech Recognition Engine for recognition. Microsoft speech toolkit (SAPI 5.3) is used in the current implementation to accomplish such a task. The imperative expressions are mapped to CAD functions using the proposed approach. The identified CAD function is executed in the SolidWorks CAD modeler [28] to create, modify and manipulate CAD models. Visual feedbacks are then fed back to the user through graphical user interfaces.

The mapping from the voice input to the CAD functions is accomplished by using the knowledge guided semantic inference. The CAD commands' behavioral glosses are one of the most important sources of knowledge. The coverage of the behavioral gloss determines the number of features in the feature template, which is of critical importance in the entire process. The description fields of the CAD commands, inclusive of the verbs, nouns, preposition phrases, have great impact on the success rates of inference, as these words contains "*signatures*" of the features to be matched. In the current implementation, over 500 commonly used CAD functions are included, and the description fields of each command are automatically constructed from the SolidWorks help manual, with minor adjustments to maintain the structural consistency. Fig. 12 shows a snapshot of the currently used behavioral glosses.

The CAD specific synonym and hyponym–hypernym databases are manually constructed and hundreds of data entries have been created and maintained in the database, as shown in Fig. 13. In addition, the WordNet [25] lexical database is used extensively to get the general synonyms and the polysemy of given words.

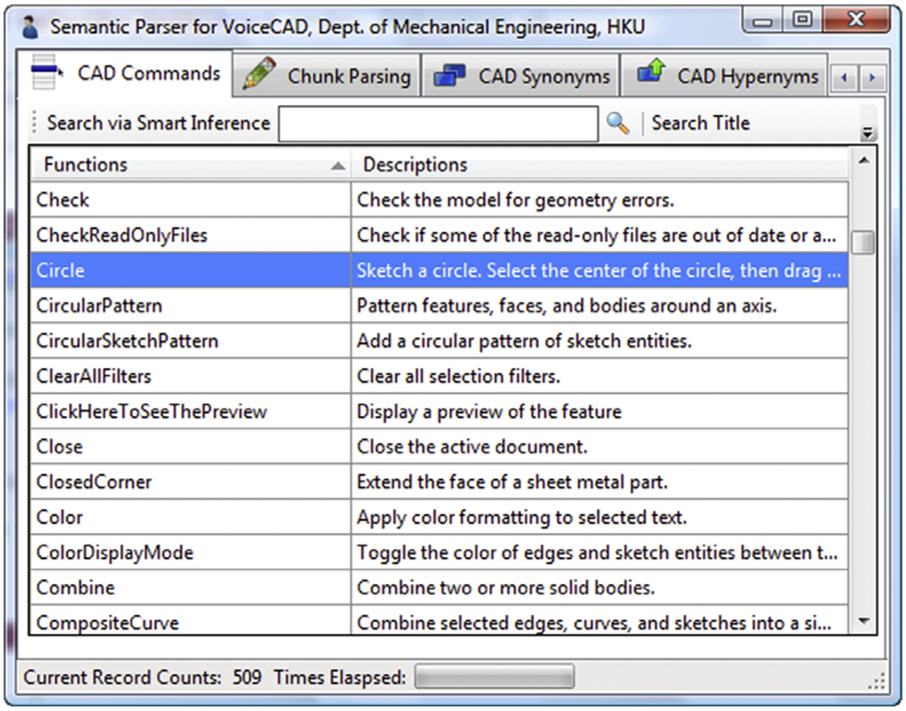
Using the proposed approach, a list of voice inputs have been tested. The input expression and the inferred CAD functions are shown in Table 7. The needed time for these semantic inferences ranges from 0.03 s to 0.75 s. The experiments are conducted with a PC with 4 GB memory and an Intel Core 2 Q6600 2.4 GH CPU on a Windows Vista sp1 platform. Experiments show that most command inferences can be finished within 1 s, and are appropriate for activating CAD commands in real time.

Figs. 14 and 15 demonstrate the application of the proposed approach in the process of a model construction. Intuitive phrases, instead of exact function names such as "lofted base" and "save documents" are successfully mapped to related CAD commands. This demonstrates the efficacy of the proposed approach.

Our experiments also show that if user intent can be effectively captured, then using the voice interface has great advantages. Quantitative evaluations have been conducted in our latest research and the results are reported in [36]. Experiments show that about 40%–50% of mouse movement can be reduced using the proposed VeCAD system [36].

5. Conclusions and future work

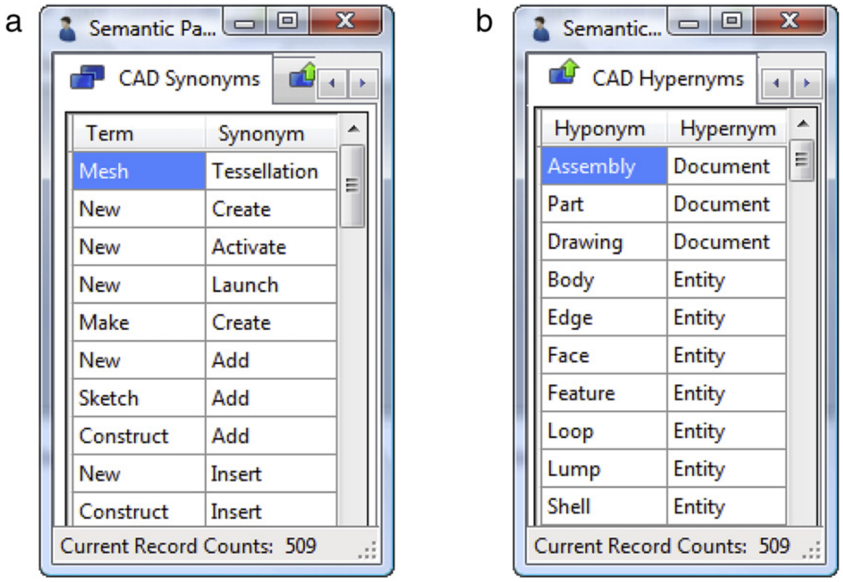
This paper investigates an approach to *infer* the semantics of users' voice input in a voice-enabled CAD system. The objective


Semantic Parser for VoiceCAD, Dept. of Mechanical Engineering, HKU
CAD Commands
Chunk Parsing
CAD Synonyms
CAD Hypernyms
Search via Smart Inference
Search Title

Functions	Descriptions
Check	Check the model for geometry errors.
CheckReadOnlyFiles	Check if some of the read-only files are out of date or a...
Circle	Sketch a circle. Select the center of the circle, then drag ...
CircularPattern	Pattern features, faces, and bodies around an axis.
CircularSketchPattern	Add a circular pattern of sketch entities.
ClearAllFilters	Clear all selection filters.
ClickHereToSeeThePreview	Display a preview of the feature
Close	Close the active document.
ClosedCorner	Extend the face of a sheet metal part.
Color	Apply color formatting to selected text.
ColorDisplayMode	Toggle the color of edges and sketch entities between t...
Combine	Combine two or more solid bodies.
CompositeCurve	Combine selected edges, curves, and sketches into a si...

Current Record Counts: 509 Times Elaspsed:

Fig. 12. The behavioral glosses used in the proposed approach.



Term	Synonym
Mesh	Tessellation
New	Create
New	Activate
New	Launch
Make	Create
New	Add
Sketch	Add
Construct	Add
New	Insert
Construct	Insert

Current Record Counts: 509

Hyponym	Hypernym
Assembly	Document
Part	Document
Drawing	Document
Body	Entity
Edge	Entity
Face	Entity
Feature	Entity
Loop	Entity
Lump	Entity
Shell	Entity

Current Record Counts: 509

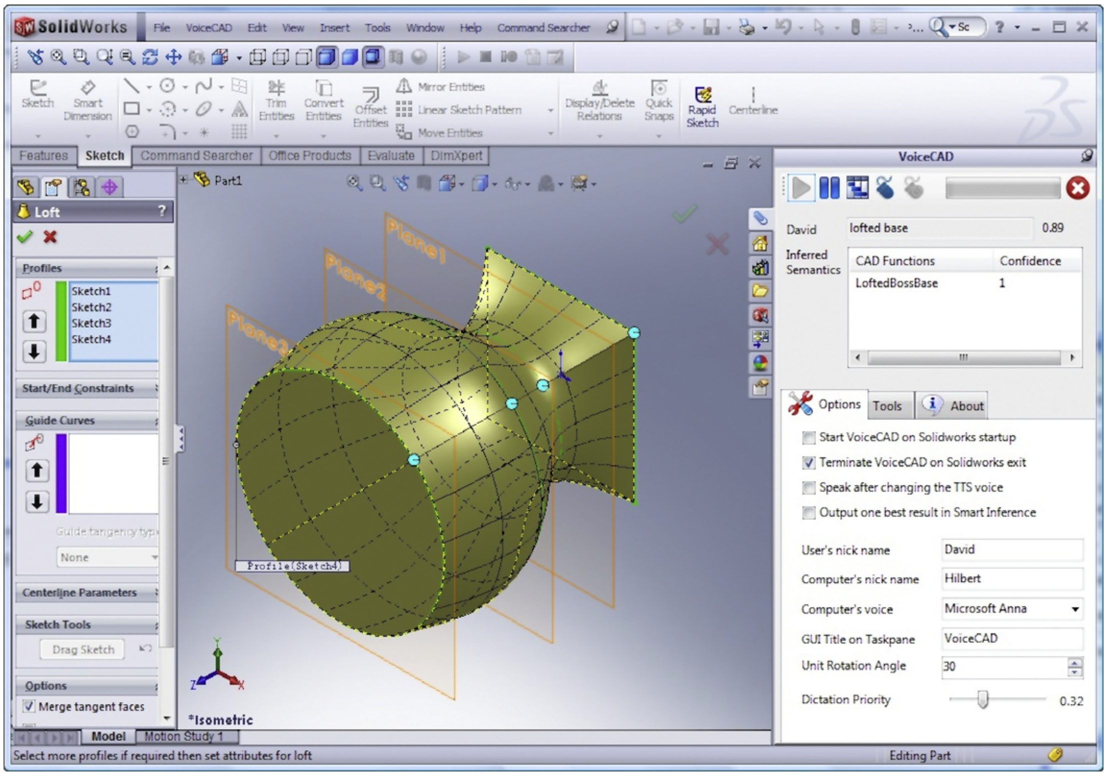
Fig. 13. CAD-specific synonyms and hypernym–hypernym pairs. (a) Synonyms; (b) hyponyms and hypernyms.

Table 7
Some voice inputs and the corresponding output CAD functions.

Voice input	Inferred CAD functions
Give me a circle	Circle, PerimeterCircle
Try rectangle	Rectangle
Delete it	Delete, DeleteFace, DeleteSolidSurface, DisplayDeleteRelations
Cut along this curve	ExtrudedCut, LoftedCut, CutWithSurface, RevolvedCut, SweptCut
Insert a bevel	Chamfer
Draw a spline curve	Spline, FitSpline
End the curve	PickMode
Draw an arc	3PointArc, CenterpointArc, TangentArc, SketchFillet

is to free the users from memorizing predefined voice commands. Designers can therefore communicate with the CAD modeler using natural language conversations to a much more flexible degree.

The novelties of the proposed approach lie in the use of a knowledge-guided approach to infer the semantics of voice input. The CAD-specific behavioral glosses are an important source of



Sketch1
Sketch2
Sketch3
Sketch4

☑ Merge tangent faces

David	lofted base	0.89

Inferred Semantics

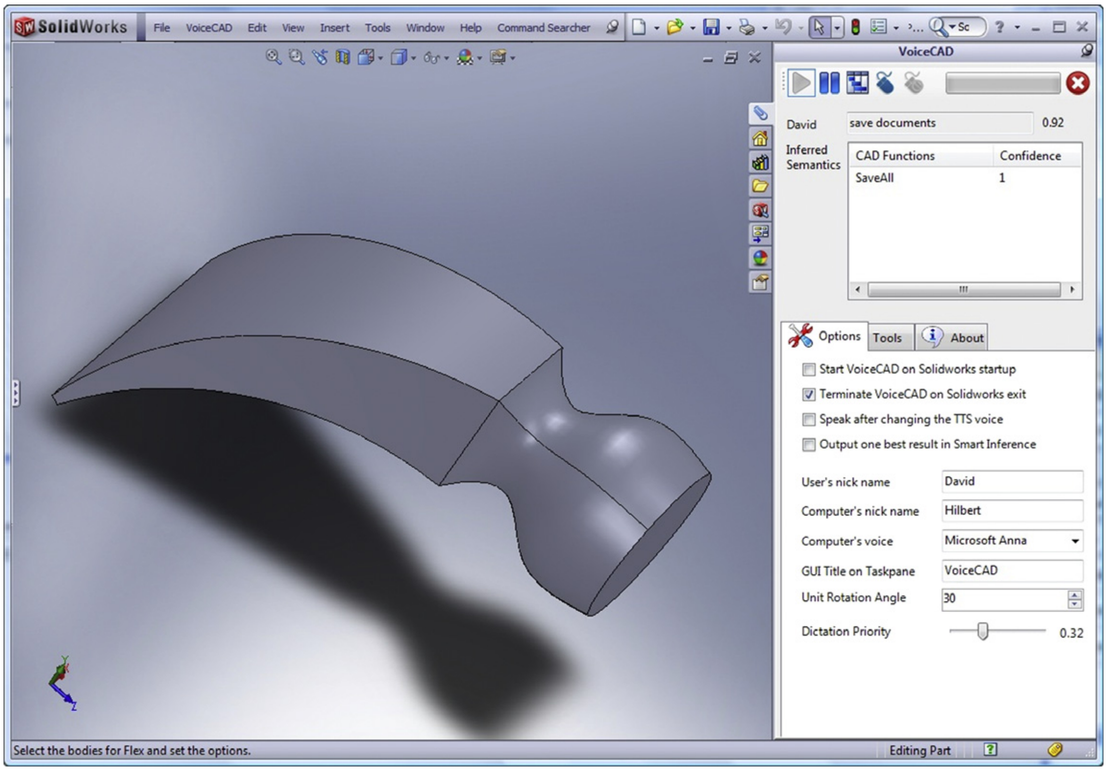
CAD Functions	Confidence
LoftedBossBase	1

☐ Start VoiceCAD on Solidworks startup
☑ Terminate VoiceCAD on Solidworks exit
☐ Speak after changing the TTS voice
☐ Output one best result in Smart Inference

User's nick name	David
Computer's nick name	Hilbert
Computer's voice	Microsoft Anna
GUI Title on Taskpane	VoiceCAD
Unit Rotation Angle	30
Dictation Priority	0.32

Select more profiles if required then set attributes for loft
Editing Part

Fig. 14. VeCAD addin for SolidWorks and an example modeled with the proposed approach.



David	save documents	0.92

Inferred Semantics

CAD Functions	Confidence
SaveAll	1

☐ Start VoiceCAD on Solidworks startup
☑ Terminate VoiceCAD on Solidworks exit
☐ Speak after changing the TTS voice
☐ Output one best result in Smart Inference

User's nick name	David
Computer's nick name	Hilbert
Computer's voice	Microsoft Anna
GUI Title on Taskpane	VoiceCAD
Unit Rotation Angle	30
Dictation Priority	0.32

Select the bodies for Flex and set the options.
Editing Part

Fig. 15. A solid model designed with the aid of knowledge-guided inference.

knowledge that contributes to the success of this smart inference. Our investigation shows that CAD-specific knowledge is also helpful to enhance the robustness of the semantic inference, and can be used to eliminate the chance of overestimation and underestimation. To the best of our knowledge, the use of such CAD-specific knowledge and the related knowledge representations has rarely been reported in literature in promoting the ease of use of current CAD systems.

It is noted that the proposed semantic inference approach is still very primitive in terms of flexibility and intelligence. Towards

Table A.1
The nine senses and glosses of the verb sweep in WordNet lexical database [25].

Word senses	Glosses in WordNet
Brush, sweep	(sweep across or over; "Her long skirt brushed the floor"; "A gasp swept cross the audience")
Sweep, sail	(move with sweeping, effortless, gliding motions; "The diva swept into the room"; "Shreds of paper sailed through the air"; "The searchlights swept across the sky")
Sweep, broom	(sweep with a broom or as if with a broom; "Sweep the crumbs off the table"; "Sweep under the bed")
Embroil, tangle, sweep, sweep up, drag, drag in	(force into some kind of situation, condition, or course of action; "They were swept up by the events"; "Don't drag me into this business")
Cross, traverse, span, sweep	(to cover or extend over [10] an area or time period; "Rivers traverse the valley floor", "The parking lot spans 3 acres"; "The novel spans three centuries")
Sweep	(clean by sweeping; "Please sweep the floor")
Sweep	(win an overwhelming victory in or on; "Her new show dog swept all championships")
Sweep	(cover the entire range of)
Swing, sweep, swing out	(make a big sweeping gesture or movement)

offering a highly intelligent and rational CAD system, there is still a very long way to go. For instance, the current system cannot properly handle such commands as "*draw a concentric circle with the last one*", "*delete the biggest green circle*" etc., as this requires even an higher level of intelligence to infer related semantics and the underlying logics: what the "last one" refers to and how to tell the "biggest" are, so far, not understandable by our prototype system.

Another limitation of the current approach is that only sequential voice inputs are allowed, as opposed to episodic one. For example, such expressions as "*give me a circle please, oops, can you zoom in first? OK, radius: 30*" cannot be properly handled by the current system. These are far more challenging, as Russell and Norvig put it, "We will see before too long that achieving perfect rationality – always doing the right thing – is not feasible in complicated environments. The computational demands are just too high" [37].

Moreover, the approach presented in this paper did not take into account using the voice to input parameter values. For instance, if the user utters "*give me a circle with a 25 mm radius and center here*", the proposed approach is still inadequate to handle such patterns. A preliminary work to resolve this problem is presented in [38], and is still under active development.

For the future work, we wish to incorporate more sources of knowledge, including common sense, contextual discourse, the user's customs and personal preferences etc, together with more variations of knowledge representations and data structures in the VeCAD system. Also, as a further goal, we hope to be able to extend similar approaches to collaborative CAD systems, where two persons talk in natural voices, discussing different geometries, topologies, materials, or modeling strategies, and the VeCAD system can accordingly simulate/update the conceived models, until both collaborators reach an agreement.

Acknowledgements

We would like to thank the Department of Mechanical Engineering and the Committee on Research and Conference Grants of the University of Hong Kong for supporting this project (Project No. 200707176061). Thanks also go to the Research Grants Council of Hong Kong SAR government for supporting the project (Project Code: 716808E).

Appendix

See Table A.1.

References

[1] Chu CC, Mo J, Gadh R. A quantitative analysis of virtual reality based computer aided design system interfaces. Journal of Computing and Information Science in Engineering 2002;2:216–23.
[2] Michael AG, David SE, Timothy WF. The integrality of speech in multimodal interfaces, vol. 5. ACM Press; 1998. p. 303–25.
[3] Holmes JN. In: Holmes W, editor. Speech synthesis and recognition. 2nd ed. London (New York): Taylor & Francis; 2001. 298 p.
[4] Liu X, et al. Virtual DesignWorks - Designing 3D CAD models via haptic interaction. CAD Computer Aided Design 2004;36(12):1129–40.
[5] Liu X, et al. Manipulation of CAD surface models with haptics based on shape control functions. CAD Computer Aided Design 2005;37(14):1447–58.
[6] Diniz N, Branco C. An approach to 3D digital design free hand form generation. In: Proceedings of the international conference on information visualization. 2004.
[7] Chu CPP, Dani TH, Gadh R. Multi-sensory user interface for a virtual-reality-based computer-aided design system. CAD Computer Aided Design 1997; 29(10):709–25.
[8] Gao S, Wan H, Peng Q. An approach to solid modeling in a semi-immersive virtual environment. Computers & Graphics 2000;24(2):191–202.
[9] Weyrich M, Drews P. An interactive environment for virtual manufacturing: The virtual workbench. Computers in Industry 1999;38(1):5–15.
[10] Kou XY, Tan ST. Design by talking with computers. Computer-Aided Design and Applications 2008;5(1–4):266–77.
[11] Becchetti C. In: Ricotti LP, editor. Speech recognition: Theory and C++ implementation. Chichester: Wiley; 1998. 407 p.
[12] Schroeder MR. Computer speech: Recognition, compression, synthesis ; with introductions to hearing and signal analysis and a glossary of speech and computer terms. 2nd ed. Berlin (New York): Springer; 2004. 375 p.
[13] Microsoft Word. http://www.microsoft.com/word/.
[14] Dragon NaturallySpeaking Medical. http://www.nuance.com/naturallyspeaking/medical/.
[15] Peacocke RD, Graf DH. An introduction to speech and speaker recognition. Computer 1990;23(8):26–33.
[16] Dani TH, Gadh R. COVIRDS: A conceptual virtual design system. In: ASME database symposium. Boston (MA, USA): ASME; 1995.
[17] Dani TH, Gadh R. Creation of concept shape designs via a virtual reality interface. CAD Computer Aided Design 1997;29(8):555–63.
[18] Bolt RA, Herranz E. Two-handed gesture in multi-modal natural dialog. In: Proceedings of the 5th annual ACM symposium on User interface software and technology. Monteray (CA, United States): ACM; 1992.
[19] Speak4CAD. http://www.speak4cad.com/.
[20] ThinkDesign. http://www.think3.com/en/default.aspx.
[21] Manning CD. In: Schtze. H, editor. Foundations of statistical natural language processing. Cambridge (Mass, London, England): MIT Press; 1999. 680 p..
[22] Brill E, Mooney RJ. Overview of empirical natural language processing. AI Magazine 1997;18(4):13–24.
[23] Berger AL, Della Pietra VJ, Della Pietra SA. A maximum entropy approach to natural language processing. Computational Linguistics 1996;22(1):39–68.
[24] Lesk M. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In: Proceedings of the 5th annual international conference on Systems documentation. Toronto (Ontario, Canada): ACM; 1986.
[25] Miller GA. WordNet: A lexical database for English. Commun. ACM 1995; 38(11):39–41.
[26] Banerjee S, Pedersen T. Extended gloss overlaps as a measure of semantic relatedness. In: Proceedings of the eighteenth international joint conference on artificial intelligence. 2003.

[27] Naskar SK, Bandyopadhyay S. Word Sense Disambiguation Using Extended WordNet. In: Proceedings of the international conference on computing: Theory and applications. IEEE Computer Society; 2007.
[28] SolidWorks, http://www.solidworks.com/.
[29] Hauptmann AG. Speech and gestures for graphic image manipulation. In: Proceedings of the SIGCHI conference on Human factors in computing systems: Wings for the mind. ACM; 1989.
[30] Brill E. A simple rule-based part-of-speech tagger. In: Proceedings of ANLP-92, 3rd conference on applied natural language processing. 1992.
[31] Marcus MP, Santorini B, Marcinkiewicz MA. Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics 1993;19(2):313–30.
[32] Gong Z, Cheang CW, Hou L. Multi-term web query expansion using WordNet. In: Lecture notes in computer science (including subseries Lecture notes in artificial intelligence and lecture notes in bioinformatics). 2006. Krakow. p. 379–88.
[33] He B, Ounis I. Combining fields for query expansion and adaptive query expansion. Information Processing and Management 2007;43(5):1294–307.
[34] White RW, Marchionini G. Examining the effectiveness of real-time query expansion. Information Processing and Management 2007;43(3):685–704.
[35] Lin HC, Wang LH, Chen SM. Query expansion for document retrieval based on fuzzy rules and user relevance feedback techniques. Expert Systems with Applications 2006;31(2):397–405.
[36] Kou XY, Liu XC, Tan ST. Quadtree Based Mouse Trajectory Analysis for efficacy evaluation of Voice-enabled CAD. In: 2009 IEEE international conference on virtual environments, human–computer interfaces and measurement systems. 2009.
[37] Russell SJ, Norvig P. In: Norvig. P, editor. Artificial intelligence: A modern approach. 2nd ed. Upper Saddle River (NJ): Prentice Hall; 2003. 1081 p.
[38] Xue SK, Kou XY, Tan ST. Natrual voice-enabled CAD: Modelling via natural discource. Computer-Aided Design and Applications 2009;6(1):125–36.