SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

July 2012 Vol. 55 No. 7: 1528–1540
doi: 10.1007/s11432-011-4436-z

Human knowledge acquisition from 3D interaction in virtual environments

CHENG Cheng*, JIANG Ru & DONG XueMei

Beijing Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, Beijing 100081*, China*

Received August 24, 2010; accepted March 31, 2011; published online January 2, 2012

Abstract Recently, virtual environment (VE) based design and planning, which is a kind of interaction intensive computing, has shown great potential. Most users of such systems are specialists or technicians and their 3D interactions with the system embed a great deal of knowledge and skills. Thus, how to integrate human knowledge and skills into VE is a great challenge to researchers in the human computer interaction field. This paper proposes a method for acquiring user knowledge from VE. The main ideas and work include, abstracting the interactive process and formalizing the interactive semantics, as well as providing a semantic model of an interactive information repository, from which user interactive processes can be retrieved and all kinds of application logics can be established. A virtual assembly planning system is introduced as an example of a practical application of this method. Experiments show that the related models can well capture user knowledge and retrieve the interaction process.

Keywords human computer interaction, interactive semantics, virtual environment, temporal logic, virtual manufacturing, virtual assembly

Citation Cheng C, Jiang R, Dong X M. Human knowledge acquisition from 3D interaction in virtual environments. Sci China Inf Sci, 2012, 55: 1528–1540, doi: 10.1007/s11432-011-4436-z

1 Introduction

Virtual manufacturing environments have the most critical requirements of direct manipulation (DM), which is a main style of operation in such virtual environments where users interact with virtual objects via a variety of input devices. Virtual assembly is a typical such an VE, allowing humans to carry out virtual prototyping, design verification, simulation, training, and assembly planning [1, 2]. Because VEs always provide a realistic environment in which the participants can interact with virtual objects in the same way as they do in reality, and most of the users of virtual manufacturing systems are specialists or technicians, the participants' manipulations reflect a great deal of knowledge and a variety of technics in the domain. It would thus be significant to capture the elements of the interactions.

One benefit of acquiring user knowledge is to derive all kinds of application logic, such as assembly planning, from interaction. Because product design is a creative activity, although we can automate routine activities, attempts to harness artificial intelligence technology to provide a design have not been successful. Another benefit is that we can use such knowledge to retrieve interaction process.

*Corresponding author (email: guoguocheng@vip.sina.com)

Virtual manufacturing applications are interaction intensive, and rework in such virtual design or planning processes is inevitable. For large scale products, which include a large number of parts, the capability of automatically simulating human interactive operation is necessary for virtual manufacturing systems.

Some of the most critical problems hindering knowledge acquisition in VE are understandings of human intents and availability of interactive semantics. Understanding user's intent will make human computer interaction more natural and harmonious, while the semantics construction can contribute to the abundance and usefulness of knowledge in VE. The former problem requires researchers to abstract user behavior, while the latter requires a formalization of DM and multimodal inputs [3–6]. Both involve complex DM spatio-temporal relationships among the objects, and should be properly organized in a semantic representation. The semantic model is a kind of abstract representation that serves as a bridge between DM and application logics.

Here we present a brief review of several research aspects that are crucial to the above mentioned problems in VEs. These aspects are scattered throughout areas that are traditionally referred to as motion description, semantic models, and system integration.

Related research on motion description can be classified into two categories: one is the task-level approach, which designs a set of task-dependent primitives and tries to recognize the subject task as a sequence of symbolic primitives. The other is the trajectory level approach, in which the demonstrated trajectory, and the applied force if necessary, is directly transformed into robot motion [7].

With respect to the task-level approach, many researchers have used Petri nets to represent the assembly process [8,9]. Cao [10], Thomas [11], and Zha [12] introduced a framework for robotic task sequence planning. McCarragher [13] modeled assembly as a discrete event dynamic system using Petri nets. The disadvantages of Petri nets are that they cannot depict object's continuous behaviors, nor have they a static description of the assembly. Temporal logic has also been used in assembly modeling. Seow [14] used temporal logic to model the assembly sequence.

The trajectory level approach has been extensively investigated in the robotics field with respect to robot motion and manipulation. A very attractive aspect is human motion imitation, which means that robot behavior is taught by demonstration and observation. The critical problem is how to specify and model the spatial trajectory. Kim [15] represented the trajectory of a humanoid using a 3rd Bezier curve. Faria [16] studied how a 3D hand trajectory can be segmented by curvatures and hand orientation for classification using a probabilistic approach. Richter [17] and Hyodo [18] used haptic and force feedback information to abstract and specify motion. To discern ambiguity among demonstrations, the trajectories corresponding to each essential interaction are generalized by calculating their mean and variance [7]. The trajectory of human manipulation can then be used to reproduce skilled behavior.

Various related researches on aspect of semantic model have focused mainly on product representation. These kinds of semantic models are application domain specific, for example, in virtual assembly they are conventionally called assembly models. Although many assembly models have been proposed in the literature, the relational model [19], hierarchical model [20], and virtual link model [21] are classic ones. These models were all designed for WIMP-based CAD environment, and provided only static descriptions of the assembly. The AND/OR graph [22,23] is a model specific to assembly planning. Horvath [24] attempted to provide a product model that integrates multiple aspect information. For interaction intensive applications, the semantics are more complicated, and should also be stored in a semantic model which is a significant framework for product data. How to represent the interactive process in such a semantic model is still an open research area.

Related studies on the aspect of system integration were mainly concerned with how to integrate different application logics. Acquiring human knowledge is also an important aspect of system integration in virtual manufacturing. Regarding system integration, Ciurana [25], Pierre [26], and Wang [27] studied a product models that include several aspects of manufacturing with the exception of human operation. The majority of assembly researches have focused on modeling an individual distinct virtual application, such as assembly planning. Algorithms for different systems are very difficult to be integrated because their data are highly self-contained. Banerjee [28] mentioned a big limitation of object behavior description, and tried to give a structure to solve the problem.

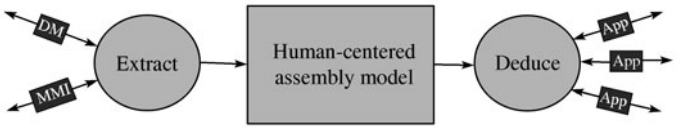

Figure 1 Outline of our research methodology.

Because virtual assembly has the most critical requirement to human computer interaction, we set out to investigate a method for acquiring knowledge from user' interaction within a virtual assembly application. We aim to present an interaction formalization method to make the interactive semantics available, and to give a semantic data model for retrieving the interaction process and acquiring knowledge. The temporal logic language XYZ [29] was used to formalize hybrid user manipulation and to construct the process semantics. We discuss in detail how to reflect and record human manipulation in VEs. Retrieving the user design process and valuable manipulations from the assembly model is demonstrated by a planning algorithm. The main contributions of this paper are that temporal relationships depicting user intents and object traces mapping user DM are constructed, and based on these concepts, DM and the semantic formalization thereof are proposed. An assembly planning algorithm based on the new semantic model is given as an example of knowledge application.

As shown in Figure 1, this paper discusses three aspects. In Section 2, we present a method of interactive system abstraction and formal specification. In Section 3, a semantic model is proposed, while in Section 4, an algorithm for deducing assembly planning is introduced as an example of how to acquire and apply human knowledge based on the interactive semantic model. In the final section we present our conclusion.

2 Interactive system abstraction and modeling

The participants and the computer together make up a VE system. In such a typical interaction intensive computing system, the capabilities of intelligence and computing should be well balanced. Object model theory and object oriented analysis and design techniques form the foundations of VE system design. Each entity in the VE is considered to be an object, and after interaction, some objects are combined to make new composite objects. Therefore, in this paper the system abstraction and interactive semantics construction are both based on object oriented design.

2.1 Perception mechanism construction

Perception is the VE computing capability that enables all specified spatial relationships among the objects to be discovered. Without a perception mechanism, the participants cannot accurately manipulate objects in the virtual space, nor can the VE produce multiple semantics. Almost all the prior perception mechanisms in VEs use a collision detection strategy. This is not sufficient for the demand of natural interaction in a virtual manufacturing application.

A new perception mechanism is proposed based on feature-based design data instead of pure VRML polygon patches. It is built as a capability of the active objects in virtual space, and a set of notations is used to represent the perception algorithms. Here p_i represents the parts that are combined to build an assembly, and fea_i represents the assembly features of a part. Assembly feature is a significant concept in assembly application and refers to a type of geometric shape with specific engineering semantics. Several auxiliary features(af_i) are established for assembly features, such as $Origin, f\_Origin, Major$ axis, $Minor$ axis, and $the3rd$Axis. $Origin$ represents the reference point of a feature. f_Origin refers to a point on the face of a feature indicating the position of the face. $Major$ axis, $Minor$ axis and $the3rd$-Axis are three axes that pass through the $Origin$ and are perpendicular to each other.

Specific perceptions are constructed to cater for particular interactive scenarios. Feature matching perception ($FeaMatch$) automatically judges the assembly feature (fea_i) pairs that the user intends to fit together. The two features relate to the target part and the current manipulated part respectively.

Aligning perception ($Align$) is responsible for discovering the alignment constraint of two axis auxiliary features on two features during the user manipulation. Face mating perception ($FaceMate$) classifies the parallel faces belonging to the currently assembled parts and calculates the potential set of matching pairs. It helps users to determine the final position of the part. Each specific perception has its own unique visual feedback.

2.2 Cognitive modeling of interactive process

The principle of cognition and behavior unity exposes the fact that cognition is not only the regulator of external behavior, but also is formed based on motor activity. In fact, users' experiences, expertise, and knowledge are embedded in their interaction behaviors in VEs. Capturing this kind of information is significant, but we should take into account the limitations of human perception, the interactive context, the intents, and the behaviors that are to be performed to reach the goals [30, 31].

Intent is the VE user's inner activity that imagines the VE reaching a given state next during a stage of interactive process. It is determined by the user's current mental state and current circumstances, and is expressed via temporal hybrid multimodal inputs. The intents of participants are extracted from their DM operations. To achieve this goal, scenarios of interactive assembly are established. The virtual assembly is composed of a scene manager (SM), a space navigator (NV), a root assembly object (AO) and subassemblies (SA_i). The navigator represents a virtual hand in the system. The virtual assembly space is discretized and object behavior is temporalized. The virtual space is separated into a deposit area for parts and an assembly workshop area, moreover the assembly workshop is uniformly partitioned into m sub-spaces, $space_i$. $CurSpace$ is the current working space in which the user is working. The system uses universal linear time. Every discrete event occurs at a discrete time. For any time t_i, we define the next time as $t_i + 1$.

We study the interactive software model from the viewpoints of both domain requirements and cognitive psychology. In fact, there is a clear workflow in every domain application. This is a macro circulation that is determined by the domain requirements. The workflow is composed of a sequence of activities. Each activity is well defined, and in a single activity there may be several critical tasks that must be fulfilled. Furthermore, each task is composed of several operations and can be depicted by a micro interactive sequence, which also involves intents, interactive patterns, and perceptions, as shown in Figure 2.

By a long term observing on cognitive process in virtual assembly, we have discovered that three intents are exhibited in the three main operation phases. These elemental operations are always performed sequentially and iteratively, and form a stable cognitive/motor cycle. The different intents in the cognitive/motor cycle are facilitated by three distinct perception types, namely, $FeaMatch$, $Align$ and $FaceMate$. As such, the intents are named after the respective perceptions used. In the cognitive/motor cycle, when a component is manipulated, these three perceptions are executed sequentially in a fixed order. After the user's affirmation ($Affirm$), the object carries out a short passage of behavior to fulfill the user's intent. This behavior passage is the object's own behavior, also called animation, other than the user's manipulation. Without perception and the object's own automatic motion, the user cannot succeed in accurately moving the object to the proper position in assembly. We combine user manipulation with object's behavior to form a behavior segment (tbs_i). Behavior segments have inherent temporal characteristics, and together constitute the whole behavior of an object. The corresponding temporal behavior segments as shown by the inner thick arced arrows, are formalized and will be created when the cycle has passed.

A user's intent is deduced from DM and multimodal inputs. Every single input modality has a concrete state at any given time. The multi-states of the modalities at a time point form a distinct state vector. A short continuous sequence of distinct modal state vectors embodies the user's intent. We use temporal inference of the short state vector sequence to discover the intent. The strategy is as follows. First, we abstract human cognitive elements and the cognitive process of a task in the real world from the psychological analysis. Second, we suggest a cognitive process and important elements of a task in the VE. Third, we establish inference rules from the obtained state vector sequences of typical interactions. Finally, the rules are applied to a state vector sequence to derive an intent.

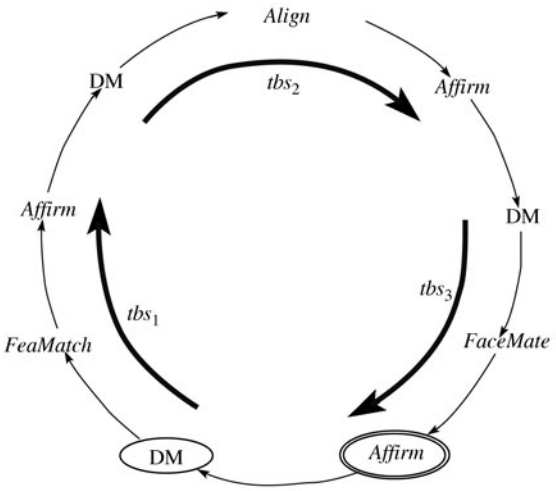

Figure 2 Specific cognitive motor cycle and relevant temporal behavior segments.

3 Semantic model of VE interaction

The above interactive process specification involves several interactive elements, intent, perception and behavior. We prefer to call the interaction specification system an interactive language. Just as a programming language has its semantics, so too does an interactive language need a semantic construction. Interactive semantics can reflect the user's interactive intents and essential elements of behaviors, and from which we can retrieve the interaction process and deduce the user's knowledge. How to design and save these semantics significantly affects the capability of the interactive system. Therefore, we focus on the construction of interactive semantics.

3.1 Temporal relationships representing intent

Relationships are used to depict intents. Because DM has spatio-temporal characteristics, naturally temporal relationships are necessary. Although there may be many different intents during the assembly process, two are essential. One is the intent assembling whereby the user aims to add a constituent to the partial assembly without taking into consideration any accurate orientation, while the other is the intent matching whereby the user aims to search for a feature pair to which certain discrete constraints can be added. Given below are the definitions of the two temporal relationships reflecting the two intents.

Definition 1 (Temporal aggregation relationship)**.** Suppose object A is a component and object B is a subassembly, where A is a constituent component of assembly B. The relationship between A and B is called a temporal aggregation relation, and has the form: (t, A, B), denoted by TAR. $TAR \in T \times O_T \times O_T$, where $T = (t_1, t_2, \ldots, t_n)$ is a set of time points, and $O_T = \cup_{t_i \in T} O_{t_i}$ is the set of all the objects in the assembly with subset O_{t_i} referring to all the objects that have been assembled at time t_i.

The temporal aggregation relationship reflects the participant's intent of assembling. When this intent is captured, a temporal aggregation relationship is constructed. In this way an object aggregation hierarchy is created. Because this creation relies strongly on the spatio-temporal scenario, it contains the assembly sequence information, which itself is a very important aspect in assembly planning.

Definition 2 (Temporal constraint dependency)**.** Let objects o_i, o_j be two components in an assembly. If o_j has a constraint that refers to o_i, the abstract relationship between components o_j and o_i is called a temporal constraint dependency relation, and has the form: (t, o_i, o_j), denoted by TCD. $TCD \in T \times O_T \times O_T$, where $T = (t_1, t_2, \ldots, t_n)$ is a set of time points.

Geometric constraint (cn) is an effective utility for building a computer aided system. A directed geometric constraint mechanism is used in our system, $Interaction3D$. The basic constraint types are mainly $CN\text{-}Pt$ (point constraint), $CN\text{-}Ln$ (line constraint), $CN\text{-}Pln$ (plane constraint) and $CN\text{-}Sph$ (sphere

constraint), while the mechanism includes constraint adding, constraint deleting, constraint propagation, and constraint satisfaction. The constraints are imposed on auxiliary features, with the parameters of the constraints obtained by referring to other parts. By separating the detailed parameters from the constraint, we obtain an important semantic element, namely the temporal constraint dependency relationship.

A temporal constraint dependency relationship reflects the participant's intent of feature pair matching. Adding constraints is necessary when a part is to be fitted to an assembly. The process is implicit and hybrid. When this intent is captured, a temporal dependency relationship can be created. In this way a directed acyclic graph (DAG) of constraint dependencies can be constructed. Because this relies strongly on the feature geometry and assembly technical expertise, it embodies the assembly path information, which is itself another very important aspect in assembly planning. Here we show only two critical temporal relationships. Various other temporal relationships can be established to represent other intents.

3.2 Trace of object behavior

Object behavior reflects the participant's operations in the interactive virtual assembly. Because participants' DM embodies their design knowledge, object behavior is destined to be a significant part of the interactive semantics. To mine a user's design knowledge from the structure of the semantics, we need to conduct a deep analysis of object's behavior as well as to provide a formal definition thereof.

Here, an object is considered to be a hybrid system, which receives both continuous events, such as moving the motion controller of a 3D space mouse, and discrete events, such as pressing any key on the space mouse. These events allow the corresponding flow and jump transitions to be made. A flow transition is a visible motion segment in the VE.

Definition 3 (Temporal behavior segment)**.** Temporal behavior segment (*tbs*) is an elementary unit of behavior, and has the form: $\langle t_i, cn_{\text{start}}, bhv_k, cn_{\text{end}}\rangle$, where $t_i \in T$ is the start time of the behavior segment, and $bhv_k \in BHV$, with BHV being a set of primitive behavior types. $cn_{\text{start}} \in CN$ is the initial condition of the behavior segment expressed in terms of constraints, and $cn_{\text{end}} \in CN$ is the terminating condition also expressed in terms of constraints. Here, CN is the constraint set.

The temporal behavior segment is a fundamental concept of object behavior. All the complicated behaviors can be discretized into such basic behavior cells in the sense of assembly. A segment of behavior is represented as definite kinds of motions with constraint satisfactions. For any such segment, the behavior type is fixed, but the constraint parameters can be varied according to the changing circumstances. This allows the object to perceive the current constraints in real-time and to adjust the parameters of the constraints in the pre- and post-conditions as the behavior is being carried out.

Definition 4 (Short behavior sequence)**.** Short behavior sequence (*sbs*) is a fixed sequence of temporal behavior segments. Each short behavior sequence reflects a basic cognitive/motor circulation in human's task oriented working process, and has the form: $tbs_1; tbs_2; tbs_3$, where ';' denotes the sequential operator between the temporal behavior segments.

A short behavior sequence is a formal specification of task oriented behavior. The temporal characteristics of a temporal behavior segment provide the short behavior sequence with the capability of specifying the starting time for performing the behavior and the explicit order of the three temporal behavior segments. The three consecutive segments must all be present. In most situations, the three temporal behavior segments can be obtained naturally, but sometimes this is not the case. In certain specific situations, we need to create a pseudo segment to fill the empty space in a short behavior sequence. During the assembly process, participants continually repeat the uniform cognitive and motor operations until a part has been fitted in the right place in the assembly. For any component, the cognitive cycle should be carried out at least once to complete the assembly operation.

Definition 5 (Object trace)**.** Object trace (*Otrace*) is a formal specification of object's complete behavior during which the object moves from an initial state to a terminal state. It is a successive sequence of short behavior sequences, and has the form: $sbs_1; sbs_2; \ldots; sbs_n$.

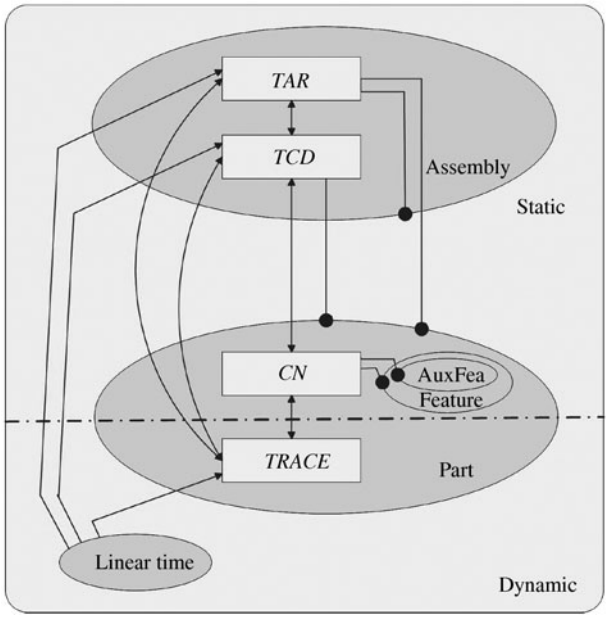

Figure 3 Composition of the semantic model.

The object trace gives a composition of a complete object behavior in virtual assembly. It shows that human interactive behavior can be discretized and that the behavior is repetitive with a universal form. For a complicated assembly process, the cognitive and motor cycle should be carried out several times, so that the trace contains all the short behavior sequences corresponding to each of the cognitive and motor cycles.

Human operations can, thus, be expressed as an object trace. Naturally human interactive behavior is amalgamated into the description of a virtual object. Object trace is the only dynamic semantics for the so-called interactive language. Other than recording all the details of the operation trail, the object trace only captures the key frames of the user's DM. Nonetheless, this description is adequate to retrieve the user's operation for application logic.

3.3 Structure of virtual assembly semantics

The above semantic elements do not exist in isolation. Together they constitute the spatio-temporal semantics of human interaction. In a typical VE, e.g., a virtual assembly, the system is composed of objects (parts) and composite objects (assemblies or subassemblies). A system state is represented as a group of objects' states in the scenario, and all the semantic elements belong to either the objects or the composite objects.

The composition of the different semantic aspects is shown in Figure 3. Although this is a semantic model for a virtual assembly application, it is not limited to this area. TAR, TCD, CN, and $TRACE$ represent the sets of temporal aggregation relationships, temporal constraint dependency relationships, constraints, and object traces, respectively. The sets of CN and $TRACE$ are encapsulated in the constituent object(part), which is modeled by geometric features (fea_i) and auxiliary features (af_i), while the sets of TAR and TCD are encapsulated in the composite object or assembly. From another perspective, TAR, TCD, and CN are static descriptions, while $TRACE$ dynamically describes the assembly process.

The discrete events at linear discrete time points trigger the creation of model elements, as depicted by the single arrow lines in Figure 3. The creation of relationships and constraints refers to other objects, as shown by the dot ended lines. A constraint refers to a part's features and also to the feature's auxiliary features, such as $Origin$, $Major$ and $Minor$. There are also relevancies between relationships and constraints, as alluded to by the double arrows. For example, the relevant temporal dependency relationships *tcd*s in TCD should be canceled when a temporal aggregation relationship *tar* in TAR is canceled, and vice versa. It is very obvious that TCD has a close relationship with CN. This means

that, when a *tcd* is produced, at least one constraint *cn* in CN is also produced. Multiple *cn*s may share a common *tcd*, and therefore, the subsequent *cn*s may not cause changes in the TCD.

The $TRACE$ set contains all the traces that are constructed using constraints in CN. The relationship between CN and $TRACE$ is actually a very close, yet complicated one, because it not only includes the relevancy that canceling a constraint results in the related trace being canceled and disabling a constraint results in the trace being disabled, but also involves the problem of perceiving changed constraints concurrently when executing a trace. The relevance between the $TRACE$ and the temporal relationships is that the trace simulation process accompanies the construction of *tar*s and *tcd*s.

Primitive operations to the model are defined to build the semantic elements and to maintain the data repository corresponding to a user's natural DM. The functions are listed below, where X can be *tar*, *tcd*, *cn*, *tbs*, *sbs* or *Otrace*:

a) $CreateX()$: Create an X object;

b) $AddX()$: Add an X object to the assembly model;

c) $DelX()$: Delete an X object from the assembly model;

d) $DisableX()$: switch the state of an X object to disabled;

e) $CnSatisfaction()$: Cause the constraints to be satisfied;

f) $CnPropagation()$: Cause a constraint to be propagated.

A deductive system is also constructed to facilitate interactive dynamics and to maintain consistency in the semantic model. In an interactive assembly process, humans do not need to express everything explicitly and in detail; e.g., if a user adds a part to a complex assembly, the temporal aggregation relationship goes down the hierarchy from the root node to an appropriate location. The deductive system can verify the model operations and ensure that the VE semantics are consistent.

3.4 Formalization of DM

In this study, our aim is to construct interactive semantics based on the interactive process abstraction. Most of the researches in the area of semantics have focused on distributed or concurrent computing systems. As such, there is not much literature on semantics of interactive system. Yet, the semantics construction for interactive systems has great significance.

An axiomatic semantics approach is used to formalize the interactive semantics. We redefine the semantic form to suit the interactive system, and axioms and propositions have the form: $P\{Manip; Perc; Affirm\}R$, where P is called the precondition, $Manip$ represents DM or multimodal inputs, $Perc$ represents the perception process that is executed by the active object in a VE, $Affirm$ represents the user's affirmation event that is related to the relevant perception feedback, and ';' represents a sequential operation. R is called the postcondition. Together $Manip$, $Perc$, and $Affirm$ compose a micro interactive structure, and after executing such a micro control structure, the relevant semantics can be obtained.

The temporal logic language XYZ is used to realize this semantics approach. This is a uniform framework that can express both commands for state transitions and formulae for logical reasoning. Rules for state transitions are specified in the form: $[LB = id_i \wedge P_i \Rightarrow Qi \wedge BHV = Bhv_i \wedge \mathrm{OLB}=\mathrm{id}_{N1};]$, where LB is the control variant of a state, id_i is the identifier of a state, $\Rightarrow$ is the deduction symbol, and P_i, Q_i are both proper temporal logic expressions. P_i describes the current state of the system, while Q_i describes the state of the system at the next time. Bhv_i represents the behavior of the objects.

A series of scenarios has been constructed in an assembly system. Due to space limitations, only one such scenario, *Aligning*, is analyzed here. The coincidence of the two principal axes of the matched feature pair is perceived and some semantic elements are built. The interactive process and semantics are specified as shown in Figure 4, where $Type()$ gives the type of the auxiliary feature, $Angle()$ evaluates the angle between two axes, $MatchedAuxRep()$ displays the matched auxiliary features, $FaceMate()$ perceives a face pair mating, '@' indicates the subject of the behavior, and cn_{original} is a constraint that defines the original position and orientation of the manipulated part. *Coincidence* and *Dist* are constraint types. The constraints are organized in a DAG. Two of the three temporal behavior segments in a short behavior sequence are created, while the third is created in the next phase. $Trans$ and Rot represent the translation and rotation operations respectively.

$$
\begin{aligned}
\Box[&LB = l_3 \wedge Scenario == Aligning@SA_1 \\
&\wedge CurSpace == Space_i \wedge Translate@NV \\
&\wedge \exists af_1 \wedge Type(af_1) == Axis \wedge af_1 \in fea_1 \\
&\wedge \exists af_2 \wedge Type(af_2) == Axis \wedge af_2 \in fea_2 \\
&\wedge Angle(af_1, af_2) \leqslant \theta \wedge MatchedAuxRep(af_1)@p_1 \\
&\wedge MatchedAuxRep(af_2)@p_2 \wedge \$O(Affirm@NV) \\
\Rightarrow\ & Scenario = FaceMating@SA_1 \wedge CurSpace == Space_i \\
&\wedge BHV = (CreateCn(cn_1, Coincidence, af_1, af_2)@p_1 \\
&\wedge AddCn(cn_1)@p_1 \wedge AddTcd(t, p_1, p_2)@SM \\
&\wedge CnPropagation()@SA_1 \wedge CnSatisfaction(cn_1)@p_1 \\
&\wedge CreateCn(cn_2, Dist, d_0, p_1.Origin, p_2.Origin)@p_1 \\
&\wedge AddCn(cn_2)@p_1 \wedge CnSatisfaction(cn2)@p_1 \\
&\wedge CnPropagation()@SA_1 \wedge DisableCn(cn_2)@p_1 \\
&\wedge CreateTbs(tbs_1, t, cn_{\text{original}}, Trans, cn_1 + cn_2)@p_1 \\
&\wedge CreateTbs(tbs_2, t, cn_{\text{original}}, Rot, cn_1)@p_1 \\
&\wedge AddTbs(tbs_1)@p_1 \wedge AddTbs(tbs_2)@p_1 \\
&\wedge FaceMate()@p_1) \wedge t := t + 1 \wedge \$OLB = l_4;]
\end{aligned}
$$

Figure 4 Specification for Aligning scenario.

4 Deriving assembly planning from the semantic model

To date, assembly planning is still a difficult problem and planning algorithms intrinsically have a high complexity. The WIMP interface interactive planning system cannot provide users with a knowledge enriched environment to conceive the assembly design. Some VE based planning methods still use algorithms to search the optimized solution, and the complexity of the algorithm is the same as that of the algorithms in WIMP. Design knowledge and expertise have not been utilized sufficiently in these methods.

The semantic model provides a data repository, based on which every conceivable application algorithm can be devised, taking advantage of the expert's knowledge and skills. Assembly planning is such a typical application. The algorithm below shows how the spatio-temporal semantic information can be used to derive a planning design. User's intents are retrieved from TAR and TCD, and the detailed interactive operations are reconstructed from $TRACE$.

The semantic model based assembly planning algorithm produces an assembly sequence Seq, which is output through a function (see Figure 5). The notation ';' is a sequential operator, e.g., '$p_i; p_j$' means p_i precedes p_j, and '[]' is the concurrent operator, e.g., $[p_{i1}, p_{i1}, \ldots, p_{in}]$ denotes that p_{ij} should be assembled concurrently.

The achieved sequence is feasible. The system provides services, such as adjustment operations, that allow the user to make local changes to the assembly quickly. Because a VE provides a realistic environment and allows users to take into consideration many factors, participants can apply their mechanical skills completely. Different sequences can be produced immediately after a local adjustment. An evaluation based on metrics can be carried out to ascertain whether the sequence has reached a certain standard and is superior to the existing assembly planning. Through continuous local modification operations, we can obtain an optimized sequence at any time.

Together with deducing the assembly sequence, the algorithm also calls a function $PathOptimize()$ to deduce an assembly path for each component. The function mainly reads the object trace and optimizes the short behavior sequence micro-cycles. Three short behavior sequences, S_1, S_2, and S_3, representing a regular assembly process and the corresponding optimized path are shown in Figure 6. The solid line segments m_i and circles r_i represent the realistic translation and rotation motions, while the dashed segments and circles denote the pseudo translation movements and rotation movements, respectively, built for formalization. The lines and circles labelled with '′' represent a reduction to S_1 and S_2 unconstrained short behavior sequences, while S_3 is a constrained one. A strategy for optimizing S_1 and S_2 is realized

AssemblyPlanning (A_0):
Step 1: Assembly sequence of A_0: $Seq \leftarrow \phi$;
Initialize the variables:
$C_1 \leftarrow \phi; C_2 \leftarrow \phi; C_3 \leftarrow \phi; arr_0 \leftarrow \phi; stack_1 \leftarrow \phi$;
$CurTAR$ is a set in the memory storing all *tar*
relations produced for assembly A_0;
$CurTCD$ is a set in the memory storing all *tcd*
relations produced for current assembly A_0
Step 2: if $Type(A_0) == part$ then Return;
Step 3: For $\forall\ tar_i \in CurTAR \wedge tar_i.B == A_0$
Classify the tar_i into groups, in each of which
the elements have the same t tag;
then arrange them in descendant order to an
array arr_0.
Step 4: Construct the constraint dependency graph DAG using the *tcd*s from $CurTCD$.
Step 5: While $arr_0 \neq \phi$ Do
Acquire a constituent component of A_0:
$C_1 \leftarrow arr_0$.Get();
For $\forall\ tar_i \in C_1$ Do $C_2 \leftarrow C_2 \cup tar_i.A$;
Analyze C_2 in the DAG by depth first searching:
If all the components which depend on the parts in C_2 have been disposed
then push C_2 into stack $stack_1$;
Else arr_0.Add(C_2);
Step 6: While $stack_1 \neq \phi$ Do
Step6.1: $C_3 \leftarrow stack_1$.Pop();
For $\forall\ p_{ij} \in C_3$ Do
Add the concurrent components to Seq:
Seq.Add ($[p_{i1}, p_{i2}, \ldots, p_{in}]$);
Step6.2: optimize the assembly path of every component
based on the object's trace:
PathOptimize(p_{ij});
Step6.3: if Type(p_{ij})==Assembly then
AssemblyPlanning (p_{ij});

Figure 5 Assembly planning algorithm based on the semantic model.

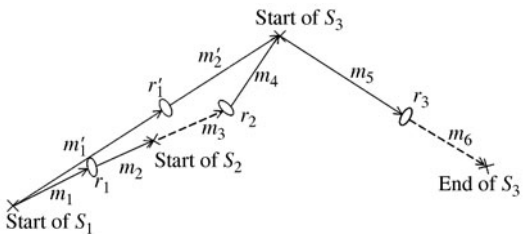

Figure 6 The diagram of a sbs micro-cycle and the optimized sbs.

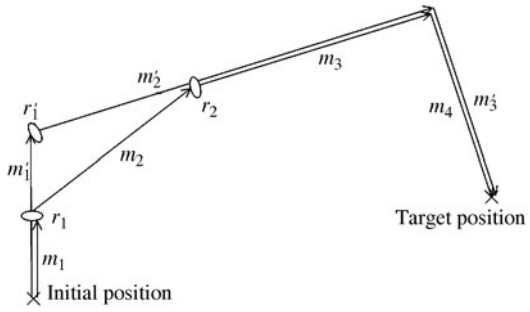

Figure 7 The real path of a sbs micro-cycle and the optimized path segment.

in the function, and the resulting path is constructed from a new short behavior sequence $m_1' r_1' m_2'$ plus S_3.

A view of the concrete movement trail of an assembly path is given in Figure 7. The inner lines show the original behavior trail, while the outer contour is the corresponding optimized path. For a complicated assembly process, the path is composed of several such trail segments with the unconstrained short behavior sequence micro-cycles being optimized.

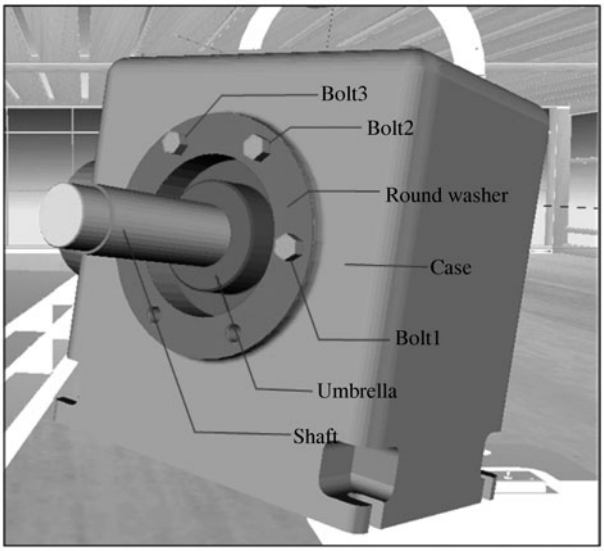

Figure 8 Partial assembly with seven parts.

The achieved assembly path is feasible. Several important points captured during the interaction process are critical for path planning and these are recorded in the object trace. The path can be optimized further if more factors are considered. In robot assembly workshops, the object's trace is used for path planning for the robot instead of an offline program.

Other than searching for an optimized planning, we aim to provide a feasible plan and to use metrics to evaluate the result. Local assembly modification is supported by the system and any change can be made quickly given that every part has been manually assembled at least once. The user can determine whether the acquired sequence and the path of the assembly are good enough from the evaluation. They can easily ascertain the progress of the recent planning compared with previous ones. If the evaluation result does not fall into the anticipated scope, local modifications can be made until the expected result is reached. Evaluation functions rely heavily on the circumstances of the plant and the goals of the designers, and we can devise a number of the metrics.

A prototype virtual assembly system *Interaction*3*D* has been built based on the semantic model presented in this paper. PCs and 3Dconnexion space mice are used. The software has been developed on the platforms of Open Inventor 6.0 and Microsoft Visual C++ 6.0. A 36 parts gear case assembly is used as a product example to verify and validate the semantics framework. Assembly and disassembly can easily be performed by DM. A new perception mechanism and a directed constraint mechanism support the DM. The abstraction of cognition and behavior and the interactive semantics guide the system in creating temporal relationships and object traces. XML is used to save and retrieve the interaction process. To simplify the example, a partial case assembly with seven parts is shown in Figure 8. The corresponding XML content with respect to the TARs, TCDs, CNs, and $TRACE$s is given in Figure 9.

5 Conclusions

A method for acquiring human knowledge from DM has been proposed. We have formalized human cognition and behavior in VEs, and devised temporal relationships and object behavior traces to reflect user's intents and behaviors. The semantic model take advantage of recording a hybrid human-computer interaction process. Experts' knowledge and the experiences of participants can be retrieved and utilized to construct application algorithms. A desktop virtual assembly system *Interaction*3*D* has been constructed based on the semantic model to demonstrate how expert users' knowledge is acquired from natural 3D DM. Practice shows that the temporal semantic model can support DM and human knowledge acquisition in a VE.

The disadvantages of this model are that currently, only a few intents are represented, and many tech-

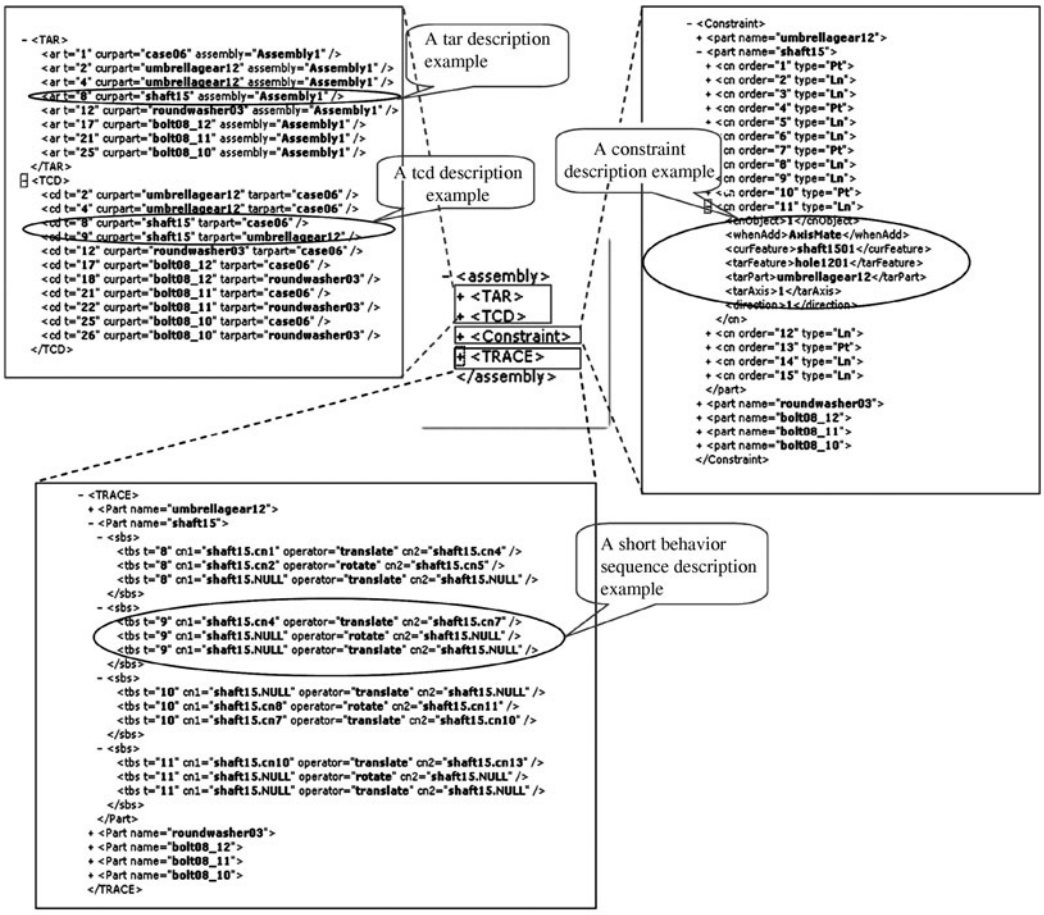

Figure 9 XML semantics description of the partial assembly in Figure 8.

nical constraints still have not been included. Future work includes continued improvement to the semantic model to support representing more complicated intents and technical experiences. In essence, this paper has proposed a semantic framework for human-computer interaction in VEs. The goal is to solve the current problem of insufficient human knowledge and computing system integration. Although there is scope of discrepancies in the understanding of application domains, we believe that such a semantic framework is indispensable on the road toward promoting virtual design and virtual manufacturing practice.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Grant No. 60773046) and Open Project Foundation of the State Key Laboratory of Computer Science (Grant No. SYSKF0905).

References

1 Yuan X B, Yang S X. Virtual assembly with biologically inspired intelligence. IEEE Trans Syst Man Cybern C, 2003, 33: 159–167
2 Smith S S F, Smith G, Liao X. Automatic stable assembly sequence generation and evaluation. J Manufact Syst, 2001, 20: 225–235
3 Prendinger H, Ishizuka M. Symmetric multimodality revisited: Unveiling users' physiological activity. IEEE Trans Indust Electr, 2007, 54: 692–698
4 Zhang J W, Knoll A. A two-arm situated artificial communicator for human-robot cooperative assembly. IEEE Trans Indust Electr, 2003, 50: 651–658

5 Rodriguez A, Basanez L, Celaya E. A relational positioning methodology for robot task specification and execution. IEEE Trans Robot, 2008, 24: 600–611
6 Chueh M, Au Yeung Y L W, Lei K P C, et al. Following controller for autonomous mobile robots using behavioral cues. IEEE Trans Indust Electr, 2008, 55: 3124–3132
7 Ogawara K, Takamatsu J, Kimura H, et al. Extraction of essential interactions through multiple observations of human demonstrations. IEEE Trans Indust Electr, 2003, 50: 667–675
8 Hsieh F S. Analysis of flexible assembly processes based on structural decomposition of Petri nets. IEEE Trans Syst Man Cybern A, 2007, 37: 792–803
9 Wu N Q, Zhou C M, Li Z W. Resource-oriented Petri net for deadlock avoidance in flexible assembly systems. IEEE Trans Syst Man Cybern A, 2008, 38: 56–69
10 Cao T H, Sanderson A C. Task decomposition and analysis robotic assembly task plans using Petri nets. IEEE Trans Indust Electr, 1994, 41: 620–630
11 Thomas J P, Nissanke N, Baker K D. Hierarchical Petri net framework for the representation and analysis of assembly. IEEE Trans Robot Autom, 1996, 12: 268–279
12 Zha X F, Lim S Y E. Assembly/disassembly task planning and simulation using expert Petri nets. Int J Product Res, 2000, 38: 3639–3676
13 McCarragher B J. Petri net modelling for robotic assembly and trajectory planning. IEEE Trans Indust Electr, 1994, 41: 631–640
14 Seow K T, Devanathan R. A temporal framework for assembly sequence representation and analysis. IEEE Trans Robot Autom, 1994, 10: 220–229
15 Kim S H, Imai H, Sankai Y. Interactive task generation for humanoid based on human motion strategy. In: Proceedings of the 2004 IEEE International Workshop on Robot and Human Interactive Communication, Hurashiki, Okayama, 2004. 485–490
16 Faria D R, Dias J. 3D hand trajectory segmentation by curvatures and hand orientation for classification through a probabilistic approach. In: The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, USA, 2009. 1284–1289
17 Richter C, Fischer M, Irlinger F, et al. A spatial path specification system for mechanism development. In: International Conference of Human System Interaction 2009, Catania, Italy, 2009. 236–241
18 Hyodo S, Ohnishi K. A method for motion abstraction based on haptic information directionality and an application to haptic motion display system. IEEE Trans Indust Electr, 2009, 56: 1356–1363
19 Li L, Wesley M A. AUTOPASS: an automatic programming system for computer controlled mechanical assembly. IBM J Res Dev, 1979, 21: 321–333
20 Eastman C M. The design of assemblies. Society of Automotive Engineers, Technical Paper Series, 1981, 0148-7191/81/0223-0197
21 Lee K, Gossard D C. A hierarchical data structure for representing assemblies: part 1. Comput Aid Des, 1985, 17: 15–19
22 Homem de Mello L S, Sanderson A C. AND/OR graph representation of assembly plans. IEEE Trans Robot Autom, 1990, 6: 188–199
23 Gu T L, Xu Z B, Yang Z F. Symbolic OBDD representations for mechanical assembly sequences. Comput Aid Des, 2008, 40: 409–520
24 Horvath L, Rudas I J, Jezernik K. Towards content oriented integration of product and robot system models. In: International Symposium on Logistics and Industrial Informatics, 2007. 7–12
25 Ciurana J, Garcia-Romeu M L, Ferrer I, et al. A model for integrating process planning and production planning and control in machining processes. Robot Comput-Integrat Manufact, 2008, 24: 532–544
26 De Lit P, Danloy J, Delchambre A, et al. An assembly-oriented product family representation for integrated Design. IEEE Trans Robot Autom, 2003, 19: 75–88
27 Wang H, Dong X, Duan G H, et al. Assembly planning based on semantic modeling approach. Comput Indust, 2007, 58: 227–239
28 Banerjee A, Banerjee P, DeFanti T, et al. A behavioral layer architecture for telecollaborative virtual manufacturing operations. IEEE Trans Robot Autom, 2000, 16: 218–227
29 Tang C S. Temporal Logic Programming and Software Engineering (in Chinese). Beijing: Science Press, 1999
30 Bedny G, Karwowski W, Bedny M. The principle of unity of cognition and behavior: implications of activity theory for the study of human work. Int J Cognit Ergon, 2001, 5: 401–420
31 Kulyk O, Kosara R, Urquiza J, et al. Human-centered aspects. In: Kerren A, et al. eds. Human-centered visualization environments, LNCS 4417. Berlin: Springer-Verlag, 2007. 13–75