Complex Event Processing for Intent Understanding in Virtual Environments

Sebai Mounir and Cheng Cheng

***Abstract*—Virtual manufacturing environments need complex and accurate 3D human-computer interactions. Current virtual environments (VEs) suffer from one major issue which is the heavy loads of the users both on cognitive aspect and motor operational aspect. In order to solve this issue, the solution presented aims to increase both machine's cognitive capability and the throughput of the system. This solution is mainly based on techniques and methods of a well-established field known as complex event processing (CEP). Our approach applies CEP to input events in multimodal systems, the events (State vector) which are produced in the system are received, filtered, aggregated or transformed into higher-level intents using a rule-based system. The experiments have shown that intent-driven software construction method and CEP (Complex Event Processing) have a great potential in both, enhancing the naturalness and efficiency of human-computer interactions (HCI) and increasing the throughput over low latency (increase the responsiveness of a system). It also can be considered as an effective analysis method for human-centered VE developments.**

***Index Terms*—Complex event processing (CEP), human-computer-interaction (HCI), intent, multimodal input, virtual environment, virtual assembly.**

I. Introduction

Communication between humans is predicated on multimodality, through both parallel and sequential utilization of multiple perceptual modalities. We communicate not only via verbal language, but additionally through our utilization of intonation, gaze, hand gesture, physical gesture and visages (facial expressions), which ascertains high precision and simplicity. Research on multimodal systems aim to analyze how human-computer interaction can profit from multiple modalities in similar ways.

Never less, the meaning of modality is ambiguous. Bellik [1] defines it as "a concrete form of a particular communication mode" where "mode" refers to the five human senses: sight, touch, hearing, smell, and taste, and to the human different ways of expression: gesture, speech (producing information). L. Nigay [2] also defined modality as: "Multimodality is the capacity of the system to communicate with a user along different types of communication channels and to extract and convey meaning automatically". This means that a multimodal system uses speech, gestures and other human input channels to allow the user to interact with the system.

Based on the definition of Bellik and L. Nigay, we redefine the modality as a "concrete form of a particular mode referring the human senses or their expression ways and using the coupling of interaction language with input devices". Modalities can also be classified as active or passive: it considered active when it is used consciously by the user otherwise it is considered as passive.

VE systems especially rely on multimodal interactions. In this paradigm users use all kinds of input devices to manipulate the virtual objects in VE. Using the eye and the hands, the participants of VE express their interactive intents. Researchers have invented many 3D input devices to manipulate virtual objects in virtual space. We mention some representative works in multimodal interaction. The researches on data glove based gesture and image processing based bare-hand motion capturing are the papers of [3], [4]. Virtual human and virtual hand can greatly enhance immersion and interactive realism. Researchers investigate virtual hand gestures to express every kind of interactive intents [5], and use virtual hand to grab virtual parts to realize accurate assembly operations [6]. Interests on eye-tracking research have been growing rapidly since 2003, and many researches about eye gazing, especially the ones using eye movements as a means of interaction with a computer, have been carried out [7], [8]. But there is not any research on eye and hand modal coordination and integration. We have investigated eye and hand coordination in order to find the principle of this multimodal integration using devices below shown in Fig. 1, those devices will be used later to help capturing the user's intent..

Eye tribe [9] is a device equipped with an eye tracker enables users to use their eye gaze as an input modality that can be combined with other input devices like mouse, keyboard, touch and gestures, referred as active applications.

Fig. 1. Input devices.

The Eye Tribe Tracker in Fig. 2 is an eye tracking system that can calculate the location where a person is looking by means of information extracted from person's face and eyes.

Manuscript received August 9, 2016; revised December 12, 2016.

Both authors are with the Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, Beijing 100081 PRC (e-mail: moumou2788@yahoo.fr, cc@bit.edu.cn/guoguocheng@vip.sina.com).

DOI: 10.7763/IJCTE.2017.V9.1135

The eye gaze coordinates are calculated with respect to a screen the person is looking at, and are represented by a pair of (x, y) coordinates given on the screen coordinate system.

Magellan/SPACE MOUSE [10] is a 3D input device that is used to control the position and orientation of 3D graphical objects in virtual space. The device controls three translational degrees of freedom (X, Y and Z) and three rotational degrees of freedom (A, B and C).

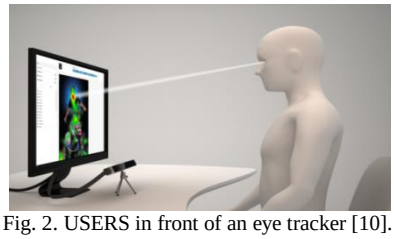


Fig. 2. USERS in front of an eye tracker [10].

II. Complex Event Processing and Multimodal Input Systems

Interactive and Reactive systems respond to the occurrence of events of interest e.g., user interaction or changes in the state of components by performing some computation, which may in turn, trigger new events.

Complex Event Processing (CEP) systems analyze large flows of primitive events received from a monitored environment to timely detect situations of interest, or composite events. In CEP, the processing takes place according to user-defined rules, which specify the relations between the observed events and the phenomena to be detected.

Due to a growing amount of different input devices as well as multimodal interaction techniques, multimodal interactive systems such as interactive surfaces and VE systems, they have seen a fast development in recent years. However, many of those systems tend to use low-level interfaces to deal with user input, leading to systems hard to extend or reuse. CEP can be applied to several domains: sensor networks for environmental monitoring [11]; payment analysis for fraud detection [12]; financial applications for trend discovery [13]; RFID-based inventory management for anomaly detection [14]. The general subject of our work – the application of CEP to interactive systems (Intent Understanding) – has not received much attention so far. The CEP research community does not consider it as a potential application area [15], Complex event processing (CEP) refers to resources that collect different kinds of data from different parts of an IT system, or other sources, to look for meaningful results that can be reported to decision makers.

The term Complex Event Processing was popularized in [16]; however, CEP has many independent roots in different research fields, including discrete event simulation, active databases, network management, and temporal reasoning. It refers to the representation process of events by computer. David Luckham [17] gave definition of complex events in the 2001 book "The Power of Events": complex event detection is a pre-defined collection of tools and techniques designed for analysis and control of a series of interrelated events. And he gives two meanings to the word event. The first meaning refers to an actual occurrence (the something that has happened) in the real world or in some other system. The second meaning takes us into the realm of computerized event processing, where the word event is used to mean a programming entity that represents this occurrence. In the followings, we show the definitions and synonyms of some event processing concepts according to this glossary and the examined articles:

1) Event: Anything that happens, or is contemplated as happening, also used to mean a programming entity that represents such an occurrence in a computing system.
2) Complex event: An event that is an abstraction of other events, (constructed event, high-level event; sometimes it means a composite and a derived event as well).
3) Event stream: a linearly ordered sequence of events.
4) Event attribute: A component of the structure of an event; (event property).
5) Event channel: A conduit in which events are transmitted from event sources (emitters) to event sinks (consumers). (Event connection).
6) Situation: A specific sequence of events.
7) Raw event: An event object that records a real-world event.
8) Detection time: The timestamp when the event was detected.
9) Event Processing Engine: A set of event processing agents and a set of event channels connecting them.

To facilitate a common understanding, we represent the analogs between the mind and CEP in Table I:

TABLE I: Human Cognitive Functions and CEP Functionality

Human Body	Complex Event Processing	Functionality
Senses	Transactions, sensors, input output devices.	Direct interaction with environment, provides information about environment
Nervous System	Chanel, information bus, digital nervous system	Transmits information between sensors and processors
Brain	Rules engines	Processes sensory information, "makes sense" of environment, formulates situational context, relates current situation to historical information and past experiences, formulates responses and actions

Only in recent years, CEP has emerged as a discipline in its own right and an important trend in industry. The founding of the Event Processing Technical Society [18] in 2008 underlines this development. CEP is a powerful technology, it offers several benefits:

1) Permit very high event throughput with low latency, as the CEP engine is specially designed for continuous event stream processing. It uses well tested and implemented algorithms and improvement methods with optimization techniques, beside any future improvement will be immediately available to our System. Latency: How long does it take until the effects of an input event appear in the output?
2) High data rates. Data rate is how many input events per second can the system process
3) Detection, aggregation, fusion and selection of input events are based on a declarative specification (using the Event Processing Language (EPL)). This declarative

approach allow to specify interactions like gestures more abstractly and lets the CEP engine perform the actual processing and maintenance of current and past events. This abstraction also makes it possible to profit from optimizations and other improvements carried by the engine without having to change the applications code.

4) As CEP solutions are also used in non-interactive scenarios such as context event processing in ambient assisted living environment [19].

III. Intents Expression in Virtual Environments

Definition 1: (intent) is a mental activity which helps the VE reaches a given state in a short term during a stage of an interactive process. It depends on user's intellectual state and current working scenario, and it is delivered through multimodal temporal hybrid events.

Intent is concrete both in reality and virtual reality. Intent expression is very natural in our life, but nearly all the intents are ignored in everyday life, we have never paid any attention to them, and some intent were only investigated by the cognitive psychologists for other goals. The reason of explicitly defining intent is that, only the fine and elaborate meanings can satisfy the requirement of design and planning in virtual environments. Intent capturing and understanding will not be an easy thing except the case where the intent is delivered by dictation or other predefined explicit manners. The amount of intents in everyday life is big. For software construction we don't need to realize all these intents. Only the significant intents which are normally used in designing and planning will be investigated here. So the amount of types of intent and the total amount of intents will not be in big numbers.

Definition 2(perception): Is the real-time computation and relevant feedback representation of an active virtual object. It relates with every kind of spatial relationship among the virtual objects. Virtual object communicates with others and computes to recognize a spatio-temporal relationship with other objects in the scene.

Definition 3 (interactive mode): Is spatio-temporal abstract of user's operation behavior in a specific mission within a virtual environment. It includes a specific timing of multimodal cooperation or a temporal sequence of interactive manipulations.

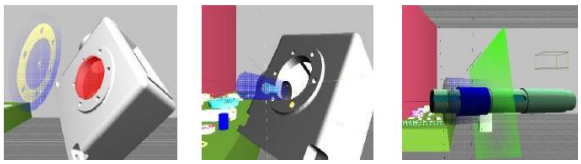
Based on the concepts defined above, we can set up a cognitive model for virtual environments. The cognitive model here means a paradigm of human-computer interaction in VEs. Here we use a virtual assembly application as an example to illustrate the cognitive model. In order to explain the model we will explain the concept of work flow. Work flow is an activity sequence in virtual environments to fulfill the whole mission of assembly. Every activity is made up of a short sequence of tasks, where a task is composed of a set of primitive operations. Every task is formalized by an intent, interactive modes and perceptions. In other words, once an intent of a task occur, the multimodal operations will follow up and normally play several definite interactive modes, and then at a time point an interactive mode will trigger a specific perception. In general, the task sequence in an activity will be completed before entering the next activity. In this situation, intent of next task will bring the VE into the next activity, and so on. But in some exception situation, the user intent will break away from the current activity and transit VE into another activity which is not in the specified work flow when a user' thinking jumping occurs.

Both intent and interactive mode need to be captured in real-time. Intents guide virtual environments to transit from task to task and deduce object's behaviors. They will trigger activity transitions in a work flow, and trigger the jumping among the tasks in an individual activity node. How to get the rules of judging intents is a problem which must be solved. It is impossible to find the solution from computer science and software engineering; instead we can find it from cognitive psychology. We analyze human thoughts and behaviors from experiments and observations.

According to behavior theory [20], human's cognitive activity and motor behavior are connected with each other. Cognitive activities are cooperated with external behaviors; meanwhile, human external behaviors reflect the inner thinking activities. User intents are expressed by their interactive behaviors in virtual environments. Because there are different virtual environment configurations, we should give a clear definition to the situation where intents are expressed. Here the authors mainly analyze intent expressions in a virtual assembly system Interaction3D which is a desktop virtual environment. The main input devices used are space mouse and a non-intrusive eye tracking instrument.

Definition 4(state vector): State vector is a multi-dimensional vector structure which depicts a state of multimodal inputs at a time point, which is expressed as: <t, Sce, Obj1, Obj2, Obj3, Aty, Tsk, Mod1, Mod2, Mod3>. Where ' t' represents the time point of the state,' Sce' represents for a current scene, 'Obji' represents for the related virtual objects, 'Aty' represents for system's current activity, 'Tsk' is a user's current interactive task, 'Modi' represents for the state of input channel i.

Based on a long term observations, we discovered that intents can be well expressed within 5 successive state vectors (multimodal event slices). So in the later part of this paper we will analyze no more than 5 state vectors for intent.



(a) FeaMatching (b) Coincidence (c) FaceMating

Fig. 3. The scenes of main intents and relevant perceptive representations.

Fig. 3 shows some perceptive representations in virtual assembly 3D Interaction. In Fig. 3(a), when the round washer approaches the case and the feature matching relationship has been perceived, the matched feature will be lighted with the red cylinder. This representation of FeaMatching perception will be the condition of collect the input data and create a new state vector. In Fig. 3(b), the Major axis of cylinder feature of the current part bolt and the Major axis of the feature hole of the target part case is lighted with red color. This is the representation of Coincidence perception. After a feature pair matching has been found, user rotate the current part and make the axis of the current feature on the current part

coincident with the axis of the target feature on the target part will make the Coincidence perceptive representation occur. In Fig. 3(c), a green square plane is used to mention the user that a feature pair surface mating is discovered, the green plane also displays the place where this mating will occur.

IV. Using Complex Events Processing in Intent Understanding

CEP engines manage event-driven information systems by employing techniques such as detecting complex patterns, building correlations, and relationships such as causality and timing between many events. Events are represented as objects which have different attributes depending on the type of the event. The event processing is responsible for receiving input events produced by the input devices, applying any processing rules registered with the middleware and providing all events (raw or processed) to the interaction layer.

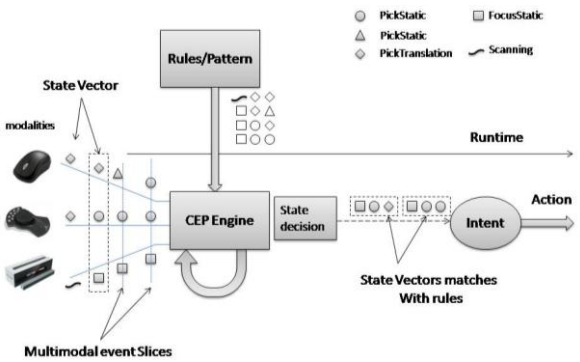


Fig. 4. Architecture of Intent understanding.

In our case as shown in Fig. 4 the inputs of event engine are a state vectors which will be issue from all function of event processing, state vectors enter the engine one by one and are matched against the stream specifications of all queries (Queries are written using the EPL). If an event matches a specification, the query is evaluated and, might generate intent as output and this one will trigger an action. The schema shows the architecture of using CEP in intent understanding.

A. Event Pattern Matching

Event patterns match when multiple events occurs that match the pattern's definition. Patterns can also be temporal (time-based). Pattern matching is implemented via state machines. Pattern expressions can consist of filter expressions combined with pattern operators. Expressions can contain further nested pattern expressions by including the nested expression(s) in round brackets.

There are 5 types of operators:

1) every: Operators that control pattern finder creation and termination.
2) and, or, not : Logical operators.
3) -> (followed-by): Temporal operators that operate on event order.
4) within: Guards are where-conditions that filter out events and cause termination of the pattern finder.
5) at: Observers observe time events as well as other events, such as timer.

The event processing language (EPL) is as follows:

[INSERT INTOinsert_into_def] SELECTselect_list
{ FROMstream_source_list /
MATCHINGpattern_expression }
[WHEREsearch_conditions]
[GROUP BYgrouping_expression_list]
[HAVINGgrouping_search_conditions]
[ORDER BYorder_by_expression_list]
[OUTPUToutput_specification]

We show Some Intents represented by temporal logic in [21] as (1), (2) and we give the equivalence of those descriptions using Event Processing Language.

$$\odot\odot (Sce==Stock) \wedge \odot (Sce==Asm) \quad (1)$$

⊙N is true at time *t*, iff *N* was true at time *t*-1;

The equivalence using Event Processing Language is in Fig. 5:

```
select * from SV[] match_recognize (
measures e1 as SV[1], e2 as SV[2] pattern (e1 e2) define
e1 as (e1.sce==stock ),
e2 as (e2.sce == Asm))
```

Fig. 5. Intent in (1) represented with event processing language.

e1 describe a state Vector at time *t*-1 with attribute sce=stock following by e2 a state at time *t* with attribute sce=Asm, means the interaction space transferred from stock scene to assembling scene,

$$O (Stt==Pick) \wedge O O (Stt==Apprch) \quad (2)$$

O N is true at time *t*, iff *N* is true at next time *t*+1;

The equivalence using Event Processing Language is in Fig. 6:

```
select * from SV[] match_recognize (
measures e1 as SV[1], e2 as SV[2] pattern (e1 e2) define
e1 as (e1.Tsk == Pick),
e2 as (e2.Tsk== Apprch).
```

Fig. 6. Intent in (2) represented with event processing language.

e1 describe a state Vector at time *t* with attribute TSK=Pick following by e2 a state at time *t*+1 with attribute TSK=Apprch, means the The system state transit from the pickup task object approach task.

B. Algorithm of Intent Capture

The EPL offers an *INSERT INTO* clause, which allows creating new events or passing existing events back into an event stream, which results in the following structure for rules:

1) INSERT INTO name of new event type
2) SELECT list of new event attributes
3) FROM stream specification

The EPL globally allows using many SQL like constructs and expressions, and also event stream processing specific treatment clauses and operators. We will explain a few aspects in order to make the algorithms easier to understand. In general, the EPL permits accessing all attributes of event objects by specifying their name (e.g., accessing the attribute 'pos' on the object 'Mouse' would be written as Mouse .pos) and the same for calling any methods defined on it.

Within the FROM clause, it is possible to use different event stream specifications: filter-based, pattern-based or window/view-based. The first (filter-based) is used for directly matching events based on their type/stream name with optional filter expressions to further narrow down the amount of events it applies to. The second(pattern-based) allows more complex matching of event patterns, which might consist of different types of events from different streams and thus also allows expressing temporal relations using the followed-by operator (->). The last (window-based) is used in combination with the first specification method and allows operating on a window of events collected over time or depending on user defined conditions.

An event is a special kind of a message generated by input devices. Analyzing event data is difficult if the data is not normalized into a common, complete, and consistent model. Therein lay the challenges for modeling — to allow virtually any type of event to be defined and to provide maximum infrastructure for supporting event handling. We define the structure of event which is a same with state of vector

StateVec V

{

t, *represents the time.*

Sce, *represents for a current scene.*

Obj[], *represents for the related virtual objects.*

Aty, *represents for system's current activity.*

Tsk, *user's current interactive task.*

Mod[].*represents for the state of input channel i.*

getGaze(), getRot(), getTrans().

}

$$\mathbf{getGaze}() = \frac{|\text{Focus} - \text{CurFearCenter}|}{|\text{Focus} - \text{CurFearCenter}| + |\text{Focus} - \text{TarFearCenter}|} \times 10 \quad (3)$$

$$\mathbf{getRot}() = \begin{cases} \mathbf{10} & \boldsymbol{Max}(\theta_1, \theta_2, \theta_3) \geq \boldsymbol{\pi}/\mathbf{2} \\ \frac{\boldsymbol{Max}(\theta_1, \theta_2, \theta_3)\boldsymbol{Mod}(\boldsymbol{\pi}/\mathbf{2})}{\boldsymbol{\pi}/\mathbf{2}} \times \mathbf{10} & \boldsymbol{Max}(\theta_1, \theta_2, \theta_3) < \boldsymbol{\pi}/\mathbf{2} \end{cases} \quad (4)$$

$$\mathbf{getTrans}() = \frac{|\text{Origin}_{cf} - \text{Origin}_{tf}|}{\text{D}_{\text{Max}}} \quad (5)$$

We explain those equations in experiments and analysis section.

Each input device is wrapped by a collector, contains all the modules and operations necessary to collect the devices specific events from the available devices, convert them into state vector and send them the CEP Engine.

We constructed the algorithm based on the results shown in the figure below (Fig. 7) which were acquired by observing the intents experiments. The metrics was calculated using (3), (4), (5).

First, as shown in Fig. 7, the algorithm will collect the vectors state using the script: "Select (*) from SV-Repository Retain batch 5 Events". This means the moment when five vectors state are collected, they will be sent immediately to the engine to be treated. We will create a vector state whenever an interaction occurs between a human and the input device.

Second, we create statements; A statement is a continuous query registered with an Esper engine instance that provides results to listeners in real-time when the stream of event matches a specification. In our case we have three intents therefore we should define three statements, a statement for each intent.

The query corresponding to Feamatch intent means whenever we get a high Gaze metric (greater than threshold defined by experiments) and a decrease of Translation Metric, the statement will invoke FeaMatchStatListener.

Algorithm : Complex Event Processing to Intent Capture
Input: State Vector
Output: Intent

Begin :

1. SV is an array representing vectors state, SV-Repository is a storage capacity for the vectors state; n is the number of vectors with a read state; ***Select (*) from SV-Repository Retain batch 5 Events.***
2. Continuous query statements can then be created and registered at runtime. We have 3 intents that need an event pattern each

   EPStatementFaceMate ← getQueryFaceMate().
   EPStatementCoincidence← getQueryCoincidence().
   EPStatementFeaMatch← getQueryFeaMatch().

3. Create a listener and attach it to the statement Receiving Statement Results
   FeaMatchStatListener()← execute Intent FeaMatch;
   CoincidenceStatListener ()← execute Intent Coincidence;
   FaceMateStatListener ()← execute Intent FaceMate;

End

getQueryFeaMatch()

" select * from SV[] match_recognize
(measures V1 as SV[1], V2 as SV[2], V3 as SV[3], V4 as SV[4], V5
as SV[5] pattern (V1 V2 V3 V4 V5) define
V1 as V1.getGaze() is > threshold,
V2 as (V1.getTran() >V2.getTran()) & V2.getGaze () > threshold,
V3 as (V2.getTran() >V3.getTran()) &V3.getGaze () > threshold,
V4 as (V3.getTran() >V4.getTran())&V4.getGaze() > threshold,
V5 as (V4.getTran () >V5.getTran ()) &(V1.getTran () <10*
V5.getTran ()) & V5.getGaze () > threshold".

getQueryFaceMate()

"select * from SV[] match_recognize
(measures V1 as SV[1], V2 as SV[2], V3 as SV[3], V4 as SV[4], V5 as
SV[5] pattern (V1 V2 V3 V4 V5) define
V1 as V1.getGaze () < threshold,
V2 as (V1.getTran () >V2.getTran ()) & V2.getGaze () < threshold,
V3 as (V2.getTran () >V3.getTran ()) &V3.getGaze () < threshold,
V4 as (V3.getTran () >V4.getTran ())&V4.getGaze() < threshold,
V5 as (V4.getTran () >V5.getTran ()) & (V1.getTran () <10*
V5.getTran ()) & V5.getGaze () < threshold".

getQueryCoincidence()

" select * from SV[] match_recognize
(measures V1 as SV[1], V2 as SV[2], V3 as SV[3], V4 as SV[4], V5 as
SV[5] pattern (V1 V2 V3 V4 V5) define
V2 as (-0.5 <V1.getTran () –V2.getTran () <0.5),
V3 as (-0.5 <V2.getTran () –V3.getTran () <0.5),
V4 as (-0.5 <V3.getTran () –V4.getTran () <0.5),
V5 as (-0.5 <V4.getTran () –V5.getTran () <0.5)".

Fig. 7. The algorithm schema of CEP intent understanding.

The script of concidence mate means whenever we get a stable Translation Metric (small change) for five state vectors, the statement will invoke CoincidenceStatListener.

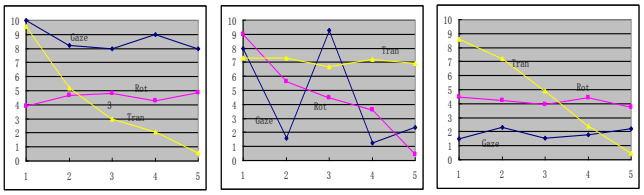
The statement corresponding to the faceMate intent is whenever we get a low Gaze metric (less than threshold defined by experiment) and a decrease of Translation Metric, the statement will invoke FaceMateStatListener.

Third, adding a Listener. When attaching a listener to the statement provided by the engine, it will be invoked by the engine in response to one or more events that change a

statement's result set. Listeners contain what intent will be executing (triggering) as actions.

V. Experiments and Analysis

The goal of experiment is to demonstrate how we designed the Fig. 6 from observation. It can be described as below: The task of the participants is to assemble a cover washer onto a transmission case. When a participant carries out the work, he/she first should pick the washer part and translate it facilitated with approaching perception, then continue to translate it in a company of FeaMatch perception, later, rotate the washer part in the company of Coincidence perception; finally continue to translate the part in a company of FaceMate perception. At every point when a perceptive representation emerges the participant will release an affirmation, which is a discrete event. The scenes of three main intents, FeaMatch, Coincidence and FaceMate, are shown in Fig. 8.



(a) FeaMatching (b)Coincidence (c)FaceMating

Fig. 8. Understanding the intent experiment results.

To compare and analyze the experiment data of distinct intents from state vectors, the experiment conductors gave three normalized metrics for three modalities, which are hand rotation metric, hand translation metric and eye movement metric. The maximal metric value is set to 10 units. First the eye movement parameter is normalized. In virtual environment two important parts are assigned when assembly is going on in the scene, one of which is the part called current part which the user is manipulating, and the other is called target part which is the part that current part is being assembled on. The feature on the current part which has relevant semantics with the current assembly operation is called current feature, while the feature on the target part which has corresponding semantics with the current assembly operation is called target feature. The eye focus during the interactive process in VE can be classified in three types, i.e., focusing on current feature, focusing on target feature, and focusing on any other where. The first type and second type are the most important ones thus the normalized eye movement metric will distinguish these two gaze types. The normalized metric GazeValue is shown in (3). Focus is a user's gaze point on screen; CurFeaCenter is the geometric center of current feature on current part, and TarFeaCenter is the geometric center of target feature on target part.

Secondly, the hand rotation parameter is normalized. For every geometric feature we give names of Major, Minor and the 3thAxis to the three local coordinate axes respectively. Suppose that the angle between the two Major axes of current feature and target feature is $\theta 1$, the angle between the two Minor axes is $\theta 2$, and the angle between the two the 3th Axis axes is $\theta 3$. The normalized metric of rotation is given by (4).

Thirdly, the hand translation parameter is normalized. The geometric centers of the current feature and the target feature are Origincf and Origintf respectively. Suppose the maximal assembly operational distance is DMax which is the possible longest translation distance in the virtual space. The normalized two handed translation metric TranValue is as shown in (5). In order to facilitate the intent analysis, the authors used a kind of line chart to visualize the multimodal metrics. In experiment, three distinct key intents, namely FeaMatch, Coincidence, Facemate, are verified. The experiment results are shown in Fig. 9. The curves labeled with signs Gaze, Rot and Tran represent the statistical data of metric GazeValue, RotValue and TranValue of the interactive assembly process respectively. For intent FeaMatch, the normalized value Gaze and Rot nearly has no change, the normalized value Tran drops linearly. For intent Coincidence, the characteristic of the figure is that, the normalized value of Tran almost has no change, meanwhile, the value of Rot decreases linearly and the value of Gaze shows a distinct jumping. While for intent FaceMate, the value of Tran decreases and the values of Gaze and Rot keep unchanged. The experiment results demonstrate the distinctions among the three main intents we used in assembly process.

In order to verify what we designed and ran our own performance evaluation in real time. We used assembly system, we define rules and patterns. Those rules were actual implementations of intents. As we only wanted to measure the impact of the CEP in intent capturing. For each state vector passed into the CEP Engine, the time was measured it took the Engine to accept the next event. This time (shown in Table II) includes all query evaluations as well as calling all registered event listeners.

TABLE II: Experiments Result

Total (state Vector) transmitted	Average time response (ms)	Intent Capture
500	80.2	98.3%
1000	114	99.1%
1500	143.6	98.8%
2000	153.6	98.5%
4000	241.2	98.7%

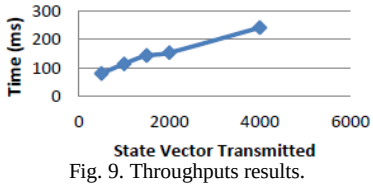


Fig. 9. Throughputs results.

The Fig. 9 shows the throughput (ms) plotted as a function of total state vector transmitted; we can see clearly after looking to the graph (scales roughly linearly) the high throughput and the scalability given by CEP engine (real time processing in intent capturing), after used a recording of different quantity raw input and configuring 3 rules (each rule for each intent). And we see also from the result (Table II) most of intent is capturing (98.5 % average) in our experiments, the accuracy is depending how we define our rules and patterns.

VI. Conclusions

We introduce the established principles and methods of

complex event processing into the domain of interactive systems especially for intent understanding. The architecture presented in this paper tailored to bring several benefits over the currently used approaches in this field. It allows for a more extensible and scalable system and increase the throughputs, which is especially important to those systems, as more and more different input modalities are added to these systems.

ACKNOWLEDGMENTS

This paper is funded by the National Natural Science Foundation of China (Grant No.61370135), and partially supported by Beijing Key Discipline Program.

REFERENCES

[1] Y. Bellik and D. Teil, "Definitions Terminologiques pour la Communication Multimodale," presented at Interface Homme-Machine (IHM), Paris, Nov. 30-Dec. 2, 1992.
[2] L. Ligay and J. Coutaz. Multifeature systems: The CARE properties and their impact on software design. [Online]. Available: http://iihm.imag.fr/publs/1995/immichapnigay95.pdf
[3] S. Andrew, K. Dinesh, and K. Pai, "Musculotendon simulation for hand animation," *ACM Transactions on Graphics*, vol. 27, no. 3, August 2008.
[4] M. Weber, G. Heumer, H. B. Amor, and B. Jung, "An animation system for imitation of object grasping in virtual reality," *ICAT 2006*, pp. 65-76, 2006.
[5] H. G. Wan, S. M. Gao, and Q. S. Peng, "Virtual grasping for virtual assembly tasks," presented at the Third International Conference on Image and Graphics (ICIG'04), Hong Kong, China, Dec. 18-20, 2004.
[6] B. F. Yi, C. Frederick, J. Harris, L. Wang, and Y. S. Yan, "Real-time natural hand gestures," *Computing in Science & Engineering*, May/June 2005, pp. 92-97.
[7] Z. W Zhu and Q. Ji, "Eye gaze tracking under natural head movements," presented at 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, June 20-25, 2005.
[8] M. Kumar, J. Klingner, R. Puranik, T. Winograd, and A. Paepcke, "Improving the Accuracy of Gaze Input for Interaction," in *Proc. the Eye Tracking Research and Application Symposium*, 2008, pp. 65-68.
[9] Digital image. [Online]. Available: http://dev.theeyetribe.com/general/
[10] K. Broda, K. Clark, R. Miller, and A. Russo, "SAGE: A logical agent-based environment monitoring and control system," in *Proc. European Conference on Ambient Intelligence*, Springer-Verlag, 2009, pp. 112-117.
[11] User's manual Logitech Edited 01/2001 by LogiCad3D.
[12] S. Møller, N. Poul, M. Migliavacca, and P. Pietzuch, "Distributed complex event processing with query rewriting," in *Proc. ACM International Conference on Distributed Event-Based Systems*, 2009, pp. 1-12.
[13] A. J. Demers, J. Gehrke, M. Hong, M. Riedewald, and W. M. White, "Towards expressive publish/subscribe systems," in *Proc. 10th International Conference on Extending Database Technology, 2006*, pp. 627-644.
[14] F. S. Wang and P. Liu, "Temporal management of RFID data," in *Proc. International Conference on Very Large Data Bases VLDB Endowment*, 2005, pp. 1128-1139.
[15] A. Hinze, K. Sachs, and A. Buchmann, "Event-based applications and enabling technologies," in *Proc. the Third ACM International Conference on Distributed Event-Based Systems*, 2009, pp. 1-15.
[16] D. C. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*, Addison-Wesley, 2002.
[17] D. Luckham, *The Power of Events*, Addison-Wesley Professional, May 8, 2002.
[18] Event Processing Technical Society (EPTS). [Online]. Available: http://www.ep-ts.com
[19] S. Lehmann, J. Schäfer, R. Dörner, and U. Schwanecke, "Towards integration of user interaction and context event processing in intelligent living environments," in *Proc. ARCS Workshops*, vol. 200 of LNI, 2012, pp. 111-122.
[20] G. Bedny, W. Karwowski, and M. Bedny, "The principle of unity cognition and behavior: Implications of activity theory for the study of human work," *International Journal of Cognitive Ergonomics*, vol. 5, no. 4, pp. 401-420, 2001.

Sebai Mounir was born in Batna, Algeria on August 27, 1988. He received engineering of computer science in polytechnic school, Algeria in 2013, and M.S. in 2016 in computer science and technologie in Beijing institute of technology, china. His research interest is human computer interaction.

Cheng Cheng was born in China in 1966. He is Associate professor, master tutor of Beijing institute of technology. His research interests include virtual assembly and virtual manufacturing technology, Human-computer Interaction and virtual environments.