A hardware and software architecture to deal with multimodal and collaborative interactions in multiuser virtual reality environments

Article *in* Proceedings of SPIE - The International Society for Optical Engineering · January 2014

DOI: 10.1117/12.2042499

A hardware and software architecture to deal with multimodal and collaborative interactions in multiuser virtual reality environments

Martin, P., Tseu, A., Férey, N., Touraine, D. and Bourdot, P.

CNRS/LIMSI, VENISE group
Université Paris Sud, Orsay, France

ABSTRACT

Most advanced immersive devices provide collaborative environment within several users have their distinct head-tracked stereoscopic point of view. Combining with common used interactive features such as voice and gesture recognition, 3D mouse, haptic feedback, and spatialized audio rendering, these environments should faithfully reproduce a real context. However, even if many studies have been carried out on multimodal systems, we are far to definitively solve the issue of multimodal fusion, which consists in merging multimodal events coming from users and devices, into interpretable commands performed by the application. Multimodality and collaboration was often studied separately, despite of the fact that these two aspects share interesting similarities. We discuss how we address this problem, thought the design and implementation of a supervisor that is able to deal with both multimodal fusion and collaborative aspects. The aim of this supervisor is to ensure the merge of user's input from virtual reality devices in order to control immersive multi-user applications. We deal with this problem according to a practical point of view, because the main requirements of this supervisor was defined according to a industrial task proposed by our automotive partner, that as to be performed with multimodal and collaborative interactions in a co-located multi-user environment. In this task, two co-located workers of a virtual assembly chain has to cooperate to insert a seat into the bodywork of a car, using haptic devices to feel collision and to manipulate objects, combining speech recognition and two hands gesture recognition as multimodal instructions. Besides the architectural aspect of this supervisor, we described how we ensure the modularity of our solution that could apply on different virtual reality platforms, interactive contexts and virtual contents. A virtual context observer included in this supervisor in was especially designed to be independent to the content of the virtual scene of targeted application, and is use to report high-level interactive and collaborative events. This context observer allows the supervisor to merge these interactive and collaborative events, but is also used to deal with new issues coming from our observation of two co-located users in an immersive device performing this assembly task. We highlight the fact that when speech recognition features are provided to the two users, it is required to automatically detect according to the interactive context, whether the vocal instructions must be translated into commands that have to be performed by the machine, or whether they take a part of the natural communication necessary for collaboration. Information coming from this context observer that indicates a user is looking at its collaborator, is important to detect if the user is talking to its partner. Moreover, as the users are physically co-localised and head-tracking is used to provide high fidelity stereoscopic rendering, and natural walking navigation in the virtual scene, we have to deals with collision and screen occlusion between the co-located users in the physical work space. Working area and focus of each user, computed and reported by the context observer is necessary to prevent or avoid these situations.

Keywords: Multimodal Fusion, Collaborative and Multimodal interaction, Immersive and Multiuser Virtual environments

Further author information: (Send correspondence to Bourdot. P)
Bourdot. P: E-mail:patrick.bourdot@limsi.fr, Telephone: +33 (0)1 69 85 81 75

1. INTRODUCTION

Recent years have witnessed the development of new complex information systems dedicated to a growing array of activities such as medical, genomics, etc. These developments were made possible by a number of progress in computer technology itself (speed, storage, bandwith), but also greatly benefited from improvements in signal processing and communication sciences. Efforts are still needed to study the use of advanced interfaces (namely Virtual Environments) for complex applications, involving the interpretation of massive and intricate databases. Virtual Reality (VR) systems cannot be promoted for such applications without creating natural and "transparent" user interfaces. While VR-experts are often needed for applications using VR technologies, intuitive interfaces could bring non-expert users to use them. Natural communication in high-tech information system exploits human modalities that have to be simultaneously perceived and interpreted, hence the need for multimodal interfaces in Virtual Environments (VEs). More and more often groups of users, as opposed to a single expert, are involved in collaborative tasks (product design, data exploration, etc). Many studies focus on multimodal or collaborative interactions in Virtual Environments, but to our knowledge none covers these topics together. This is what motivated the present work.

2. RELATED WORK

2.1 Multimodal VR interactions

Since Bolt[1] with his *"Put that there"* paradigm, multimodality has been a growing field of Computer Science. Few studies focusing on specific VR systems were conducted. But before focusing on these works, it is important to recall the main concepts that our approach is addressing. Oviatt et al.[2] proposed a classification of architectures for multimodal interactions: *early-fusion* and *late-fusion*. The first one integrates signals at the *feature* level, which is generally appropriate for closely coupled and synchronized modalities. The second one integrates information at a *semantic* level, which allows direct integration of existing recognition techniques and easy extension. From another point of view, Martin[3] classified multimodal interactions in five categories: *transfer* (pieces of information produced by a modality are used by another modality), *equivalence* (pieces of information may be processed as an alternative, by either of modalities), *specialization* (a specific kind of information is always processed by the same modality), *redundancy* (the same pieces of information are processed by several modalities) and *complementarity* (different pieces of information are processed by each modality but have to be merged to form an actual command). Consequently, any truly generic system should be able to handle all five categories. The challenge of a multimodal interface is to analyze and interpret multiple communication channels in real-time. Moreover, in order to combine the many heterogeneous modalities available in a VR system, one must rely on a generic and versatile framework. While most of works focus on multimodal equivalence, our system wants to address also redundancy and complementarity. Ladry et al.[4] and Mendonça et al.[5] focused on the requirements for the design and implementation of fusion engines, while Dumas et al.[6] and Lalanne et al.[7] presented surveys on multimodal interfaces.

Multimodal VR systems integrating continuous signals such as gestures and tracking produce asynchronous events and require passive co-reference[8] — while active co-reference combines two deliberate input events, passive co-reference cannot properly interpret an input event without knowing the state of another device. There is a lack of a generic approach that would achieve multimodal fusion with asynchronous events or passive co-reference and that is one of the problems that we are addressing. Various systems are relevant for VR interactions but some are application dependent and others are platform dependent. Flippo et al.[9] proposed a framework, managing two stages of multimodal fusion. The first one aims at interpreting the events coming from the devices, while the second one merges the resulting interpretations in a single command. Sun et al.[10] presented an approach to merge multi-sensory data, using a unification-based engine (MUMIF) and Krahnstoever et al.[11] presented a framework managing audio and visual cue information (speech, natural gestures and face detection), to be used in large screen displays. However data fusion is only invoked after a speech event, which reduces possibilities of interactions. Kaiser et al.[12] introduced a multimodal interaction architecture, finding the highest scoring multimodal interpretation, given a set of events among natural language, gaze direction and four different gestures. Pfeiffer and Latoschik[13] and Latoschik[14] proposed multimodal key elements, such as semantic environment representation and semantic interaction description. This system can rout continuous data streams asynchronously, but there is no indication for a reconfigurability or multi-user feature. These works are comparable to ours but none deals with the collaborative aspect.

2.2 Collaborative VR interactions

When we talk about CVE, several aspects may be considered: display (VR system), interaction (rules, type) and user study. Our work mainly focus on the display and interaction aspects. Benford et al.[15] introduced a spatial model of interaction including concepts such as *aura*,*focus*, *nimbus*, *adapters* and *boundaries*, to manage natural social communication skills in shared VE. Broll[16] reviewed existing models of distributed VE and proposed a method to manage concurrent interactions in such environments. Collaborative interactions can be classified in three levels (Margery et al.[17]):

- Basic cooperation (level 1) : users can perceive each other (via avatars) and can communicate
- Parallel manipulations (level 2) : users can act on the scene individually. This level is divided in two subdivisions, one for constrained manipulation (i.e. by scene design) and the other for free manipulations
- Cooperative manipulation (level 3) : users can cooperate on objects. A division is also made between independent operations on objects (same object but different properties) and competitive operations (co-dependent manipulations)

To allow the highest cooperation level (level 3), one must define rules to manage inputs from the users.[18] Some works were made in this direction, leading to several frameworks[17] and.[19]

With the dawn of CVE displays and interaction frameworks, studies have been conducted to evaluate users in collaborative tasks (e.g. in terms of presence and performance). Schroeder et al.[20] found that asymmetry between users of different systems affects collaboration. Moreover, Steed et al.[21] showed that a differential in level of immersion between distant users leads to the more immersed user being the dominant one. Fussel et al.[22] argued that a shared visual space and a physical (or virtual) co-presence are essential for collaborative tasks. Immersive aspect were covered by Narayan et al.[23] who showed that task performance is improved with a stereoscopic visualization. In parallel with studying collaborative interactions, work on shared displays has been carried out. One of the key issues of CVE is to immerse groups of users instead of single ones. Accordingly multi-user immersive systems have been created and developed, for instance, by Agrawala et al.,[24] Arthur et al.[25] and Blom et al.[26] Fröhlich et al.[27] developed a multi-viewer stereo display using shutter projectors and polarization for local collaboration of two users. They evaluated three different configurations and concluded that active separation (shutter) of users and passive separation (polarization) of left and right eyes is the most promising approach. All these systems allow two-users or more — up to six at our knowledge — to be physically and virtually co-located.

Evaluations of multi-user collaboration scenarios in immersive systems have just begun. Dang et al.[28] presented a framework using Virtools*— CVRNR† — which supports distant collaboration between immersive CVE (with MultiUser Pack) and evaluations of each level of the platform. It allows users to manipulate objects in a collaborative (simultaneous) way. On the other hand, Olson et al.[29] and Teasley et al.[30] pointed out that co-located teamworks are more productive and more efficient. Salzmann et al.[31] evaluated two-user co-located interactions in immersive systems with two scenarios, one using HMD for *face-to-face* interactions, the other using a multi-stereoscopic wall for *side-by-side* interactions.

2.3 Discussion

Several critical points of the multimodal process can be inferred from studies on multimodal interaction: *disambiguation* — comparing user's inputs —, *decision* — generating actions desired by a user — and *dialog* (or *decision management*) — identifying additional treatments to be applied on incoming actions. Previous multimodal approaches have been carried out on single user and non-cooperative scenarios. But now, the improvements of VR technology allow, for instance, cooperation of co-located users into a shared virtual scene. Therefore the multi-user context is a key point in the multimodal process. Such a system must be able to handle interactions from several users and thus one must know when data fusion may occur, but also what data can be merged. What happens if pieces of information coming from several users have to be combined ? At which level, the pieces of information have to be merged ? These new important questions about multi-user multimodal systems reveal that there are similarities between collaborative and multimodal processes. We should not restrict the notion of collaboration only to simultaneous activities, but we need a broader view including the notions of dialog and continuity. It is not solely a question of solving some constraints at a given time, but rather of an

*http://www.virtools.com/

†Collaborative VRNR. VRNR (Virtual Reality Normalized Resources) is provided with the VRPack

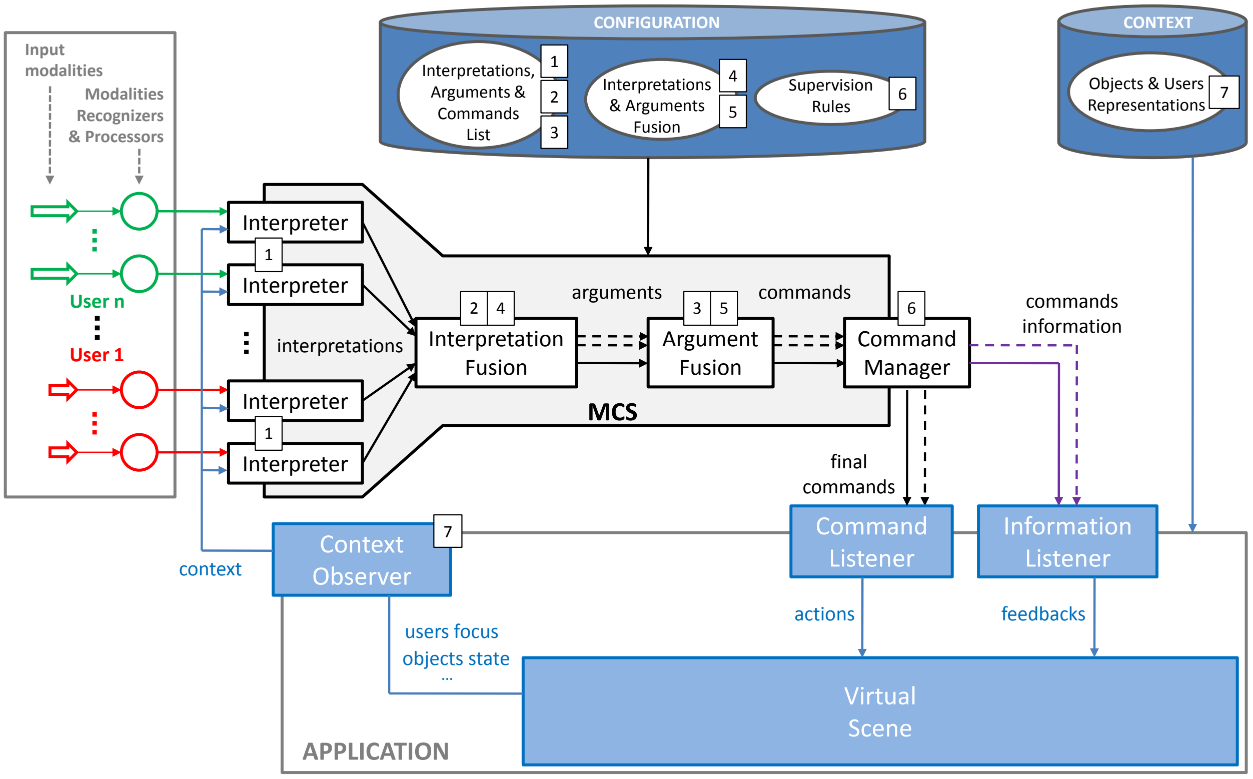

Figure 1. The framework's architecture. The Multimodal and Collaborative Supervisor (MCS) is configured and initialized using XML files. MCS processes users' inputs, taking into account rules and context, and transmits results to the application. Context of application is also configured and is constantly feeding the MCS.

ongoing management over time. Thus, to bring closer these two processes, we have chosen to extend the classification for collaborative interactions proposed by.[17] Actually, the previous classification of Margery et al. was mainly focusing on user's collaborative manipulation. Our extended classification is *task-based* oriented, which is more appropriate to a multimodal approach:

- Basic cooperation (level 1) : users can perceive each other and can communicate. This level is composed of two general situations:
  - (a) co-located interaction : users are immersed in the same place or in the same display. Depending on the technology used (full or limited cohabitation), they can have natural communication. For instance, in a multi-user CAVE with multi-stereoscopy visualization system, users have a full cohabitation. They can see or touch each other and can have natural dialog. In case of HMD systems, users have a limited cohabitation. They are able to have natural conversations, but they do not see others physically.
  - (b) remote interactions : users are immersed in a virtual world but are distant (virtual presence). Therefore, multimedia communication and avatars are required.
- Parallel tasks (level 2) : the notion of *task* includes manipulation of scene's objects (cf. Margery's classification) but also command aspect to control the application. Here, the users can act on the scene individually: a given task is performed by only one user. This level is divided in two subdivisions, (a) constrained tasks (by scene design) and (b) free tasks. Users' interactions are completely independent of each other, whatever the different processes.
- Cooperative tasks (level 3) : users can cooperate on the same object or within the same task. This level is divided in three sub-divisions:

- (a) independent tasks : users generate tasks with a similar target (but independent properties) which can be performed independently of each other.
- (b) synergistic task : users' interactions can be combined, at any level of treatment, to generate one single task. This concept includes the previous co-manipulations and is close to the complementarity concept of Martin.[3]
- (c) co-dependent task : users generate dependent tasks (i.e. similar task or competitive task on a same target). These co-dependent tasks can not be performed immediately because ambiguities have to be solved. With the "similar tasks" comes a concept of redundancy, also close to the one of Martin.[3]

It is obvious that level 1 of collaboration does not depend on the system used but rather on technology. But we have also noticed that level 2 and level 3 could be introduced in a multimodal process: merging separately events of each user ensures the level 2, while combining events of all users ensures level 3. This is one of the key points of our system.

3. OVERVIEW

The purpose of the present work is the design of a generic, device-independent multimodal system that could handle inputs typically found in VR setups: 3D tracking, speech recognition, gesture recognition, haptic device, flysticks, etc. Following Oviatt's classification,[2] our model performs *late-fusion* and treats inputs following *typed-feature-structure unification*. The general principle of this multimodal and collaborative supervisor (MCS) is based on three semantic augmentation stages (*interpreters*, *interpretation fusion* and *argument fusion*) and one supervision layer (*command manager*) as shown in Figure 1. This process transforms incoming events of different sensors into information understandable by the application. In other words, it builds abstract and sensor independent data structures from users inputs and generates meaningful representations in terms of actual commands. In order to ensure genericity as well as reusability, the MCS has been implemented in C++, using pluggable components for each processing step.

As previously remembered, a truly efficient system needs to address some cooperations between modalities (cf. section 2.1). Our architecture is designed to process *equivalence*, *redundancy* and *complementarity*. One of the main differences is the splitting in four stages whereas others multimodal systems generally have two stages. The first step intends to abstract users inputs from the application and transmit them — as device independent *interpretations* — to the multimodal process. The *interpretation fusion* and the *argument fusion* stages are respectively a disambiguation and a decision phase producing *arguments* and *commands*. The last stage — *command manager* — ensures the validity and robustness of the commands coming from the three previous steps before sending the final fusion results to the application in an appropriate format. Our first and last stages — *interpreters* and *command manager* — can be regarded as respectively "input interface" and "output interface" with the application. The other major difference is that our system deals with the collaborative aspect, which to our knowledge, makes it unique. Actually, this is possible because the multimodal and collaborative processes share similarities. Moreover, we will see below that these similarities are happening into specific stages, justifying the architecture of the MCS.

4. RECONFIGURABLE MULTIMODAL AND COLLABORATIVE SUPERVISOR

One could have chosen to implement a *state-machine-based* system to represent ongoing multimodal interaction. In fact, in some cases, the state of 3D objects may change depending on the interaction. However, interpretations of inputs may suffer inherent delays due to recognition processes, forcing the state-machine to manage an historic and to store events toggling objects status when needed. This is not a problem per se, but issues may arise when stored information is more complex, in terms of structure or time stamping. For instance, information depending typically on the tracking rate (e.g score computed to determine the "best" current object pointed) could produce a very large state machine and overload the process of objects state update. From a conceptual view, a "per object" state machine transfers the focus of processing multimodality on the objects. Our purpose is to stick to the user behaviour and goal, and this is why a data flow implementation was preferred. Each level can produce one or several results that can be stored in a time-based global event queue (see Fig. 2). Developer can trigger the change of state of objects inside the queue with the specification of interpreters.

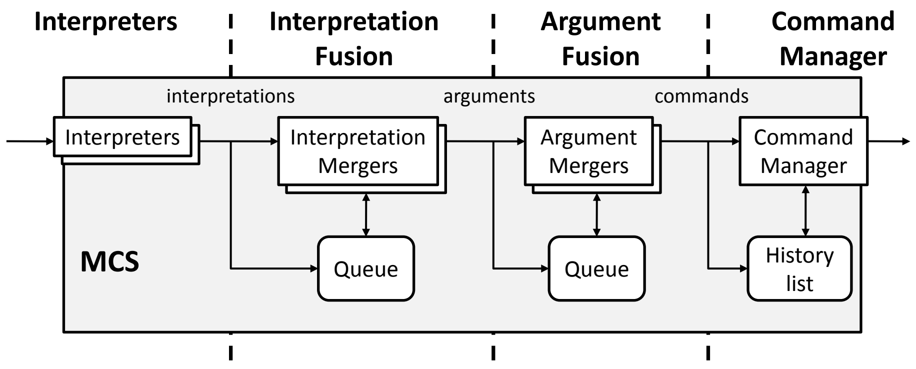

Figure 2. Three semantic augmentation stages (interpreters, interpretation fusion and argument fusion) and one supervision level (command manager).

4.1 Interpreters

A user immersed in a virtual world sends continuous or non-continuous events to the application, through VR peripherals (speech, gestures, flystick, etc.). By analyzing the various incoming signals, the semantic of these events can be enriched. The first stage of our MCS is an interface between real world (users) and application (virtual world). Thus, each interpreter, specific to a user and to a modality, will produce an *interpretation* — possible *interpretations* being configured *a priori* for each modality. In this way *equivalence* as well as *specialization* (cf. section 2.1) is performed. An interpreter can simultaneously generate several *interpretations*, depending on its inputs. For instance, when a user is pointing, the system may produce at least two *interpretations*: one is the designation of the axis defined by the pointing gesture (visible through a laserbeam), the other may be the designation of a potential object (intersected by the laserbeam). Each interpreter receives informations from a given modality and processes it adequately. *Interpretations* are characterized by a type — e.g "INT-INDICATE-ENT" for the indication of an entity — and a sub-type — e.g "AXIS" or "CUBE" depending on the type of the entity (see Fig. 3 and 4). The generated *interpretations* contain important book-keeping informations: start and end times of the event, the user ID — user from who the event is coming —, a timeout — maximum delay for being processed by next level — and a capsule of data associated to the event. As a matter of fact, we can say that *interpretations* describe the potential intentions of users. This stage is connected to a *context observer* (cf. section 4.5) to treat events according to all necessary information. These interpreters can evolve, depending on the desired scenario and they can be interfaced easily with the platform used, thanks to their genericity — structure and functions.

4.2 Fusion stages

Generated pieces of information have a limited lifespan. In fact they are placed in the related level's queue, during a specified timeout, to be processed. In the case of explicit and non ambiguous pieces of information, when no timeout is specified, they go directly to the following stage without staying in the related queue (see Fig. 2). Interpretations are handled by specific entities generating arguments (cf. section 4.2.1) and arguments are transformed, also by specific entities, into commands (cf. section 4.2.2). All these specific entities are described in the XML file, to determine which interpretations (respectively arguments) are expected to be merged and the type of the result.

The fusion is based on three main rules:

- *Type compatibility* : interpretations (respectively arguments) must have compatible types and sub-types.
- *Time compatibility* : interpretations (respectively arguments) must occur in a same time interval. Time stamping of events and synchronization of nodes are ensured by the implemented platform, which was developed in earlier work.
- *User compatibility* : by default, fusion occurs if pieces of information have the same user ID (events coming from the same user). This rule is flexible compared to the two others, as we may choose to merge any pieces of information (no user-filter) to obtain a "collaborative disambiguation" (interpretation fusion) or a "collaborative decision" (argument fusion).

4.2.1 Interpretation Fusion

To characterize ”concrete” intentions of users, it may be necessary to gather several interpretations (”potential” intentions). This is known as a ”disambiguation phase” which consists in solving *co-references* (cf. section 2.1) and is managed by specific entities called ”interpretation mergers”. In fact, there is an ”interpreter merger” for each specified combination of interpretations. This way, *complementarity* and *redundancy* are addressed in the multimodal fusion process (cf. section 2.1). Merging any interpretations (no user-filter) produces a ”collaborative disambiguation” (see Fig. 4). *Arguments* contain the same kind of information as interpretations do: start and end times, user ID, a timeout and the same data capsule of interpretations. But conversely to the interpreters stage, this interpretation fusion level is the first that makes a complete abstraction of the VR devices. *Arguments* represent the ”purpose” of users at a given time and act as a temporal sampling of users actions.

4.2.2 Argument Fusion

While interpretation fusion solve co-references, argument fusion achieves a process of completeness, known as a ”decision phase”. Arguments are subject to merging to build *unitary commands* (orders of users), following the fusion rules described above. This fusion is handled by entities called ”argument merger” and one exists for each defined combination of arguments. Merging any arguments (no user-filter) produces a ”collaborative decision”. The *commands* still include previous pieces of information transmitted by interpretations and arguments. In fact, *commands* are the actions dictated and truly determined by users at a given time, justifying the existence of this stage — while the previous stage can be seen as a preparation/clarification phase. Finally, the argument fusion stage manages also the level ”3(b)” of our classification (cf. section 2.3): combining pieces of information coming from several users is a synergistic process.

4.3 Command Manager

The last stage is a supervision level (or ”dialog phase”) which is one of the main feature of the MCS. It can be seen as an ”intelligent” level which ensures robustness and validity of users commands. This supervision layer finishes the complete management of collaborative tasks in a multimodal process. Then, several users can act in/on a shared virtual scene, while being physically and virtually together. Although users’ events are treated separately in the first level, all generated pieces of information can be merged (multi-user process) in the previous fusion stages: this is a collaborative process. More precisely, this supervision layer has been designed to address two main situations:

- Group of commands : a group of simultaneous commands may be received by this stage. History and context will define in which order commands have to be processed, resulting in a group of *ordered commands*.
- Collaborative commands : several commands can be generated in a same small time interval. Depending on their type, they can be treated as *collaborative commands*. In that case, the MCS has to check in the history or in the queue (see Fig. 2) if similar commands co-exist. If commands are found, an adapted treatment has to be applied to transform these ”single commands” in a global *collaborative command* (see Fig. 3).

In fact, this stage manages the levels ”3(a)” and ”3(c)” of our classification (cf. section 2.3): independent commands will be scheduled and co-dependent commands will be either scheduled if they are competitive, or processed as collaborative ones if they are not competitive. The two situations described above are not independent: there might be a ”group” of ”collaborative commands”. To handle these situations, we use the currently algorithm: when pieces of information are entering into the supervisor stage, they are added in a temporary queue and a timeout is triggered. If other pieces of informations arrive before the end of the timeout, they are also added in the queue and the timeout is relaunched. When the timeout ends, the queue is processed. Obviously the timeout becomes a truly critical point for this stage: outgoing final commands of the MCS are the actions that users really want to do at a given time and ideally, they should not wait for the system to respond to their demands. Thus, the timeout has to be set long enough so that events can be sequenced. On the other hand, users should not wait too long and so a compromise has to be found between processing time and latency in the definition of timeouts.

Once we have collected the ”simultaneous” commands, we have to process them. Two different kinds of management have to be done: scheduling grouped commands and modifying collaborative ones. The commands can indeed be organized following their priority (cf. section 4.4). If we deal with collaborative commands, then we may have to modify them. Implemented with genericity in mind, the supervisor layer of the MCS contains generic processes to modify commands:

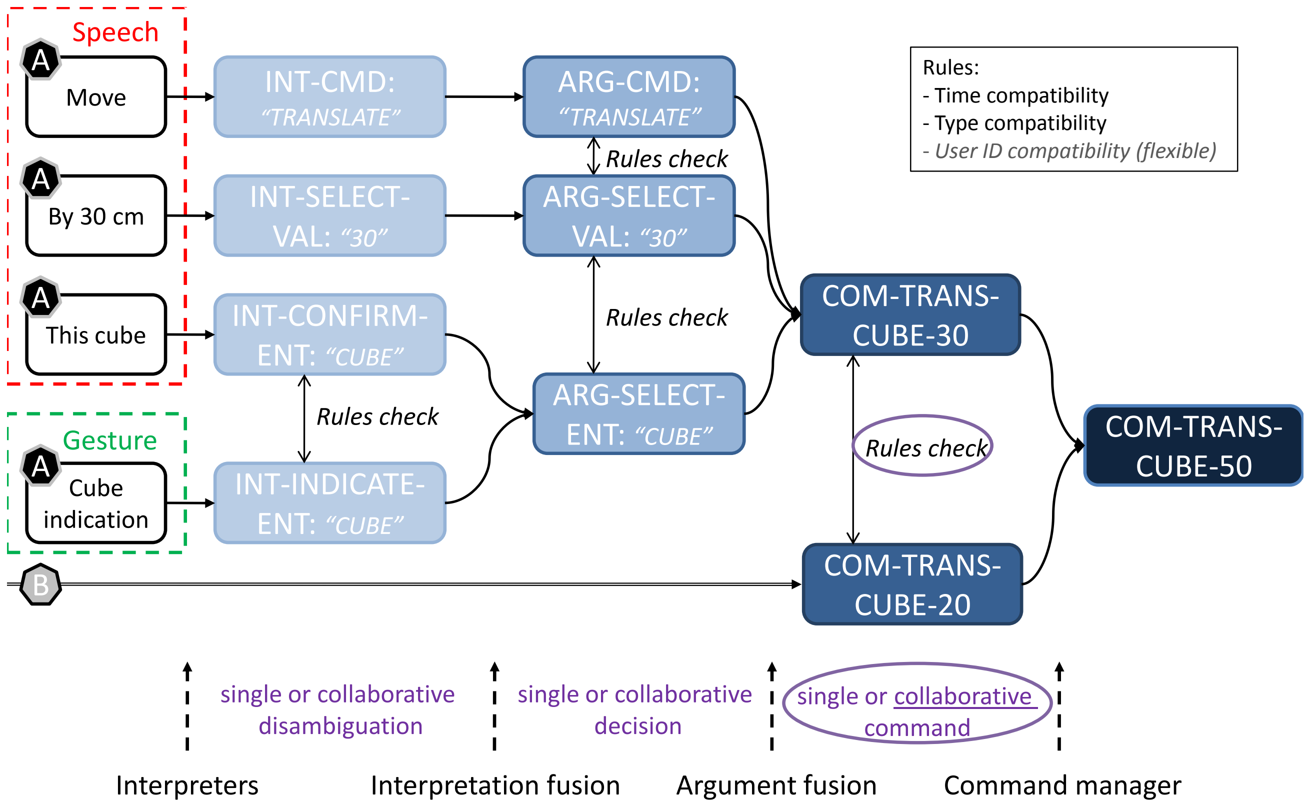

Figure 3. Example of a multimodal/collaborative command. Rules check are defined respectively in 4.2.1, 4.2.2 and 4.3.

choose among several commands, *replace* a command by another one, *combine* several commands and *none*. Obviously it is possible to add more processes.

4.4 Configuration

Versatility is insured by an ASCII configuration file used to describe multimodal interactions (see Listing 1). The XML format provides an appropriate and flexible way to clearly represent interaction. Thanks to its availability on common operating systems and platforms, XML is useful to ensure the interoperability of our system. Since we dont want neither to distribute the multimodal and collaborative process – potential significant network and interprocess communication consumption – nor to parse unnecessary XML, we chose to transmit the pieces of information between stages by using binary structures. Then the configuration file is divided in seven parts: *interpretation*, *argument*, *command*, *interpreter*, *interpretation_merger*, *argument_merger* and *command_manager*.

As previously explained, interpretations, arguments and commands have the same structure: they are defined by a type and a subtype (see lines 3-16 of Listing 1 for an example of interpretation). The types of interpretations are the potential intentions that the user may have in the desired scenario, while the subtypes explicits these intentions. In a same way, the types of argument are the real intentions a user may have and the types of commands are the action a user can do (lines 17-20 and 21-23). The interpreter part defines which interpretations can be generated by the inputs — gesture and speech recognition, ontology and haptics (lines 24-31). This part will be explicited in sections 5.2 and 5.3 by showing examples of configuration and evolutions from one scenario to another. The interpretation_merger part (lines 32-37) characterizes the transformations of interpretations into arguments. Each "intm" structure owns a set of waited interpretations (one or more), a *creator*, which is the kind of treatment to be applied, and a resulting *argument*. For single interpretation, a "direct" creator is called: no tests and no treatments are made and the corresponding result is immediately generated. On the contrary, a "generic" creator is called when several interpretations are merged (cf. section 4.2.1). It is exactly the same scheme for the argument_merger part (lines 38-44), with "argm" structures — a set of waited arguments, a creator and a resulting *command*. Lastly, the command_manager part (lines 46-60) defines commands which have to be validated (cf. section 4.3).

Listing 1. Configuration File of MCS

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<MCS>
  <interpretation><!--Just few examples, more in full version-->
    <int name="INT-INDICATE-ENT">
      <sub name="CUBE"/><sub name="AXIS"/><!--Other sub-interpretations-->
    </int>
    <int name="INT-CONFIRM-ENT">
      <sub name="CUBE"/><sub name="AXIS"/><!--...-->
    </int>
    <int name="INT-SELECT-VAL">
      <sub name="DISTANCE"/><sub name="ANGLE"/>
    </int>
    <int name="INT-CMD">
      <sub name="TRANSLATION"/><sub name="ROTATION"/><!--...-->
    </int>
  </interpretation>
  <argument><!--Just two examples, more in full version-->
    <arg name="ARG-SELECT-ENT"/>
    <arg name="ARG-CMD"/><!--List of sub-arguments-->
  </argument>
  <command>
    <cmd name="COM-TRANS"/><!--...-->
  </command>
  <interpreter>
    <inter name="GESTURE RECO" creator="gesture">
        <!--List of interpretations-->
    </inter>
    <inter name="SPEECH RECO" creator="speech"><!--...--></inter>
    <inter name="ONTOLOGY" creator="ontology"><!--...--></inter>
    <inter name="HAPTICS" creator="haptics"><!--...--></inter>
  </interpreter>
  <interpretation_merger> <!--Just few examples, more in full version-->
    <intm name="SELECT-ENTITY" creator="generic" argument="ARG-SELECT-ENT:A">
      <int>INT-INDICATE-ENT:A</int>
      <int>INT-CONFIRM-ENT:A</int>
    </intm>
  </interpretation_merger>
  <argument_merger><!--Just few examples, more in full version-->
    <argm name="TRANSLATE-ENTITY" creator="generic" command="COM-TRANS">
      <arg>ARG-CMD:TRANSLATE</arg>
      <arg>ARG-SELECT-ENT:A</arg>
      <arg>ARG-SELECT-VAL:B</arg>
    </argm>
  </argument_merger>
  <command_manager>
    <coms name="SUPERVISOR" creator="generic" timeout="1.5" manager="decode" history="25">
      <cmds priority="0" dialog="choose">
        <cmd>COM-CMD:SELECT</cmd>
      </cmds>
      <cmds priority="1" dialog="combine">
        <command>COM-CMD:TRANS</command>
      </cmds>
      <cmds priority="-1" dialog="replace">
        <command>COM-CMD:RECTIF</command>
      </cmds>
      <cmds priority="-2" dialog="none" direct="true">
        <command>COM-UNITCMD</command>
      </cmds>
    </coms>
  </command_manager>
</MCS>
```

There is only one ”coms” structure which owns a set of properties: a creator, a timeout (in seconds), the length of the history and a manager to process the MCS’s outputs. The ”coms” structure contains groups of commands identified by ”priorities”: 0 (higher) to N for all groups — except -2 for direct commands and -1 for rectification commands. These priorities are used to order commands. Finally, for each group a ”dialog” is specified which is in fact the process applied on the commands.

4.5 External modules

Aside from the main structures described above, there are three additional external modules acting as intermediaries between the MCS and the external 3D application (see Fig. 1):the *command listener*, *information listener* and *context observer*.

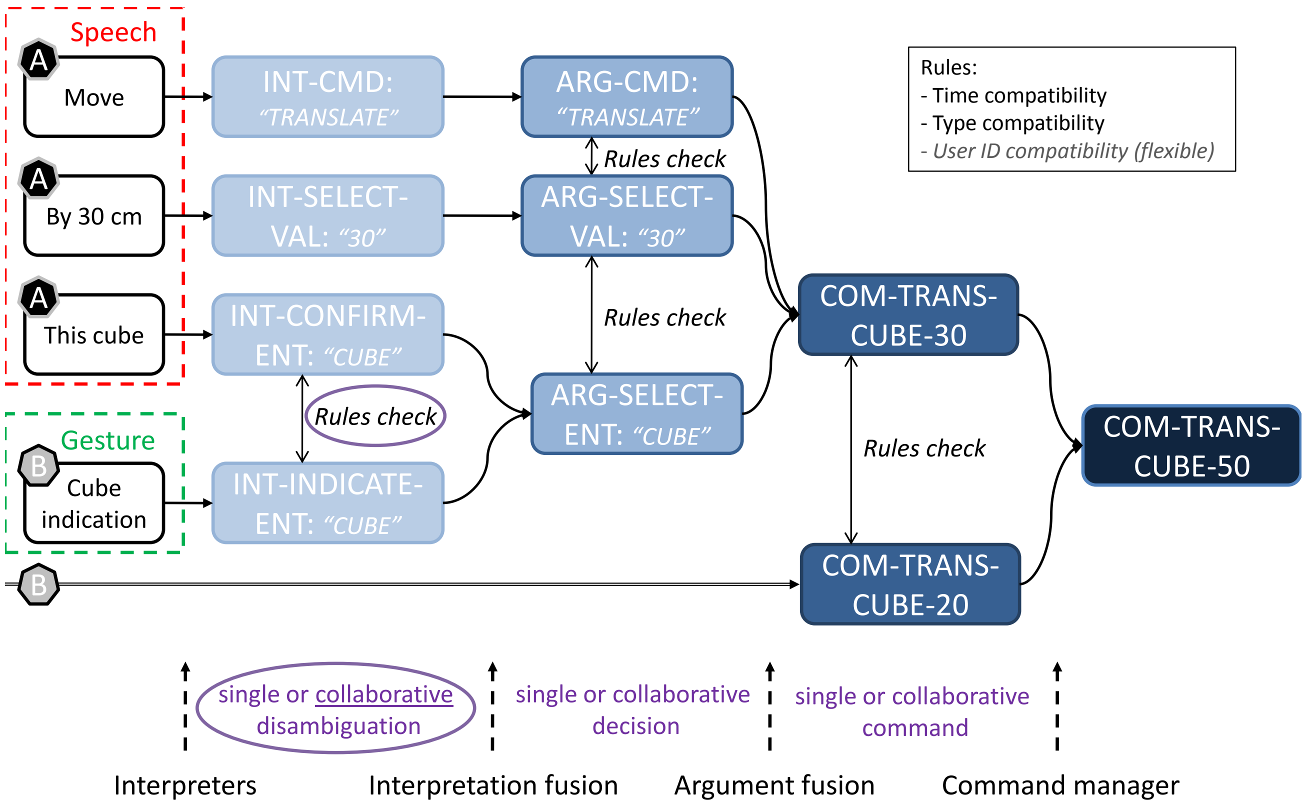

Figure 4. Example of a multimodal/collaborative command: a collaborative disambiguation (passive co-reference).

4.5.1 Information and Command listeners

The *command listener* is a part of the ”output interface”. It receives all the commands generated by the MCS in an adapted structure — e.g. an array — and will trigger the corresponding actions inside the application. The *information listener* is also a part of the ”output interface”. The MCS transmits informations to the application — if a command is aborted or generated, additional feedbacks, etc. — that the user needs to be informed of, so as to stay focused on their tasks.

4.5.2 Context observer

The *context observer* provides contextual events needed by the MCS. The main goal of the *context observer* is to translate according to the raw data coming from the tracking and devices used in the virtual environment according and the virtual scene content, to high level events, The are two kinds of event generated by the *context observer*. On the one hand, several events are related to the behavior of several users in the real space into the immersive device, that are espcecially important in collaborative scenarii. It allows the MCS to know when a user looks at his collaborator to disable for example the voice command interpretation during the natural communication between the two subjects. Moreover, as the users has their own head-tracked stereoscopic point of view, it is also important to prevent collisions between the users and/or between the users and screens, promoted by the physical co-localisation of the users in the immersive environment. On the other hand, several events are related to the interaction between the users and the virtual scene. The context observer is able to provide to the MCS for each user which virtual objects are in his focus, or which object is pointed. To implement this feature, as the *context observer* was designed to be independent from the MCS and the application, an XML configuration file explicits the contents of the virtual scene managed by the targeted application, by defining the properties of its 3D objects (position, bounding box, ...), as well as the informations concerning the users and their associated tracking and interaction devices in the virtual environment. The application must also be adapted to send to the *context observer* the updated positions and orientations of the dynamic virtual objects, in order to compute the interaction events for each users. Thus the *context observer* feeds the first stage of MCS with all the necessary contextual informations used in the interpretation stage (see Fig. 1).

5. SCENARIOS

The reconfigurability feature of the proposed supervisor allows it to be easily applied in different contexts along with different kinds of inputs and outputs. Here we detail two applications: a single-user assembly task dedicated to cognitive evaluations, and a collaborative assembly task for the automotive industry. In each scenario, the XML configuration was modified in order to handle the necessary interactions while interfaces of the MCS (interpreters and command manager) also present some notable differences of implementation. These applications are developed into Virtools — a commercial VR software based on visual programming.

5.1 Inputs and Outputs

Inputs and outputs mainly depend on the supporting VR immersive system but also on the application needs. In our case, at least three input modalities — speech recognition, gesture recognition and haptics — and two output modalities — haptics and visualization systems — are proposed.

5.1.1 Gesture recognition

The generic gesture recognition used in our applications was developed within our team. This kind of systems consist in a preliminary training phase followed by a statistical recognition process. Currently, the system recognizes nineteen mono-manual or bi-manual gestures. The hardware part of this gesture recognition is based on the ART Fingertracking [‡] two-hand system.

5.1.2 Speech recognition

Our applications use two different systems. The first one is a speech recognition system provided by *Vecsys*[§]. The vocabulary of this system has been configured in a previous collaboration and is dedicated to the automotive context. It was designed from the beginning as a multi-speaker system and allows natural language. An additional process, *Ontology*, analyzes the speech results and transforms them into generic sentences. The other one, the *Microsoft Speech API*, is characterized as a mono-speaker system and requires explicit user training. Nevertheless, in most cases, several different users can use the same profile without much performance loss.

5.1.3 Haptics

Haptic devices — in our case the Virtuose 6DoF haptic arm by Haption[¶] — allow direct and simple manipulation of objects. Advantages of these devices are their usability — they are found easy to manipulate by a large number of people — and their input/output features. Inputs are the users' movements/gestures with the device and outputs are the different states in which the devices can be, giving to users alternate force feedbacks sensations and information — collision detection, haptic guides, and so on.

5.1.4 Visual

Visual outputs are important ones because they largely ensure immersion. Our two applications use two extremely different visualization systems. In application 1, a *standard visualization* is exploited. This visualization is a monocular projection on a wall (see Fig. 5). The dimensions of this screen — 2m x 2m — were enough to immerse the users, and stereoscopy, although possible, was not deemed necessary. The application 2 uses a *double stereoscopy visualization system* of BARCO[‖] — active separation of users and passive stereoscopy. With the help of appropriate customized software components, two users (or two groups of users) are fully immersed in a CAVE-like set-up, each with his own adaptive stereoscopic view (see Fig. 6). This feature allows us to explore new situations such as the collaborative one described below.

[‡]http://www.ar-tracking.com/

[§]http://www.vecsys.com

[¶]http://www.haption.com

[‖]http:://www.barco.com

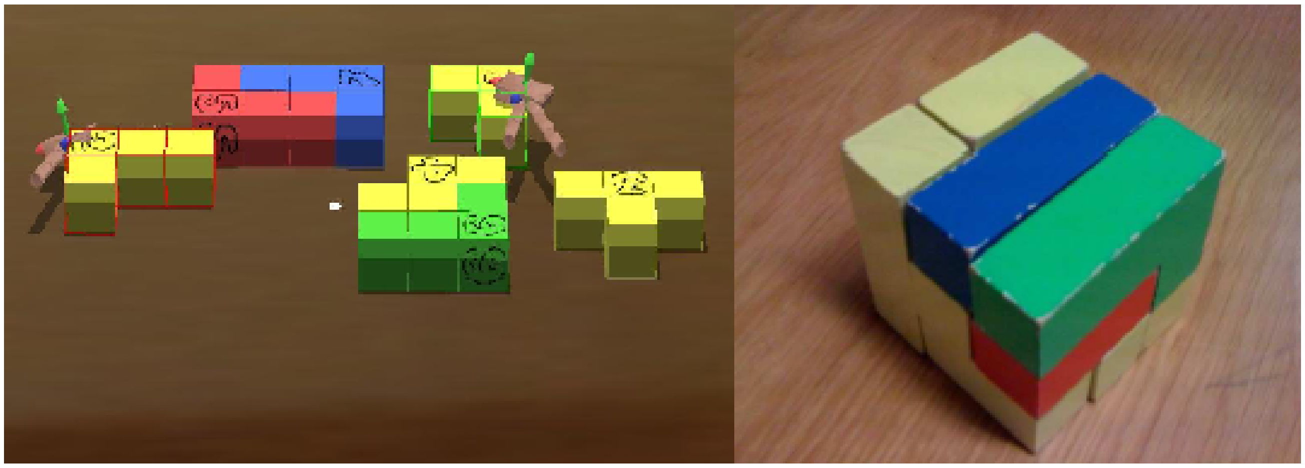
Figure 5. MVPuzzle application (virtual world / real-world)

5.2 Scenario 1 : Single-user assembly task

This first application was motivated by the need of cognitive ergonomists to take advantage of VR systems to study strategies of users confronted to a complex task — a three-dimensional (3D) puzzle. Several evaluations were carried out with a real version of the puzzle, and the new possibilities offered by the multimodal supervisor made it possible to set up experiments with a virtual version called MVPuzzle, for multimodal and virtual puzzle. These experimentations used our generic gesture recognition system, the Microsoft Speech API for command language (e.g ”move left” or ”select cube one”) and a standard visualization on a large wall. These inputs — speech and gesture — are then configured in the XML file.

Listing 2. Configuration of Intepreters for Scenario 1

```
 1  <?xml version="1.0" encoding="ISO-8859-1"?>
 2  <interpreter>
 3    <inter name="GESTURE RECO" creator="gesture">
 4      <gesture int="INT-INDICATE-ENT:CUBE">ind cube one</gesture>
 5      <gesture int="INT-INDICATE-ENT:AXIS">ind axis</gesture>
 6      <gesture int="INT-INDICATE-VAL:DISTANCE">ind distance</gesture>
 7    </inter>
 8    <inter name="SPEECH RECO" creator="speech">
 9      <word int="INT-CMD:TRANSLATION">translate</word>
10      <word int="INT-CMD:SELECT">select</word>
11      <word int="INT-SELECT-ENT:CUBE" object="Cube1">Cube one</word>
12      <word int="INT-CONFIRM-ENT:CUBE">this cube</word>
13      <word int="INT-SELECT-ENT:AXIS" vector="1.0,0.0,0.0">X</word>
14      <!-- ... Y & Z ... -->
15      <word int="INT-CONFIRM-ENT:AXIS">this axis</word>
16      <word int="INT-CONFIRM-VAL:DISTANCE">this distance</word>
17    </inter>
18    <inter name="ONTOLOGY" creator="ontology"><!--not used--></inter>
19    <inter name="HAPTICS" creator="haptics"><!--not used--></inter>
20  </interpreter>
```

The gesture interpreter (see lines 3-8 of Listing 2) can in particular generate interpretations of 3D indication (pointing a cube or an axis, etc.) and commands (catching or releasing cubes, etc.). To ensure a more natural speech with the Microsoft Speech API, the speech interpreter (lines 9-20) must include an exhaustive list of interpretations: only the combinations of words written in this interpreter will be processed and this is why there are no spoken number in this scenario. This puzzle was made of seven blocks of various shapes which have to be assembled to form a 3 x 3 x 3 cube (see Fig. 5). We wanted to study the strategies of users trying to solve the problem, depending on the modality used — speech only, gestures only, or both. Ergonomists were interested by user strategies — in which order blocks are manipulated, what is the affordance of blocks, time to complete the task, relationships between tests of mental rotation and success in completing this puzzle, etc. — while we wanted to study interactive aspects — how subjects make use of the modalities, overall performance of the system, and so on. Analysis of experimentations results — statistics and videos — are still in progress and will be reported in a subsequent article.

5.3 Scenario 2 : Two-user collaborative task

We are conducting research on VR interaction with industrial partners in order to evaluate the potential use of the proposed approach in a ”Product Lifecycle Management (PLM)” context (automotive field). We applied our supervisor to a collaborative situation, entitled *MalCoMIICs*, for multimodal and co-localized multi-user interactions for immersive collaborations. Specific developments of the Virtools VRPack allow us to handle a double stereoscopy visualization. Moreover, this application uses our generic gesture recognition system, the Vecsys speech recognition system and a haptic device coupled with IPP[**]. Then the XML configuration file has to evolve to take into account the desired interactions and the new modalities (see Listing 3). The gesture interpreter (see lines 4-9 of Listing 3) can now generate the appropriate interpretations of 3D indication (e.g. pointing a seat). The speech interpreter has been replaced by the ontology interpreter (lines 11-33).

Thanks to ontology, we can now process spoken numbers to specify distances or angles. Moreover, an other new interpreter appears: the haptics interpreter (lines 34-42). It can generate interpretations such as commands (e.g. catching or releasing seats), indications and selections of objects. These evolutions of interactions induce changes in several sections of the XML file: new interpretations and their linked arguments, new combinations for fusion stages, and finally new treatments of commands. However, modifying the XML file is easy and absolutely not time consuming. Within a virtual assembly chain, two users cooperate to define the insertion trajectory of a seat into the shell of a car. For instance, user A selects a seat and manipulates it by using vocal commands combined with the haptic device to analyze the possible trajectory paths. At the same time user B is constraining the task of user A by defining virtual guides (i.e. axes and/or planes) with vocal inputs and two hands gesture commands. We are currently setting up experimentations based on this scenario, with the help of automotive design engineers.

Listing 3. Configuration of Interpreters for Scenario 2

```
 1 <?xml version="1.0" encoding="ISO-8859-1"?>
 2 <interpreter>
 3   <inter name="GESTURE RECO" creator="gesture">
 4     <gesture int="INT-INDICATE-ENT:SEAT">ind seat 1/3</gesture>
 5     <gesture int="INT-INDICATE-ENT:SEAT">ind seat 2/3</gesture>
 6     <gesture int="INT-INDICATE-ENT:AXIS">ind axis</gesture>
 7     <gesture int="INT-INDICATE-VAL:DISTANCE">ind distance</gesture>
 8   </inter>
 9   <inter name="SPEECH RECO" creator="speech"><!--not used--></inter>
10   <inter name="ONTOLOGY" creator="ontology">
11     <keyword int="INT-CMD:SELECT">select</keyword>
12     <keyword int="INT-CMD:TRANSLATION">translate</keyword>
13     <keyword int="INT-CMD:CONSTRAINTTRANSLATION">moveAxis</keyword>
14     <parameter name="OBJECT">
15       <param int="INT-SELECT-ENT:SEAT" object="Seat 1/3">seat 1/3</param>
16       <param int="INT-SELECT-ENT:SEAT" object="Seat 2/3">seat 2/3</param>
17       <param int="INT-CONFIRM-ENT:SEAT">SELECT</param>
18     </parameter>
19     <parameter name="AXIS">
20       <param int="INT-SELECT-ENT:AXIS" vector="1.0,0.0,0.0">X</param>
21       <!-- ... Y & Z ... -->
22       <param int="INT-CONFIRM-ENT:AXIS">SELECT</param>
23     </parameter>
24     <parameter name="DIST">
25       <param int="INT-SELECT-VAL:DISTANCE" have="DIST">cm</param>
26       <param int="INT-SELECT-VAL:DISTANCE" have="DIST">mm</param>
27       <param int="INT-CONFIRM-VAL:DISTANCE">SELECT</param>
28     </parameter>
29   </inter>
30   <inter name="HAPTICS" creator="haptics">
31     <haptics int="INT-CMD:SELECT">select</haptics>
32     <haptics int="INT-CMD:ATTACH">attach</haptics>
33     <haptics int="INT-CMD:DETACH">detach</haptics>
34     <haptics int="INT-INDICATE-ENT:SEAT">ind seat 1/3</haptics>
35     <haptics int="INT-INDICATE-ENT:SEAT">ind seat 2/3</haptics>
36     <haptics int="INT-SELECT-ENT:SEAT">sel seat 1/3</haptics>
37     <haptics int="INT-SELECT-ENT:SEAT">sel seat 2/3</haptics>
38   </inter>
39 </interpreter>
```

[**]Interactive Physical Pack, a physical simulation library for Virtools developed by Haption

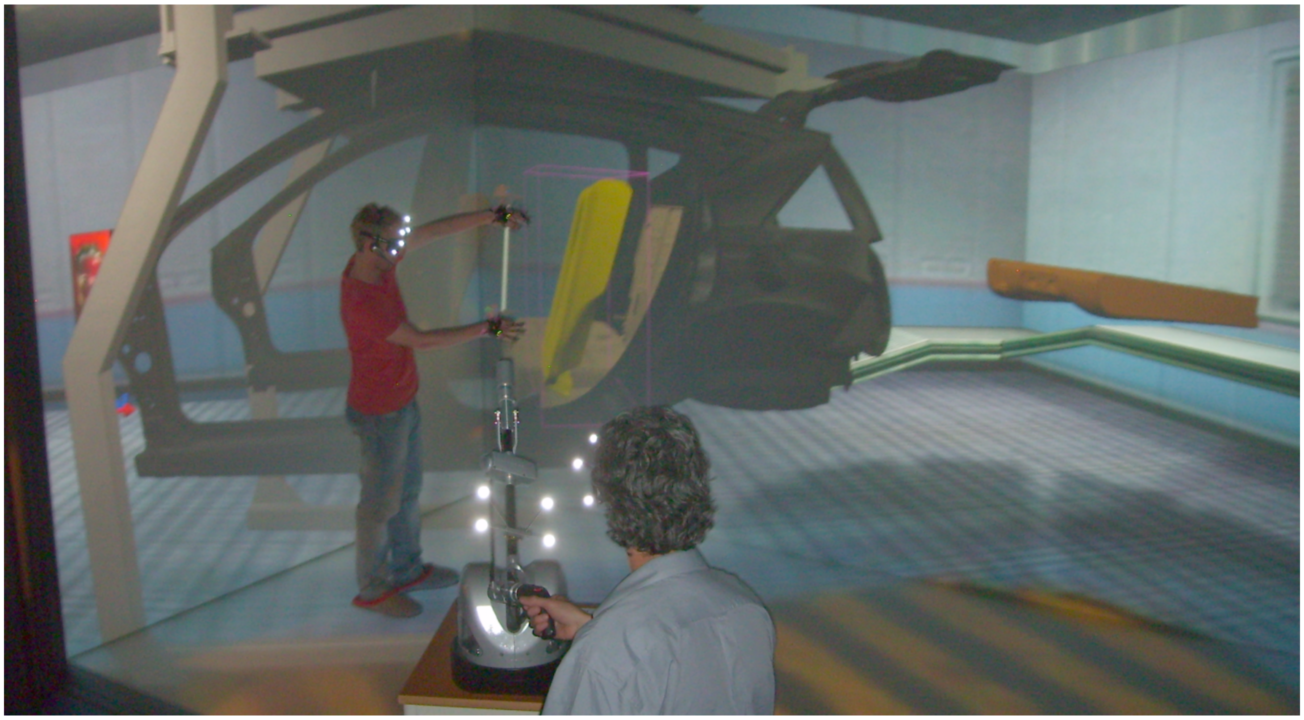

Figure 6. MalCoMIICs application

6. CONCLUSION AND FUTURE WORK

This paper focuses on the design and the implementation of a reconfigurable multimodal and collaborative supervisor (MCS) for VE applications. This work was motivated by the similarities found between multimodal and collaborative processes. The MCS performs *late fusion* to integrate at a semantic level, informations coming from a tracking system, gesture and speech recognition systems, and haptic devices. We demonstrate the modularity and genericity of the proposed multimodal fusion supervisor. An XML-based configuration describes each processing stage by specifying possible combinations of data. This configuration allows it to be easily applicable to many different contexts such as the two described applications. The MCS provides several components required for its complete integration with a VR application/platform: an ”input interface” (*interpreters*), a processing core (*interpretation fusion, argument fusion, command manager*) and an ”output interface” (*command manager*). The ”input interface” processes users inputs such as speech, gestures and haptics, the MCS core handles multimodal and collaborative treatments and the ”output interface” packages the final commands. Data synchronization and time stamping are also taken care of throughout the process. Thanks to its splitting in four stages, the MCS covers the *disambiguation*, *decision* and *dialog* phases, which are three critical points of collaborative and multimodal processes. A *context observer*, configured by a database comprising objects and users representations, feeds the MCS with all the necessary pieces of information.

The MCS is now used in several applications, including some pre-industrial evaluations, and we plan to launch a set of evaluations, such as performance of the system and user studies of the add-value of multimodal and collaborative interactions in VEs. Two different aspects of performance have to be experienced. First the core of the MCS must be benchmarked,[32] to quantify response time — preliminary evaluations show that the existing applications exhibit a very satisfactory response time. On the other hand, complete evaluations of the input systems have to be conducted. We are currently setting up experimentation scenarios to evaluate collaborative interactions. Many complex aspects have to be considered: immersion, usability and utility. We are especially focusing on collaborative interactions between two co-located users: we want to evaluate which level of complexity of collaboration the users may achieve by using a MCS approach. Experimentations will be conducted with ergonomists having a specific expertise in collaborative tasks.

REFERENCES

[1] Bolt, R. A., ""put-that-there": Voice and gesture at the graphics interface," in [*SIGGRAPH '80: Proc. of the 7th annual conference on Computer graphics and interactive techniques*], 262–270, ACM, New York, NY, USA (1980).

[2] Oviatt, S., Cohen, P., Wu, L., Vergo, J., Duncan, L., Suhm, B., Bers, J., Holzman, T., Winograd, T., Landay, J., Larson, J., and Ferro, D., "Designing the user interface for multimodal speech and pen-based gesture applications: state-of-the-art systems and future research directions," *Hum.-Comput. Interact.* **15**(4), 263–322 (2000).

[3] Martin, J.-C., "Tycoon: Theoretical framework and software tools for multimodal interfaces," in [*In John Lee (Ed.), Intelligence and Multimodality in Multimedia Interfaces*], AAAI Press (1998).

[4] Ladry, J.-F., Navarre, D., and Palanque, p., "Formal description techniques to support the design, construction and evaluation of fusion engines for sure (safe, usable, reliable and evolvable) multimodal interfaces," in [*ICMI-MLMI '09: Proc. of the 2009 international conference on Multimodal interfaces*], 185–192, ACM, New York, NY, USA (2009).

[5] Mendonça, H., Lawson, J.-Y. L., Vybornova, O., Macq, B., and Vanderdonckt, J., "A fusion framework for multimodal interactive applications," in [*ICMI-MLMI '09: Proc. of the 2009 international conference on Multimodal interfaces*], 161–168, ACM, New York, NY, USA (2009).

[6] Dumas, B., Lalanne, D., and Oviatt, S., "Multimodal interfaces: A survey of principles, models and frameworks," 3–26 (2009).

[7] Lalanne, D., Nigay, L., Palanque, p., Robinson, P., Vanderdonckt, J., and Ladry, J.-F., "Fusion engines for multimodal input: a survey," in [*ICMI-MLMI '09: Proc. of the 2009 international conference on Multimodal interfaces*], 153–160, ACM, New York, NY, USA (2009).

[8] Bellik, Y., "Media integration in multimodal interfaces," in [*Proc. of the IEEE Workshop on Multimedia Signal Processing*], 31–36 (1997).

[9] Flippo, F., Krebs, A., and Marsic, I., "A framework for rapid development of multimodal interfaces," in [*ICMI '03: Proc. of the 5th international conference on Multimodal interfaces*], 109–116, ACM, New York, NY, USA (2003).

[10] Sun, Y., Chen, F., Shi, Y. D., and Chung, V., "A novel method for multi-sensory data fusion in multimodal human computer interaction," in [*OZCHI '06: Proc. of the 18th Australia conference on Computer-Human Interaction*], 401–404, ACM, New York, NY, USA (2006).

[11] Krahnstoever, N., Kettebekov, S., Yeasin, M., and Sharma, R., "A real-time framework for natural multimodal interaction with large screen displays," in [*ICMI '02: Proc. of the 4th IEEE International Conference on Multimodal Interfaces*], 349, IEEE Computer Society, Washington, DC, USA (2002).

[12] Kaiser, E., Olwal, A., McGee, D., Benko, H., Corradini, A., Li, X., Cohen, P., and Feiner, S., "Mutual disambiguation of 3d multimodal interaction in augmented and virtual reality," in [*ICMI '03: Proc. of the 5th international conference on Multimodal interfaces*], 12–19, ACM, New York, NY, USA (2003).

[13] Pfeiffer, T. and Latoschik, M. E., "Resolving object references in multimodal dialogues for immersive virtual environments," in [*VR '04: Proc. of the IEEE Virtual Reality 2004*], 35, IEEE Computer Society, Washington, DC, USA (2004).

[14] Latoschik, M. E., "A user interface framework for multimodal vr interactions," in [*ICMI '05: Proc. of the 7th international conference on Multimodal interfaces*], 76–83, ACM, New York, NY, USA (2005).

[15] Benford, S., Bowers, J., Fahlen, L., Mariani, J., and Rodden, T., "Supporting cooperative work in virtual environments," *The Computer Journal* **37**(8), 653 (1994).

[16] Broll, W., "Interacting in distributed collaborative virtual environments," in [*VRAIS '95: Proc. of the Virtual Reality Annual International Symposium (VRAIS'95)*], 148, IEEE Computer Society, Washington, DC, USA (1995).

[17] Margery, D., Arnaldi, B., and Plouzeau, N., "A general framework for cooperative manipulation in virtual environments," in [*Virtual Environments*], **99**, 169–178 (1999).

[18] Ruddle, R. A., Savage, J. C. D., and Jones, D. M., "Symmetric and asymmetric action integration during cooperative object manipulation in virtual environments," *ACM Trans. Comput.-Hum. Interact.* **9**(4), 285–308 (2002).

[19] Pinho, M. S., Bowman, D. A., and Freitas, C. M., "Cooperative object manipulation in immersive virtual environments: framework and techniques," in [*VRST '02: Proc. of the ACM symposium on Virtual reality software and technology*], 171–178, ACM, New York, NY, USA (2002).

[20] Schroeder, R., Steed, A., Axelsson, A., Heldal, I., Abelin, Å., Widestr ”om, J., Nilsson, A., and Slater, M., “Collaborating in networked immersive spaces: as good as being there together?,” *Computers & Graphics* **25**(5), 781–788 (2001).

[21] Steed, A., Slater, M., Sadagic, A., Bullock, A., and Tromp, J., “Leadership and collaboration in shared virtual environments,” in [*VR ’99: Proc. of the IEEE Virtual Reality*], 112, IEEE Computer Society, Washington, DC, USA (1999).

[22] Fussell, S. R., Kraut, R. E., and Siegel, J., “Coordination of communication: effects of shared visual context on collaborative work,” in [*CSCW ’00: Proc. of the 2000 ACM conference on Computer supported cooperative work*], 21–30, ACM, New York, NY, USA (2000).

[23] Narayan, M., Waugh, L., Zhang, X., Bafna, P., and Bowman, D., “Quantifying the benefits of immersion for collaboration in virtual environments,” in [*VRST ’05: Proc. of the ACM symposium on Virtual reality software and technology*], 78–81, ACM, New York, NY, USA (2005).

[24] Agrawala, M., Beers, A. C., McDowall, I., Fröhlich, B., Bolas, M., and Hanrahan, P., “The two-user responsive workbench: support for collaboration through individual views of a shared space,” in [*SIGGRAPH ’97: Proc. of the 24th annual conference on Computer graphics and interactive techniques*], 327–332, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (1997).

[25] Arthur, K., Preston, T., Taylor, R., Brooks, F., Whitton, M., and Wright, W., “Designing and building the pit: a head-tracked stereo workspace for two users,” in [*Proc. of 2nd International Immersive Projection Technology Workshop*], (1998).

[26] Blom, K., Lindahl, G., and Cruz-Neira, C., “Multiple active viewers in projection-based immersive environments,” *Immersive Projection TechnologyWorkshop* (2002).

[27] Fröhlich, B., Blach, R., Stefani, O., Hochstrate, J., Hoffmann, J., Klüger, K., and Bues, M., “Implementing multi-viewer stereo displays,” in [*WSCG (Full Papers)*], 139–146 (2005).

[28] Dang, N., Chatelain, C., Pergandi, J., and Mestre, D., “A framework for design and evaluation of collaborative virtual environments,” (2008).

[29] Olson, J. S., Covi, L., Rocco, E., Miller, W. J., and Allie, P., “A room of your own: what would it take to help remote groups work as well as collocated groups?,” in [*CHI ’98: CHI 98 conference summary on Human factors in computing systems*], 279–280, ACM, New York, NY, USA (1998).

[30] Teasley, S., Covi, L., Krishnan, M. S., and Olson, J. S., “How does radical collocation help a team succeed?,” in [*CSCW ’00: Proc. of the 2000 ACM conference on Computer supported cooperative work*], 339–346, ACM, New York, NY, USA (2000).

[31] Salzmann, H., Jacobs, J., and Fröhlich, B., “Collaborative interaction in co-located two-user scenarios,” in [*JVRC ’09: Proc. of Joint Virtual Reality Conference - the 15th Eurographics Symposium on Virtual Environments*], 85–92 (2009).

[32] Dumas, B., Ingold, R., and Lalanne, D., “Benchmarking fusion engines of multimodal interactive systems,” in [*ICMI-MLMI ’09: Proc. of the 2009 international conference on Multimodal interfaces*], 169–176, ACM, New York, NY, USA (2009).