An easy-to-use Framework to integrate Data Processing and Data Fusion in VR Applications

Pierre Boudoin, Samir Otmane, Malik Mallem and Hichem Maaref
IBISC Laboratory
CNRS FRE 3190
University of Evry, France

Abstract

One of the most important aspect when designing a VR application is to preserve the 3D interaction continuity. In order to respect this point, during the last years, VR systems need more and more sensors to provide the most efficient 3D interaction in the virtual world. But, with the multiplication of sensors in these systems, new constraints has emerged, especially how to process the huge amount of incoming data and how to estimate a correct interpretation from it. However data processing is often a complex approach. In this paper, we present to you a framework that can be used such an upper-layer on the hardware layer to provide data processing and data fusion to a virtual reality application.

1. Introduction

One of the aim of the research in 3D interaction in virtual environment is to provide the best user's experience in virtual reality. We call best user experience, the fact that 3D interaction is natural, the nearest from the reality. The main solution to reach this objective is to multiply the number of devices and sensors to use them in the best way as possible. This seems to be a good approach because by multiplying the numbers of sensors and devices we offer to the system the possibility to interpret what the user's doing.
For example,
-many sensors measuring the same type of data permit to minimize the ambiguity or uncertainty on noise.
-many identical sensors working in parallel allow a faster data extraction and so diminish the processing time.
-many different sensors can be complementary and so can create new information, deduce information unapproachable for only one sensor, or extract one information from a high-level symbolic layer.

But, there is a serious problem with this approach: how to interpret this huge amount of data? Indeed, multiple sensors give huge data flow and the aim is not to study each data from devices in an independent way. It's where data processing and data fusion may be useful.

Another problem, is the duration of the data processing because it can be very time-consuming. So it can be not very adapted to put this process into the VR application because it could slow down the rendering and the interaction. It is more cautious to offset the data processing part from the VR application.

2. Related Work

It seems that no one works on a framework for using data processing and data fusion in virtual reality. However many researches have been made on multimodal framework. The study of their architecture is very interesting and can be adapted to match our objectives. All these works are generally based on XML [1] files and a knowledge base. But, each architecture has an interesting feature.

Thus, the Ed Kaiser's team [2] worked on mutual disambiguation. Their work testified an interesting approach to manage multimodality with the use of what they called Multimodal Integrator. The aim of this integrator is to find the best multimodal interpretation with the preliminary rated inputs. The principle is to unify inputs data in:
-Amalgamating redundant or complementary data via a logical test set;
-Taking care about the spatio-temporal aspect of data;
-Taking care about data hierarchy.

Every multimodal architecture which supports the gesture or speech recognition uses one or many components to realize data processing. Latoschik [3] presented a user interface framework for multimodal VR interaction which uses a Knowledge Representation Layer (KRL) and another component for gesture processing. The KRL component was designed to represent the environment's knowledge about objects and their features, possible(inter)actions as well as lexical bindings for these representations. In the same way , Touraine et al. proposed a framework to manage mutimodal fusion of events [4]. This framework manages many multi-

ICAT 2008
Dec. 1-3, Yokohama, Japan
ISSN: 1345-1278

sensorial devices. But, the main point of this framework is that it has a distributed structure allowing it a complete dispatching of device services and their client on as many machines as required. It's also used dated events and a synchronization system to process them adequately. Irawati et al. also proposed a multimodal framework [5] and they defined one interesting component: the interaction manager. The interaction manager has been designed to integrate the interpretation results of multimodal to be a single complete interpretation.

Leonhardt and Magee proposed an additional layer of indirection between sensors and application [6], dedicated for location-aware application. This layer supports acquisition and integration of location data from multiple heterogeneous devices. Another interesting point is that his architecture can be recursive, allowing for multi-stage acquisition trees.

The Virtual-Reality Peripheral Network (VRPN) [7] is a set of classes within a library and a set of servers that are designed to implement a network-transparent interface between application programs and the set of physical devices (tracker, etc.) used in VR systems. The idea is to have a PC or other host at each VR station that controls the peripherals (tracker, button device, haptic device, analog inputs, sound, etc). VRPN provides connections between the application and all of the devices using the appropriate class-of-service for each type of device sharing this link. The application remains unaware of the network topology. VRPN also provides an abstraction layer that makes all devices of the same base class look the same.

3. Proposition

In a classical VR system we generally have got this type of architecture. In one hand we've got the different devices and their dedicated server or API. In the other hand we've

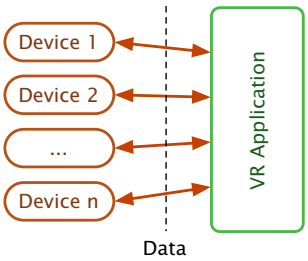

Figure 1. Classic VR system architecture.

got the application which try to communicate with these devices, somehow or other. We could also encounter this type of architecture, which is based on a hardware abstraction layer in charge of the unification, and standardization of data incoming from the different devices. The most popular is VRPN. The problem with VRPN is that it's quite complex to add a device, which is not yet compatible with it. Moreover, VRPN doesn't interpret the incoming data. So,

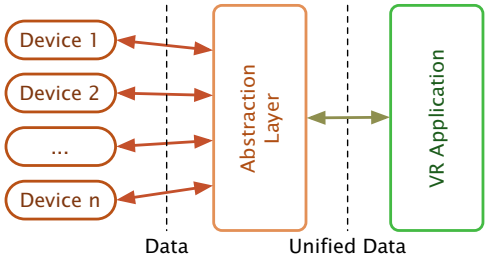

Figure 2. VR system architecture with abstraction layer.

we propose a new VR system architecture (see fig.3), using what we call an Adaptative 3D Interaction Framework (A-3DI Framework), by adding a new component: data processing and data fusion, between the abstraction layer and the VR application.

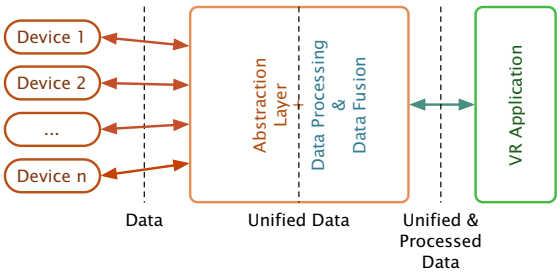

Figure 3. VR sytem architecture with AI3D framework.

4. A-3DI framework in a VR system

4.1. Devices

The user is interacting with the virtual environment through many devices, sensors or interfaces. All of these communicate with PCs via drivers, API or network.

4.2. VR application

The VR application represents the part of software where the 3D interaction techniques are implemented and where the rendering of the application is done.

4.3. A-3DI Framework

The A-3DI Framework is situated between the devices part and the VR application (see fig.4). It is composed of three elements: A-3DI Core, a configuration file and a web application.

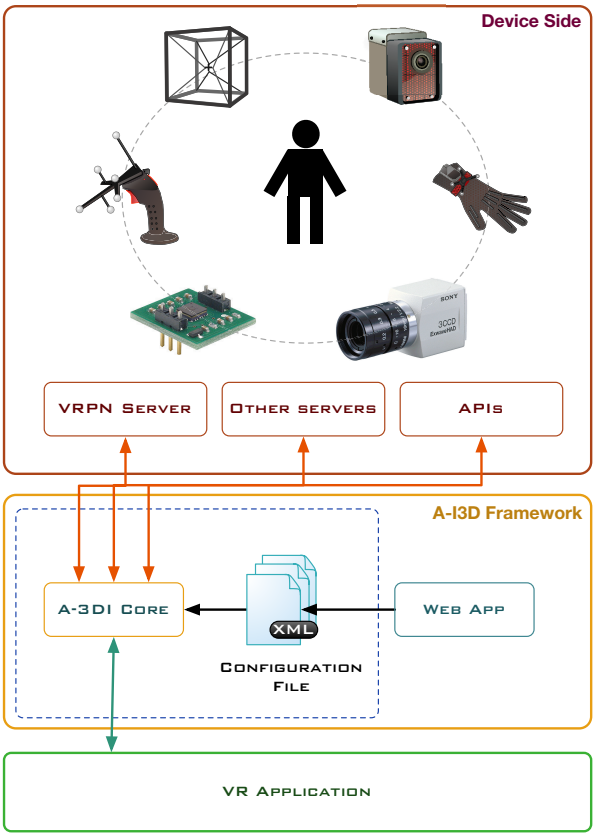

Figure 4. VR sytem with AI3D framework.

4.3.1 Web Application

We planned to use a web application to permit the user to configure the AI3D framework through a web form which generates a XML configuration file.

4.3.2 Configuration file

All configuration is described in one XML file. This file is composed of three main part: devices initialization description, virtual device description and virtual events description.

•Device initialization description

The device initialization description (see fig.5) is the part in the configuration file, in which the user describes which devices he wants to listen and which data will be extracted.

•Virtual devices description

In the virtual devices description (see fig.6) are described combinations of devices and how are computed the combined data. So it's here that the fusion criteria, combination law and the corresponding methods can be defined.

•Event description:

Here (see fig.7) are listed all the events the user wants to catch in the VR application. For doing this, the user must indicate which data from which device to catch or what condition on data must be verified to trigger the event.

```
<initialization>
    <name>
        EVR@ Default Configuration
    </name>
    <description>
        Default configuration for our VR platform
    </description>
    <usedDevices>
        <vrpn>
            <device class="tracker" type="VRPN" name="ART-Hand" />
            <device class="button" type="VRPN" name="ART-Flystick" />
            <device class="button" type="VRPN" name="Data Gloves" />
        </vrpn>
        <others>
            <device class="tracker" type="other" name="SPIDAR" />
            <device class="tracker" type="other" name="Wiimote" />
        </others>
    </usedDevices>
</initialization>
```

Figure 5. Device initialization description part.

```
<devices>
    <device class="tracker" type="virtual" name="Hybrid Tracking">
        <method name="correspondance">
            <device name="ART-Hand" />
            <device name="SPIDAR" />
        </method>
    </device>
</devices>
```

Figure 6. Virtual device description.

```
<events>
    <event type="virtual" id="0" name="Closed fist" method="test">
        <device class="analog" type="VRPN" name="Data Gloves">
            <value>0</value>
        </device>
        <device class="button" type="VRPN" name="Flystick Button">
            <value>0</value>
        </device>
    </event>
</events>
```

Figure 7. Event description.

4.3.3 A-3DI Core

A-3DI Core is the main part of all the framework. Fig.8 shows the architecture of A-3DI Core. From the XML configuration file, it will be able to establish a connection with the needed devices using the appropriate clients. Then to declare virtual devices and events using the descriptions inside the configuration file. These virtual devices or events will be able to call the mathematical methods to compute the desired data.

Globally, A-3DI Core acts like a virtual device and the VR application only communicate with it and not with all the other physical devices. We call this part A-3DI Core. In this server we designed different parts which all have a

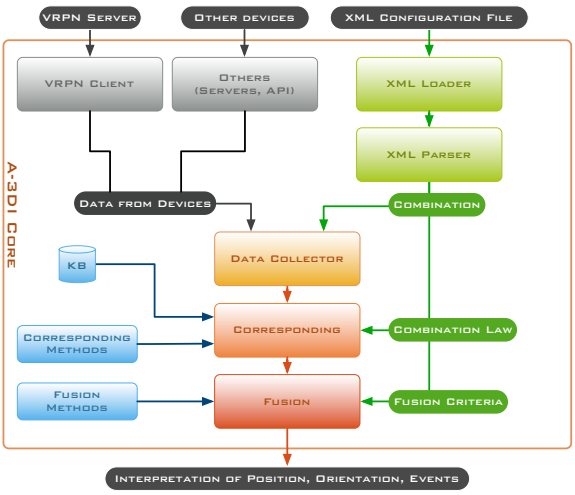

Figure 8. AI3D framework architecture.

dedicated task. The architecture of this framework has been thought in order to be the most modular as possible.

A-3DI Core is based on a multithreaded multi-users TCP server. It allows it to release the processed data on network and assuring compatibility with all VR application. Indeed, just a TCP client is needed for retrieve data from A-3DI Core. It's also assure that the data processing doesn't interfere with the rendering of the VR application, and thus prevents from lowering the performances of the application.

5. A-3DI Core

5.1. A-3DI Core components

5.1.1 Client Side

The device communication component allows the A-3DI framework to exchange data with the device. It is composed of a VRPN client in order to communicate with devices compatible with VRPN. But is also include other clients in order to communicate with third-party devices. This component is the first data abstraction layer.

5.1.2 Corresponding Methods

This component acts like a library of mathematical methods for doing correspondence

5.1.3 Fusion Methods

This components is also a library of mathematical methods for realizing data fusion.

5.1.4 XML Loader

The XML Loader open the configuration file and check is validity. If it doesn't respect the W3C standard, this component display a warning to the user.

5.1.5 XML Parser

The XML Parser component parse the configuration file in order to retrieve the three description seen earlier in order to dispatch them to the others components:
-the initialization description,
-the virtual device description,
-the events description.

5.1.6 Knowledge Base

A knowledge base can be defined which contains some useful informations about the device we use. But, depending from the application context, this is not necessary.

5.1.7 Data Collector

The role of the Data Collector is to retrieve the data from the different devices using the dedicated clients or API following the initialization description in the configuration file. From the output of this component, each data are formatted following if they came from a tracker, a button, an analog or if they are events. This component is the second abstraction layer in the framework.

5.1.8 Corresponding

Data to be fused are often heterogeneous. These data must be re-designed because it's impossible to fused them in their primary form. So we must find a new common representation space. A first process is necessary to transform some of the initial data into equivalent data in a common space in which the fusion will be made.

5.1.9 Fusion

It's inside this component that the fusion operation is realized. The readjusted and modeled data are combined following a combination law induced by the chosen theoretical framework. It's also here that decision making can occurred in order to solve conflicts resulting from the fusion of contradictory data incoming from the different sources.

5.2. Algorithm

Fig. 9 shows a simplified algorithm representing the unfolding of the process in A-3DI Core. We can see the computing loop in which:
-Firstly, data are retrieved from the real devices.

-Then, all data are processed following methods described in the configuration file and the computed data are injected in the virtual devices.
-Finally, data from virtual devices are released on the network

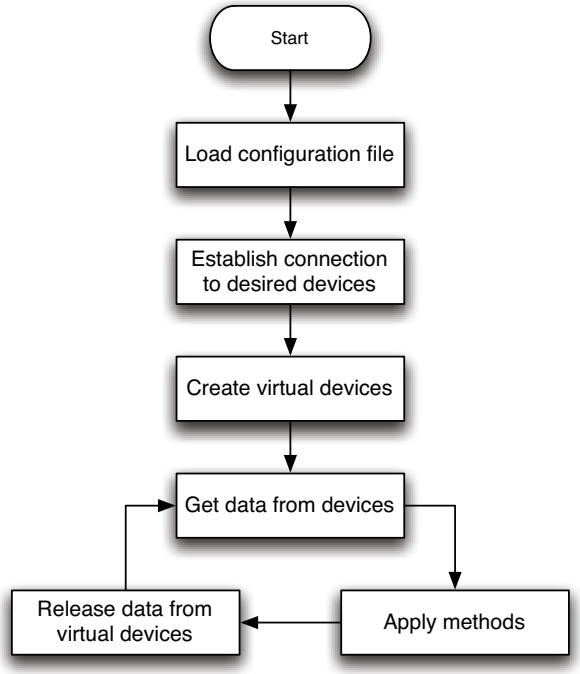

Figure 9. Simplified algorithm.

5.3. Process study

Data providing from A-3DI Core are sent on the network, so it's very easy to study in real time what the A-3DI Core has computed using a math software which allows to use TCP connection such MathWorks™ MATLAB. It can be very interesting to monitor incoming data or visualize them in near real-time, in order to improve the data processing and, so, reach our objectives.
All data incoming from each device or computed after each step of the data processing can be retrieved through date stamped packets by activating a 'verbose' mode in A-3DI Core. In order to not overload the network the 'verbose' mode is optional and comprises different level of details.
The whole process study is described in figure 10. Initially, the user has got a first idea of what he wants to do. Then he designs his idea by using the web application, which generates the XML configuration file. At launch the A-3DI Core loads the configuration file and data exchange is done with the VR application. If the 'verbose' mode is activated, the user can monitor data incoming from the A-3DI Core with a math software and eventually improve his algorithm and a new cycle begins.

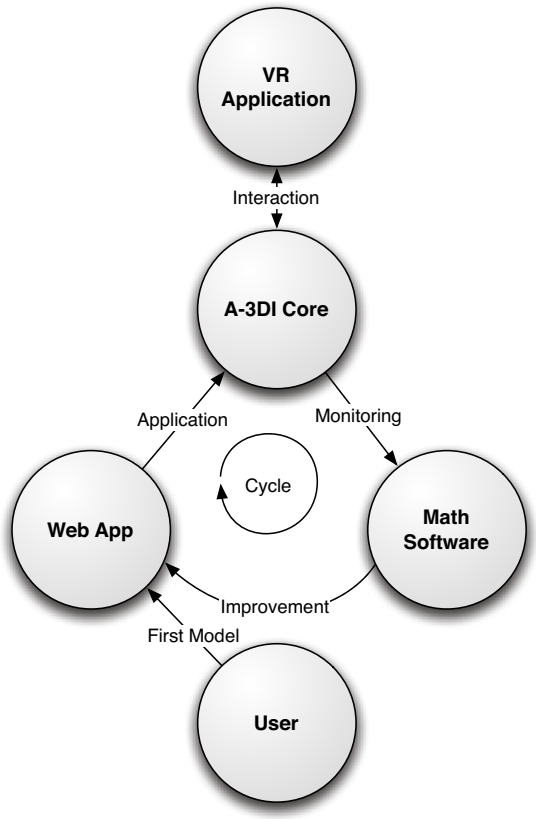

Figure 10. Process study cycle.

6. Experimental Framework

6.1. VR Platform

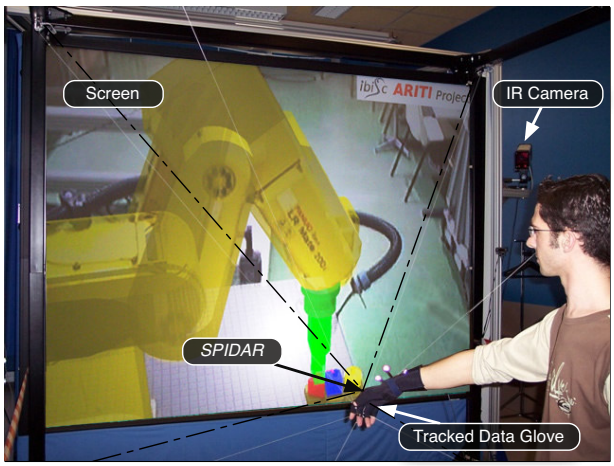

Figure 11. Our semi-immersive VR/AR platform. We can see a user working with optical tracked wireless Data Gloves and a SPIDAR.

Our laboratory owns a semi-immersive VR/AR multi-

modal platform called EVR@ (see fig.11). It permits stereoscopic display, wireless hand/head gesture tracking and force feedback. We have ART optical tracking system composed of two ARTTrack1 infrared cameras, two wireless Flystick, one head marker and one hand marker. We also have a 6D force feedback device SPIDAR-G [8] and 5DT Data Gloves. Each device is associated to a specific server which is accessed via the C++ VRPN library [7] by clients. The interactivity between the user and the VE is done by using Virtools™ 4.0 [9] as a front-end. Virtools™ is efficient software for prototyping and testing 3D IT because it offers a fast and graphical way to compute them and link them with hardware devices and VEs by connecting specific building blocks to each other.

6.2. Application

6.2.1 Context

Everybody who use an optical tracking has been confronted to the situation where markers are occulted from IR cameras and therefore the tracking system cannot compute the position of them, creating bad body position informations and a jump when they are not occulted any more. In these situation it is interesting to switch to another tracking device, the duration of the optical tracking isn't available. But still it is necessary to be sure that the other tracking device has got the same referential and measure that the optical tracking. If not, jumps of position measurement effect will occurs.

Depending on the application, it could be serious consequences and this is our situation. Indeed we currently teleoperate a 6 degrees of freedom robot with a Flystick but this is not very natural and easy for the user. We wish to teleoperate it with a tracked Data-Glove. But optical hand-tracking is more sensitive to the occultation situation, so we would like to use them in combination of a SPIDAR to prevent from occultation and also for providing a haptic feedback.

6.2.2 Methods to apply

We know that the SPIDAR we own, have many weaknesses and these prevent us from using its tracking capabilities in an optimal way. Especially, we know that there is a linear shift between the measure perceived by the SPIDAR and the reality. We wished to readjust the SPIDAR tracking in order to match the reality.

For this we used another device which is very accurate: the ART optical tracking which offers a less than 10^{-3} millimeter error. When the optical tracking is available we used it and calculate the scale matrix A and the shift matrix B, both defined in (3), which allow us to pass from SPIDAR position (P_{SPID} - Eq.2) to optical tracking position (P_{ART} - Eq.1) from the relation defined by (4). Then, by resolving a system, we can compute the adjusted SPIDAR position (5)

$$P_{ART} = [X_{ART}, Y_{ART}, Z_{ART}]^T \tag{1}$$

$$P_{SPID} = [X_{SPID}, Y_{SPID}, Z_{SPID}]^T \tag{2}$$

$$A = \begin{pmatrix} a_x & 0 & 0 \\ 0 & a_y & 0 \\ 0 & 0 & a_z \end{pmatrix}, B = \begin{pmatrix} b_x \\ b_y \\ b_z \end{pmatrix} \tag{3}$$

$$P_{ART} = P_{SPID} * A + B \tag{4}$$

$$P_{SPID_{adj}} = P_{SPID} * A + B \tag{5}$$

However, we also must to define priority for these two devices. The highest priority has been given to the ART optical tracking because it has the best information quality. Following this priority list the A-3DI Core will listen the highest priority device if it has got an activity. Else it will listen the following in the priority list and etc (see fig.12). A-3DI Core tests the activity from a tracking device by watching the data variation. If data are constant, the device is declared without activity, else the device is declared with activity. Fig 13 shows an example of device activity.

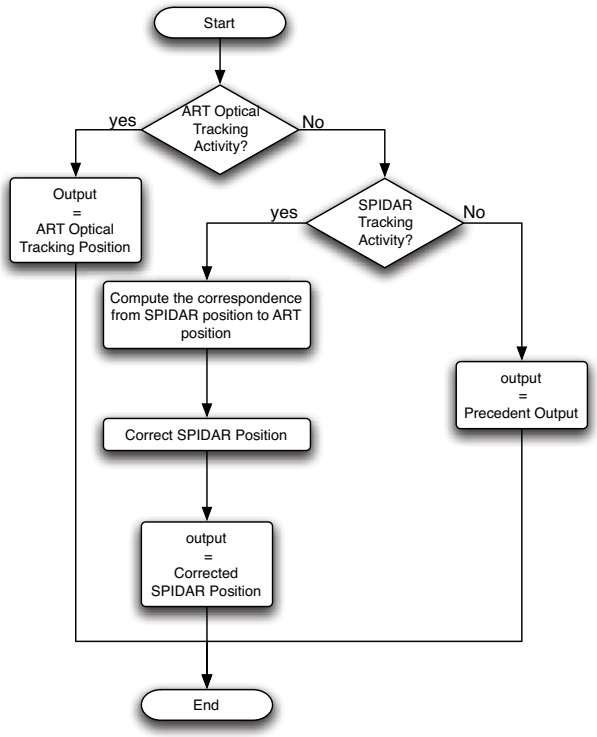

Figure 12. Priority switch algorithm

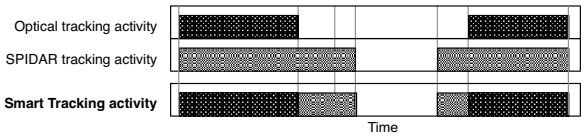

Figure 13. Example of device activity

6.3. Using our framework

To use our framework for this application, we need to generate the XML configuration file using the web application (Currently the web application hasn't been developed, we create the XML file with a notepad application). In the first part which contains the initialization part (see fig.14) we define that we use two devices:
-one VRPN-type: the ART tracking
-one other-type: the SPIDAR

In a second time we define the virtual devices and the process made on data. Here we need to make a correspondence from ART tracking's position to SPIDAR's position. We do this by calling the correspondence method (see fig.15).
There is no need of the third part: event description, because we don't have to manage any event in this application.

This is the only settings, we ask to the developer or user of the application. After doing this, the A-3DI Core is launched and the VR application can retrieve data from it via a simple TCP client. All data processing made is transparent to the user's eyes.

```
<initialization>
    <name>
        EVR@ Default Configuration
    </name>
    <description>
        Default configuration for our VR platform
    </description>
    <usedDevices>
        <vrpn>
            <device class="tracker" type="VRPN" name="ART-0"/>
        </vrpn>
        <others>
            <device class="tracker" type="other" name="SPIDAR"/>
        </others>
    </usedDevices>
</initialization>
```

Figure 14. Configuration file - Initialization part

6.3.1 Results

The two following figures show the tracking positions using Optical tracking and SPIDAR before (16) and after (17) being processed through A-3DI Core. We use MATLAB for monitoring these data.

As you can see in 17 the SPIDAR tracked position overlaps the optical tracked position. With this method, we reached a mean variation of 3.4 mm on the X axis, 3.2 mm on the Y axis and 4.1 mm on the Z axis.

```
<devices>
    <device class="tracker" type="virtual" name="Hybrid">
        <method name="correspondance">
            <device name="ART-0"/>
            <device name="SPIDAR"/>
        </method>
    </device>
    <device class="tracker" type="virtual" name="Switch">
        <method name="priority">
            <device name="ART-0" priority="2"/>
            <device name="Hybrid" priority="1"/>
        </method>
    </device>
</devices>
```

Figure 15. Configuration file - Virtual device part

So if the user isn't tracked by the optical tracking anymore the system can switch transparently to the SPIDAR tracking without notAs you can see the SPIDAR tracked position overlaps the optical tracked position. So in the user isn't tracked by the optical tracking anymore the system can switch transparently to the SPIDAR tracking without notable jump in the position, providing a continuity in the 3D interaction jump in the position, providing a continuity in the 3D interaction.

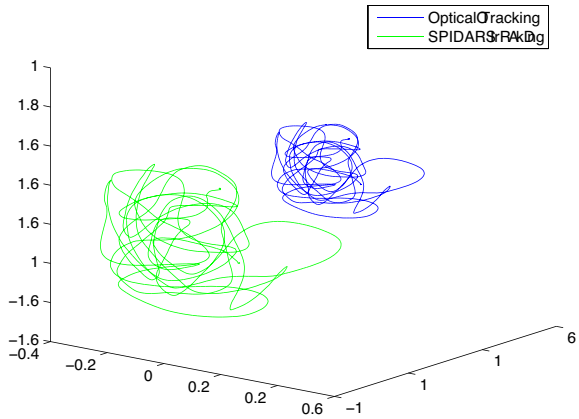

Figure 16. Tracked position from optical tracking and SPIDAR without correspondence

7. Conclusion and Future Works

The main objective of our system is to get a natural 3D interaction in virtual environments. in which the user could choose the device(s) which he should be the more efficient, without worrying about the hardware layer.

In this paper, we would like to answer to a problem which is: How to manage efficiently many devices in order to provide to the user, a natural 3D interaction in virtual environments? In order to answer this question, we proposed an Adaptive 3D Interaction System (A-3DI). A-3DI allows developers of VR applications to manage data from many devices and to process them using data process-

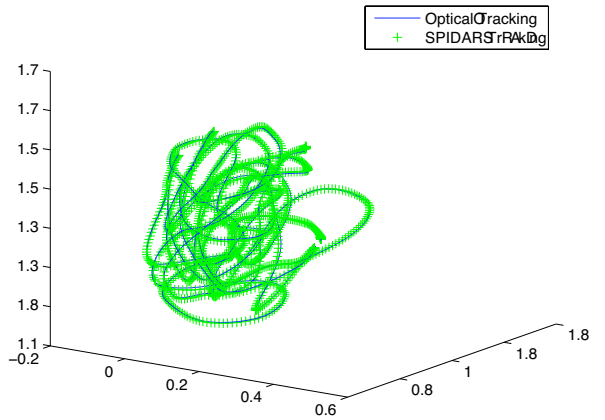

Figure 17. Tracked position from optical tracking and SPIDAR with correspondence using the A-3DI framework

ing and data fusion methods. The developer can easily and quickly create virtual devices, combining many physical devices, which may call mathematical methods and uses them in the VR application.

This A-3DI system has been developed in C++ and tested on Virtools™ with a preliminary experiment that serves to switch from ART optical tracking to a SPIDAR tracking smoothly without jumps.

We plan to enrich the library of mathematical methods with more methods. We also want to use this framework with more devices in order to integrate it in our Augmented Reality Interface for Teleoperation on the Internet (ARITI) [10]. ARITI consists in the control of a 6 degrees of freedom robot for teleoperation task that uses the ARITI framework to assist the user. This teleoperation is made from our VR/AR semi-immersive platform EVR@ which allows 3D interaction.

8. Acknowledgments

We wish to thank "Le Conseil Général de l'Essonne" and "Le C.N.R.S." for funding this work.

References

[1] T Bray, J Paoli, and C Sperberg-McQueen, "Extensible markup language (xml) 1.0 (third edition)", *World Wide Web Consortium Recommendation*, 2004. 1

[2] Ed Kaiser, Alex Olwal, David McGee, Hrvoje Benko, Andrea Corradini, Xiaoguang Li, Phil Cohen, and Steven Feiner, "Mutual disambiguation of 3d multimodal interaction in augmented and virtual reality", *ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces*, pp. 12–19, Nov 2003. 1

[3] James Myers and Robert McGrath, "Cyberenvironments: adaptive middleware for scientific cyberinfrastructure", *ARM '07: Proceedings of the 6th international workshop on Adaptive and reflective middleware: held at the ACM/IFIP/USENIX International Middleware Conference*, Nov 2007. 1

[4] Damien Touraine, Patrick Bourdot, Yacine Bellik, and Laurence Bolot, "A framework to manage multimodal fusion of events for advanced interactions within virtual environments", *EGVE '02: Proceedings of the workshop on Virtual environments 2002*, vol. 23, pp. 159–168, May 2002. 1

[5] Sylvia Irawati, Daniela Calder, and Heedong Ko, "Spatial ontology for semantic integration in 3d multimodal interaction framework", *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, pp. 129–135, 2006, 1128944. 2

[6] Ulf Leonhardt and Jeff Magee, "Multi-sensor location tracking", *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pp. 203–214, 1998, 288291. 2

[7] Russell Taylor, II, Thomas Hudson, Adam Seeger, Hans Weber, Jeffrey Juliano, and Aron Helser, "Vrpn: a device-independent, network-transparent vr peripheral system", *VRST '01: Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 55 – 61, Nov 2001. 2, 6

[8] Seahak Kim, Masahiro Ishii, Yasuharu Koike, and Makoto Sato, "Development of tension based haptic interface and possibility of its application to virtual reality", *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 199–205, 2000, 502428. 6

[9] Dassault Systemes, "Virtools", http://www.virtools.com/. 6

[10] Samir Otmane, Malik Mallem, A Kheddar, and Florent Chavant, "Ariti : an augmented reality interface for teleoperation on the interne", *Advanced Simulation Technologies Conference 2000, High Performance Computing*, pp. 254–261, 2000. 8