Design and Implementation of a wearable, context-aware MR framework for the Chloe@University application

Xavier Righetti
xavier.righetti@epfl.ch

Achille Peternier
achille.peternier@epfl.ch

Mathieu Hopmann
mathieu.hopmann@epfl.ch

Daniel Thalmann
daniel.thalmann@epfl.ch
VRLab - EPFL
1015 Lausanne
Switzerland

Abstract

In this paper, we present the technical details and the challenges we faced during the development and evaluation phases of our wearable indoor guiding system which consists of a virtual personal assistant guiding the user to his/her desired destination. The main issues that will be discussed can be classified in three categories: context detection, real-time 3D rendering and user interaction.

1 Introduction

Wearable computing and mobile augmented reality are often considered as complex topics because of the amount of difficulties that need to be overcome during the implementation of such systems. Computational resources, device size and weight, battery power, comfort, robustness, QoS, etc. are all factors that need special attention and balancing in order to build a working framework that fits our research requirements.

A typical wearable augmented reality system features three main elements: user worn electronic devices responsible of data retrieval, communication and processing (ranging from networking to image rendering), a optical display to feed user eyes with visual feedback and a tracking system to ensure an adequate registration between the real and virtual worlds[11].

We introduced a global overview of our project in [12] and first results of our system in [10]. In this paper we expose technical details about the design and implementation of our wearable mixed reality framework embedded in a jacket and featuring real-time rendering, geolocalization and a hands free interface. We built this system in the scope of the Intermedia Chloe@University project, summarized in section 3. In sections 4 and 5 we first describe the hardware and software architecture we adopted, followed by a detailed descriptions of each component we implemented and the results we obtained. Section 6 is dedicated to a two-ways evaluation of our framework, one *in vitro* and one during a real demonstration scenario. Section 7 contains our remarks concerning the lessons learned and future work.

2 Related work

Advances in mobile computing, thanks to fusion of wearable computers, wireless networking and mobile AR interfaces, have widely increased the creation and adoption of wearable augmented reality systems [11]. Augmented Reality-based frameworks require a very accurate localization of the user within the environment, including not only his/her position but also orientation and current sight direction. For example systems relying on fiducial markers like [15] or [6] require a former preparation step of the surroundings wherein the AR experience will take place. Despite the fair accuracy offered by such tracking systems (ARToolKit in this case), scenarios are limited to areas previously filled with fiducial markers. Our solution aim at being less accurate but more generic, by reducing or eliminating this step in order to cover also wider areas with less effort.

Many researchers moved to handheld devices to reduce obstruction and weight of wearable frameworks (as described in [18]) by using mobile phones [8] or Personal Digital Assistants (PDAs) [17]. In [5], DeVaul et al. describe MIThril: a platform for wearable computing research based on a Sharp Zaurus PDA. Since 2000, this linux-based platform is used to facilitate the development of distributed real-time multimodal and context-aware applications. It supports a HMD, a user-input device, some physiological sensors, and a wireless communication unit. Similarly, in [14], we also developed a very compact and lightweight MR wearable system using a PDA as core device. Thanks to recent improvements in mobile hardware, we decided to switch to a slightly bigger but less limited machine in this project, preferring a Ultra-Mobile PC (UMPC) instead. In [1], Oliver Amft et al. chose a

1-4244-1506-3/08/$25.00 ©2008 IEEE

different approach by developing their own core unit and hiding it inside a belt, making it completely invisible and unobtrusive.

Indoor localization and tracking of persons has been an active research subject for many years, which began with specialized hardware requirements (infrared, ultrasound, electromagnetism, etc). Recent systems tried to avoid these costs and deployment problems by using technologies already deployed such as GSM or Wi-Fi. One of the first works in this direction was RADAR [3] [4], a tracking system based on signal strength information given by radio frequency devices. RADAR system worked in two phases: the off-line phase which consisted in collecting signal strength data in order to build a signal map, and the real-time phase, corresponding to the localization phase and based on the previously acquired signal map. This system provided a median resolution of nearly 3 meters. This two-phase model has inspired several papers, which tried to enhance this system in different manners and for different purposes: enhance accuracy and precision [2] [9], localize in high signal fluctuation areas [7], use only one access point [19]. Section 5.5 will describe our choices and implementation.

3 Chloe@University project

The Chloe@University project is part the a european research project, interactive media with personal networked devices (INTERMEDIA) in which we seek to progress beyond home and device-centric convergence toward truly user-centric convergence of multimedia. Our vision is The User as Multimedia Central: the user as the point at which multimedia services and the means for interacting with them converge.

4 Hardware architecture

In this section, we will define the hardware design specifications for the Chloe@University project.

4.1 System overview

As shown on figure 1, our system consists of a jacket, a integrated circuit (IC) board, a core unit, an optical monocular see-through Head-Mounted Display (HMD), an orientation tracker, a Bluetooth module, a WiFi antenna, a self-powered USB hub and an extra battery. Each component will be thoroughly described in the following sections.

4.2 Core unit

We adopted a Sony Vaio UX-70 Ultra Mobile PC as core unit of our wearable system. This devices is a very compact PC closer to a notebook for performances (Intel Centrino 1.06 GHz processor) and a PDA for sizes (150 mm x 32 mm x 95 mm, weighting only 0.5 kg). This device is equipped with 512 MB RAM, 30 GB hard-disk and perfectly fulfills our needs in communication (embedded

Figure 1. Overall view of the hardware

bluetooth and WiFi adapters), connectivity (via a powered USB port) and visualization (featuring a Intel 945GMS graphics chipset and a VGA out).

The core unit is responsible for centralizing the sensor's data and for running the application that generates the 3D environment on the HMD. The Vaio's battery supplies a 2600mAh current and has an autonomy of 3.5 hours according to the manufacturer. From our experience, the battery only last a maximum of two hours when all peripherals are attached and the software is running.

4.3 See-through Head-Mounted Display

The Liteye-500 see-through monocular head-mounted display (HMD) provides information to the user thanks to its 852x600 OLED microdisplay. It is connected to the central processing device using a VGA cable and is powered by a standard USB plug. The HMD works better in low light environments, but for a better visibility, it features a sliding panel behind the display to switch from the see-through modality to a non-transparent one.

4.4 The I-Jacket

In our project, the iJacket has two main functions: first, it is a wearable garment used for storing various hardware devices. Second, it is a User Interface (UI) responsible for getting the user input. The UI for user input consists of a flexible membrane keyboard with 8 buttons. It is connected to a dedicated IC board developed internally. The board houses an embedded PIC microcontroller in order to detect when and which buttons are pressed . The Integrated Circuit (IC) board is then connected by USB to the core unit. A battery powers the IC board with at least 5 hours of autonomy. This 4.8V NiMH battery provides 1800mAh and can be recharged in one hour.

The microcontroller has been programmed using PCWH Compiler from Custom Computer Services, Inc. This third party software lets the programmer code in standard ANSI C. In our software architecture, the PIC18 mi-

Figure 2. User interacting with the i-jacket.

crocontroller is 100% interrupt-driven. It is a commonly used technique for real-time computing. The information is then sent to the core unit via an RS-232 serial link over USB thanks to the embedded FT232BL USB UART chip from FTDI.

4.5 XSens inertial sensor

The MTx is a small and accurate 3DOF inertial Tracker. It provides drift-free 3D orientation as well as kinematic data: 3D acceleration, 3D rate of turn (rate gyro) and 3D earth-magnetic field. The MTx is an excellent measurement unit for orientation measurement of human body segments. A highly dynamic response combined with long-term stability (no drift) outputs the accurate full 360 degrees 3D orientation. Furthermore, the sensors can be individually calibrated for temperature, 3D misalignment and sensor cross-sensitivity.

In our project, the inertial tracker is used to map the orientation of the virtual map displayed in mixed reality (MR) with the physical orientation of the user's head. This functionality guarantees the user to see the virtual map with an intuitive point of view.

A dedicated DLL has been created in order to get real-time information from the orientation sensor. Hence, an additional thread constantly listens to the serial port and updates the orientation of the 3D architectural model.

4.6 Bluetooth modules

We chose Bluetooth for detecting surrounding multimedia equipments such as a carPC, a printer, a mobile phone or a digital camera as it is the standard protocol for communicating wirelessly between electronic devices. The Bluetooth device currently used is a WT11 module from Bluegiga which is class 1 and Bluetooth 2.0 compatible. Also, we opted for the Bluesoleil[1] stack as it has a simple API for detecting and communicating with Bluetooth devices.

[1]http://www.bluesoleil.com

In our current implementation, the jacket can be connected to any Bluetooth-enabled computer and is able to receive and store the name and the elapsed time of the song currently playing with the Itunes software.

The multimedia devices that can connect to the jacket are constantly looking for Bluetooth devices (Inquiry) and once the jacket is found, a connection to the Serial Port Profile (SPP) service is established. Then, the application constantly checks if the connection is alive. If not, the Inquiry routine is relaunched until the jacket is found again. This approach reduces the energy consumption of the jacket as its Bluetooth module is discoverable but not always active. Additionally, upon successful connection, the jacket detects the approximate location of the user and thus derives appropriate functionalities or events related to the context.

4.7 Hardware for localization

Unfortunately, the WiFi chipset embedded on the Ultra Mobile PC was not satisfying to perform localization: the signal strength values range was too short (between -50 and -90dB) compared to the chipset embedded on an Intel Centrino laptop for example. For this reason, we chose the Linksys WUSB200, which provides a better value range (-30 to -90dB). Another advantage of having a second WiFi chipset is the network connectivity. Indeed, when you are associated with an access point, the values are updated less quickly or only for the linked access point. Next paragraph will describe more precisely these limitations.

Different techniques are available in order to acquire the surrounding access point BSSIDs (Basic Service Set Identifier) and their associated RSSI (Received Signal Strength Information). Under Windows XP, we first though about NDIS (Network Driver Interface Specification)[2]. This interface is only accessible by kernel-space programs, but Microsoft provides a bridge to this interface : NDISUIO. Unfortunately, there is a lack of documentation about NDISUIO, so we gave up this solution. Alternatively, Netstumbler, a tool which facilitates the detection of 802.11 networks, allows the user to add a script, which will for example write WiFi information in a file. However, this method presents some disadvantages : you must turn on Netstumber each time you want to read WiFi information, and Netstumbler disables the wireless connection management utility (no data traffic possible). Fortunately, retrieving WiFi information under the Vista OS is much easier as Microsoft provides along with its operating system a new API, the Native Wifi API [3]. This new interface thus offers C/C++ functions to access and update WiFi information, but the only drawback is a slower WiFi refresh rate when the adapter is connected to an access point. This latter is not an issue for us as we are using two different WiFi chipset : the WUSB200 to read WiFi information, and the UMPC chipset for the network connection. Concretely, with the Windows Native Wifi

[2]www.ndis.com

[3]http://msdn2.microsoft.com/en-us/library/ms706556(VS.85).aspx

API and our Linksys WUSB200, the WiFi information (BSSID and RSSI) are updated at one hertz.

5 Software architecture

In this section we describe first a general overview of the software-side of our system, followed by detailed descriptions of each component.

5.1 System overview

The software system is managed by a stand-alone application running under Windows Vista on the core unit. Our framework is based on a main stand-alone application, called ChloeClient, running on the UMPC. This software is responsible of managing the user human/machine interface, data handling from external devices (sensors, WiFi localization) and generating real-time visual feedback. Sensors and external devices are connected via a plugin architecture, based on dynamic loaded libraries (DLL). We adopted for a .dll oriented approach, thus each framework add-on should be released as a stand-alone .dll (or more .dlls) to be integrated in the ChloeClient via a .h and .lib file. Software side, each additional functionality will then require just a very simple, minimal and robust interface to their contribution. Each module will then be initialized, updated and released by the ChloeClient through the methods declared in the .h file, like InitSensorModule(), updateSensorModule(), etc.

5.2 Core application

The ChloeClient is a C++ application developed in Visual Studio 2005. It uses the Simple DirectMedia Library (SDL)[4] as basic windowing/events manager and the MVisio graphics engine for rendering (see 5.3). The ChloeClient is intended to run continuously when the i-Jacket is worn by a user. This application is then responsible to centralize and manage locally (that is, natively and directly on the UMPC) the different tasks that affect the mobile user. Among these tasks, we enumerate:

1. Generating and displaying visual feedback to the user, ranging from a graphics user interface system to 3D rendering of the environment and the virtual human acting as guide.

2. Reproducing multimedia content, like music, pictures and videos.

3. Retrieving and analyzing information from the different sensors, like current orientation, position (via the RFID module, Wi-Fi, Zigbee, GPS) and environmental conditions (via Zigbee: light availability, temperature, etc.).

4. Using Wi-Fi and Bluetooth to connect the ChloeClient to a remote server, web services, other users or other devices.

[4] http://www.libsdl.org

5.3 Graphics engine

We used the Mental Vision 3D Graphics Engine (MVisio) [13] as rendering software to perform real-time 3D generation of images directly on the core device. MVisio natively optimizes itself for best performances according to the hardware it is running on by automatically deciding which optimized parts of the pipeline to activate.

The Sony UX-70 features a mid-level embedded graphics adapter, Intel 945GM Express, supporting OpenGL 1.4.0 thus no GLSL shading language and forcing the Graphics Engine to use only basic OpenGL optimizations available on the fixed pipeline. An interesting comparison among UMPC implementations of OpenGL under Vista has been published by MicroPCTalk[5].

5.4 User interface

Our user interface is composed of two main parts: the visual feedback (output), which is displayed on the user's HMD, and the tangible interface (input), which is responsible for controlling the system. Further subsections will discuss about the communication between the user and the system in more details.

5.4.1 Visual feedback

We adopted a mixed reality approach instead of a full augmented reality one because of the limitations AR systems usually have, such as the limited area of action because of the very accurate tracking required and the calibration and preparation of an environment to fit into an AR scenario. Our MR system is conceptually similar to a car GPS display: we show to the jacket wearer a world-aligned 3D view of the surrounding. When the user turn his/her head, the 3D orientation of the scene changes accordingly. If the user looks up/down, the 3D camera rises consequently offering a wide fly-by view of the surrounding. The localization system determine which area of the scenario the user is currently moving on and centers the current viewpoint to this sector. The system automatically adapts the camera zoom and depth of view according to the accuracy of the localization system, in order to make sure that the user current real physical position will always be located within the 3D virtual environment.

A virtual guide (concretely a virtual 3D human) gives additional information to the user. The virtual human is substituted for the plain GUI system in order to offer to the user a more user-friendly interface, other than just plain text or abstract feedback (like arrows or signs) and to give a better readability of the information on the HMD, requiring otherwise very large fonts and images to be easily read. The virtual guide leads for example the user from point A to point B by preceding him/her in the virtual 3D environment, showing the way to follow. Thanks to the localization system feedback, the guide waits for the user if he/she is too far or come back and readapt the path if

[5] http://www.micropctalk.com/2007/12/17/ux-models-benchmarked/

he/she get lost or on a wrong way. In order not to hide the virtual guide nor the environment when the user is behind walls or doors, a circle of transparency has been activated around the localization system position returned. This way, the area surrounding the user's position is semi-transparent and fades progressively to solid according to the distance, improving user vision and matching between the virtual visualization of the environment and reality.

5.4.2 Graphics User Interface (GUI)

User input for mobile applications generally requires special attention because of the mobility constraints excluding standard acquisition supports like mouses or keyboards. In our system, we are using three different solutions to let the user interact with the framework: the i-Jacket forearms buttons, the voice recognition and the sensors.

Forearms buttons are directly embedded in the i-Jacket and interfaced with the ChloeClient through a plugin DLL using a standard serial port for data reading. Forearms buttons have the advantages of letting both user hands free when user input is not required. Because of the low amount of buttons available (only 5) we adopted a contextual menu interface, similar to the one available on mobile phones or TVs. According to the operation users want to perform, like selection a destination, changing the guide aspect or point of view, etc. each button assume a different functionality. By adding a 2D graphics interface reproducing the jacket forearm and labeling each button with its function on the top of the 3D view of the surrounding, users can easily interact with the ChloeClient application (see figure 3).

Voice recognition is an alternate way of input that can be used with or in the place of the forearms buttons. Because of the limitations of the current voice-based approaches, featuring good results with a small vocabulary and a good calibration phase, we decided to use only a very small set of voice commands. Each command simulate a forearm button click, thus allowing us to reuse the same interface without major modifications and freeing the user from requiring to free his/her right hand to interact with the software (for example when he/she is holding objects or riding a bicycle). The use of a very small vocabulary also has the advantage of performing pretty well on different user voices without requiring a calibration step, perfectly fitting real-case contexts like demo cases where several users may be wanting to try the i-Jacket by themselves.

Sensors, like the X-Sens orientation tracker or the localization system, are performing passive user feedback. Passive feedback means that users don't need to directly inform the framework about their position or orientation because this information is automatically gathered and used by the system. This passive input is also used to determine context-events according to the user position, like showing complementary information when entering some

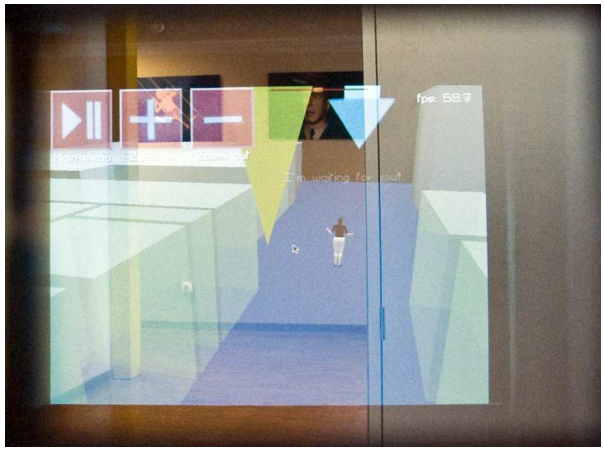

Figure 3. MR view of the graphic UI during selection of destination.

interest points or reaching intermediary destinations.

5.5 Software for localization

First, we have started by enumerating initial requirements for our application. Indeed, we wanted our localization technique to be:

- **Light**: As the UMPC hardware is limited and our program is running resource consuming tasks such as the 3D rendering
- **Fast**: As we want to track moving people, so we need to update user position as fast as possible
- **Easy to deploy**: As the application is intended to run in various places.
- **Robust**: As reliability is more important than precision for our application.

As other positioning systems presented in section 2, our system works in two phases: the off-line phase which consists in collecting signal strength data in order to build a signal map, and the real-time phase, corresponding to the localization phase and based on the previously acquired signal map. During the off-line phase, the user creates a navigation graph: each room (or part of corridor) is represented as a vertex of the graph, then the user has to link neighbor vertices between them and records signal data (access point identifiers and signal strengths) for each vertex.

After this quick training phase (15 seconds per vertex to capture signal data), the localization system is ready to be used: the application takes WiFi samples every second during three seconds, smooths the received signal strengths by removing the values too far from the mean if the standard deviation is too high, locates the user (by a simple distance calculation) according to the signal map acquired during the off-line phase. The system uses the

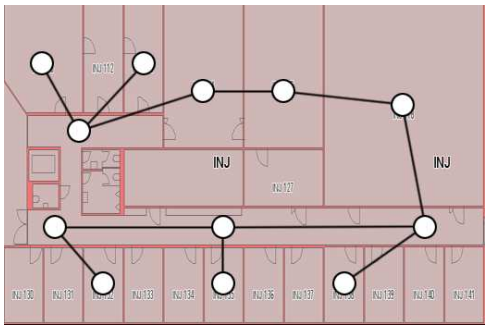

Figure 4. Example of a graph created with the WiFi editor during the training phase

navigation graph to ensure that the new position is realistic (close to the previous one), and then avoid the biggest mistakes due to signal noise.

6 Evaluation

First informal tests have been conducted internally with the help of our lab's staff. Then, we demonstrated our system at the annual review of the INTERMEDIA project in Madrid where we had the occasion to collect feedback from several users.

6.1 In vitro tests

For the first evaluation of our Bluetooth link, we manually paired the devices in order to avoid wasting additional time during the connection process. Even so, our software needs at least 6 seconds to detect the bluetooth device and two more seconds to connect to the Serial Port Profile (SPP) service in order to send the required information. Unfortunately, this timing cannot be reduced as it is a technical limitation of the bluesoleil stack. We also investigated the use of the more complete widcomm stack but inquiry process revealed itself even slower.

3D graphics rendering is a very computational expensive task that difficulty fit into the low resources offered by mini or wearable devices. Because of this aspect, rendering speed is a very important factor when deciding which platform/system as core unit for a wearable 3D framework including realtime computer graphics. Consequently, our graphics support has been tested through a performance benchmark among different platforms and devices in order to measure differences and limitations of each one. We used an unified benchmark application using three different models: the Stanford bunny (downloaded from Aim@Shape repository[6] as example of a complex single mesh object, a 3D building (see fig. 3) with many transparent walls to stress pixel fillrate, and a skinned character with a full HAnim[7] bones system. Each scene is lit by a dynamic omnidirectional light source.

[6] http://www.aim-at-shape.net/
[7] http://www.h-anim.org/

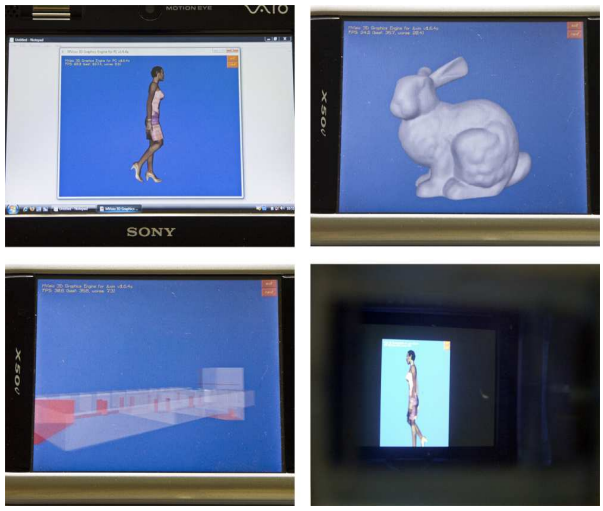

Figure 5. Benchmark overview

Table 1. Graphics benchmark results: average framerates

Device/Model	Bunny	Building	Character
Desktop PC	1280	1050	525
Vaio UX-70	36	160	63
Axim x50v HW	22	33	15
Axim x50v SW	6	9.3	7
HTC Polaris	4	6.7	4.5

The bunny model is made by a single mesh and counts 10680 vertices and 21348 faces. No textures are applied on the mesh. The building model is made by 27 meshes for a polycount of 506 vertices and 375 faces. No textures are applied but materials are semi-transparent and requires to be depth-sorted at each frame. The human character is a single mesh made by 3803 vertices and 6904 faces. The mesh is animated through a skeletal animation system with 78 bones. Each vertex position is computed as the resulting influence of up to 4 bones. The model is textured by a 256 by 256 pixel map.

We ran this benchmark on four different devices: a desktop PC (Intel Core2Quad 2.4 GHz, NVidia Gefore 8800 GT, Windows XP), a Sony Vaio UX-70 UMPC (Intel U1300 1.06 GHz, Intel 945GM, Windows Vista), a Dell Axim x50v PocketPC (ARM 624 MHz, Intel 2700G, Windows Mobile 5), and a HTC Polaris mobile phone (Qualcomm 7200 400 MHz, Windows Mobile 6). This test has been performed at a screen resolution of 640 by 480 pixels on the Desktop PC, Vaio UX-70, Axim x50v (hardware mode) and 320x240 on the HTC Polaris and Axim x50v in software mode. On the Axim x50v we performed the test twice using both the hardware 3D acceleration and software mode. Results are reported in table 1. To improve test robustness and comparison between different devices , we forced the use of OpenGL and OpenGL|ES fixed pipelines.

Considering requirements such as the adequate respon-

Table 2. First localization experiment result

Good	Near	Wrong
81%	15%	4%

Table 3. Second localization experiment result

Good	Near	Wrong
79%	19%	2%

siveness of the system and the mobility and size constraints imposed by wearable applications, Vaio UX-70 and Axim x50v appear as best candidates. Despite the very good ratio between the size and the results of the Axim x50v, we finally decided to adopt the slightly heavier Sony UMPC, as it has much better computational resources for a relatively bigger form factor.

The protocol applied for testing our system consisted on having different users walk around the test buildings using the UMPC. Users were asked to check the position shown in the UMPC and qualifying the system's accuracy as wrong, near or good. The good qualification was applied if the localization system shown the exact room or corridor part, the near qualification when the system indicated not the exact position but a juxtaposed position, and the wrong was for all other cases.

The system was tested in our laboratory. The surface is 817 square meters (43m19m), representing 18 offices of variable size. The floor is directly covered by 4 access points (2 Cisco Aironet 1200 and 2 NETGEAR WGR614), but we can acquire up to 7 different signals in some rooms (from access points placed on the nearest floors or buildings).

Table 2 shows the result of an experiment in the first building, our lab. This experiment just followed the off-line phase, which was done the same day, to ensure a good link between signal data acquired during the off-line phase and signal data acquired during the localization phase.

During these tests, we have noticed that the 15% good and 4% bad was mostly obtained in the corridor, because of the lack of important changes in the signal compared to pass a door for example. Another reason of having a good rate rather than a very good could come from the latency of the system : the system works with recent Wi-Fi data, not current, causing a latency of 1 to 3 seconds. These first results are quite satisfying for our goal, which was to have a room-sized precision system with a very low error rate.

One of the other point of our requirements was a short off-line phase. Last section explained why our system provides a short off-line phase in terms of time. But if the system needs to be retrained every day to keep its precision and accuracy capabilities, the off-line phase is not as short as we want. That's why we performed a second test in our lab, more than two weeks after the off-line phase, without any modification on the data.

Table 3 shows the result of this experiment : the accuracy doesn't change since the first experiment. This is a really satisfying result because fluctuating Wi-Fi signals over time is one of the biggest problem of Wi-Fi localization systems. These good results could be attributed to our precision : we are not confronted to the same problems with a room-sized precision system and with a 2 or 3 meters precision system. A system with a high precision is influenced by little changes in the signal (people moving around, opened and closed doors) whereas our system only detects important changes in the signal.

6.2 The Madrid demo

In this subsection we described the results we obtained by using our framework on a real-case demo in Madrid in October 2007.

In our latest configuration, one button could have multiple functions according to the context. Unfortunately, users reported that they needed to think few seconds before pressing the correct button. Indeed, users needed to check on the HMD which button is associated with the desired function. Hence, this configuration has been proved not very practical and intuitive as buttons are marked with fixed symbols on the jacket as seen on figure XX. An ideal solution will be discussed in section 7. On the whole, buttons are easy to press but unfortunately, tactile feedback is missing. One way to remediate this issue is to add audio feedback such as an audible "click" to inform the user that the button has been pressed.

Another issue that appeared at the Madrid ongoing demonstration is the magnetic sensitivity of the inertial sensor. Indeed, when approaching magnetic disturbances, the sensor behaves erratically. The 3D architectural model of the building isn't aligned with the reality anymore and thus the user loses his orientation. However, despite these issues, most users still managed to follow the virtual guide and reach the destination.

Results in Madrid were quite the same than our lab experiments. Indeed, the building is organized as our laboratory : mainly long corridors and small rooms. However, we noticed two things which disturb signal data.

The first thing were fire doors in corridors : during the signal acquisition phase, they were closed. But during our demo, people walked around and open these doors, and this way modify significantly signal data.

The second thing were the people around us during the demo. Indeed, to show our results, more than 10 persons were permanently standing next to the UMPC, and then altered the received signal strength values.

7 Conclusions and future work

For future work, we would like to improve the user interaction by embedding a tactile flat screen in the sleave of the jacket. Furthermore, we will develop a more modular approach towards wearable ubiquitous computing in a sense that the user will not have to decide between several garments according to their fixed respective functionali-

ties, but he/she will rather select and add modules that will suit his/her needs. Thus, the functionalities of this modular system become entirely personalized as they will depend on the selected wearable modules and the available surrounding devices. This approach will also be improved by using different core units in a scalable way, according to the context. For example we will keep the relatively heavy Sony UMPC for the full version, featuring all modules, but we could use mobile phones or PDAs combined with only a subset of the modules to create lighter frameworks. Furthermore, in terms of privacy, more questions need to be raised, such as using user preferences for determining the appropriate balance of personal information disclosure[16].

Acknowledgment

This work was performed as part of the Intermedia NoE[8], funded by the European Union as part of the Framework VI research program. The authors wish to express their gratitude to the members of the Intermedia Consortium for their valuable contributions.

References

[1] O. Amft, M. Lauffer, S. Ossevoort, F. Macaluso, P. Lukowicz, and G. Troster. Design of the QBIC wearable computing platform. *Application-Specific Systems, Architectures and Processors, 2004. Proceedings. 15th IEEE International Conference on*, pages 398–410, 2004.

[2] J. Astrain, J. Villadangos, J. Garitagoitia, J. de Mendívil, and V. Cholvi. Fuzzy location and tracking on wireless networks. *Proceedings of the international workshop on Mobility management and wireless access*, pages 84–91, 2006.

[3] P. Bahl and V. Padmanabhan. RADAR: an in-building RF-based user location and tracking system. *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2, 2000.

[4] P. Bahl, V. Padmanabhan, and A. Balachandran. Enhancements to the RADAR User Location and Tracking System. *Microsoft Research*, 2000.

[5] R. DeVaul, M. Sung, J. Gips, and A. Pentland. MIThril 2003: applications and architecture. *Wearable Computers, 2003. Proceedings. Seventh IEEE International Symposium on*, pages 4–11, 2003.

[6] F. eddine Ababsa and M. Mallem. Robust camera pose estimation using 2d fiducials tracking for real-time augmented reality systems. In *VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry*, pages 431–435, New York, NY, USA, 2004. ACM.

[7] W. Ho, A. Smailagic, D. P. Siewiorek, and C. Faloutsos. An adaptive two-phase approach to wifi location sensing. In *PERCOMW '06: Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops*, page 452, Washington, DC, USA, 2006. IEEE Computer Society.

[8] P. Honkamaa, J. Jäppinen, and C. Woodward. A lightweight approach for augmented reality on camera phones using 2d images to simulate 3d. In *MUM '07: Proceedings of the 6th international conference on Mobile and ubiquitous multimedia*, pages 155–159, New York, NY, USA, 2007. ACM.

[9] R. Kumar K., V. Apte, and Y. Powar. Improving the accuracy of wireless lan based location determination systems using kalman filter and multiple observers. *Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE*, 1:463–468, 0-0 0.

[10] N. Magnenat-Thalmann, A. Peternier, X. Righetti, M. Lim, G. Papagiannakis, T. Fragopoulos, K. Lambropoulou, P. Barsocchi, and D. Thalmann. A virtual 3d mobile guide in the intermedia project. In *The Visual Computer, International Journal of Computer Graphics*. Springer (to appear).

[11] G. Papagiannakis, G. Singh, and N. Magnenat-Thalmann. A survey of mobile and wireless technologies for augmented reality systems. *Comput. Animat. Virtual Worlds*, 19(1):3–22, 2008.

[12] A. Peternier, X. Righetti, M. Hopmann, D. Thalmann, M. Repettoy, G. Papagiannakis, P. Davy, M. Lim, N. Magnenat-Thalmann, P. Barsocchi, T. Fragopoulos, D. Serpanos, Y. Gialelis, and A. Kirykou. Chloe@university: an indoor, mobile mixed reality guidance system. In *VRST '07: Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pages 227–228, New York, NY, USA, 2007. ACM.

[13] A. Peternier, D. Thalmann, and F. Vexo. Mental vision: a computer graphics teaching platform. In *Edutainment 2006*, Technologies for E-Learning and Digital Entertainment. First International Conference, Edutainment 2006. Proceedings (Lecture Notes in Computer Science Vol.3942), pages 223–232, 2006. Virtual Reality Lab., Ecole Polytech. Fed. de Lausanne, Switzerland.

[14] A. Peternier, F. Vexo, and D. Thalmann. Wearable Mixed Reality System In Less Than 1 Pound. In *Proceedings of the 12th Eurographics Symposium on Virtual Environment*, 2006.

[15] G. Reitmayr and D. Schmalstieg. Location based applications for mobile augmented reality. In *AUIC '03: Proceedings of the Fourth Australasian user interface conference on User interfaces 2003*, pages 65–73, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[16] I. Smith, A. LaMarca, S. Consolvo, and P. Dourish. A social approach to privacy in location-enhanced computing. *Proceedings of the Workshop on Security and Privacy in Pervasive Computing, 2004*, 2004.

[17] D. Wagner, T. Pintaric, and D. Schmalstieg. The invisible train: a collaborative handheld augmented reality demonstrator. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Emerging technologies*, page 12, New York, NY, USA, 2004. ACM.

[18] D. Wagner and D. Schmalstieg. First steps towards handheld augmented reality. In *ISWC '03: Proceedings of the 7th IEEE International Symposium on Wearable Computers*, page 127, Washington, DC, USA, 2003. IEEE Computer Society.

[19] G. V. Zàruba, M. Huber, F. A. Kamangar, and I. Chlamtac. Indoor location tracking using rssi readings from a single wi-fi access point. *Wirel. Netw.*, 13(2):221–235, 2007.

[8] http://intermedia.miralab.unige.ch/