Institut Eurécom
2229, route des Crêtes
06900 Sophia-Antipolis, France

Research Report

RR-98-043

Perceptual User-Interface in a Media Space

Alain Karsenty

September 1st, 1998

Perceptual User-Interface in a Media Space

Alain Karsenty

Eurecom Institut

2229, route des Cretes

06904 Sophia-Antipolis

+33 - 4 93 00 26 63

karsenty@eurecom.fr

ABSTRACT

In the context of a MediaSpace, we describe a way to enhance the interaction between users through the use of a perceptual user-interface. The principle is to use agents that are aware of the user's physical state (e.g. is the user typing on the keyboard, meeting with other people, on the phone, etc.) in order to detect presence, and to use this information to improve connectivity through the MediaSpace. We will first describe an application we developed on top of a MediaSpace, EasyMeeting, based on user-aware agents. Then, we will present the implementation (multi-agent architecture, language). Finally we will discuss the different type of agents as well as privacy issues. We believe the perceptual user-interface is a step toward a better communication man-machine. Instead of the usual approach in which users consciously interact with the machine, we make the computer aware of the users and thus make users unconsciously interact with the computer.

KEYWORDS: groupware, Computer-Supported Cooperative Work, Computer-Human Interaction, Intelligent Agents, Media Spaces.

INTRODUCTION

The keyboard and mouse are today's computers main source of input. Considering the human capabilities and the technology available this is quite limiting of what could be done to improve human-computer communication. In this paper, we propose to add as an input source to the computer, knowledge of the user's presence in the computer office. This is achieved by providing an "eye" and an "ear" that informs the computer about what is happening at a particular place, thus allowing the computer to perceive its environment, hence the term " perceptual user-interface ". As opposed to the traditional (mouse, keyboard devices) and less traditional (multimodal interfaces, speech recognition, gesture based interface...) input sources, this input is of a passive form since the computer is aware of the user's actions whereas the user is not aware of the computer listening to his actions. We chose a passive input approach since we did not want to add a burden to the user (e.g. carry badges, fill up information in a file...). This way, we hope the user will benefit from the system without having extra-work to do, which avoids a failure of groupware systems pointed out before [9]. Such an approach to human-computer interaction is sometime referred to indirect manipulation [12].

We call the agents that listen to the user's physical activity user-aware agents. To illustrate the use of such agents, we can imagine a simple screen-saver utility that would benefit from it. Whenever the user is not working in front of the computer, a user-aware agent informs the computer which turns the screen-saver on and locks the screen. As soon as the user is back in front of the computer, the user-aware agents recognize the user and the screen automatically unlocks itself. However, this work focuses more on the CSCW (Computer-Supported Cooperative Work) field. Indeed, knowledge of the user's state can be quite valuable in CSCW to help improve distant communication between users. We investigated the use of such agents in a MediaSpace (multimedia system allowing co-workers to communicate through audio/video) with the purpose to make meetings between co-workers easier through the concept of a perceptual user-interface. The application we developed is called EasyMeeting, and allows to easily set up meetings between users distantly located. When a meeting is requested, the computer connects the users whenever possible (e.g. they are in front of the computer, not talking on the telephone, etc.).

We will first describe EasyMeeting, the application to a MediaSpace, the implementation and in particular the architecture, and finally we will discuss the use of agents, and privacy issues.

RELATED WORK

Both the VideoDesk [11] and the DigitalDesk [25] link the real world to the computer world, by enabling the computer to "see" (video analysis). In the VideoDesk, a camera is mounted on top of the physical desk and analyzes the scene. It can for instance recognize the text written on a paper (using Optical Character Recognition techniques) and paste it into an application. The VideoDesk recognizes the user's hand movements in order to manipulate objects on the screen. Such work, however, focuses on single-user application and active input, whereas our work focuses on CSCW and passive input.

Multi-agent systems [14, 16] are also related to our work except that our focus is more on Computer-human interaction.

Software agents [12] share a similar approach to our work. The notion of indirect manipulation is close to the idea of perceptual user-interface, but they do not apply it to a MediaSpace, and focus their research on Artificial Intelligence.

On the MediaSpace aspect of our work, many systems exist that study the interaction through video [8, 3, 1, 21, 23, 15, 2]. For instance, Portholes [6] uses the MediaSpace in an asynchronous way to develop group awareness. Snapshots from people's office are taken at a regular interval and displayed as a background mosaic picture. Thus, co-workers unconsciously develop a stronger feeling of co-presence. Portholes' approach is different from EasyMeeting in the sense that it brings to the user the knowledge of who's in the office and who's not but does not bring this knowledge to the computer. This could have been done, for instance, by some image processing on the mosaic. In a similar fashion, VideoWindow [8] connects two distant rooms through a large video-display, thus improving communication between distant sites.

Finally the concept of active office [10] and reactive environment [4] are the most closely related to our work. The active office builds application based on knowledge of the location of people wearing active badges. Our systems differs in the sense that we do not want users to wear any device, and we also have different sort of agents, not only location ones. In the reactive environment a meeting room is equipped with sensors to automatically perform the various tasks. The user-aware agents approach is more generic in that it provides a general architecture for such systems and our experience deals more with interaction between users than with room equipment.

EASYMEETING: PERCEPTUAL USER-INTERFACE IN A MEDIASPACE

First, we describe an example of an office situation, that can be improved through the use of a perceptual user-interface. Part of every day's office work often consists in informal/formal meetings between co-workers. However, those meetings, especially informal, do not happen so easily. As an example, let's consider John who wants to meet Sally. John dials Sally's phone number and gets a busy tone. A moment later, John passes by Sally's office, to find the office closed. John then leaves a note asking Sally to get back in touch with him. When Sally returns, the inverse scenario happens where John is out of the office making it difficult for her to get back to him. This scenario can happen indefinitely! Although reality is not so negative, people in offices still spend a lot of time running after each other. To improve communication, electronic mail is often employed. It's asynchronous, therefore allows the exchange of information to be made much easier. However, many situations need to be discussed in face-to-face meetings, or at least through telephone or video. For this purpose we developed EasyMeeting, an application that allows to easily meet through a MediaSpace.

Definitions

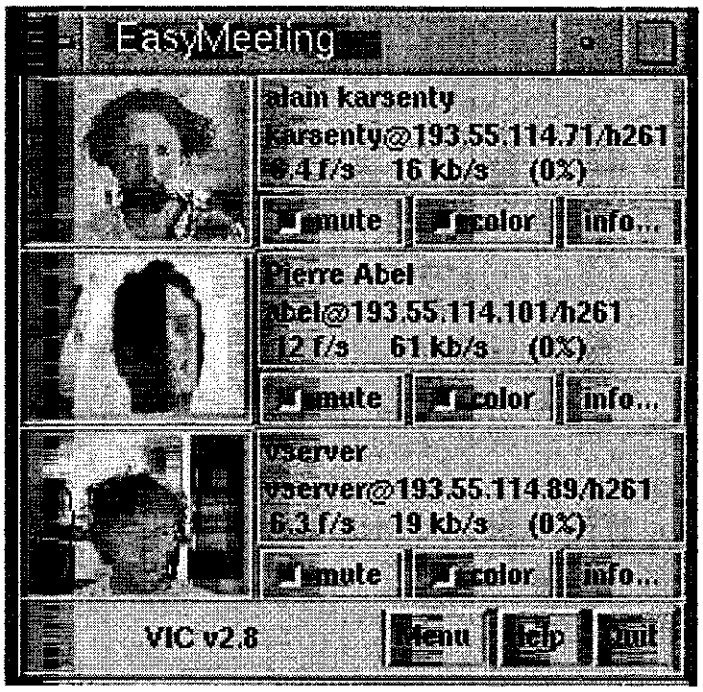
In the next sections, we will provide definitions of the concepts used throughout this paper, then we will describe the EasyMeeting application.

MediaSpace

A MediaSpace is a system that integrates video/audio and computer networking technology in order to provide a rich cooperative environment. A number of systems have been built at different sites [8, 3, 1, 21, 23, 15]. They range from simple systems similar to videophones, which connect users through audio/video to more research oriented prototypes. The latter kind looks for more original use of audio/video than simply a telephone with video. Our research focuses on this direction by using user-aware agents to enhance the interaction through the MediaSpace.

Our first version was analog. The advantage of such a MediaSpace is that the audio/video quality is excellent, and the technology old enough to be reliable. However we are now using a digital MediaSpace, based on the mbone, using the existing vic/vat tools. The advantage is that it is much easier to maintain (e.g. when users change offices, there is no wire to set up, since everything is transmitted through the internet), it can be used worldwide and a large group meeting is as easy to set up as a two-way communication since broadcasting is part of the mbone.

An example of a MediaSpace session between three users is shown on Figure 1, it uses the mbone tool called vic to establish the video link.



alain karsenty
karsenty@193.55.114.71/h261
8.4 f/s 16 kb/s (0%)

Pierre Abel
abel@193.55.114.101/h261
12 f/s 61 kb/s (0%)

vserver
vserver@193.55.114.89/h261
6.3 f/s 19 kb/s (0%)

Figure 1: meeting between three users on the mbone.

User-aware agents
The purpose of user-aware agents is to collect as much information as possible about the users' physical activity. User-aware agents, are software/hardware that provide this information. Such agents are aware of the user's physical state, e.g. who the user is, what he/she is doing, where he/she is, etc. The way to get such information can be very diverse: infra-red motion detectors, video analysis, pressure captors under the chair, cell sensitive to light, badges. User-aware agents are low-level agent in the sense that they provide raw data. For instance, they can detect motion, or change of light but do not provide higher-level information such as a user is present or not.

Perceptual user-interface
A perceptual user-interface is an interface based on the user-aware agents. The computer perceives the surrounding environment, therefore obtains information about the users, in order to enrich the human-computer interaction.

Features
Based on those previous concepts of user-aware agents and perceptual user-interface, we now describe the features of theEasyMeeting application.

Meeting
EasyMeeting's most basic feature allows a user to create a meeting. The user specifies a group of people that will meet in between two dates t_1 and t_2, as well as the private reason of the meeting, which will only be shown to the initiator of the meeting, and the public reason which will be shown to all participants. When time t_1 is reached, the system starts the agents for all participants, and waits for their state to be present. If all users' state is present before t_2, the system displays for each participant a dialog box asking them if they want to participate in the meeting. Users can accept, reject, or postpone

to later. If all agree, an audio/video connection is then established between all participants and the agents are back to an idle state. If time is passed t_2, the initiator of the meeting can optionally postpone the meeting or cancel it.

System feedback
Those functionalities allows the user to know the current status of agents and meetings. The " list meetings " functionality shows the user a list of meetings that are scheduled. Users can delete or modify those planned meetings if they wish to. The " agent status " functionality shows the agents available on the system, and if they are currently active.

Access control
It has been shown that access control is an important aspect of CSCW [17] and MediaSpaces in particular [7,18]. We cannot let a few agents decide everything, it is obvious that we must provide users with a means to control what other people see and when they can access it. Control to what people see is provided with the mirror mechanism, whereas control to when people can see is provided by the door mechanism (see CAVECAT [13]).

The mirror provides a video feed-back. It allows one to see his/her own image on the monitor in order to be aware of the image seen by the other users. Users often employ the mirror function to center their image.

It is also possible to control one's availability through the door metaphor. User can set their door to the open or closed state. When the door is closed, users are not available for meetings, whereas an open door means they can participate in a meeting. Optionally, when closing the door, they can specify a message that will be provided to the other users. This is similar to the post-it note that we stick to the door when going out of the office, indicating the place were we can be found. We plan to extend the current version with a " door by users " mechanism. Users can open doors for certain users, and close it for others, thus giving more control to the users.

Experience showed an unexpected way for users to control access to their environment, which is to simply close the cap of the camera. This is a hardware features that ensures privacy to the user. EasyMeeting is not aware of the state of the camera cap, thus still try to use audio/video agents which can result on erroneous data depending on what the audio camera generates when the cap is closed. One way to solve this would be to analyze the image and detect whether the cap is closed or not, and link it to the door state: a closed camera cap could mean a closed door state. Another option would be to ask the user after closing the cap if he/she also wants the door state to switch to closed, otherwise we only use the keyboard agent to detect presence. User-testing will be needed to find the best suited solution.

IMPLEMENTATION
To manage such a distributed systems we chose to use a hierarchical multi-agent architecture and to implement the communication between agents with TCP (tcl currently allows communication using TCP). The interface was implemented in Tcl-tk. Tcl-tk and its various extensions allowed us to quickly implement a prototype that is easy to modify. The multi-agent architecture allowed us to quickly add/remove user-aware agents and to clearly separate the modules of the system. The agents are implemented both on Sun workstations and Silicon Graphics, thus the system can run on a heterogeneous system since tcl/tk runs on both.

In this section we describe the implementation first from the logical architecture point of view, then from physical one.

Agent-based architecture

The architecture is based on a number of agents that collect information about users' activity in order to connect them at the best appropriate time. Given the heterogeneous nature of the agents, it seemed suited to organize them in a hierarchical way. We thus based our architecture on three kind of agents: lower-level agents (user-aware agents) that collect raw data, higher-level agents (Intelligent Agent) that analyze the information provided by the various user-aware agents and the server agent (i.e. the "brain" of the system) which collects information from the Intelligent Agents in order to make connections between users. We describe this architecture in Figure 2. As we see, information flows from raw data provided by the user-aware agents, to user data provided by the intelligent agent and finally group data stored in the server.

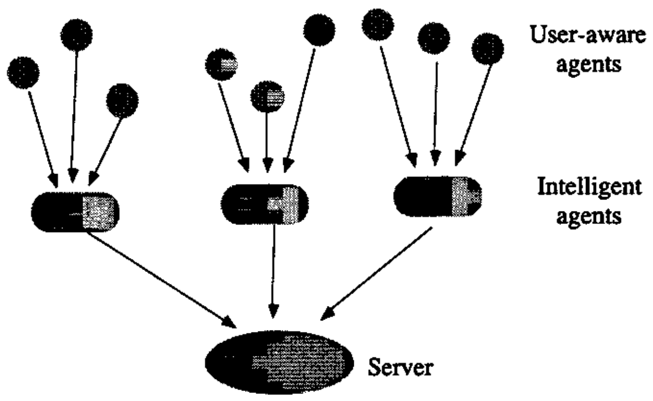


Figure 2: Logical architecture

User-aware agent

In our case, we decided to use what we had available, i.e. the video camera that is part of the MediaSpace and the keyboard/mouse attached to each user's computer. The information thus provided is motion detection, audio activity and keyboard/mouse activity. Motion detection is done via a fairly simple program that analyzes digitized pictures taken from the camera and consider that motion happened whenever the pixel sum value between pictures is higher than a given threshold. The audio agent consider audio activity when the audio volume goes beyond a given threshold. The keyboard/mouse activity is done via a Xlib program that intercepts keyboard and mouse events. Those agents are enough to detect fairly accurately whether a user is at his desk or not. Indeed, if a user is typing, he/she is rather motionless, but is still detected, when talking, audio is detected, and when not typing text, the user is often in motion (unless reading/writing on the physical desk).

Intelligent agent

The Intelligent Agent is the one that centralizes the information from the user-aware agents in order to infer higher-level information. For instance, in the previous example, an intelligent agent attached to three user-aware agents (keyboard activity, audio activity and motion detection) will infer that a given user is present or not, whether this information is provided by the keyboard activity agent or the motion detection agent. Intelligent Agents' knowledge are only about a particular physical place which is usually an office. Therefore, they collect user data and don't have any knowledge about the group. The Intelligent Agent has knowledge also about the reliability of agents. In the case of audio or motion detection agents, it does not take into account one time event. For instance, a door banging will generate only one audio event, or a change of luminosity (light switched on, cloud covering the sun) will also generate only one video event. Those one-time event are often not significant, and should be discarded to improve the reliability of the system.

Server
The server has global knowledge of the system. It is the one that CSCW applications will interact with. As an example, if we want to connect user A and B together, the server knows which computer they are usually logged on and will send a query to the appropriate Intelligent Agents. Knowing A and B's respective status, the server will guess whether or not to connect them.

Language
The system has been implemented using mainly Tcl-tk for the interface, TCP (part of tcl) for communication between processes, and C languages for lower-level routines (image analysis, keys activity agent...). We show in Figure 3 the main components of the physical architecture and the languages used.

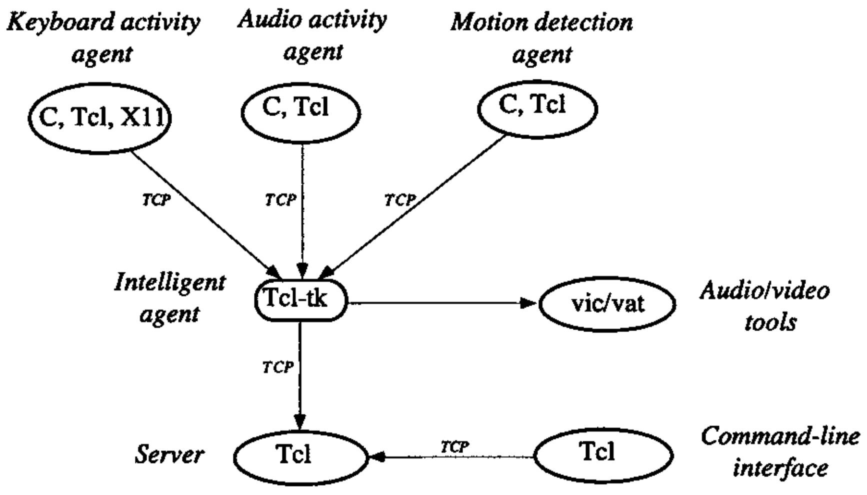


Figure 3: Physical architecture

For each user's workstation we had three process running. The keyboard activity agent implemented in C/X11 scans all the windows and reports when keys are typed. The motion detection is implemented in C and uses the sun Xil library to do real-time motion detection. The algorithm is very basic in order to run fast, pixel difference between successive pictures is calculated, after a set threshold, the agent considers that there is motion and communicate the event to the intelligent agent. The audio agent is implemented in C. The Intelligent Agent is implemented in Tcl-tk in order to communicate both to the agents and to the server. In fact, it could have been implemented in tcl, but it needs to launch vic/vat and control the interface, which requires a function part of tk. The server is implemented in Tcl. It communicates both to the EasyMeeting command-line interface and to the intelligent agents, and makes call to the mbone tools to establish the MediaSpace connections. Finally, the EasyMeeting command-line interface is implemented in Tcl and communicates with the server.

User-testing: statistical analysis

In order to evaluate how reliable the agents are to detect the user's presence, we conducted a few tests. We gave a user a simple tool with two buttons " Out " and " In ", and told the user to click on " Out " when out of the office, and " In " when in the office. This way, we recorded in a file the " real " state of the user. In the meantime we had the agents running and loggin in a file every 10 seconds the state of the user. Two set of agents were used in parallel:

- set A : audio/video detection and keyboard activity.
- set B : keyboard activity.

This was done in order to evaluate how much more information the audio/video agents add to the system in comparison to a keyboard agent alone.

We used to criteria:

- Sa = Success on absence : number of time where agent detects absence successfully.
- Sp = Success on presence: number of time were agent detects presence successfully.

Which gave the following results:

	Sa	**Sp**
set A (all agents)	69.7%	70.2 %
set B (keyboard)	83.2 %	50.3 %

The criteria Sp is the one that improves the chance of successful meeting. Using all agents (set A) we obtain a reasonable percentage. Using only the keyboard agents, the result drop to 50%, which shows that audio/video agents are quite useful.

The criteria Sa show the system's ability to detect absence. Both set A and B show good results, however we remark that the keyboard alone yields better result than using all agents. This is due to the fact that the keyboard is very reliable at detecting absence, since it is rarely the case that someone will use someone else's keyboard/mouse.

Regarding using set A versus set B, we can say that set A is more appropriate to the needs, since it improves the chances to meet, but using set B has the advantage of being easy to implement and more reliable in the case of absence. We must also outline that criteria Sa is not fundamental since when a connection is being established, the system always asks the users, through a dialog box, if they agree to meet, thus if the user is not present, the system will still be reliable.

DISCUSSION

In this last section we discuss two issues, one technical, the pros and cons of various user-aware agents, and the second social, namely the " big brother issue ".

What kind of user-aware agents?

There is many ways to get information about a user's state. Various hardware/software alternatives can be combined to obtain the best result. So far, EasyMeeting uses keyboard/mouse activity and motion detection. It turned out to be enough for simple cases, when users always log on the same computer and do not want to be contacted in other places. However, one of EasyMeeting's initial goal was to make connections available anywhere in the office. This goal cannot be achieved without some sophisticated software such as face-recognition, which are not very reliable and do not provide real-time recognition. In the following, we discuss different alternatives and the pros and cons of each one.

Motion detection (video)

This is a simple efficient captors that can detect any motion in real-time. Simple algorithms provide reliable results. However, the result is very limited, we cannot know who is moving, or how many users are moving.

Face recognition (video)

Face-recognition is a very active field of research [5, 22], however no real-time efficient algorithm exists for this complex problem. This is the reason why we haven't used it. User identification is of crucial importance for the system, since this is the feature that allows one to be contacted wherever he is. The most reliable solution is probably badges however we do not want at this point users to wear any special device.

Scene analysis (video)

This is similar to face recognition but more generic. We may want to know, for instance, the user's location in the office, how many users are present, etc. Except for a few simple cases, it cannot be achieved in real-time, which might not always be a problem. In the case where we want to know the number of users present in the office (i.e. trying to find out if the user is busy holding a meeting), we can tolerate a few minutes delay.

Speech recognition

Such agent has the interesting real-time feature. However, the drawback, is from the human-computer interaction side. One has to speak to be identified, which goes against the passive approach of our system. However, such agent could be used as a complement to other captors.

Telephone activity

This captors can easily be implemented and provide very reliable information. In fact, it could also be coupled with the previous speech recognition module. Such agent can tell the user is busy on the telephone, and moreover knows who the user is calling (using caller Id). With such information, the interaction could be customized. For instance, if a co-worker wants to meet user B, and user B is on the telephone talking to a friend, the system could be customized in order to authorize the agent to make a video connection, despite the conversation on the telephone. Or, user B may customize the system not to be interrupted when talking to customers.

Keyboard/Mouse activity

The simplest form of monitoring is the keyboard/mouse activity. Such activity informs the server that a user is present in front of the computer. Information provided by this agent is however partial: a user might be in front of the computer simply reading, the software agent will not detect any activity.

Software agents
By software agents, we mean the many software that run on a machine which can provide useful information about the user. For instance, if a user runs a calendar manager application, the server can extract from it useful information (e.g. the user is out of the office this afternoon, etc.). The screen lock on signifies that a user is probably not at his desk. If user logs on somebody else's display, his/her login can be detected making it easy to detect where the user is located. Finally, EasyMeeting itself can provide useful information, since it monitors who's meeting who at different time. From an implementation point of view, the software agents need to communicate with the server, which means to either modify the applications (e.g. implement a custom shell, calendar manager, etc.) or, if possible, use existing API (Application Programming Interface) to get the information.

	presence	*who*	*how many*	*Real-time*	*what*
Speech recognition	*	*		*	
Motion detection	*			*	
Face recognition		*			
Scene analysis			*		*
Telephone activity	*			*	*
Keyboard activity	*			*	
Software activity	*	*		*	*

Figure 4: Pros and cons of various user-aware agents

We summarize the pros and cons of the various user-aware agents in Figure 4. We gave five criteria:

- presence: the agent can detect someone's presence
- who: the agent is able to identify the user.
- how many: detect the number of users present.
- real-time: the agent is able to provide the information in real-time.
- what: the agent is able to detect what the user is doing (e.g. the user is writing at his desk, on the telephone, etc.)

As a conclusion, it is obvious that there is no one all powerful agent, but each with a specific advantage. A good system will combine in the most clever way many agents in order to build a reliable system.

Privacy issue
A fundamental issue to the whole system, is the "Big Brother" issue. Agents scattered through the office, monitoring the actions of the users reminds too much of a telesurveillance system...We ought to design a system that users will trust. Nobody will use a system where it is possible to spy on everyone's actions.

To overcome the big brother issue, we present a number of rules we followed during the design and implementation of the system.

No access to other users' state data
One of the fundamental design of the system is the impossibility for a user to know what another user is doing. When requesting a connection with someone, if the user cannot be reached, the reason why is completely hidden to other users. If one wants to find out the reason he/she cannot get in touch with someone, traditional methods needs to be employed. For instance, one should ask the secretary if the college is out of the office, or go physically to the college's office.

To illustrate how EasyMeeting tries to respect this principle, let's take the example of user A who requests to meet with user B. When the system detects that B is present, we could present both users with the dialog box requesting if they are ready to meet, however, if the meeting fails, the user A initiator of the meeting would then obtain the information that user B doesn't want to meet with him, or if the meeting is canceled after a time out, user A would know that user B is not in the office. To avoid that someone would use EasyMeeting as a way to " spy ", the system asks first user B if he/she is ready for the meeting, and if yes, it asks the initiator of the meeting.

We would like to strike the difference with other systems such as Portholes. They aim at developing group awareness, a sense of co-presence, by explicitly making users graphically co-present, whereas our aim, in the case of EasyMeeting, is to ease the interaction through the MediaSpace. Therefore, privacy can and should be respected.

Evanescent data
This is the computer aspect of privacy. Data, such as who is where, doing what, should not be stored in a file, unless necessary. For instance, when a connection is requested, the server will simply ask the various agents to find the user. Whenever the connection is established, there is no need to keep the information about the user.

By doing so, we ensure that in the design of the system itself, there is no way to break privacy rules. We therefore make this technology more trustable.

Ubiquitous/invisible agents
From a hardware point of view, the agents can be numerous and scattered through the building. In order to make the users not feel "watched", ubiquitous agents should be made part of the environment, thus invisible. In that sense we move toward Weiser's view of ubiquitous computing [24].

In many analog MediaSpace, a node consists of a monitor, on top of it a video camera, a microphone on the table or attached to the camera, and next to it the workstation. The whole system takes up a lot of physical space, and is far from being invisible...In our case, using a digital MediaSpace we only have a small camera on top of the workstation, which is already an improvement although we can argue about the choice to centralize all communication through the computer screen. Systems such as Hydra [19] provide interesting alternatives to be used in MediaSpaces: a hydra unit consists of a small box made of a small camera and a mini-display, which can be easily moved around. A meeting consist in using many hydra units next to each other.

Control over the agents
Users should be able to know what agents are on the computer, if they are running, and basically be able

to control them. This reduces the feeling of being spied, since one has more control over the " spies ". This is achieved in EasyMeeting with two methods. First, the agents are processes which are run by each users, not by another user or a super-user. This allows the users to kill the processes if they want, and feel that those agents belong to them. Second, the users are provided with an interface showing which agents are available and running, thus avoiding the feeling that something is going on they don't know about.

When You See Is When I See

We can derive from the acronym WYSIWIS (What You See Is What I See) often used in groupware [20] the same acronym but with a different meaning : When You See Is When I See. A WYSIWIS MediaSpace respects privacy, since one cannot see without being seen. One-way connections, which would allow someone to spy in somebody else's office, are thus not possible. However the WYSIWIS concept can be extended in many other way. For instance, we can extend it in the time dimension, allowing both synchronous and asynchronous communication. In the case of the glance feature, WYSIWIS can be applied for face-to-face as well as digital communication. When someone glances in a college's office, a snapshot may be taken to be replayed later. In a similar fashion, an agent could detect when someone physically glance in a college's office and take a snapshot of the user peeking through the door. When the college comes back to the office, snapshots can be replayed, thus the college can find out who is trying to reach him/her. This way we also respect privacy, since a user cannot use the glance feature to "spy" in somebody else's office.

However, experiences shows that ensuring a WYSIWIS MediaSpace is not easy. As an example, when using the analog MediaSpace, there was a case where a user had shutdown the monitor. At some point a two way connections had been established, however the user did not know that he/she was being watched since both the audio and the video would not go through the monitor. This case is resolved though, since we now use a confirmation dialog box before establishing any audio/video connection.

CONCLUSION

Human-computer interaction research is focused on active mode of communications, in which the user is aware when communicating to the computer. In this paper we described what we call a passive approach to human-computer interaction, in which the user is not aware to communicate to the computer. On the contrary, it is the computer that is made aware of the user, hence the term " perceptual user-interface ". On this principle, we described EasyMeeting, a digital MediaSpace improving connectivity among users.

This work is continuing in different directions.

First, we need to further experiment. This is still a prototype, and we are currently making it more robust. Experimentation will allow us to validate some of the ideas discussed in this paper.

Second, we plan to add more captors than keyboard/mouse activity and audio/motion detection. The most important one is user identification. Even if face recognition is not reliable yet, it will still be useful to do some basic experiments.

Finally, we are interested to apply the user-aware agents architecture to other applications, CSCW or single-user. Many applications can take advantage of this concept (e.g. the screen-saver described earlier, teleteaching, shared editing, etc.) and this will also help us refine our architecture.

ACKNOWLEDGMENTS
We would like to thank Armand Sanier and Fréderic Veynachter who greatly contributed to the implementation of the prototypes. Pascal Gros for the exchange of thoughts and the help in setting up the MediaSpace. Bernard Merialdo for the useful discussion. Nabil Madrane for helping implementing the motion detection agent. And finally the many people who participated in testing the system and gave useful feedback, Jean-Michel Sadoul, Katia Fintzel, Florence Dubois and Khaled Boussetta.

REFERENCES

1. Buxton, W. and Moran, T., "Europarc's Integrated Interactive Intermedia Facility (IIIF): Early Experiences," in Multi-User Interfaces and Applications, Gibbs, S. and Verrijn-Stuart, A.A., Eds. North Holland, 1990, pp. 181-188.
2. Buxton, W., "The three mirrors of interaction : A holistic approach to user interfaces." In L. W. MacDonald & J. Vince, editors, Interacting with Virtual Environments. Wiley: NY, 1994.
3. Sarah A. Bly, Steve R. Harrison, and Susan Irwin, "Media Spaces: Bringing People Together in a Video, Audio and Computing Environment", *Communications of the ACM*, 36(1), p.28-47, January 1993.
4. Cooperstock, J. R., Tanikoshi, K., Beirne, G., Narine, T., Buxton, W., "Evolution of a Reactive Environment", In Communications of CHI'95, 1995.
5. Davies, E. and Shepherd (eds), "Perceiving and Remembering Faces", Academic Press, London, 1981.
6. Paul Dourish, Annette Adler, Victoria Bellotti, and Austin Henderson, "Your Place or Mine? Learning from Long-Term Use of Video Communication", Europarc, Technical Report EPC-94-105, EuroPARC, Cambridge 1994.
7. Paul Dourish, "Culture and Control in a Media Space", in Proceedings of ECSCW'93.
8. Fish, R. S., Kraut, R. E., Chalfonte, B. L. "The VIdeoWindow System in Informal Communications", in Proceedings CSCW'90, Oct. 7-10, 1990, pp.1-11.
9. Grudin, J, "Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces," in Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW'88), Portland, OR, September 26-29, 1988, pp. 85-93.
10. Harter, A, Hopper, A, "A Distributed Location System for the Active Office", In IEEE Network, vol. 8, n.1, pp. 62-70, Jan/Feb 1994..
11. Krueger, Myron, W. Artificial Reality. Reading: Addison-Wesley, 1993.
12. Pattie Maes, « Humanizing the Global Computer », in IEEE Internet Computing, July-August 1997.
13. Mantei, M., Baecker, R., Sellen, A., Buxton, W., and Milligan, T., "Experience in the Use of a Media Space," in CHI'95, 1991, pp. 203-208.
14. Martial, F. V., "Coordinating plans of autonomous agents. LNAI 610, Springer-Verlag: Berlin.

15. Margrethe H. Olson and Sara A. Bly. "The Portland experience: A report on a distributed research group." International Journal of Man Machine Studies, 34(2):211-228, 1991.

16. Rizzo, P., Cesta, A., and Miceli, M., "On Helping Behavior in Cooperative Environments", in International Workshop on the Design of Cooperative Systems, June 1995.

17. Tom Rodden, Gordon Blair, "CSCW and Distributed Systems: The Problem of Control" in Proceedings of the Second European Conference on Computer Supported Cooperative-Work (ECSCW'91), September 25-27, 1991, Amsterdam.

18. Salber, D. and Coutaz, J., "Fenetres sur groupe: des mediaspaces pour collaborer et communiquer." In Proceedings of Interface des mondes reels et virtuels, Montpellier, pp. 309-318, Feb. 1994.

19. Sellen, A., Buxton, W. and Arnott, J. "Using spatial cues to improve videoconferencing" in Proceedings of CHI'92, pp. 651-652, 1992.

20. Stefik, M., Bobrow, D.G., Foster, G., Lanning, S., and Tatar, D., "WYSIWIS Revisited: Early Experiences with Multiuser Interfaces," ACM Trans. on Office Information Systems, vol. 5, no. 2, 147D167, April 1987.

21. Thomas, R. H., Forsdick, H. C., Crowley, T. R., Schaaf, R. W., Tomlinson, R. S., Travers, V. M., Robertson, and G. G. "Diamond: A multimedia message system built on a distributed architecture." In IEEE Computer, volume 18, pages 65-78. 1985. Reprinted in Greif, 1988.

22. Turk, M. A. and Pentland, A. P., "Face Recognition Using Eigenfaces" in Proceedings IEEE, 1991.

23. Kazuo Watabe, Shiro Sakata, Kazutoshi Maeno, Hideyuki Fukuoka, and Toyoko Ohmori. "Distributed multiparty desktop conferencing system: Mermaid." pages 27-38. ACM Press, New York, NY, October 1990.

24. Weiser, M., "The computer for the 21st century." Scientific American, pp. 66-75, Sept. 1991.

25. Wellner, P.. "Interacting with paper on the DigitalDesk". In Communications of the ACM, July 1993, vol. 36, no 7, pp. 87-97.