Blending Games, Multimedia and Reality

Chris GauthierDickey
Department of Computer Science
University of Denver
chrisg@cs.du.edu

ABSTRACT

Games, multimedia, and reality are on a converging path as motion capture, 3d video, and high performance networking and computing become more affordable and common place with consumers. Next generation video game systems are planning on including many aspects of motion capture and facial recognition algorithms. As such, we present an analysis of system issues surrounding the amalgamation of these technologies in the future and then propose a system architecture where augmented reality can become a part of our every day learning, computation, and gaming. We conclude with a discussion of the advantages and disadvantages of our design.

Categories and Subject Descriptors

H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Artificial, augmented, and virtual realities*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms

Design, Human Factors

1. INTRODUCTION

As game and multimedia experiences have improved, they have become increasingly used on a daily basis by consumers as a source of entertainment, communication, training and education. Given that the next generation of devices will likely include advances such as 3D video, audio capturing, motion capture, and wearable displays, we need to consider how an architecture for these future multimedia systems should be designed so that it effectively takes advantage of its resources while providing the services needed for future games and multimedia experiences.

To date, most of the multimedia research has been concerned with allowing individual devices to provide one or more of these services. We are instead interested in a holistic approach to allowing mass interaction and interoperability between games, multimedia and reality in a unified architecture. In this architecture, users can interact with each other in physical spaces, whether or not the user is actually physically present, but it requires sensors capable of tracking multiple targets, singling out voices in a crowd, and wearable displays for maximum efficacy.

The advantages of an architecture which provides this type of connectivity are many-fold. If we can network motion capture (*mocap*) devices that are capable of tracking multiple targets, then we can mocap a space such as an entire building floor. This in turn gives the people in the building the ability to interact with virtual objects (via wearable or fixed displays) and other people, whether it be for collaboration or entertainment purposes.

We present an architecture which takes into account recent research into areas of games, multimedia, and augmented reality. Our assumption is that users will:

1. Expect mobility: users are already used to phones and PDAs that allow them to stream audio and video, play games, and run applications.

2. Need to stream data: given user mobility, we expect that we will need to be able to stream data to appropriate resources for processing, whether it is a CPU, a GPU, or a cluster for computation.

3. Need new input methods: carrying around a mouse and/or keyboard for computer interactions is extremely both cumbersome and error prone when shrunk for mobility purposes. Motion capture, on the other hand, would allow us to recognize gestures and give users virtual input devices appropriate to their application. Further, we expect multi-user interactions from these input devices.

4. Need to capture audio/video: we expect resolution and bandwidth to increase as users desire more fidelity and we capture additional information such as that needed by 3D video. Further, we expect users will need/want to be able to stream the data from their capturing devices to other devices.

These requirements present some multimedia research hurdles to be overcome and include multi-target, full-body motion capture, computer vision, targeted listening and parsing of voices within crowds, and gesture recognition. Within the context of our architecture, we discuss these research issues.

We note that the architecture described in this paper is one possible way of organization that would allow for the incremental improvement of sensors, output devices, and computational infrastructures as it is interconnected by a stream processing language. As new hardware is added, data streams can be redirected to appropriate resources with little reconfiguration through the language. We conclude with a discussion of the possibilities and advantages and drawbacks of such an architecture and design.

The contribution of this paper is an architectural design for this type of multimedia system that includes authentication, processing and interactions. Further, we identify important areas of research for the realization of this system and we also identify important drawbacks that must be considered for future multimedia systems that track users with the detail required by the desired types of interactions within our system.

2. BACKGROUND AND RELATED WORK

The background for this work really encompasses a large portion of multimedia research, especially over the last several years, which has been focused on improved stream processing and data streaming methods, and improved multimedia interfaces. In this section, we cover some of the more recent work that is related to our architecture here. This section is not meant to be a survey on all work, but just touches on those areas closest to our architecture. We do note that in particular, much of the multimedia architectural work has been recently focused on P2P streaming of video and audio. While this is useful for internet-scale transmissions, it is less useful for our architecture which, while distributed, is still local to a particular physical *space*.

Capra et al. discuss methods for interacting with a fictional system similar to the one we present here [4]. Their methods are interesting because they allow users to interact with sensors via requests, sending data streams to recording or mobile devices. Our architecture would support such interactions by setting permissions on authenticated users allowing the recording of sensor streams.

IrisNet, by Campbell et al., is an architecture where multimedia sensors are attached to significant computing sources which can store and preprocess feeds and then allow queries on the stored data by the network [3]. This architecture is interesting because it specifically tried to address the issue that multimedia sensors collect large streams of data compared to typical mote-sensors. However, it differs from our proposal in that it is concerned primarily with sensors and not interactions between users as ours is.

Wu et al. describe developing a theoretical framework for understanding the quality of the user experience in a system such as ours [10]. This framework uses models of the system and sensors to predict how well it will both track and monitor participants. A framework such as this could be used to evaluate our architecture prior to implementation.

Bernardin and Stiefelhagen developed an audio/visual multi-person tracking system capable of identifying people in a room using microphones and video cameras once trained with voice and facial features [2]. Their system works well in conversation style settings, where one person talks at a time.

Roussel and Gueddana make the argument that research into video mediated communication should focus on the human aspects of communication and allow participants to choose their level of participation and fidelity [9]. While this is somewhat focused on changing the behavior of video, we believe that another dimension could be important and useful in communication–that of virtual presence.

Kwon and Candan descibe DANS, which uses a distributed hash table to provide distributed workflow routing, resource location and processing [7]. While the workflow graph is in essence static (since it is defined by what processing must be done on the data being processed), the chosen nodes and routing on the network are performed dynamically, allowing flexibility in node choice for load balancing and redundancy.

DANS is perhaps the closest related work to a system we use called DUP [5], which is a distributed stream processing language that connects sensors, computational infrastructure, and outputs. However, using a DHT causes unnecessary overheads due to routing through an overlay on the network.

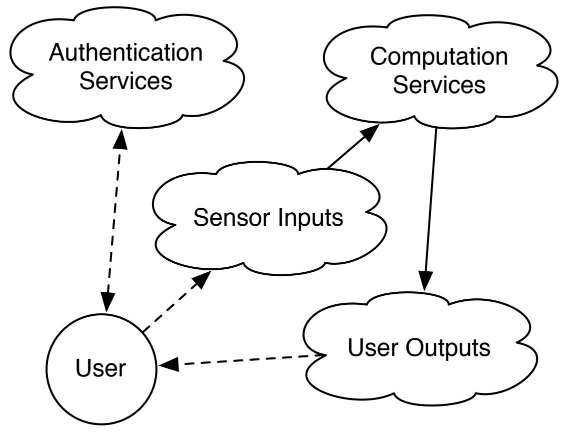

Figure 1: ***High-level Architecture:*** **Upon coming in contact with this architecture, users authenticate to determine which services they have available to them. Sensors, located in the physical space, collect motion capture data, audio and video when users are present. Collected data is streamed to computation services for processing and understanding user requests. Results are streamed back to fixed user output devices, such as displays or speakers, or streamed back directly to the user, in the case of wearable output devices like AR goggles.**

Benko argues that we should not only investigate multi-touch surfaces, but also non-flat and depth-aware interfaces [1]. This would allow more natural interactions in the physical world with virtual objects. We only add that multi-user interfaces are a must for future architectures such as ours. Of course many researchers have investigated a plethora of interfaces, we only mention one here for brevity purposes.

3. ARCHITECTURE

Our architecture is comprised of three main aspects: authentication, a computing and networking infrastructure, sensors for data collection, and *output* devices (such as wearable or fixed displays). We diagram the major pieces in Figure 1. From this figure we can see that the user first connects with the architecture by authenticating to determine available services. The user then interacts with the system through the sensors, (which we broadly define as input devices or actual sensors), recognized gestures or voice. Given that sensors typically have low computational power, the data is streamed to computational services which process the input. Assuming the user is identified and authenticated, the processed input is used to interact with the system in some fashion and the output of their actions are streamed to either output devices or to the user.

Of course, this architecture probably seems quite familiar. If we replace the sensor inputs with 'keyboard and mouse', and the user outputs with with 'LCD monitor' we have a typical computing platform. The primary difference here really comes in several areas:

1. *We expect this to be a* space*-oriented architecture, where a space encompasses an entire floor or building, a designated outdoor space, or a vehicle.* As a space-oriented architecture, it is naturally distributed, so that if, for example, sensors in a building fail, they do not affect interactivity in another area. Furthermore, the space-oriented concept naturally leads to a hierarchical organization of spaces: a campus, for example,

is a collection of spaces (the buildings) with which a user can travel to and interact with.

2. We assume that the architecture is multi-user. This is equivalent to modern multi-user OSes that allow remote login. The difference here is that *we expect users will interact with each other and their programs for collaboration*. Current OSes make this a somewhat difficult task at best, using the file paradigm to exchange interactions.

3. *We assume user mobility*. This is equivalent to logging on to different machines in the same administrative domain. In our case, we expect users to be able to migrate from space to space fairly seamlessly. Sensors allow a user to interact with the system regardless of where they are in the space. In addition, we expect that users will want to log on to a space remotely.

4. *Unlike a typical computing platform, we model spaces within the computation services for user-computer interactions and user-user interactions.* This, for example, allows remote avatars to be present in a building roaming the halls or remaining by the water-cooler to catch and converse with colleagues.

3.1 Assumptions

The main assumptions in our design are related to user desires and available hardware. First, we assume that users would be interested in this type of architecture if it worked effectively. They may be willing to use wearable displays, for example, if the result was a significantly improved experience. We assume that all users will carry a mobile device capable of some computation and at least a WiFi connection. We do not make assumptions about the capabilities of the mobile device beyond that it can be used to help identify the users, though if it provides location services, it simplifies processing required to locate and identify individuals. Last, we assume that we can provide enough networking hardware capable of streaming the data from the sensors to other sinks.

3.2 Authentication

Authentication is a key aspect of this architecture for several reasons. We acknowledge that current authentication technologies are sufficient to be used with our architecture. However, we have a few requirements beyond simple authentication to particular machines in a work place. Because users are mobile, we do not want them to have to type in passwords on a virtual keyboard or perhaps mobile device frequently as it increases the chance for their passwords to be compromised—especially when one considers the fact that we are advocating video cameras and motion capture devices to be placed *throughout* a space).

Instead, we assume that users will have a mobile device with them that can handle their identification authentication as follows. Spaces will broadcast a periodically changing and certified session ID. Their mobile device will first connect to the network and be assigned an IP address. They can then receive the session ID wirelessly or during DHCP. Their device can then use public-key cryptography to identify themselves to the system: they encrypt the session ID, time, and device address with their private key and send this over the wireless network. The server can then use their public key to decrypt the packet and determine which mobile address is associated with the user. We diagram the authentication packet in Figure 2. The use of the session ID prevents replay attacks on the authentication server and allows us to determine which device(s) belong to an authenticated user.

User ID	E(Session ID, timestamp, device address)

Figure 2: ***Authentication packet:*** **The authentication packet includes a clear-text user ID, to aid the server in determining which public key to use, and the session ID, current time, and device address, which are encrypted with the user's private key. Once authenticated, a shared key can be sent to the user allowing the exchange of encrypted information between the computation services and the user.**

Once the authentication server has the user's address, it can exchange a shared session key for encryption with the user. This allows data to be encrypted more efficiently than with public/private key pairs. For example, devices which are capable of calculating their own position or orientation can send updates to the system concerning their current location and/or orientation. In addition, further authentication can take place securely using the shared key.

We note that the computational services will contact the authentication services as needed to validate that a user has sufficient credentials to request and use a given service.

3.3 Sensor Inputs

Sensor inputs form the primary method for interacting with the spaces. A significant amount of research in games, multimedia, and virtual/augmented reality has gone into designing sensors, which we broadly define as *any* input device. However, the majority of this research has been single-user specific. In our architecture, we assume that multiple people will want to interact both with the computational services and with each other. Thus, as researchers we need to consider how we can measure and understand multi-party interactions.

Sensors additionally have important security requirements. Because they are capable of collecting live information, sensors should be required to authenticate with the system so that only administratively placed sensors can feed information to the system. In addition, they need some computational power for encryption and processing. We believe encryption of their data streams is required to prevent eavesdropping (discussed in Section 5). Further, the computational services needs to communicate with sensors. Services may need to turn sensors on or off, move their orientation (for those which can move), or increase or decrease sensitivity.

For interaction purposes we imagine that microphones, cameras, touch devices, motion capture, and simple portal threshold sensors are required, though clearly the idea behind this architecture is to allow future senors to be easily integrated with the system. Significant research has been accomplished in each of these sensor types (obviously portal threshold sensors are available to consumers in the form of automatic light switches–for our purposes, we would use these to turn on and off rooms of sensors based on whether or not someone was in the room).

We argue that microphones are required to allow voice commands to the system to be processed. The challenge here is 1) locating where the voice is originating from, and 2) distinguishing voices from each other in group settings. Bernardin and Stiefelhagen showed that voices could assist in locating the originators [2], but research needs to be done in separating voice streams and then analyzing their contents. Presumably, knowing the physical locations of the microphones in a room, and assuming we have a sufficient number of microphones, a combination of fast-fourier transforms and physical modeling would allow us to triangulate voice positions and separate the voice streams. At the sensor level, all we need is to stream the voice data to the computational services for analysis–voice analysis is not performed at the sensors, though one could imagine sensors having enough computational power in the future to perform the analysis locally.

Video cameras are also needed in this system for two primary reasons. First, video cameras stream data to the computational services part of our architecture for analysis. Computer vision techniques are frequently used to determine gestures or recognize faces and moods, thus giving users new ways to interact with the system. However, if someone is logging into the system remotely, techniques in 3D video could be used to build a 3D scene that they could interact with (i.e., texturing objects automatically with real images).

We also believe that touch devices, including touch sensitive displays, are important for this architecture because of the new ways they allow users to interact with the system. Recent work, such as that by Benko [1], argue that while the flat display will probably never go away because it is convenient for a large number of applications, other shapes and forms may provide more natural methods of interaction.

Finally, we argue that large scale motion capture is essential for this architecture. First, computer vision is quite computationally expensive. Adding motion capture data gives us a 3D image of the scenes, allowing us to more easily distinguish people. Clearly motion capture is going to be part of the next generation of game systems (e.g. Microsoft's^{TM}Project Natal, Sony's^{TM}PlayStation Motion Controller, Nintendo's^{TM}Wii Remote). Thus, commodity motion capture is on the horizon for consumers. On the other hand, only Microsoft's^{TM}technology appears to capture motion without markers or additional devices beyond the motion sensor, and it is not clear how many people it can capture simultaneously. In addition, we not only need to capture human motions, but we need to associate them with individuals. One might do this through face recognition or through mobile devices which give fine-grained location information (i.e. less than the width of a human in accuracy). Continued research in this area is necessary to allow us to capture motions in an entire *space*, especially in terms of subdivision of the processing tasks, unifying capturing devices, and human identification.

Input sensors should be both addressable and searchable, perhaps through some type of service from the computational services. For example, security may want to monitor camera feeds. Remote users may want to have access to a remote camera for interaction purposes. However, clearly access to these devices should be restricted to authenticated users only.

Being able to search for sets of sensors is also important for privacy reasons. For example, if one wanted to turn off all sensors in a room for a private conversation, one would rather issue commands to the system such as "turn off all sensors in the conference room" instead of having to physically disable the devices.

3.4 Computation Services

The computation services are considered the backbone of the architecture as they provide the computational power necessary to process the streams of data being sent by the sensors. These computation services may be computer systems with CPUs and/or GPUs. In Figure 1, we show the authentication and computation services as clouds primarily because they may or may not be performed on the same machine. Note that the sensors do need some networking and computation ability to encrypt their data streams on route to the computation backbone, and as such are not just simple data collection devices.

Distributed Stream Processing: Providing the glue between the sensors, computation services, authentication and output is a stream

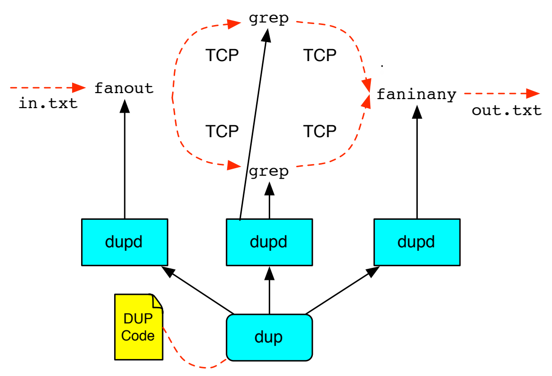

Figure 3: In this figure, the DUP code, which specifies the workflow, calls the DUP executable which contacts DUP daemons running on specified machines. Each DUP daemon gets a copy of the DUP code, which is parsed and which determines what stages are run on the local machine. This simple example shows 3 different machines: the first is running `fanout`, which duplicates the input to all outputs; the 2nd machine is running parallel `greps` for different things; the final machine is running `faninany` which merges the results of all inputs to a single output.

processing system. For our purposes, we need something that is lightweight and has low latency. Simple socket connections suffice for this purpose, but preclude significant amounts of code reuse. Instead, we use DUP, which is a parallel and distributed stream processing language [5]. DUP is an extension of Unix pipes, except that they allow any number of inputs and any number of outputs, instead of just 3 that standard Unix pipes allow (`stdin`, `stdout`, and `stderr`). DUP stages are similar to CMS pipelines [6]. We call these multi-stream pipes *stages* and stages can be written in any language appropriate to their functionality. With DUP, one designs a data graph that represents the flow of data in the system. This concept is similar to other workflow processing systems, such as DANS [7], except that stages are simply processes that run on any system and started (and stopped) by a `dupd` daemon. A simple example of DUP is shown in Figure 3.

An important feature that DUP gives us is the ability to write the stage in any language that supports reading and writing to files. Since POSIX compliant machines view even sockets as files, we can read from multiple sources and write to multiple sources across the network fairly easily and in a composable manner. Stages can then be written in a language appropriate to the problem. For example, GPUs provide massive parallelism at the expense of poorly performing branch statements. However, many computer vision and graphics problems are solvable in real-time using a GPU. Thus, a stage for processing images can be written in CUDA that runs on the GPU and by using DUP, we can easily forward the data streams to this stage for processing from any other machine. In essence, DUP would allow us to leverage the computational services we have available and easily change the workflow as new hardware and software is developed.

Modeling and Simulating the Physical Space: In addition to DUP, we expect that spaces will be *modeled* in 3D. As new users approach and authenticate with a space, they will be streamed the 3D model of the space. This will allow both their mobile systems and the computational services to have an understanding of the layout of the space. Within the computational services, a virtual simulation will be executing using the 3D model of the space and the positions of all recognized humans in the space. This is akin to a game server running where the virtual world is the 3D representation of the physical world.

The modeling and simulation of the physical world within the computational services is important because it allows users to interact with the physical world in a meaningful way and it allows users to remotely interact with people present physically. Of course, furniture and objects are often moved around a building. We expect that modeling will be done primarily of the walls, floors, doors, and other permanent features. Computer vision could be used to recognize other objects in the physical world and allow additional modeling.

For example, if a user wishes to share a video with another user, they could display it virtually on the wall. The simulation would occur not only in the virtual representation, but also with wearable displays or on real wall-mounted displays (if wearable displays were not available). Two users with wearable displays could look at a physical wall and augment the reality with a video stream.

In fact, a physically present user could interact with a virtual user if the physical user had a wearable display. As their voice would be captured by microphone sensors when they spoke, the voice data would then be streamed to any users within the vicinity in the virtual space so that the remote user could hear and communicate with the physically present user. In addition, when the virtual user spoke (perhaps into a headset at home), their voice would be streamed to the physical location corresponding the the virtual location of their avatar. Given this kind of physical and virtual interaction, remote collaboration would in theory be easier.

Processing Audio, Visual, and Motion Data: However, beyond modeling and simulating the spaces, the computational layer must also process voice data from group settings to interpret user commands (i.e., "Computer, play the video on this display"). DUP would allow us to easily stream voice data to CPUs or GPUs responsible for interpreting voice commands. Data streamed from motion capture sensors and video cameras would also be analyzed on appropriate and available resources using DUP. The analysis of this data is, and most likely will continue to be, the subject of much research in efficient and accurate computer vision. Understanding the motions of users in the space and interpreting gestures correctly is key to giving them the ability to interact effectively with each other and the provided computational services.

We emphasize the usage of stream processing primarily because high latency can cause undesirable user effects, such as motion sickness, if displays are not updated promptly. Thus a method for streaming processing is necessary for the architecture.

3.5 User Outputs

In terms of outputs, users are typically accustomed to audio and visual output. Audio output is fairly simple once we know the physical/virtual location of the user. For example, if a space has speakers located around it, we can search for the closest speaker and stream audio to them. This does cause some difficulty in calculation if we are trying to present them with 3D audio, especially in the presence of other users in different locations. On the other hand, if they have headphones attached to their mobile device, we could stream audio to it fairly trivially.

Visual output is slightly more complex, depending on whether or not the user has wearable displays. Without wearable displays, we can only display 2D video on fixed screens mounted throughout the space. Lacking a wearable display, users that wish to interact with virtual participants or 3D objects, do so through a fixed display.

Given wearable displays (either in HUD or screen format), we can present them with an augmented reality by streaming 3D video to their wearable displays (and probably through their mobile device). Since mobile devices still do not have significant processing power (and using it drains the batteries in them), we assume that we will stream already processed frames to them from the computational services. As mobile devices become more powerful, we can simply stream data to them and allow the mobile device to perform the rendering operations.

Tactile and other outputs can also easily be added to the architecture, though olfactory output systems are probably least desirable in shared spaces since smells are difficult to contain in a fixed area.

3.6 Network Architecture

With this set of devices, from sensors, to computers, to output devices, we now consider the network architecture. We have assumed so far that users will carry mobile devices which connect to the network. One issue that arises in practice is that the hand-off between WiFi access points can take up to 1 second. Research has addressed some of these issues [8], but connection issues with sockets may arise when streaming audio and video to a mobile device that is switching access points.

Furthermore, increasing the number of WiFi users in an area typically has a degrading effect on performance. On the other hand, we assume that the computational services layer will provide the majority of the data processing and compression so that only a single video and audio feed will need to be streamed to a mobile user.

Because the architecture is, for the most part distributed (i.e., you cannot physically interact with the first and second floors, therefore we can separate the simulation between floors of a building), adding sensors mainly requires adding network bandwidth in the form of switches and possibly routers so that the computational infrastructure has sufficient bandwidth for the multimedia streams it must process and respond to. Virtual avatars will move from simulation to simulation by simply connecting to the source of simulation for a particular space.

Therefore, except for increased bandwidth to handle the multimedia streams from numerous sensors and faster and stable handoffs between WiFi access points, we do not foresee major architectural changes required in a typical local area network.

4. ADVANTAGES

Given this architecture, we list the advantages of having such a design. We assume that participants have, at minimum, wearable displays (which is reasonable given current technology).

For the business and academic world, computational spaces allow us to collaborate with colleagues more efficiently. We can meet together and interact with 2D and 3D objects without the encumbrances of laptops and current OSes. Further, an ill colleague can simply connect virtually without spreading germs, though some may consider attending meetings even when sick to be a drawback instead of a benefit. Roussel and Gueddana suggest that at least in video mediated communication, we should be able to customize our 'presence' [9]. Such a system as ours allows us to even appear virtually as an avatar of any shape or size.

In addition, this type of system would allow for augmented classrooms. Again, students unable to make an early morning class could attend virtually. We could also present a more engaging education by changing the classroom settings to more interesting places. Perhaps learning math on the beach is more engaging to a student than a classroom with white walls and small windows. Of course this brings to light an entire other area of research which relates to content creation for such a system.

This architecture would help advance the state-of-the-art in computer games. As noted before, companies are already investigating and using some form of motion capture as input. Since we are fully tracking human motions throughout a building, games could be truly mixed with reality. From puzzle type games, to role-playing games, to typical zombie hack-and-slash games, the settings could be known locations, adding an interesting element to play and new challenges to game design. Given pervasive motion capture, we could have multiplayer games with both physically and virtually present players.

Finally, this type of architecture could advance the way we interact with computers allowing us to use voice, gestures, motions and augmented reality. Obvious computing paradigms for fixed-flat displays, such as windows and desktops, may make less sense in an augmented reality.

5. DRAWBACKS

Given the advantages, we would be remiss not to mention drawbacks to an architecture such as this. First, adding pervasive sensors increases the chance for eavesdropping on conversations or interactions. Security is extremely important, as is the ability to 'shut off' sensors in a room for privacy. Given the abundance of sensors makes it easier for malicious users to place false sensors that *always* record and stream to their own storage as they would be less likely to be recognized. Thus, any setup such as this would require physical security measures to audit devices in a building.

Some people get motion sickness from using wearable displays, especially if there is sufficient latency between image updates (often referred to as image swimming). As technology advances, the swimming issues may be overcome, but this still does not mean that it would prevent motion sickness from the devices. As such, we would suggest some set of fixed displays that didn't require wearable displays for interaction.

Hacking is another problem with this type of system. Anyone gaining improper access to the computational system would have the ability to read incoming data feeds. Similarly, they may be able to inject false information into the system with this kind of access. One can only imagine the result of a hacker taking control of a wearable display.

Finally, privacy issues are clearly involved when anyone proposes fully tracking and recording individuals in a space and are important issues for future research. Clearly at least some method for making oneself 'untrackable' is important, especially when major deadlines are approaching. Perhaps being able to participate remotely even when physically being at the space is necessary to allow users to remain in their offices while attending meetings.

6. CONCLUSION AND FUTURE WORK

In this paper, we have presented an architecture that allows multiple users to interact in both collaborative (i.e., work-related) and entertaining (i.e., computer gaming) ways. Our architecture consists of an authentication component, a sensor layer, a computational infrastructure, and an output layer. We use DUP to provide the stream processing required for the many multimedia streams which must be processed, analyzed, and possibly forwarded to remote locations.

Multimedia systems research is needed to make several parts of this architecture a reality, but we argue that the future of multimedia systems will be computationally distributed, space-oriented, and allow physical and virtual participation in experiences. However, given the many advantages of such a proposed system, researchers need to carefully consider the consequences of a system that fully tracks and records human movement and propose techniques to mitigate potential issues.

As part of our ongoing and future work, we are developing voice stream recognition algorithms for crowds and building-wide motion capture sensors using lasers. We are also developing a high-level language for DUP to make building workflows easier since most of the stream processing designs take form an acyclic graph.

7. REFERENCES

[1] H. Benko. Beyond flat surface computing: Challenges of depth-aware and curved interfaces. In *Proceedings of ACM Multimedia*, October 2009.

[2] K. Bernardin and R. Stiefelhagen. Audio-visual multi-person tracking and identification for smart environments. In *Proceedings of ACM Multimedia*, September 2007.

[3] J. Campbell, P. B. Gibbons, and S. Nath. Irisnet: An internet-scale architecture for multimedia sensors. In *Proceedings of ACM Multimedia*, November 2005.

[4] M. Capra, M. Radenkovic, S. Benford, and L. Oppermann. The multimedia challenges raised by pervasive games. In *Proceedings of ACM Multimedia*, November 2005.

[5] DUP: A distributed stream processing language. `http://dupsystem.org`, January 2009.

[6] J. P. Hartmann. *CMS Pipelines Explained*. IBM Denmark, `http://vm.marist.edu/~pipeline/`, September 2007.

[7] G. Kwon and K. S. Candan. DANS: Decentralized, autonomous, and network-wide service delivery and multimedia workflow processing. In *Proceedings of ACM Multimedia*, October 2006.

[8] I. Ramani and S. Savage. Syncscan: Practical fast handoff for 802.11 infrastructure networks. In *Proceedings of IEEE Infocom*, March 2005.

[9] N. Roussel and S. Gueddana. Beyond "beyond being there": Towards multiscale communication systems. In *Proceedings of ACM Multimedia*, September 2007.

[10] W. Wu, A. Arefin, R. Rivas, K. Nahrstedt, R. M. Sheppard, and Z. Yang. Quality of experience in distributed interactive multimedia environments: Towards a theoretical framework. In *Proceedings of ACM Multimedia*, October 2009.