In the *International Symposium on Augmented Reality (ISAR 2000)*, Oct 5-6, 2000, Munich, Germany.

Adapting to Dynamic Registration Errors Using Level of Error (LOE) Filtering

Blair MacIntyre and Enylton Machado Coelho

Graphics, Visualization and Usability Center
College of Computing
Georgia Institute of Technology
Atlanta, GA, USA
{blair, machado}@cc.gatech.edu

Abstract

In this poster we describe our initial work on generating Augmented Reality (AR) displays in the face of dynamically changing errors in the pose (position and orientation) of both the user and objects in the world. Dealing with this problem is particularly important in mobile AR environments, where the tracking accuracy of the user's head can change frequently and dramatically as she moves between areas with radically different tracking systems, such as in and out of buildings. We introduce the notion of "level of error" filtering, analogous to "level of detail" culling in 3D graphics systems, to help programmers build interfaces that automatically adapt to changing registration errors.

Keywords: augmented reality, human-computer interaction, adaptive interfaces, mobile computing, wearable computing.

1. Introduction

Augmented Reality (AR) systems must be able to *register* the computer generated visual and auditory media with the real world. For example, a virtual arrow created to point at a resistor on a circuit board must appear to point at the correct resistor, not some other component near it. Registration requires accurate *tracking*: the system must know where the objects (i.e. the resistor) are with respect to the user and her displays in order to accurately register the graphics or sound with the appropriate objects in the world. Unfortunately, even small tracking errors will result in registration errors. It is our belief that the tracking problem will not be solved in the near future, especially in the context of wearable AR systems that must work predictably and reliably as mobile users move through unfamiliar, uncontrolled and even dangerous environments. Therefore, we believe that AR systems should take these errors into account rather than assume they will go away. The goal of our research is to develop techniques for creating AR displays that adapt their visual and auditory augmentations to account for changes in pose errors.

In this poster, we introduce an approach to automatically adapting to changing registration errors, which we refer to as *level of error* (LOE) filtering. This approach is similar to (and inspired by) *level of detail* (LOD) culling used in 3D graphics systems: in real time, the system computes a value that is used to select one of a set of alternate representations of a virtual object, where each representation corresponds to a range of the computed value. In LOD culling, the value used is the distance between the viewer and the objects, with the goal being to allow less computationally expensive representations to be used as the distance increases. In LOE filtering, the value is the registration error, which is in turn a function of the distance from the user to the object of the augmentation and the errors in the pose data of the user and objects. The goal of LOE filtering is to allow different representations of an augmentation to be automatically used as the registration error changes.

A long-term goal of this research is to develop high-level programming toolkits for AR applications. In our previous work, we focussed on support for rapid prototyping of distributed AR applications, but provided no explicit support for sophisticated display techniques [6, 8]. Like most other researchers, we encoded all the display techniques in the applications themselves, making them difficult to build on in subsequent systems, and difficult to change. We are in the process of developing toolkits that support sophisticated techniques for displaying, arranging and filtering augmentations across a full range of AR applications. LOE filtering provides the low level support for responding to registration error changes in real-time, and will be a key component of such toolkits.

2. Background and Related Work

Many technical problems must be solved before wearable AR systems will become practical. One of the most important is the tracking problem, which has received significant attention from AR researchers (e.g., [1, 5]). The tracking problem is so important because, for many applications, the perceived quality of an AR display is a direct function of how well the computer-generated material is registered with the user's perception of the world. The accuracy of the registration is in turn a function of the accuracy of the position and orientation (or *pose*)

information used by the system to generate the augmentations, including the pose of both the user and the objects in the environment. For an analysis of the issues related to registration error, see [4]. As mentioned in Section 1, we believe that the tracking problem will not be solved in the near future, especially in the context of wearable AR systems that must work predictably and reliably as mobile users move through unfamiliar, uncontrolled and even dangerous environments.

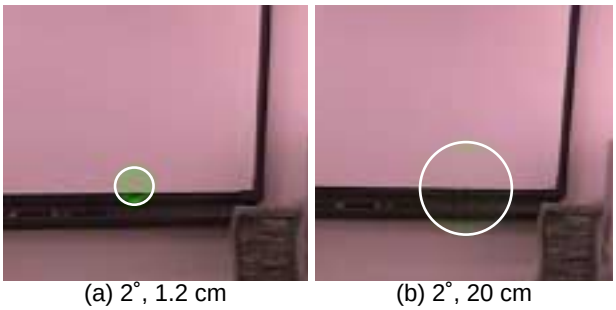
In contrast to those researchers interested in the tracking problem, most of the remaining AR researchers (including ourselves) are interested in examining the application and UI issues associated with AR. Because of the poor quality of tracking technology, and the difficulty of building even the most basic AR application prototypes [6], UI researchers usually assume some amount of accuracy in the tracking data and integrate these assumptions into their application and interface designs. At one end of the spectrum, many projects assume, implicitly or explicitly, that the tracking problem will one day be solved and near-perfect registration will be possible (e.g., [3, 11]). Others have assumed lower quality tracking and built systems that perform reasonably under those poor conditions. Many mobile systems fall into this category, typically because of the low positional accuracy available from differential GPS (i.e., usually about 2 meters) and wide-area indoor tracking systems (such as Active Badges) (e.g., [2, 9]).

In contrast to these systems, we believe that mobile AR systems should be built assuming the tracking problem will never be completely solved, and that different parts of the world, and different objects within it, will exhibit different tracking errors. Furthermore, we believe that the magnitude of the tracking errors will change dynamically as the user moves from location to location, so mobile AR applications will need to continually adapt to these changes [7].

2.1. Registration Error Ranges

It is important to remember that, for a pose measurement in a real system, the actual error cannot be accurately measured; if the error was known, the pose measurement could be corrected to eliminate it. Instead, for any given pose measurement, we use the term *error* to refer to the range of possible errors. An AR system will try to model the range of possible errors that affect a pose measurement, and use this model to determine the error range for that measurement.

Typical tracking system manufacturers provide specifications of the worst-case maximum translation and orientation errors of their devices, which can be used as a starting point for modelling the tracker's error. In Figure 1, we show the error volumes that correspond to possible end-to-end errors in our prototype, both of which are much larger than the manufacturer specifications for the tracker we used.



(a) 2˚, 1.2 cm (b) 2˚, 20 cm

Figure 1. Error volumes around a 3D point, with different tracking error (each volume was rendered using a cloud of green points, which has been outlined here in white for clarity). The error volume is intended to be centered around one of the buttons on our electronic whiteboard (near the bottom of the sphere in (a)). The error values are much larger than the manufacturer specification for our tracker (1/4˚, 6mm) because the total end-to-end error in the system is larger than just the tracker error. (a) corresponds to a realistic estimate of the error in our prototype; the button lies within the volume, just on the edge. (b) shows the volume if we had a less accurate measurement of the location of the whiteboard.

In our initial work, we are assuming that all pose measurements can have their error range modelled as an angular orientation error and a linear translation error. We also assume that the error values can change with each new measurement. While this is a simplification, it captures the gross behavior of measuring pose error, without getting bogged down in the details of individual tracker behavior. In our current experiments, we are modelling the errors by using the simple, manufacturer specified worst-case error of our trackers. However, some trackers, such as magnetic or ultrasonic position sensors, exhibit non-uniform error over their tracking space. We are assuming that, if such error could be measured, a more robust model could be built that would not violate any of this work.

More importantly, for some trackers the worst case error and typical error are significantly different. For example, a magnetic compass may have a typical worst-case error of a fraction of a degree, but may exhibit significantly greater error in the presence of strong magnetic fields [12]. A GPS system may have a worst-case error of 100 meters, but a typical error of less than half of that. Dealing with such problems is a topic for future work, and will probably require the integration of multiple sensors.

3. The LOE Filtering System

As discussed in Section 1, the solution we are pursuing is to collect a variety of information (e.g. pose errors and semantic information about the desired augmentations),

and use this information to guide the display techniques for each augmentation in an AR system.

Our first step toward a solution to this problem is the notion of *level of error* filtering, analogous to the concept of *level of detail* culling (LOD) used in 3D graphics systems. LOD is based on the observation that, as an object moves farther from the viewer, it takes up less pixels on the screen, and therefore simpler models can be used to render the object without changing its appearance. Systems that use LOD automatically switch between the different representations of an object based on the distance to the object from the viewer.

LOE is based on the observation that, as an augmentation moves with respect to the viewer, or as the error ranges for the viewer or the object change, the registration error of pixels on the augmentation may change, which may require that the representation of the augmentation change as well.

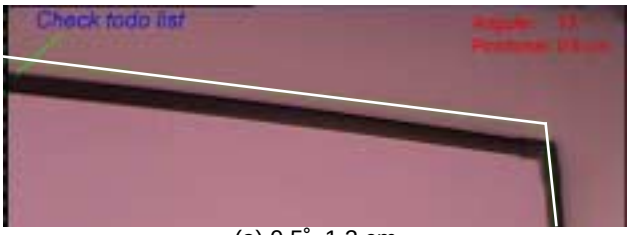
To test these ideas, we implemented LOE filtering objects as an extension to Java3D [10]. These LOE objects extend the Java3D LOD objects, and support dynamically switching between any number of representations of an augmentation based on the registration error. In this implementation, the semantics of the augmentation are implicit in the alternative representations implemented by the programmer, and are not explicitly used by the system.

3.1. LOE Object Interface

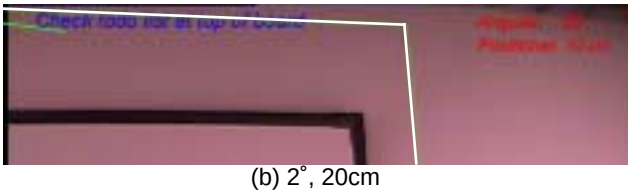
When creating an LOE filtering object for an augmentation, the programmer provides a set of *N* representations ($\texttt{representationObject}_i$, $\texttt{i=1..N}$) along with *N-1* numbers ($\texttt{maxError}_i$, $\texttt{i=1..N-1}$) representing the registration error boundaries. The system arranges for $\texttt{representationObject}_1$ to be used when the registration error is less than $\texttt{maxError}_1$, $\texttt{representationObject}_N$ to be used when the registration error is greater than $\texttt{maxError}_{N-1}$, and for $\texttt{representationObject}_i$ to be used when the registration error falls between $\texttt{maxError}_{i-1}$ and $\texttt{maxError}_i$.

Internally, the LOE objects maintain information on the eye position of the viewer and the pose error characteristics of the tracking system. Currently, we have one set of error characteristics maintained, but eventually the programmer will need to be able to specify if the target of the augmentation is subject to any additional pose error.

The registration error is computed as an angular error that combines the angular and translational errors of the user's viewpoint with those of the object being augmented.



(a) 0.5˚, 1.2 cm



(b) 2˚, 20cm

(c) 8˚, 60 cm

Figure 2. Three different augmentations to help the user locate a todo list on the electronic whiteboard in our lab. In (a), an arrow is pointing at a box highlighting the board, and is labelled with text telling them to check the todo list. In (b), the arrow is pointing to a box that encompasses that larger area the board should fall in, and the text is more specific. Finally, in (c), the error is sufficiently large that no meaningful spacial augmentation can be used, so an event more detailed textual description guides the user to the hardboard and the todo list on it.

3.2. Example: *Locate* an object

One of the most commonly used augmentations is pointing at a location in space with a text label and arrow. In this example, we created an *locate* augmentation that has three different representations, as shown in Figure 2. In (a), the whiteboard is outlined with a green box and a simple text label is used to tell the user why the board is outlined. Unfortunately, once the registration error approaches approximately an inch in local whiteboard coordinates (or 1/60th of the width of the board), the green box may no longer meaningfully outline the whiteboard. Figure 2(b) shows the second representation of the augmentation, which is similar to the first except that the rectangle is big enough to contain the whiteboard as long as the error falls within 10 inches in whiteboard coordinates (or 1/6 the width of the whiteboard), and the text label is a bit more detailed. Finally, if the registration error exceeds 1/6 the width of the whiteboard, the third representation of the augmentation is used; this version only contains a text label

that explains where the todo list is without pointing to the real world.

In this example, the LOE object takes care of switching between the three representations as the user moves or the pose error ranges change.

4. Discussion and Future Work

In this poster, we have described out initial work on automatically adapting to dynamically changing registration errors, which we refer to as *level of error* (LOE) filtering. This approach is similar to (and inspired by) *level of detail* (LOD) culling used in 3D graphics systems: in real time, the system computes a registration error value that is used to select one of a set of alternate representations for an augmentation. The goal of LOE filtering is to allow different representations of an augmentation to be automatically used as the registration error changes, either because the user moves or because the pose error range of the tracking system changes.

An interesting difference we have noticed between LOD and LOE objects is the need to support some amount of hysteresis in the function that chooses the representation to display. In LOD objects, the system can switch between representations as often as desired, including toggling back and forth between two representations if the user is sitting on a boundary. This works because the objects are typically designed to be are indistinguishable at these boundary points. In LOE objects, on the other hand, the alternate representations may be very different, so the system should not rapidly toggle between representations when a user is on a boundary. Modifications to the choice function, such as changing only if the boundary has been crossed by a certain amount, would fix this problem.

A long-term goal of this research is to develop high-level programming toolkits for AR applications. We are in the process of developing toolkits that support sophisticated techniques for displaying, arranging and filtering augmentations across a full range of AR applications. LOE filtering provides part of the low level support for adapting to registration errors in real-time, and will therefore be a key component of our toolkits.

5. Acknowledgments

The authors would like to acknowledge the members of the Augmented Environments Lab and Future Computing Environments Group at Georgia Tech for their influence on this work. We would especially like to thank Rob Kooper for his involvement, and his help with the support software. This work was supported by Siemens via a GVU Industrial Affiliate Grant, ONR under Grant N000140010361, and equipment and software donations from Sun Microsystems and Microsoft.

6. References

[1] Azuma, R. and Bishop, G. (1994) "Improving Static and Dynamic Registration in an Optical See-through HMD" In *Computer Graphics Proc. ACM SIGGRAPH '94*, pp 197–204.

[2] Feiner, S., MacIntyre, B., Höllerer, T., and Webster, A. (1997) "A Touring Machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment" Personal Technologies, 1(4):208–217.

[3] Feiner, S., MacIntyre, B., and Seligmann, D. (1993) "Knowledge-based augmented reality," Communications of the ACM, 36(7):52–63.

[4] Holloway, R. (1997) "Registration Error Analysis for Augmented Reality", *Presence*, 6,(4), Aug. 1997, pp. 413–432.

[5] Jacobs, M., Livingston, M. A., and State, A. (1997) "Managing Latency in Complex Augmented Reality Systems." In Proceedings of 1997 Symposium on Interactive 3D Graphics, Providence, RI, April 27-30, 1997, pp. 49–54.

[6] MacIntyre, B. (1999). "Exploratory Programming of Distributed Augmented Environments," Phd Dissertation, Columbia University, Department of Computer Science, 1999.

[7] MacIntyre, B. and Feiner, S. (1996) "Future multimedia user interfaces," Multimedia Systems, 4(5):250–268.

[8] MacIntyre, B. and Feiner, S. (1998) "A Distributed 3D Graphics Library." In Proc. ACM SIGGRAPH 98, pages 361-370, July 19-24, 1998, Orlando, Florida.

[9] Mynatt, E. D., Back, M., Want, R. and Frederick, R. (1997) "Audio Aura: Light-Weight Audio Augmented Reality," In Proceedings of ACM UIST'97 Symposium on User Interface Software and Technology, pp. 210-212.

[10] Sowizral, H., Rushforth, K., and Deering, M. (1998). *The Java 3D API Specification*. Addison Wesley, Reading, MA.

[11] Webster, A., Feiner, S., MacIntyre, B., Massie, B., and Krueger, T. (1996) "Augmented reality in architectural construction, inspection and renovation" In Proc. ASCE Third Congress on Computing in Civil Engineering, pages 913–919, Anaheim, CA.

[12] You, S., Neumann, S., and Azuma, R (1999). "Orientation Tracking for Outdoor Augmented Reality Registration." In IEEE Computer Graphics and Applications 19(6): 36-42 (Nov/Dec 1999).