Adaptive User Interfaces in Augmented Reality

Simon Julier* Mark A. Livingston† J. Edward Swan II† Yohan Baillot*
Dennis Brown†
Naval Research Laboratory
Washington D.C.

Abstract

In this position paper, we consider the issue of developing a flexible software architecture which will allow multiple interaction techniques to operate simultaneously and in parallel with one another. We describe a set of interaction techniques, identify their architectural requirements, and consider methods for prioritizing different techniques and arbitrating between them when they have conflicting objectives. The discussion is made within the context of a mobile AR system for urban situation awareness.

1 Introduction

Augmented reality (AR) presents information in its context in the 3D environment. This capability raises a set of questions such as: what information needs to be shown, what is the most appropriate way to show it, and how do the display elements interact with one another and the real world? These questions become even more complex and interesting when we utilize the AR display metaphor of “X-ray vision” (a capability that can *only* be provided by AR), and when we visualize over a large region such as a city.

The human mind possesses vast capabilities for processing and integrating information. One could argue, on this basis, that the AR system should simply present all available information and let the user decide for themselves what is the necessary data to perform the task at hand. This argument pre-supposes that the information is presented in a physically realistic (and photo-realistic) manner—i.e. that the AR system can present all the visual cues that we use in human vision. Most AR systems, however, are neither capable nor likely to be capable of producing real-time photo-realistic visualizations of their information databases. Indeed, there is no constraint that such information even have a physically realistic representation. This will become acute when we discuss the physically *unrealistic* representation of occluded surfaces in Section 2.2.

Even if photorealistic visualizations could be produced in real time (with proper lighting and shadows and other difficult rendering tasks), it would require tremendous computational power. Much of this effort would be wasted; the information would be filtered out by the human visual system. While it is difficult to know a priori what portions of the image will be filtered out when the user examines the image for information, it is the domain of user-centered design to determine what information is necessary, and exactly what set of cues conveys that information precisely.

Recent research into AR systems have developed a number of user interface techniques including *information filtering* [8], *occlusion representation* [11, 7], *adaptation to registration error* [12], *adaptive label placement* [3], and *3D multimodal interaction* [9]. Despite the complementary nature of these techniques, no AR system has, to date, has attempted to combine all of these techniques into a single display.

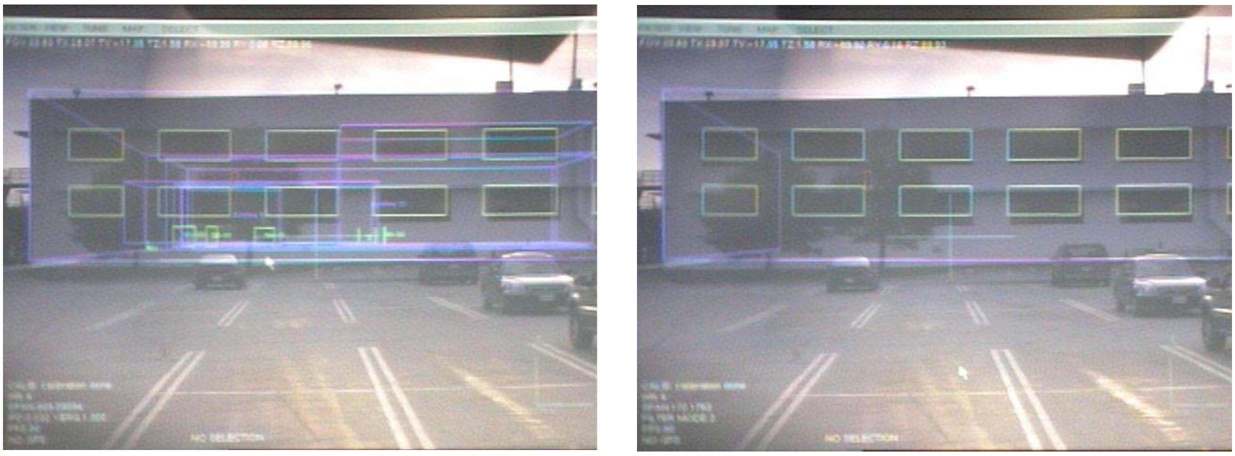
This position paper considers the problem of developing an architecture to allow the use of multiple techniques simultaneously. Section 2 describes each of the techniques in more detail and classifies them in terms of their effects on the rendering pipeline and the architectural support components which are required. A proposed architecture is described in Section 3. Section 4 summarises the paper and suggests open topics for discussion.

2 Visualization and Interaction Techniques

This section describes five user interface algorithms. Each algorithm is described in terms of the *need* for that algorithm, an *implementation* of it, and a characterization of the architectural *requirements*. After the individual algorithms are presented, we categorize them by their effects on the geometry in the rendering pipeline from the database to the screen.

[*] ITT Advanced Engineering and Sciences / Naval Research Laboratory. Corresponding email: `julier@ait.nrl.navy.mil`

[†] Virtual Reality Laboratory, Naval Research Laboratory



(a) Unfiltered display.

(b) Filtered display.

Figure 1. An example information filtering. From [8].

2.1 Information Filtering

2.1.1 Need

In large and complicated environments *information overload* is a significant problem. Overload occurs when the amount of information shown to the user is so large that it cannot be meaningfully understood. Furthermore, we may assume that visual search will always be a primary task of AR users, and there is ample psychophysical evidence that complex visualizations negatively affect visual search and other measures of visual performance [17]. While the problem is not simply volume of information, but rather design [18], it is reasonable to limit the amount of information shown in order to produce a better user interface. Figure 1(a) shows an example of this problem.

2.1.2 Implementation

To meet the requirements for our application, we developed a hybrid filtering algorithm which utilizes two steps — a culling step followed by a detailed refinement step [8]. The purpose of the culling step is to rapidly identify, from a potentially large database, only a relatively small number of objects which might be relevant. Because relevance contains a strong spatial component, we adopted concepts from the spatial model of interaction [4]. Each user is surrounded by a *focus*, the area with which the user is concerned. Each object is surrounded by a *nimbus*, the area that the object influences. The focus could, for example, be a “zone of awareness” which surrounds the user. Only an object whose nimbus intersects the user’s focus is considered in more detail.

Given a list of intersecting geometry, the refinement step does a more precise scoring to determine which objects are most relevant. We use a vector-space model to calculate the dot product between a vector of object attributes and a vector of characteristics of features which are important for a particular task [8]. However, many other algorithms such as the Query Set Architecture [2] or the rule-base used in KARMA [6] could be used.

2.1.3 Requirements

A semantically rich, object-oriented database. The system must be capable of showing individual parts of objects separately from all other objects. Therefore, the models cannot be described as a “polygon soup”, but must be a set of distinct, separate, addressable objects. Furthermore, objects need to have meta-data associated with them, such as identity and classification, as well as represent abstract relationships such as ownership.

Ability to perform intersection operations rapidly and efficiently. Efficient action of the culling step is vital if the filtering algorithm is to be efficient. Therefore, databases with efficient intersection operations should be created. However, these databases need to operate with the foci and nimbi. Therefore, the geometry of objects within that database do not necessarily bear any direct relationship with the actual physical size and shape of the object. Furthermore, because the foci and nimbi are task and context dependent, they can abruptly change in a “step wise” fashion.

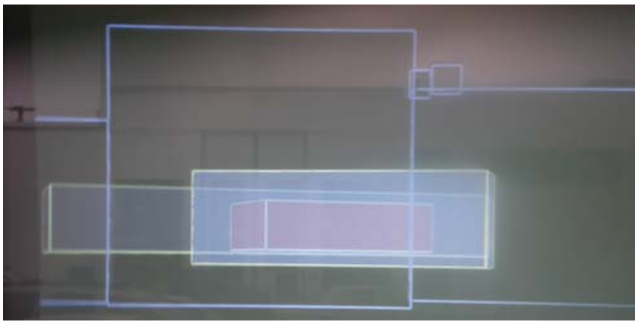


Figure 2. Occlusion representation. Showing different objects with different levels of "occlusion." From [11].

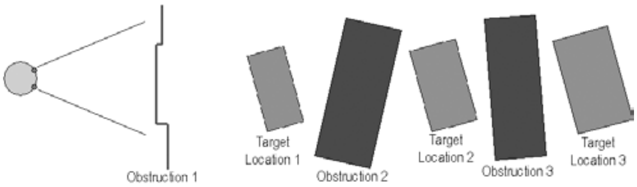


Figure 3. Overview of the potential configurations of buildings and targets in Figure 2, which shows target location 2. From [11].

2.2 Occlusion Representation

2.2.1 Need

Using the metaphor of "X-ray vision", AR can show information about objects which are physically present but are not visible from the user's view. This is one of the most powerful uses of AR. For our urban situation awareness application, we determined that users need to identify the 3D locations of other users, mobile vehicles, and structures when they are occluded by large urban structures such as buildings (see Figure 2). We can provide the user with an overhead map (Figure 3), but we would prefer that the user not need to make such a contextual switch in order to visualize the relative locations of these objects and personnel. However, heads-up visualization of occluded surfaces also introduces one of the greatest challenges for interpreting the display. Occlusion provides powerful depth cues to determine ordering. When an AR system displays surfaces which are physically occluded, alternative cues must be inserted into the display to replace the lost cues.

2.2.2 Implementation

Occlusion is a fundamental problem and, to date, no AR system has been developed which fully accounts for its presence. However, recent studies have been conducted to explore appropriate user interfaces [10, 7, 11]. An example of a depth cue representation is shown in Figure 2 [11]. Drawing inspiration from technical illustration techniques such as using dashed lines used to denote relative surface depth, this work developed the notion of a set of "occlusion layers." As the number of layers increases (and more intermediate objects are placed between the user and target object), the study varied various parameters such as drawing style, intensity, and opacity, until the user could quickly and accurately identify the depth order. Other techniques include the use of cut-away views [7] to denote depth order.

2.2.3 Requirements

Identify and classify occluding contours. The system must be able to determine which objects and parts of objects in an environment can affect the level of occlusion using the subset of the database to be drawn.

Automatically deduce the level of occlusion. Given a target object, the system should be able to calculate what the level of occlusion of that object will be. If the occlusion levels are different for different parts of the object, the occlusion levels need to be parameterized in such a way that this difference is noticed.

Encode objects. Use the perceptually identified encodings to draw the objects at different levels of occlusion. Potentially this includes the use of shading, transparency, line style, line thickness, and other rendering parameters.

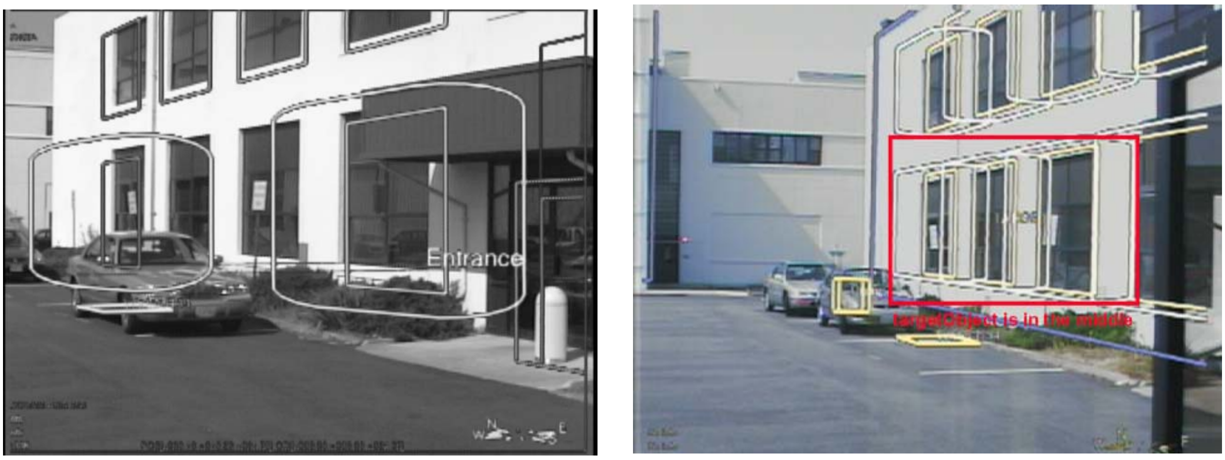
2.3 Registration Error Adaptation

2.3.1 Need

Trackers are imprecise and devices and displays can never be calibrated perfectly. As a result, the generated graphics will never perfectly align with their physical counterparts. Instead, there will always be some kind of time-varying registration error. Although improved tracking and calibration procedures will reduce the magnitude of these errors, perfect alignment cannot be achieved. Therefore, the user interface must be capable of dynamically adapting the way in which it shows information. We believe that the primary problem is *ambiguity* — it is not clear how a particular graphical object relates to the environment.

2.3.2 Implementation

One means of adapting to registration error is to use a *Level-Of-Error* (LOE) object [12]. Analogous to *Level-Of-Detail*



(a) Non overlapping error convex hulls.

(b) Overlapping error convex hulls.

Figure 4. The registration error convex hulls for two sets of windows on the side of a building. In the first case, the windows are sufficiently far apart that each can be drawn unambiguously using the hulls. In the second case, the hulls overlap and an aggregate display is used. Adapted and extended from [1].

(LOD) that changes the geometric representation of an object as a function of (for example) screen size, LOE changes the appearance of an object as a function of registration accuracy. If the graphics can be aligned precisely, standard AR representations (such as wireframe) might be sufficient. As the alignment degrades, such detail may be ambiguous with respect to the environment.

The following method for calculating registration error was described in [1]:

1. Identify a *target* object, which is the object to be shown to the use. Identify a set of *confusers*. These are objects which, from the user's point-of-view, might be similar to the target object.

2. For the target and the confusers, calculate the registration errors for the target and all confusers. These are described as convex hulls which are constructed from the covariance ellipses placed at the mean value of each projected vertex.

3. If there are no confusers, the geometry of the object can be bounded by the registration error convex hull.

4. If there are confusers, automatically generate a description of the location of the target object within the region.

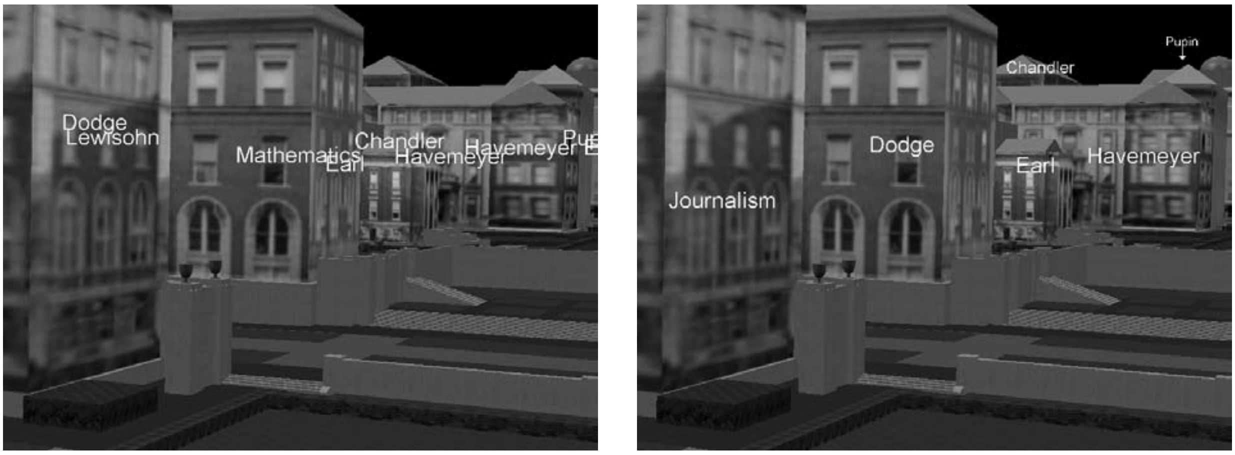
This technique is illustrated in Figure 4 which considers the example of highlighting a window to a user. Figure 4(a) shows the hulls which have been constructed for two disjoint objects in the presence of substantial yaw error. Although the registration region is stretched horizontally, the hull surrounding each object together with a suitable label is sufficient to direct the user to the correct object. Figure 4(b) shows what happens when multiple objects overlap. The display excludes the annotations for the individual objects and alters the representation to include a new summary which could be a bounding box and a textual description.

2.3.3 Requirements

Ability to automatically identify and classify potential confusers. From the database, automatically classify which objects, from the user's current perspective, are "similar" to one another. This projection operation should ideally take account of the size, shape, and potential object color.

Ability to automatically calculate the error regions for an arbitrary number of objects. Given the set of potential confusers, calculate the error regions for a set of objects. The error calculation could should be capable of taking into account errors from an arbitrary number of transformations which can change in real-time.

Ability to aggregate multiple objects together using LOEs. When confusion conditions arise, the representations of the targets and the confuser must be aggregated together.



(a) Näive label placement puts building labels on building centroids.

(b) Managed label placement considers many legibility issues when placing building labels.

Figure 5. The effect of view management on building label placement. From [3].

2.4 Adaptive Label Placement

2.4.1 Need

In many situations AR annotations cannot be treated as static 3D objects which can be rendered in the environment. Instead, some kind of active view management is required. The issue is illustrated in Figure 5(a) [3]. This figure shows a system which attempts to label buildings. If the labels are drawn näively (at the centroid of each building) the results are confusing, ambiguous, or even wrong. Rather, labels have to be drawn to respect the part of each object which is visible.

2.4.2 Implementation

The algorithm proposed by Bell computes axially-aligned approximations of the projection of objects, then determines visibility with simple depth ordering algorithms (e.g. a *z*-buffer). Labels are allowed to flow into the projections. Significant work has been carried out to provide appropriate moving label dynamics to ensure that the temporal behavior of the moving labels facilitates legibility.

2.4.3 Requirements

Identify visible objects. Be able to determine which objects, or parts of objects are visible.

Parameterize free and open space in the view plane. Be able to determine the parts of the view plane which are open. This determines where and how labels can be placed.

Real-time animation. On a per-frame basis it must be possible to update drawing characteristics such as label size and placement.

2.5 Multimodal Input

2.5.1 Need

Users need to be able to query and interact with the information they are presented. However, the usual keyboard and mouse interfaces, designed for 2D desktop use, do not work well with wearable computers [15]. Other more natural interaction paradigms must be supported. By far the most powerful interaction paradigms are multimodal, fusing a number of natural input modalities together.

2.5.2 Implementation

Pittman et al. [14] have shown that more accurate multimodal input can be achieved if probabilistic methods are used to combine probability-weighted sets of hypotheses for speech and gesture recognition. Kaiser et al. [9] describe a multimodal speech and gesture system for AR and VR. Both systems use an "adaptive agent architecture", which offloads the problem of speech and gesture recognition to outside agents. A rule-based agent called the integrator weighs all of the given data and choose the most likely meaning of a user's action. However, the object selection is done on an untracked 2D map interface, so the system assumes there is no error in that regard — the user is always able to correctly designate the selection object.

However, spatial operations, such as pointing, are complicated by tracker error in the AR system. For example, it is similar to the error adaptation, but this time, selection volumes are being calculated and the errors have to take account of the user's head tracking error and tracking errors in the selection device as well. Using these measurements, weights can be attached to objects in the selection volume. Those weights are used by probabilistic multimodal integration methods, along with inputs of other modalities (particularly speech), to determine the intended object.

2.5.3 Requirements

Input modality recognizers. As discussed above, each input modality requires a recognizer.

Probabilistic selection. Given knowledge of the environment and the direction in which the user is indicating, the system should be able to generate a list of potentially selected objects and the probability that each object has been selected.

Detailed semantic model. The model must be rich enough to perform queries such as descriptions based on object class (building, etc.) and potentially things like prepositions as well.

2.6 Summary

Having worked with each of these algorithms, we separate them by their effect on the geometry in the rendering pipeline. The algorithms can *exclude* (reject geometry in the database from the current frame of the display), *require* (force geometry in the database to be included in the current frame of the display), *alter* (change or insert new geometry into the database) and *encode* (change the graphical representation used for display). These operations can occur on the 3D geometry (actual real-world coordinates) in the database or in the 2D (view plane) representation that appears on the display from a given viewpoint.

Table 1 summarizes the properties of the techniques described in this section in terms of these categories. As can be seen, the interface techniques overlap in their domain of operation. Certainly, since certain operations can directly conflict (filtering excludes, but occlusion representation can require), there is a need for a comprehensive architecture.

But the interactions are more complex than simple conflicts. For example, the occlusion algorithm may be able to operate on the "raw" spatial database, while the registration error representation algorithm needs to use the 2D projections of some set of 3D objects–but is that set the same "raw" set, or is it a set modified by the occlusion algorithm? The answer changes, in this case, based on whether the representation would spread across multiple occlusion layers.

Technique	Action
Information filtering	exclude–3D
Occlusion representation	require–3D, encode–3D
Error adaptation	encode–2D, alter–2D
Label placement	require–2D, alter–2D
Multimodal interaction	exclude–3D, require–3D alter–3D

Table 1. Summary of the properties of the different techniques.

3 Proposed Architecture

At first sight, one might assume that a pipeline architecture is sufficient. From the database, the information filter selects only the most relevant objects. Occlusion properties for those objects would be calculated, the effects of errors introduced and labels drawn into open areas on the screen space. However such a pipeline explicitly assumes that there is no interaction between the different techniques. However, as Table 1 shows, the different techniques directly interact in the same space and so there is a possibility of conflict. For example, suppose the information filter determines that an object I is important but a second object U is unimportant. If U lies in front of I, and occludes it from the user, the occlusion system must still take account of the fact that U is there and might have to draw some part of the geometry of U to make sure that the position of I is viewed correctly. Similarly, when placing the label for I on the screen, the label needs to take account the projection of U into the display.

Therefore, simple pipeline architectures are not sufficient. Rather, we believe that a mediator architecture, such as the one shown in Figure 6, is required. This architecture is composed of four main components: the context, the display techniques, the mediator, and the display.

3.1 The Context

The common factor which ties the different techniques together is the need for the *context* within which the system is operating. Context is defined to mean the set of all quantifiable information about all of the exogenous inputs which act on the user interface system. The context includes:

Tracker data. This specifies position and orientation as well as tracker error.

User preferences. User specific choices about field-of-view and other types of adjustable tuning parameters

Ambient conditions / user psychological state. The first part refers to the environment within which the system is

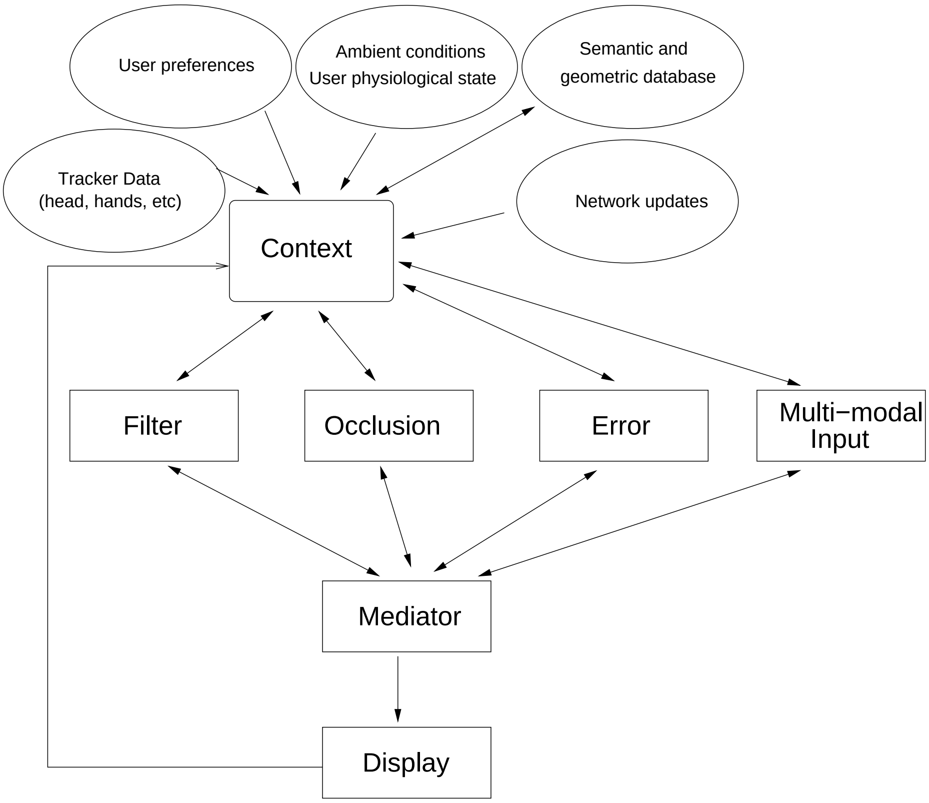


Figure 6. The proposed architecture.

operating. If the system is, for example, rendering graphics on a see-through display, what does the background look like? The second part refers to the user's current mental state. As a user becomes more stressed, their ability to process information changes.

Semantic and geometric databases. The databases store information about the contents of the environment. The semantic data is domain-specific and stores information about the meaning of objects (and hence their importance). The geometric database is a physical description of the environment and specified the actual, sizes, shapes and locations of objects.

Network updates. Almost any real system will receive network updates from objects being created, modified or deleted [5].

The context also includes the current state of the display itself.

3.2 The Interaction Techniques

This is composed of the set of techniques which were described in Section 2. Each technique can construct its own "view" of the context. For example, the culling step of the filter does not work with the geometric model of the environment. Rather, it works with the foci and nimbi of the objects. Similarly, the occlusion system might develop a view of the environment which only consists of occluding contours such as entire buildings. Each display technique issues high level instructions which indicate what display activities should be carried out. These high level instructions could be based on the taxonomy described in Subsection 2.6.

3.3 The Mediator

Given a set of inputs from the different techniques, the mediator is responsible for resolving conflicts and developing a final output. The output to the display will constitute a series of graphical behaviors, such as animations.

3.4 The Display

The display is responsible for the low level rendering of the graphical state to the user. It receives a series of behaviors and continues to render each object with those behaviors until a new behavior is specified.

4 Discussion Topics

The last section described an architecture which is very different from that encountered in traditional VR and AR systems The path between the context information and the final display can be affected by many factors. Each algorithm can read context information from the database as well as add back its own information, in case an algorithm needs to know what another is doing. These algorithms send their requests for data display to the mediator, which prepares the final display for the user to see, and may also feed control information back to the algorithms[1]. However, this architecture has a number of open issues which need to be addressed:

- How should the different components of the context be modeled? What fields are sufficient to describe tracker data, user preferences, conditions, psychological state, the databases and the network updates? The sophistication of each description determines

- How should the output from each display technique be described? Although the taxonomy provides some guidance, it does not describe specific encoding techniques which could be used.

- What is the correct choice for a mediator? There are a number of different candidates including agent-based approaches (for example, bidding or blackboards), expert systems (which can use completely arbitrary and unstructured rules) and

- What set of behaviors and capabilities should be supported by the display? The types of behaviors determine the complexity of the display system which can be implemented.

References

[1] B. MacIntyre, E. M. Coelho and S. J. Julier. Estimating and Adapting to Registration Errors in Augmented Reality Systems. In *Proceedings of the 2002 IEEE Virtual Reality Conference*, pages 73–80, Orlando, FL, USA, March 2002.

[2] P. Baudisch. *Dynamic Information Filtering*. PhD thesis, Darmstadt University, 2001.

[3] B. Bell, S. Feiner and T. Höllerer. View management for virtual and augmented reality. In *Proc. ACM UIST 2001 (Symp. on User Interface Software and Technology)*, pages 101–110. ACM Press, 2001.

[4] S. Benford and L. Fahlén. A Spatial Model of Interaction in Large Virtual Environments. In *Proceedings of ECSCW '93*, Milan, Italy, September 1993.

[5] D. Brown, S. Julier, Y. Baillot, and M. Livingston. An event-based data distribution mechanism for collaborative mobile augmented reality and virtual environments. In *Proceedings of the 2003 IEEE Virtual Reality Conference*, pages 23–29. IEEE, March 2003.

[6] S. Feiner, B. MacIntyre and D. Seligmann. Knowledge-based augmented reality. *Communications of the ACM*, 36(7):52–62, July 1993.

[7] C. Furmanski, R. Azuma and M. Daily. Augmented-reality visualizations guided by cognition: Perceptual heuristics for combining visible and obscured information. In *Proceedings of IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2002)*, pages 215–224, Sept. 2002.

[8] S. Julier, M. Lanzagorta, S. Sestito, L. Rosenblum, T. Höllerer and S. Feiner. Information Filtering for Mobile Augmented Reality. In *Proceedings of the 2000 IEEE International Symposium on Augmented Reality (ISAR 2000)*, October 2000.

[9] E. Kaiser, A. Olwal, D. McGee, H. Benko, A. Corradini, X. Li, S. Feiner, and P. R. Cohen. 3d multimodal interaction in augmented and virtual reality. In *Proc. International Conference on Multimodal Interaction-Perceptual User Interfaces (ICMI-PUI 2003)*, Vancouver, BC, Canada, 5–7 November 2003.

[10] M. A. Livingston, L. J. Rosenblum, S. J. Julier, D. Brown, Y. Baillot, J. E. Swan II, J. L. Gabbard, and D. Hix. An augmented reality system for military operations in urban terrain. In *Proceedings of Interservice/Industry Training, Simulation, and Education Conference*, page 89, Dec. 2002.

[11] M. A. Livingston, J. E. Swan II, T. H. Höllerer, D. Hix, S. Julier, Y. Baillot and D. Brown. In *Proceedings of the 2003 IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2003)*, Tokyo, Japan, October.

[12] B. MacIntyre and E. Coelho. Adapting to dynamic registration errors using level of error (loe) filtering. In *Proceedings of the 2000 IEEE International Symposium on Augmented Reality (ISAR 2000)*, October 2000.

[13] B. MacIntyre and S. Feiner. A distributed 3d graphics library. In *Proc. ACM SIGGRAPH 98*, pages 361–370, Orlando, FL, USA, 19–24 July 1998.

[14] J. Pittman, I. Smith, P. Cohen, S. Oviatt, T. Yang. Quickset: A multimodel interface for military simulations. In *Proceedings of the 6th Conference on Computer-Generated Forces and Behavioral Representation*, pages 217–224, 1996.

[15] B. J. Rhodes. WIMP Interface Considered Fatal. In *Proceedings of the IEEE VRAIS 98 Workshop Interfaces for Wearable Computers*, 15 March 1998.

[16] D. Schmalstieg and G. Hesina. Distributed applications for collaborative augmented reality. In *Proceedings of the 2002 IEEE Virtual Reality Conference*, pages 73–80, 24–28 March 2002.

[17] W. H. Teichner and J. B. Mocharnuk. Visual search for complex targets. *Human Factors*, 21:259–275, 1979.

[18] E. R. Tufte. *Envisioning Information*. Graphics Press, Cheshire, CT, 1990.

[1] This is in sharp contrast to Repo-3D [13] or *Studierstube* [16] in which the emphasis is to place all application data within the scenegraph.