Authoring of Physical Models Using Mobile Computers

Y. Baillot, D. Brown, and S. Julier
Virtual Reality Laboratory, Advanced Information Technology Division
Naval Research Laboratory, Washington DC, USA
{*baillot,dbrown,julier*}*@ait.nrl.navy.mil*

Abstract

Context-aware computers rely on user and physical models to describe the context of a user. In this paper, we focus on the problem of developing and maintaining a physical model of the environment using a mobile computer. We describe a set of tools for automatically creating and modifying three-dimensional contextual information. The tools can be utilized across multiple hardware platforms, with different capabilities, and operating in collaboration with one another. We demonstrate the capabilities of the tools using two mobile platforms. One of them, a mobile augmented reality system is used to construct a geometric model of an indoor environment which is then visualized on the same platform.

1 Introduction

One of the most important consequences of the recent advance in wearable computing systems is the ability to provide context aware computing. Using a suite of sensors, computers can detect and tailor their output to the user's current state. To achieve this adaptability, the system must possess *user models* and *physical models* [9]. A user model describe a user's "internal state" (such as time of day, topic of conversation, identity of the participants [7]). A physical model is a description of the environment and might contain information such as the location and size of objects of potential interest. The required level of detail for each model depends on the application in question. In [9], Starner *et al.* discuss how a set of wearable computing projects can be classified in terms of their reliance on these different types of models. At one extreme, systems such as the Remembrance Agent [7] (a text-based prompting system) relies almost exclusively on the user model. At the other extreme mobile augmented reality systems such as Stochasticks [5] depend on physical models to achieve accurate registration.

Current mobile context aware systems are largely spatially oriented. Many PDAs come with applications which show a user's position on a 2D map. The Context Compass [10] extends a 2D map with spatialized information. Mobile augmented reality systems, such as the Mobile Augmented Reality System [4] and the Battlefield Augmented Reality System [6] require extremely accurate 3D physical models. However, despite the importance of physical models, the issue of building these models has received scant attention. Indeed, a literature survey suggests that no current wearable computer platforms have been utilized to directly build 3D models.

We believe that a mobile computer is an ideal platform from which models can be built for three reasons. First, the surveying and modeling can happen at the same time and place, eliminating the transfer of data between a surveyor and the modeler, which can involve corruption or misinterpretation of the data. Second, because the user is constructing a model directly in the context of the environment where it will used, the model can be checked right away against the environment to avoid the verification and re-measuring cycle between the modeler and surveyor. Finally, a multi-user collaboration can achieved by naturally extending the system from one mobile computer to multiple mobile computers.

In this paper, we develop a series of methods for creating and maintaining 2D and 3D maps using mobile computing systems. Our target platform is the mobile augmented reality system which is shown in Figure 1. However, the approach described here can be generalized in two important respects. First, the system is *not* restricted to a mobile augmented reality system. Many of the techniques can be used on opaque 3D displays (such as a laptop computer) and opaque/see-through 2D displays (such as a Xybernaut system) with or without precise tracking systems. Second, because the metaphors rely on the application of a series of constraints, these constraints can be provided by multiple users collaborating with one another

Figure 1. The BARS mobile augmented reality system. On the right of the user is the computer. The HMD is equipped with a GPS antenna and a USB camera. The insert shows an image which was taken by a camera placed at the eye piece of the head mounted display.

to construct a model using a heterogeneous set of mobile computing platforms. Conversely, a single user can construct a model over a period of time.

The structure of this paper is as follows. Section 2 describes the problem of modelling in more detail. The new modeling approach is described in Section 3. The initial implementation is described in Section 4. Results are given in Section 5 and conclusions are drawn in Section 6.

2 Problem Statement

Consider the following scenario. A set of users wish to construct a 3D physical model of an urban environment. Each user possesses a mobile computer which has a tracking sensor (position and/or orientation) and is capable of generating 3D graphics. All computers are assumed to be networked together. The user needs to be able to construct simple graphical primitives (points, polygons, boxes, labels and so on) and have a capability to change the attributes of each primitive while performing data acquisition.

Some of the earliest work in model construction was carried out using CAD systems. However, CAD systems are typically designed for desk top computers and their modeling paradigms cannot be directly transfered to a wearable platform because the input capabilities are different. For example precise hand tracking is currently a problem even if emerging technologies may soon provide a solution (e.g WearTrack [2]) In addition, the input paradigm of a CAD system do not fully exploit the capabilities of a mobile AR system. For example, with a mobile AR system a user is able to directly align the generated graphics with the real-world to validate the model.

The authoring methods used in Virtual Reality introduces other difficulties and considerations. Direct manipulation methods use a metaphor similar to a sculptor — using a variety of tools a user can create and place objects out of simple primitives. Constraint-based methods apply constraints to directly manipulated objects. Some example of such systems are the Juno-2 constraint-based drawing system [3] or the VR system Sketch [11] which extends the concept of a constraint-based user interface by mapping various types of gestures to types of constraints. However, direct manipulation leads to imprecision because it relies on the perception of the user to judge how close the virtual object matches the real object, rather than its direct visual comparaison to real data. Furthermore, this type of manipulation needs accurate 6DOF tracking of the user's hand which is a problem. Rather, it is likely that most gestures can only define selection rays or cones in the real world. Perception issues have been reduced by system using constraints.

3 Authoring Techniques

The authoring of the model is conducted in two steps. First, temporary construction points are located in the model using one set of techniques. Then, using another set of techniques, primitives can be built using these construction points as anchor points. Finally, models can be modified using additionnal techniques.

A traditional pointing device (mouse, track pad, touch screen, etc) is used to select points by moving a cursor on the visible part of the model. When necessary, values can be entered using either a physical or a virtual transparent keyboard overlaid on the view. The switching between the different authoring mode is currently achieved by a menu interface. The model database is shared among several users equipped with the system so that the construction can be done collaboratively. Several users can work on different parts of the same model in parallel, or they can help each other to construct common features. If collaboration is used, the requests are carried out on a first-come-first-served basis and interpreted as if coming from a single user.

3.1 Viewing and Manipulating the Model

Our authoring techniques utilize two visualization modes for the 3D model: overlay mode and map mode. The overlay mode (Figure 2) provides the user with a 3D view of the model as seen from the user's viewpoint in the real world. In this mode and if a see-

Figure 2. Normal view of the system. For clarity, we use opaque mode. When the display is see-through, the virtual objects are superimposed on the real world. The three boxes on the right hand side (trash, create1 and create0) are part of a direct manipulation interface and are not utilized by the model construction system.

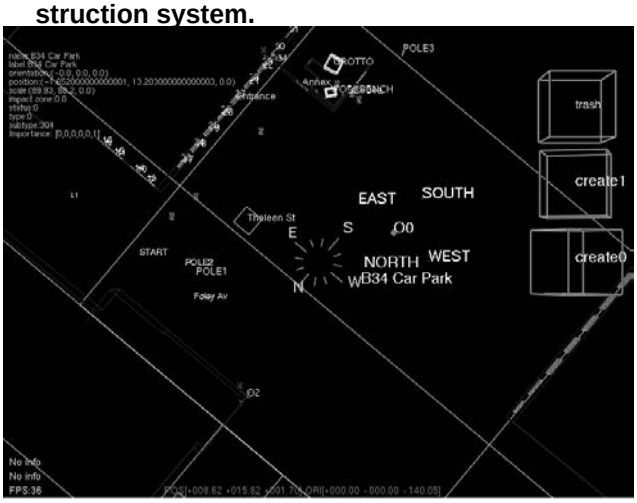


Figure 3. The map mode shows the current model in 3D (shown here in top view) in miniature.

through display is used, primitives can be constructed directly on the model to provide the user a direct view of the final result in place in the environment. The map mode (Figure 3) represents a god's eye view showing the world in miniature. The position of the map is such that the user's location on the map is always centered on the screen and the map's initial orientation is such that the user's viewpoints on top of model looking down. In addition the map can be tilted and zoomed to provide a view of the model from any vantage point.

The system is most effective with accurate tracking of the user's position and head orientation in the area being modeled. However, the system is still useful with a single or no tracker. In the absence of position and head orientation tracking sensors, the user's position and orientation in the model can be modified using the mouse. Dragging the mouse up and down moves the user's position respectively forward and backward, while dragging the mouse left and right modifies the azimuth of the forward direction to change orientation. In this configuration, the system is basically used as desktop authoring tool but has the advantage that it can be used on the modeling site on a wearable computer.

By adding a position sensor, like a GPS or a pedometer, the position of the user's viewpoint in the model can be constrained to the position of the user in the world. In this mode the mouse is still used as described previously to control the orientation of the user in the model. In this configuration, features can be placed in the model by just walking to the location and adding the feature. For example, the footprint of a building can be added to a map just by walking on the roof of the building with a GPS and adding a point on the model at the location of each corner.

If position and orientation trackers are both available, the system can be used as a complete mobile AR system, which can register the model on the environment as the user moves in the real world. The user can focus solely on building the model and not using a mouse to tweak her viewpoint of the model, as the viewpoint is automatically updated to be registered with the real world.

In this mode, the tracking systems must be properly calibrated, or else the registration between the model and the real objects will be poor. Calibration involves calculating transformation to be applied to the position and orientation sensor data to properly register the graphics with the real world. We have developed a calibration framework which allows us to use any type of sensor and calibrate the system simply using calibration landmark and virtual/real registration. This is the object of a future paper [1].

3.2 Creating Construction Points

The user must construct some points and lines at diverse locations that represent the vertices of the primitives to be built. These primitives are created temporarily during the model construction and are later removed. We will now list the techniques we have implemented to specify the locations of the points. In the following, "selection ray" refers to the line expressed in model space that passes through the current viewpoint and the cursor location on the screen.

- **Direct coordinates entry.** In this mode the user enters the three-dimensional coordinates of the new construction point using the keyboard (or virtual keyboard) in a textbox. The vertex appears in the model and can be selected like any other vertices of the primitives in the model. This mode can be used to enter the first vertices of a model to use as reference points, which user will most likely have measured with a tape measure or range finder.
- **Surface intersection.** A vertex can be created at the intersection of the selection ray and the surface it first intersects in the model. This provides the user with way to locate new vertices on the ground or on any surface of the model. For example, the corners of a window on a wall of a building already in the model can be located using this capability.
- **Lines intersection.** This mode allows the user to create a vertex at the intersection of two construction lines. Each construction line is added by pointing on the screen to cast a ray (Figure 4). Each new line is intersected with the previous lines to eventually create potentially several intersection points at a time. A line intersection is detected when the distance between two lines is below a user tunable value. This mode requires that rays must be cast from at least two locations to provide an intersection. Because the database is shared this can be done by two cooperating users from their respective viewpoints. For example, two users performing just four selections can identify the corners of the face of a building.
- **Distance from two points.** This technique works on the following theory. In a 2D plane, given two point locations and a distance from each of them, one can create circles around the points that have radii equal to those distances. These circles will intersect in two places, giving two possible new points. In the 3D model, the user can select two vertices of the same height–we assume the plane is parallel to the ground. Then the user can enter a distance, and by using the above technique, the system presents two new points. The user can easily choose which of the new points is correct. This mode is best used when creating a floor plan and new vertices can be located in relation to other vertices using a measuring device.
- **Distance from three points.** This mode is an extension of the previous one applied to points in space instead of on a plane. The user chooses three vertices in the model and specifies a distance. Using trilateration, two points are identified which verify the distance constraint (Figure 5). This mode is convenient when building new vertices without any assumptions on the coplanarity of the selected vertices with the new vertex.

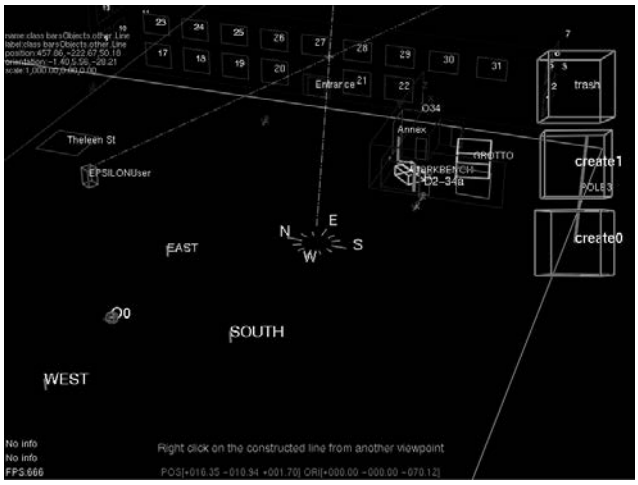


Figure 4. Lines intersection technique. Here two users (the current user's position is denoted by the compass rose at the center of the screen, the remote user's avatar is shown as the light gray box with the label "EPSILONUser") are shooting rays in the environment by pointing a common point on the real world (not shown here). The result is that the point they identified (bottom right corner of window 27) is created in 3D. Note that a single user could do this operation from two different viewpoints.

3.3 Adding Geometric Primitives

Once construction points have been located, they can be used to build new model primitives. We will now list the techniques we have implemented to construct new primitives. These techniques can also be used to select existing primitives for removal. Primitive attributes like position, orientation, scale, color, label or texture can be modified using the keyboard to enter the new values.

- **Lines.** A three-dimensional line can be built in the model by selecting two anchor vertices as its endpoints. Connected line segments can be constructed by specifying a series of points.
- **Contour extrusion.** After the user enters the extrusion height in a textbox, the contour to extrude vertically is traced as a series of line segments described above. Then the contour is extruded along the line segments. The model of a room can be constructed rapidly using this technique when using the overlay mode in the room for which a model is built. This can be simulated

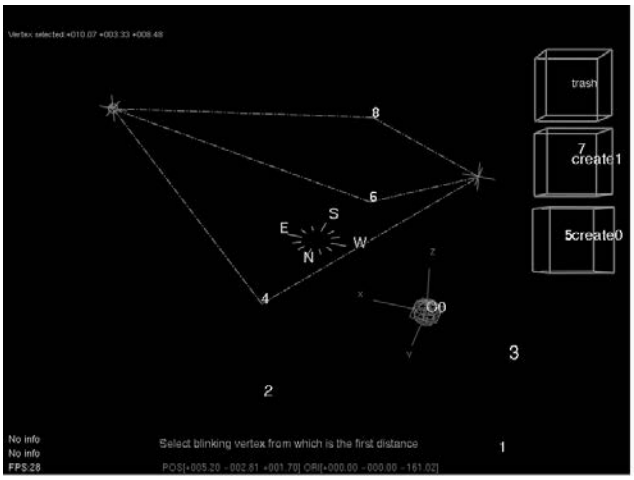


Figure 5. By giving the distance from three points (4, 5 and 8), two points are possible solutions. The lower point in the image is currently selected.

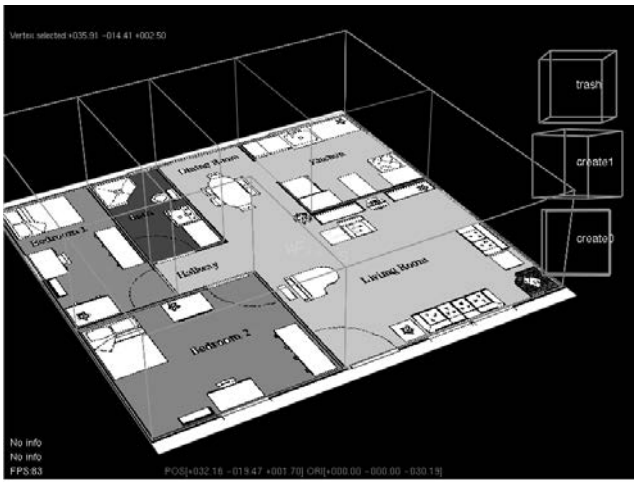


Figure 6. By loading a bitmap on the ground representing a floor plan, the user can locate vertices by creating intersection points with the bitmap, drawing the contour and extruding the walls.

if a bitmap representing the floor plan of the room is loaded on the ground first. The user can then draw the contour on the map and extrude the wall to produce the room (Figure 6).

- **Quad.** A quad is created by selecting three points. The first point is one corner of the quad. The second point belongs to one of the side of the quad that is adjacent to the first point. The last point is on the opposite corner of the quad from the first point. This constraint guarantees that a unique orthogonal quad can be found. The size and position of the quad is constrained by the first and third point, while the second point constrains its orientation. This mode is useful for building doors or windows on buildings because the constraints help prevent location errors and allow for non-orthogonality of the wall's corners.
- **Box.** A new box is created using a similar technique to that described for the quad, but with four vertices. The first face of the box is created using the first three points as if the user is creating a quad. The fourth vertex is placed anywhere on the side of the box opposite to the first side. The box is then constructed using the location of the fourth point to determine the depth of the box. This again guarantees that the box created is orthogonal.

3.4 Modifying The Model

- **Bitmap mapping.** A bitmap can be used to texture a polygon. The bitmap is loaded and is centered at a point determined by the intersection of a selection ray with a surface of the model and its orientation is such that bitmap is on the same plane as the surface. The bitmap can be arranged on the surface by mapping any point of the bitmap to any vertex of the surface of the model on which the bitmap is applied. The first vertex selected on the bitmap is mapped to a vertex of the model by translation only. Any further vertex is remapped keeping the last vertex fixed. The system perform the mapping by scaling and rotating the map so that the new vertex is correctly remapped.
- **Attributes modification.** Once the geometry of the model is constructed, the parameters of the objects can be modified. Example of such parameters can be color, label, type and so forth. Geometry (scale, position, orientation) can be modified as well while it is not the main use of the system.

4 Implementation

Our research in outdoor mobile AR system required us to construct a dense model of the environment. In

addition to buildings, we need to model distinct features such as windows, doors, electrical switches, or valves. We have implemented our modeling system on two systems, a Pentium class pen-based computer from Fujistsu and a custom 3D stereo mobile AR system built for the Battlefield Augmented Reality System (BARS) project shown Figure 1. The modeling application is built using the main API we are developing for BARS.

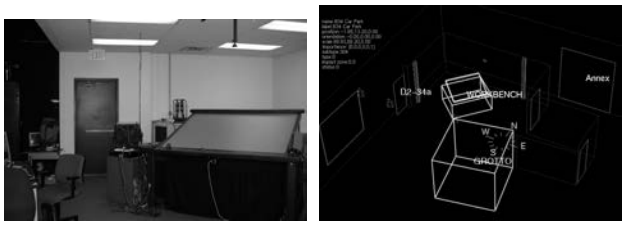
The pen computer is a standard commercial computer which is able to run 3D graphics at a reasonable speed in our case. Three-dimensional features are more conveniently modeled using an AR version of the system such as our BARS wearable system. The BARS wearable is built using an EBX form factor motherboard equipped with AGP hardware able to produce stereo imagery. A gyro-based mouse is used as pointing device. A see-through HMD is used to overlay the model in stereo on the real world. A kinematics-differential GPS and a gyro-based sensor track the user's head position and orientation. Finally, a wireless network device allows the system to communicate with others like it to share the model database. The user can do all the modeling alone in this case, or she can collaborate with others using similar units to build the model faster.

Several authoring method requires individual vertices of the model to be selected. A selection cone whose axis passes through the viewpoint (**A**) and the cursor (**B**) is formed in the reference frame of the model. We have the problem that the user could be pointing at a number of different vertices in this cone. We choose a vertex as follows. The vertex **C** of the model that allows forming the smallest angle between the vector **AB** and **AC** is considered as selected. This compensates for pointing errors, which could occur if only the distance of the vertex to the pointing ray was considered. This selection is easily implemented using the selection buffer in the graphics hardware to identify the limited quantity of vertices that project on the screen in the neighborhood of the cursor.

5 Results

Most of the techniques presented in this paper are exact in the sense that what the user input is what is going to be created. For example, the direct coordinate entry method involves the user entering coordinates which precision relies only on the way the coordinate has been measured. We do not focus of the result of using these techniques because they are not novel in their concept, rather they have been gathered from different systems and adapted into a unique mobile system.

We used the pen computer to create the vertices



(a) The laboratory. (b) 3D map.

Figure 7. The test environment. The 3D model was constructed for an augmented reality test bed and the geometry of the laboratory is known very precisely.

of buildings floor plan on the NRL base. One user was equipped with a laser range finder and the computer, while another was standing on the other side of the dimension to measure with a mirror. We used the distance from two points and three points methods described earlier to compute the positions of new vertices. Using this technique we created the vertices of the contours of four buildings and extruded them to their measured height in less than an hour. This method gave us good results for the contour of the building floor plan because the range finder gave us centimeter accurate measurements.

We tested our wearable BARS system by constructing the model of the interior of the NRL virtual reality laboratory shown in Figure 7. At the time of this publication, we chose this model over an outdoor model for several considerations. First an accurate model of the inside of the lab has been constructed using tape measuring, therefore, a precise comparison of the quantitative performance of the modeling techniques can be made. In contrast, our outdoor model is still inexact because of the large distances to measure. Second, our tracking system inside (InterSense IS900VET) is much more precise that our GPS/inertial hybrid outdoor. However, because the model and the operating area indoor is scaled down, one can expect that the errors on the resulting model will be scaled comparably with respect to a system operating outdoors. Finally, operating inside allows us to eliminate hardware problem such as capturing video outdoors or the inadequate contrast of our HMD (Sony Glasstron) when used in a sunny day.

The main technique used on the mobile AR system to create vertices is line intersection. Figure 8 shows the user creating a point on the corner of the workbench through line intersection and incorporating that point into a quad which corresponds to the face

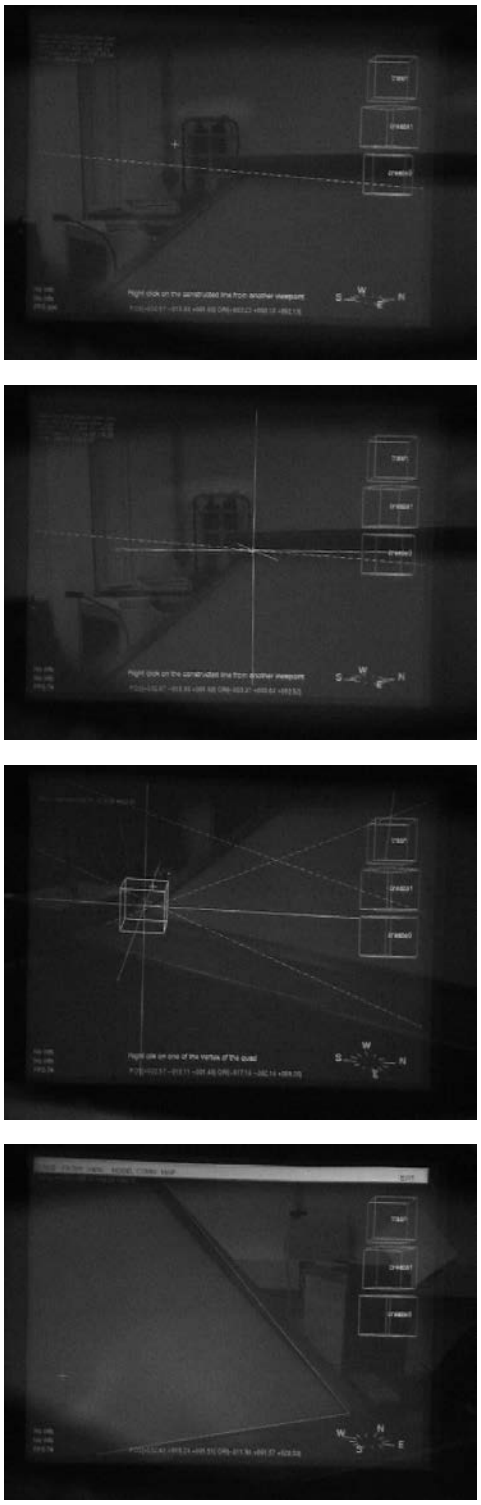


Figure 8. Creating a model of the workbench. (a) The user aligns cross with corner of workbench and a previously entered intersection line; (b) A second line is made and a construction point formed; (c) A construction point selected as part of the quad; (d) The final model.

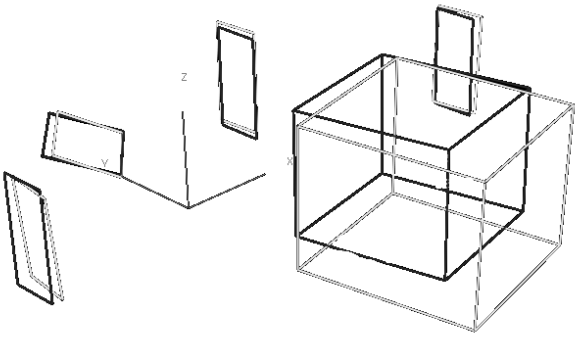


Figure 9. The resulting model. The true model is the lighter gray line, the measured model the darker gray line.

of the workbench. The final model of the VR lab is shown in Figure 9. These results show that the model is accurate, given the perspective of the mobile AR user. However, the GROTTO (large cube-like structure) shows significant errors. These arise from the fact that, within the indoor environment, the user is restricted to stand in a small area. As a result, parallax is limited. Therefore, small angular errors can lead to large modeling errors. However it should be noted that, from the user's operational area, such modeling errors are small and not visually disturbing. We are working on a formal description of the errors involved in the model which will be the object of another paper.

Modeling errors are not only due to head tracking accuracy. We realized that using a mouse cursor to point object can lead to several problems in fact. If the field of view of the graphical projection is not exactly equal to the one of the display, the pointing angle will be incorrect. If the distortion of the display is not negligeable or if the centering of the graphical viewport on the center display is biased then the pointing angle will be incorrect as well. The problem of the field of view and the distortion can be simply solved by using a crosshair at the middle of the display rather than a mouse cursor to point features. This has also the advantage to avoid the user to have his hand fixed when pointing. The centering of the viewport is a manufacturing problem but can be solved programatically using viewport tunning calibration techniques.

6 Conclusions

This paper has described an authoring toolkit which can be used to create 3D physical models of an urban environment by one or mobile users who are in that environment. The toolkit, which employs a variety of point and constraint techniques, can be utilized across

different types of mobile platforms with different capabilities.

While the AR system is more intuitive and easier to operate than the pen system, problems occur due to inadequacies of current mobile tracking systems. Errors in the tracking involve errors in the displayed model and in the selection rays, which in turn produce errors in the model created. The modeling errors are more sensitive to orientation tracking errors, which are unfortunately also the most important. The influence of orientation errors is reduced as the user approaches the target points because the angular displacement is reduced and that produces a smaller position error on the point. However, that distance cannot be too small in many cases. For example, if the target points are defining a window on a building, and the user approaches the building to reduce error due to a high angular displacement, the GPS-based position tracking becomes imprecise because it does not see as many satellites when too close to walls.

The system which was described here can be improved in a number of important respects. First, more sensing modalities can be used. For example, if a user is given a camera visual information can be obtained. This could be used to generate textures or could be combined with a photogrammetric system (such as PhotoModeler) to assist in the model building process. Head-mounted laser range finders can also be used to more precisely localize the point positions.

In addition it is not clear that WIMP-type interfaces are effective with mobile systems. As Rhodes notes [8], WIMP-type interfaces assume fine motor control, large screen area and a concentration on the user interface. To solve this issue, we are working to improve the usability of the system through the use of multimodal inputs which fuse both gesture and speech.

Finally, while our database support hierachy of objects, we currently do not support the capability to constrain the attitude of a modeled object with respect to a parent object. This capability can be usefull if an object attitude has to be changed. It would be prefereable in this case to manipulate only a local referential to which this object is attached instead of changing the attitude of every primitive of the object. We are investigating different techniques to solve this problem.

References

[1] Y. Baillot. A calibration and modeling framework for mobile augmented reality systems. In preparation, 2002.

[2] E. Foxlin and M. Harrington. WearTrack: A Self-Referenced Head and Hand Tracker for Wearable Computers and Portable VR. In *Proceedings of the Fourth International Symposium on Wearable Computers (ISWC '00)*, pages 155–162, Atlanta GA, USA, 16 –17 October 2000.

[3] A. Heydon, and G. Nelson. The Juno-2 Constraint-Based Drawing Editor. Technical Report SRC-131a, Digital Systems Research Center, Palo Alto CA, USA, December 1994.

[4] T. Höllerer, S., Feiner, T. Terauchi, G. Rashid and D. Hallaway. Exploring MARS: Developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computers and Graphics*, 23(6):779–785, December 1999.

[5] T. Jebara, C. Eyster, J. Weaver, T. Starner, A. Pentland. Stochasticks: Augmenting the billiards experience with probabilistic vision and wearable computers. In *Proceedings of First IEEE Internationonal Symposium on Wearable Computers (ISWC '97)*, Cambridge MA, USA, 13–14 October 1997. IEEE Press.

[6] S. Julier, Y. Baillot, M. Lanzagorta, D. Brown, and L. Rosenblum. Bars: Battlefield augmented reality system. In *Proceedings of the NATO Symposium on Information Processing Techniques for Military Systems*. NATO, October 2000.

[7] B. Rhodes. The wearable remembrance agent: a system for augmented memory. In *Proceedings of First IEEE Internationonal Symposium on Wearable Computers (ISWC '97)*, pages 123–128, Cambridge MA, USA, 13–14 October 1997. IEEE Press.

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

[9] T. Starner, B. Schiele, B. J. Rhodes, T. Jebara, N. Oliver, J. Weaver and A. Pentland. Augmented realities integrating user and physical models. In *Proceedings of the 1998 International Workshop on Augmented Reality*, San Francisco CA, USA, November 1998.

[10] R. Suomela and J. Lehikoinen. Context compass. In *Proceedings of the 2000 Fourth International Symposium on Wearable Computers (ISWC '00)*, pages 147–154, Atlanta GA, USA, October 16–17 2000. IEEE Press.

[11] R. C. Zeleznik, K. P. Herndon and J. F. Hughes. SKETCH: An Interface for Sketching 3D Scenes. In *Proceedings of SIGGRAPH '96*, pages 163–170, Los Angeles CA, USA, 1996. ACM.