Creating Meaningful Environment Models for Augmented Reality

Category: Research

Abstract

This article introduces a framework to generate three-dimensional models for Augmented Reality (AR), which include semantics. Augmented reality applications often associate 3D models with some aspect of reality. While the addition of semantics to 3D models has been studied previously in the field of intelligent virtual environments, AR is different because the semantics are attached to objects existing in the real world. Moreover, little attention has been paid to the process of creating real-world models containing such information. This work describes a method for the creation of “meaningful” models of the environment in an AR application named InventAry. Assisted by an ontology, InventAry enforces the creation of combined geometric and semantic environment model.

1 Introduction

Augmented Reality (AR) extends the user’s experience of the real world through the addition of virtual objects that appear to coexist and interact with the real counterparts. In most cases, the application is limited to interact with a few physical objects, and their meaning is expressed procedurally as part of the application logic. As AR systems tend to integrate activities dispersed in wider areas they require additional information about the surrounding world, and enhanced mechanisms to manage such information.

Ontologies have been widely used for context management purposes (among others) in ubiquitous computing [2]. Ubiquitous computing (Ubicomp) applications, like AR applications, require information about entities in the real world. However, while AR mainly deals with precise 3D geometry, Ubicomp tends to use topological concepts instead. This work combines the two approaches by merging geometric models with topological and other semantic information.

Semantic representations have also been the focus of various studies in Virtual Reality (VR). Luck and Aylett [1] aimed to incorporate intelligent agents and avatars in virtual environments, and introduced the concept of Intelligent Virtual Environment (IVE). Latoschick et al. [7] presented a symbolic representation to capture semantic and geometric knowledge in a unified knowledge base, later extended to support multi-modal interaction [8]. Similarly, Irawati et al. [5] explored how an ontology of the environment can be used to constrain multimodal interaction. Lugrin and Cavazza proposed a framework integrating reasoning, graphics, and physics on top of a commercial game engine [10]. Gutierrez et al. [3] added semantic descriptors based on MPEG to scene graphs in order to select different models for the same object depending on the hardware. Pitarello [13] introduced semantic zones in X3D worlds using the Web Ontology Language, OWL. Kalogerakis et al. [6] suggested an ontological representation combining knowledge of different domains and VR scenes. In their approach, 3D entities with their corresponding 3D representations are described using OWL graphs.

While all these projects use approaches that are related to our problem domain, VR deals by definition only with virtual objects and therefore virtual semantics. There is no need to strictly model aspects of the real world or deal with the dynamics of the real world. There is little work on modeling non-geometric aspects of AR application. One exception is the work by Newman et. al [12], which describes a relational database incorporating geometric and semantic aspects of an office building as well as an AR tool to update portions of the database. However, the semantic interpretation of the relational data is left to the application. The work described by Höllerer et al. [4] uses a semantic database to infer topological information relevant for a mobile AR user, but the authors do not describe how to create this database. To the best of our knowledge, no AR modeling system specifically aimed at creating semantic information exists to date.

In contrast, our work focuses on the creation process for AR models. Semantics has been treated mostly as a separate asset of environment models in the past, adding it in a separate annotation process . Our approach takes into consideration both geometric and semantic attributes when creating the environment model. An ontology assists the workflow by specifying the objects that may be found in the environment and how they relate. The conceptual descriptions are defined in advance with the help of an ontology editor, while the creation of instances representing the concrete entities and their relations is part of the runtime process. The ontology is exploited at runtime to guide the user interface operations of the InventAry application, the purpose of which is used to model the physical environment. Topological restrictions derived from the semantic relations are used as constraints for the user interface operations, which makes the required spatial interaction easier.

2 Semantics for Augmented Reality

The extension of virtual environments with knowledge brings about a representational conflict between the graphical data structures which provide visual representation, and the semantics of the given entity. Following several approaches [8] [6] the system uses a unified representation of an entity at runtime. In order to support interoperability and extensibility of the knowledge base, the ontology and the knowledge models are represented in OWL format, which is a standard for the semantic web. Scene graph files containing geometrical representations are linked to entities in the OWL files through descriptors.

InventAry aims at creating virtual models from wide areas of the real world, extending the scope of AR applications. Therefore, some commonalities and discrepancies with ubiquitous computing can be identified. While both require to know the location of entities in the real world, ubiquitous computing and augmented reality deal differently with location and spatial information. Augmented reality is mostly concerned with the geometrical aspect of spatial relations, specifying entities as collections of points and polygons. Queries and searches for objects in AR often mean casting a ray into a scengraph and looking for collisions. Ubiquitous computing, on the other hand, relies on topological information about locations, stating that certain entities are roughly in a particular location is often enough for Ubicomp. Queries and searches are required to find co-located users, devices and services in such topological networks. One contribution of our ontology is that it bridges these two views of space and provides a way to go from one to the other enabling new applications to be developed that can consider both aspects of reality.

A geometry ontology defines transformations, coordinate frames and shapes, and the relations to express that a certain shape refers to some coordinate frame. Other relations include those that deal with composition of shapes, the $partOf$ and *containedBy* relations. The relations can be used to infer that a certain shape is $partOf$ another shape, and therefore has a reference to its coordinate frame. An ontology describing space extends the geometry domain (Figure 1). It deals with entities that occupy space, defining the general concepts subsuming mobile and fixed entities and the central *SpatialThing*.

The *SpatialThing* subsumes two other powerful concepts, the physical object, and the spatial region. The latter is bound by a shape and it is used to define the operations of region connection calculus [9], which take place between spatial regions. Region connection calculus is used to infer topological information about regions, (i.e. what regions overlap, whether a region is connected to another, etc.). These concepts help to deal with indoors locations and to describe spaces topologically. Topological relations between physical objects are used during modelling to simplify interaction while editing the model. Relations such as *onTopOf*, *on*, *adjacentTo* and others are expressed in terms of the geometrical frame of reference, influencing the associated transformations. They are further explored in the next section.

Domain knowledge is incorporated through extension ontologies, which define new entities and relate them to physical objects in a similar approach to the "represents" relation introduced by Kalogerakis et al. [6]. Domains of interest for AR applications that have been encoded in ontologies include tracking, sensors and locations for activities. A total of over 80 concepts have been implemented throughout the different ontologies, many of which are still being extended as new requirements become evident.

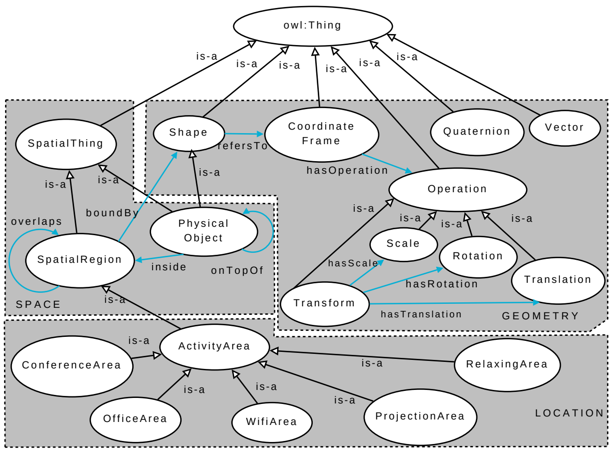

Figure 1: Example concepts connect the Geometry, Space and Location ontologies. Examples of relations other than $is-a$ are shown in blue.

3 InventAry: modeling with semantics

InventAry takes advantage of the ontological description of the world to assist the user in creating a knowledge base of it. The knowledge base contains entities that unify semantic and graphical representation.

3.1 Model Creation

The ontologies define what objects exist in the environment and how they may relate. The user creates instances of those objects encountered while exploring using the InventAry application.

The main activity starts by identifying a potential object as belonging to some category represented in the ontology. At all times, an ontology graph displays the available categories, once an object is recognized, it can be instantiated. This process creates an instance with the default properties indicated by the category. The *Shape* class, in the geometry ontology, defines its default geometry to be a box. Inheriting classes can change their geometric representation.

InventAry changes to edit mode after instantiating an entity, relying in the ontology to simplify the task of editing its properties. In edit mode, the constraints imposed on the classes restrict the properties that instances can have. Based on such restrictions, InventAry shows a list of possible properties and also hints the values for them from the entities already available. For example, the *isTrackedBy* relation of the *TrackingTarget* is restricted to instances of *TrackingSystem*, instances of *Door* can only be placed *on* instances of *Wall*. Taking advantage of such information, InventAry highlights the possible targets for relations (*on*, *isTrackedBy*, etc.) directly in the scene, simplifying the task of editing semantic information.

3.2 Assisted interaction

Regarding manipulation, InventAry supports primitive transformations for placing the object in the environment model (rotation and translation), and to modify its shape to match the real entity (scaling). However, the use of such operations is time-consuming particularly in a mobile setup. For this reason InventAry confines them to topological relations defined in the ontology (Figure 2). Taking advantage of such relations, the user interface simplifies the task of object manipulation. By using such relations as constraints, it is possible to reduce the degrees of freedom needed for interaction, in an approach similar to the one presented by Stürzlinger [15] with the difference that InventAry does not require the offline labeling of every object explicitely with the possible constraints, but allows the in situ selection of constraints based on the location ontology. Topological relations specify, for example:

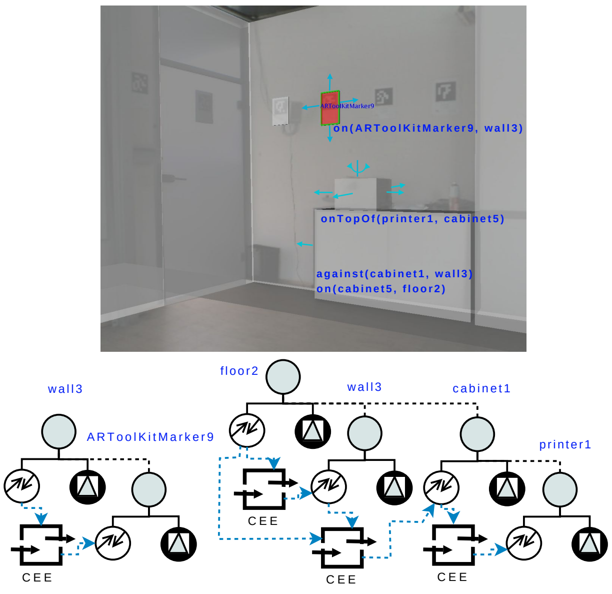

Figure 2: Simple scenes showing topological relations. The left scene shows the relation between $wall3$ and $ARToolKitMarker9$, while the right one includes more objects related to $wall3$; some of which ($cabinet1$) are constrained by more than one relation(on, $against$). Constraint enforcing engines (CEE) connect the objects involved in such relations, and constrain the degrees of freedom for manipulation accordingly. The screen-capture shows possible directions of movement of the objects involved.

$on(i,j)$**:** i refers to j, and placement has to be on a surface of j. Therefore, when manipulating i, the system makes sure that it stays on the surface of j. Once an object is restricted to be *on* another, it has a maximum of three (2 translational + 1 rotational) degrees of freedom, it is possible to further constrain objects, for example a wall can be connected to another wall. This relation is quite useful, since many objects sit on top of others.

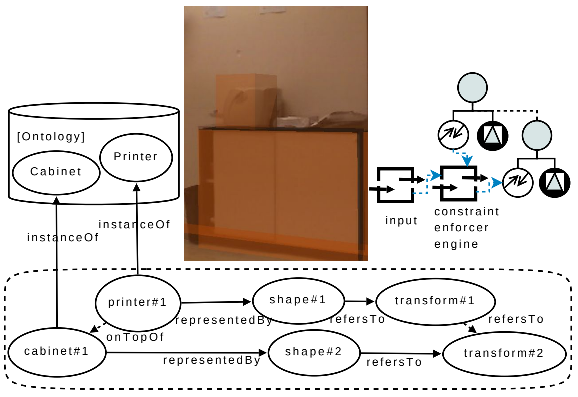

Figure 3: *printer*1 sits *onTopOf cabinet*1. A portion of the semantic net shows some of the relations and constraints between the two objects. In the associated scene, a constraint enforcer engine receives input data to manipulate the object and actualizes the scenegraph.

$onTopOf(i,j)$: i refers to j and the $+y$ component of the frame of reference of i is restricted to the $+y$ component of the frame of reference of j. It is a particular case of *on*, where the surface to which objects are restricted is the top surface. Once an object is restricted to be *onTopOf* another, has also a maximum of three degrees of freedom, however it can not be moved out of the surface of the objects it sits upon.

$inside(i,j)$: i is bound by j.

$against(i,j)$: i refers to j, such that $on(i,j)$ but not $onTopOf(i,j)$. That is, it refers i to one of the vertical surfaces of j. Therefore also restricting movement to three degrees of freedom.

$hangingFrom(i,j)$: similarly, i refers to j, and its y component is restricted to the $-y$ component of j. However, in this case the object can be moved in the y axis, as long as it does not colide with j.

Taking advantage of location relations such as the ones defined above combined with class restrictions, it becomes much easier to position an instance of *Door* (an *on* relation applied to *Door* instances can only target *Wall* instances). First, the instance is created, then an *on* property is selected relying on semantics to find and highlight the possible entities that could be related through that property. Once the application highlights the available *Wall*s, one of them is selected either by clicking on it or by selecting from the drop down menu. The spatial relations then serve as constraints, since now the geometric operations are restricted to the surface of the *Wall* instance reducing the translations and rotations needed to position the *door*1 accurately in space. Further domain ontologies might introduce new class restrictions, even to already existing classes.

Once the instance satisfies the user's intention at modeling reality it can be added to the environment model. The semantic model of the environment is extended with the new entity. The addition of a new instance to the environment model may trigger subsequent tasks depending on the ontology and whether it is able to find relations with environment objects that the user did not consider. For example, when adding an instance of *Projector* and an instance of *Screen* the ontology derives the fact that a *DisplayDevice* is available in the area.

3.3 Searching with Semantics

A core task supported by the extra knowledge is that of searching. The rich description of the environment allows querying at different semantic levels. Once the environment model has some content, it is possible to use the semantic description to find objects. The use of ontologies to define "entities" of the environment in a "structured" way, allows using advanced querying mechanisms to find objects in the environment model. Usual queries could be "show me all monitors larger than 19 inch", "show me the route to the building exit". These queries require, of course, that the entities be defined in the ontology. The result of a query, be it a single object or a collection thereof, can be then used as input to a filtering mechanism for visualization. Using advanced visualization techniques such as those suggested by Mendez [11], it is possible to assign visualization styles to objects categories in order to visually discriminate them.

Perhaps the main advantage of having a representation of knowledge is the possibility to associate it to a reasoner and let it automatically derive new facts. A reasoner can be used to infer the existence of new *entities*, as the model is created. For example, in an ontology that encodes information about activities, several zones are defined from the physical objects that can be found in them. An *OfficeArea* is defined as containing some *Seat*, *Desk*, *NetworkAccess*; while a *TrackingArea* contains some *TrackingSystem*. During model creation, the instantiation of entities derived from *Seat*, *Desk*, *NetworkAccess* will trigger the addition of an *OfficeArea* instance. The removal of such entities, will trigger the automatic invalidation of the *OfficeArea* instance, producing a sort of automatic maintenance. On the other hand, a reasoner can also be used to infer new *relations* between entities. An ontology about tracking describes several concepts associated to tracking systems. During model creation, stating that some physical object *isTrackedBy* a tracker (e.g. an instance of *ARToolKitMarker*) will add a relation stating that the object *refersTo* the tracker, associating thus the object's shape to the transform of the tracker.

4 Implementation

InventAry is implemented as an application of the *Studierstube*[1] framework. Studierstube and its associated libraries have components to handle functionality associated with AR applications, such as video input from a camera, reconfigurable tracking based on a dataflow network, all within a scenegraph framework. A proof-of-concept scenario has been implemented using an outside-in infrared tracking system in an indoor setup. We are working towards extending this scenario to use a hybrid tracking system for wide area coverage.

Figure 4 shows the architecture of InventAry. The Pellet reasoner is used, at the current stage, to check the validity of the model after a user inserts an instance in it, and to instantiate entities and relations derived (by inference) from the ones in the world database. Special scenegraph nodes have been developed to serve as repositories for semantic entities.

Custom engines (i. e., functional constraints in the scene graph) have been devised for the purposes of constraint enforcing. The input that modifies the object in 3D space is captured by one a constraint enforcing engine, which analyses the relations of the given object and corrects the positioning accordingly. Although it would be possible to carry out the necessary calculations for constraint enforcing in the reasoner based on accurate descriptions of the operations, it is infeasible to accomplish real-time response in matters related to geometric transformations during interaction. Therefore they are carried out as an extension connected to the interaction component.

5 Conclusions and Future Work

This article takes a first step towards the systematic creation of semantic annotations for AR applications, build "meaningful" models

[1]http://www.studierstube.org

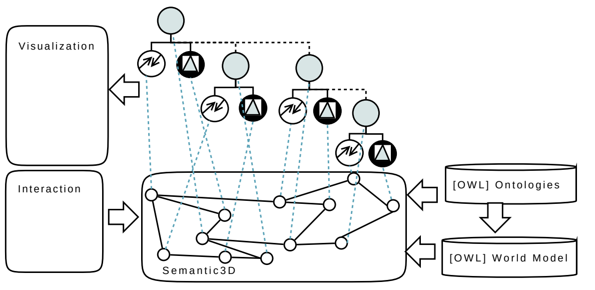

Figure 4: Special scenegraph nodes directly related to semantic entities are used to represent the world objects. These nodes are also part of the semantics, assuring that a unified representation is maintained at runtime.

of the environment. First it has been used as a form of classification for the primitives available, providing proper taxonomy organization for them. Second, by representing topological relations explicitly and derive geometric restrictions, the task of positioning new objects is reduced to relating them to already existing objects and then using constraint-guided spatial interaction for the final placement.

The resulting model need no longer be annotated separately with semantic markup, as used in other approaches. The semantic markup is readily accessible and can be exploited by other applications. For example, AR information systems such as [4] can apply visualization strategies based on semantic filtering.

The ontologies used by InventAry are being extended to address new issues, and to find new ways to support the modeling task. We are working towards extending the geometry ontology, in order to represent all aspects of a scenegraph. Another goal of InventAry is to support the creation of tracking model through inference. For example, registration procedures can be aided by making use of the existence of geometric constraints. Sensor fusion configurations can be automatically derived from the existence of certain sensor patterns [14], for which ontology-based inference may provide a suitable tool.

References

[1] R. Aylett and M. Luck. Applying artificial intelligence to virtual reality: Intelligent virtual environments. *Applied Artificial Intelligence*, 14(1):3–32, 2000.

[2] H. Chen, F. Perich, T. W. Finin, and A. Joshi. Soupa: Standard ontology for ubiquitous and pervasive applications. In *MobiQuitous*, pages 258–267, 2004.

[3] M. Gutierrez, D. Thalmann, and F. Vexo. Semantic virtual environments with adaptive multimodal interfaces. In *MMM '05: Proceedings of the 11th International Multimedia Modelling Conference (MMM'05)*, pages 277–283, Washington, DC, USA, 2005. IEEE Computer Society.

[4] T. Höllerer, H. Navdeep, and T. Steven. Steps toward accommodating variable position tracking accuracy in a mobile augmented reality system, 2001.

[5] S. Irawati, D. Calderón, and H. Ko. Spatial ontology for semantic integration in 3d multimodal interaction framework. In *VRCIA '06: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, pages 129–135, New York, NY, USA, 2006. ACM Press.

[6] E. Kalogerakis, S. Christodoulakis, and N. Moumoutzis. Coupling ontologies with graphics content for knowledge driven visualization. In *VR '06: Proceedings of the IEEE Virtual Reality Conference (VR 2006)*, page 6, Washington, DC, USA, 2006. IEEE Computer Society.

[7] M. Latoschik, P. Biermann, and I. Wachsmuth. Knowledge in the loop: Semantics representation for multimodal simulative environments. In *Proceedings of the 5th International Symposium on Smart Graphics 2005*, LNAI 3638, pages 25–39, Berlin, 2005. Springer.

[8] M. E. Latoschik and M. Schilling. Incorporating VR Databases into AI Knowledge Representations: A Framework for Intelligent Graphics Applications. In *Proceedings of the Sixth IASTED International Conference on Computer Graphics and Imaging*. IASTED, ACTA Press, 2003.

[9] S. Li and M. Ying. Region connection calculus: its models and composition table. *Artif. Intell.*, 145(1-2):121–146, 2003.

[10] J.-L. Lugrin and M. Cavazza. Making sense of virtual environments: action representation, grounding and common sense. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pages 225–234, New York, NY, USA, 2007. ACM Press.

[11] E. Mendez, D. Kalkofen, and D. Schmalstieg. Interactive context-driven visualisation tools for augmented reality. In *International Symposium on Mixed and Augmented Reality 2006*, pages 208–218. IEEE Computer Society, Oct. 2006.

[12] J. Newman, D. Ingram, and A. Hopper. Augmented reality in a wide area sentient environment. In *Proceedings ISAR'01*, pages 77–86, 2001.

[13] F. Pittarello and A. D. Faveri. Semantic description of 3d environments: a proposal based on web standards. In *Web3D '06: Proceedings of the eleventh international conference on 3D web technology*, pages 85–95, New York, NY, USA, 2006. ACM Press.

[14] D. Pustka, M. Huber, M. Bauer, and G. Klinker. Spatial relationship patterns: Elements of reusable tracking and calibration systems. In *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR'06)*, October 2006.

[15] W. Stuerzlinger and G. Smith. Efficient manipulation of object groups in virtual environments. In *VR '02: Proceedings of the IEEE Virtual Reality Conference 2002*, page 251, Washington, DC, USA, 2002. IEEE Computer Society.