INTERACTIVE MODELS FROM IMAGES OF A STATIC SCENE

Eleanor G. Rieffel, Don Kimber, Jim Vaughan, Sagar Gattepally, Jun Shingu

FX Palo Alto Laboratory, {rieffel, kimber, jimv, sagar, jun}@fxpal.com

ABSTRACT

FXPAL's Pantheia system enables users to create virtual models by *'marking up'* a physical space with pre-printed visual markers. The meanings associated with the markers come from a markup language that enables the system to create models from a relatively sparse set of markers. This paper describes extensions to our markup language and system that support the creation of interactive virtual objects. Users place markers to define components such as doors and drawers with which an end user of the model can interact. Other interactive elements, such as controls for color changes or lighting choices, are also supported. Pantheia produced a model of a room with hinged doors, a cabinet with drawers, doors, and color options, and a railroad track.

Index Terms— Interactive virtual content, marker-based model creation, semantic markers

Fig. 1. Automatically generated model of IKEA Bookcase cabinet. Users may interact with the model in a VRML browser by clicking on the drawers or door to open or close them, or clicking on color buttons to chose material color.

1. INTRODUCTION

The creation of virtual models is complex and cumbersome, even when the virtual model is based on a physical one. FXPAL's Pantheia system [1] enables users to create virtual models by *'marking up'* a physical space with pre-printed visual markers. The markers have associated meanings from a markup language that helps the system create models from a relatively sparse set of markers. Virtual spaces are most compelling when users can interact with their contents in interesting but understandable ways. This paper describes extensions to our markup language that enable users to create interactive virtual content whose dynamic capabilities mimic those of actual physical objects. Advances in the markup handling algorithms are also described, along with features of our viewer that aid users in creating the desired model.

To create interactive content using the Pantheia system, the user marks up a real world scene with pre-printed markers whose meanings indicate the dynamic behaviors that will be made available to the end user of the model. For example, markers may indicate the desired location for a drawer, or the desired location of a virtual switch that enables end users to control color options in the virtual model. Once the markers are in place, the user collects still images or video of the scene. From this input, Pantheia automatically and quickly produces a model. An attractive feature of these marker-based methods is that interactive models with a rich set of dynamic capabilities can be constructed from a static scene that remains unchanged throughout the image gathering process; the user does not need to move anything to indicate dynamic behavior. Pantheia was used to produce three virtual models, one of a room with hinged doors, one of a cabinet with drawers, doors, and color options, and one of a model railroad.

In the resulting virtual environment, users can interact with these virtual objects in ways that reflects the capabilities of physical objects. Typically the behavior of the virtual model will be caused by users interacting directly with the virtual space in a way that is independent of any activity in the original physical space. Dynamic capabilities may mirror the physical world to a greater or lesser extent depending on the capabilities of the virtual space and the goals and aesthetic criteria of the designers. For example, a door may be modeled simply as a portal from one virtual room to another or as a virtual door with a hinge that moves like a door when a user pushes on it. Similarly, a drawer may only be able to toggle between open and closed, or can be modeled using slider joints. Generally the markers suggest motions that reflect the possible motions of the physical objects in the original scene, but we allow for the possibility of suggesting motions not possible for the physical object.

Section 2 describes related work. An overview of the Pan-

Thanks to FXPAL for supporting this work.

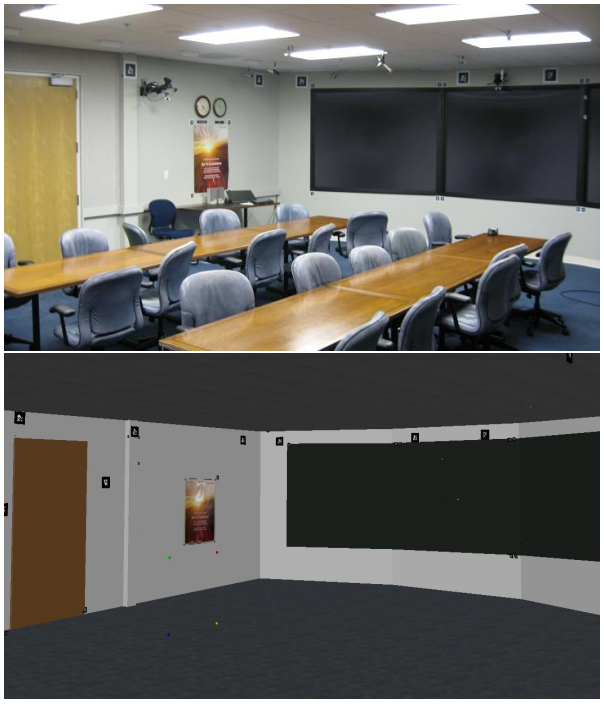

Fig. 2. Photograph of FXPAL's main conference room with markup, and an image of the model Pantheia generated from the same vantage point.

theia system is given in Section 3. Section 4 explains the markup for four types of interactive content. Our heuristic algorithm for determining geometric structure from the planar and shape markers has matured since [1] and the new algorithm is described in Section 5. Our three experiments are described in Section 6. We conclude in Section 7.

2. RELATED WORK

A number of research groups, such as [2, 3, 4], work on non-marker-based methods for constructing models from sets of images. Their work advances progress on solving the hard problem of deducing geometric structure from image features. Instead, we make the problem simpler by placing markers that are easily detected and have meanings that greatly simplify the geometric deduction. Furthermore, a marker-based approach enables users to specify which parts of the scene are important for the model. In this way, Pantheia handles clutter removal and certain occlusion issues easily since it renders what the markers indicate rather than what is seen. In addition, markers enable the specification of deviations from the physical scene on which the model is based; markers placed in the same relative positions on a rough cardboard construction, without doors or drawers, would create the same model of the cabinet we obtained by marking up an actual cabinet. Finally, without object recognition or further user input, image-based approaches cannot produce interactive content, such as drawers and doors, that mimics the behavior of physical objects.

Markers such as QR codes often support augmented reality applications by indicating what models, or other information, should be shown where in an augmented reality display. ARToolKit, for example, was designed for this application [5]. Wojciechowski *et al.* [6] use markers to position objects in virtual exhibitions. The markers used by the 3D Murale project [7] indicate information, such as what strata is currently being excavated, to support virtual reconstruction of an archaeological site. Markers are frequently used for camera pose estimation in 3D reconstruction tasks; for example Rekimoto and Ayatsuka use QR codes in this way [8], as do Rohs and Gfeller [9].

Madden *et al.* [10] patented a system that enables users to markup a model obtained from images, or markup the images, with additional information that is then used to improve the model. They give as examples markup that defines a plane and markup that indicates that two planes are at a 90 degree angle. Their approach is complementary to ours in that all of their markup occurs after the images are taken whereas we are taking images of a physically marked up space.

Chella *et al.* [11] developed an ontology for indoor environments and implemented an XML-based language based on this ontology. They applied their ontology to the situation in which a group of roving robots are trying to communicate in order to build a model of their environments. While parts of their ontology overlaps with ours (they have notions of planes, walls, doors), their robots are not physically marking up the space as they attempt to create their model.

There is a rich history of work on reconstructing polygons and polyhedra from partial descriptions. Some of this work is reviewed in Lucier's thesis [12]. Biedl *et al.* discuss several polygon reconstruction problems based on sensor data which resembles ours [13], except that while their sensor models are 3D, their reconstruction analyses are limited to 2D polygons. The area of 'point based graphics' provides methods for representing surfaces by point data, without requiring other graphics primitives such as meshes [14]. These methods have been used as primitives for modeling tools [15]. Amenta *et al.* [16] describe the point based notion of 'surfels' which are points and normals. Our markers specify one dimension over a surfel: the orientation of the marker within its plane. Generally, point based methods use large numbers of surface points, and aim to produce smooth surfaces.

3. OVERVIEW OF PANTHEIA SYSTEM

Pantheia takes as input a set of images (possibly including video frames), identifies the markers in each image, and determines the relative pose of the marker to camera for each marker in each image. If the pose of a marker in the world is known then the position of the camera can be determined. Conversely, if the pose of a camera is known then the pose of

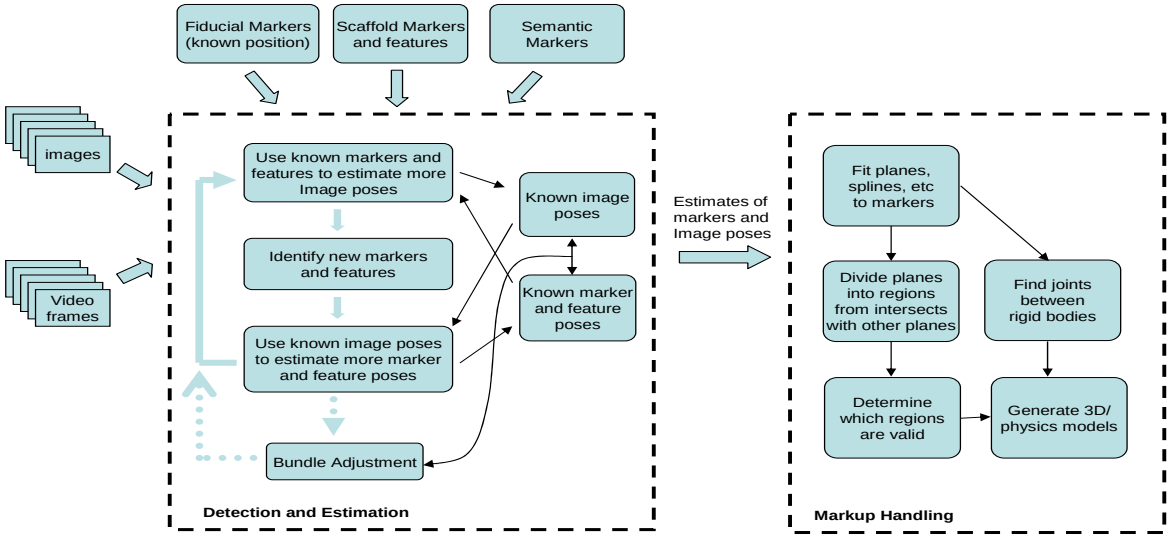

Fig. 3. System Overview

any marker identified in that image can be calculated. From a set of images that meets a few simple conditions, the pose of every marker and every image can be estimated. Pantheia obtains good estimates using ARToolKit [5] together with the sparse bundle adjustment package SBA [17] that globally optimizing the pose estimates for all markers and images.

Pantheia can create static models using a markup language [1] that includes the following elements: planar (plane, wall, ceiling, floor, door); shape (parametric shape, extrusion); modifier (line, corner); appearance (color group, color sampler, texture group, texture sampler, image extractor); submodel insertion (object insertion, definition, reference, replication). We extend this language here to support the creation of interactive content.

The behavior of appearance markers is straightforward; it is described in [1], along with experiments creating models of two rooms at FXPAL. Object insertion markers define the pose according to which a stored model of an object can be inserted in the larger model. Definition markers enable a user to specify that a set of markers defines a submodel. A reference marker, usually one of the definition markers, sets the origin and local coordinate system for the submodel, and a corresponding replication marker tells the system to place a copy of the submodel so that it is in the same relative position with respect to the replication marker that the original model is in with respect to the reference marker. This replication is accomplished by interpreting the replication marker as an instruction for the system to behave as though there were additional markers, one for each of the markers that define the model, in the same relative position with respect to the replication marker that the original markers had been in with respect to the reference marker.

We are in the process of defining a rich markup language that supports the creation of a large class of interactive virtual models. Using this language, the system would support creation of a limited, but still rich, class of models suitable as input for a physics engine such as ODE [18]. In addition to rigid bodies connected by joints, the system would support notions such as conveyor belts and motors, material properties such as mass and bounciness, and color, texture and lighting choices. The markup language we have defined so far has much in common with a specification scheme for the physics engine ODE. The output of our system can be thought of as the description of a virtual model and its dynamic capabilities in terms of a language such as COLLADA, which supports the expression of physics, or of a language such as VRML together with physics specifications for a physics engine such as ODE. Pantheia currently saves models as VRML together with metadata files that specify relations between named parts of the VRML scene graph. The metadata includes descriptions of rigid bodies, and articulation constraints between bodies, and is suitable as input to a rigging system or API, or to a physics engine such as ODE or Newton [18, 19]. We have also exported models generated by Pantheia into SketchyPhysics [20].

3.1. Interactive Pantheia Viewer and Tools

Pantheia is implemented as a collection of C++ and Python modules. The steps of the process used by Pantheia to generate models can be controlled by Python scripts, or via interactive tools. The PantheiaTool shown in Fig. 4 allows a user to create a 'project', define and print out the necessary markers, import images or video into the project, estimate image and marker poses, and run markup handling to create models. It

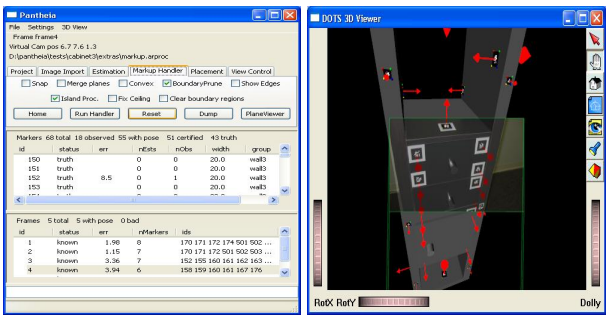

Fig. 4. Some interactive tools of the Pantheia system.

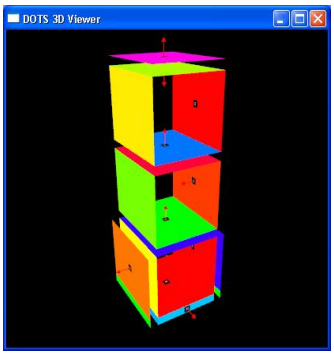

Fig. 5. Interactive Pantheia viewer showing partially completed bookcase markup.

can also control interactive viewers such as the OpenInventor based viewer also shown in Fig. 4. Users may select markers and find the images in which those markers are visible, may move the virtual camera to the position from which the images were collected, and may see those images, with an adjustable alpha value, in alignment with the model.

The interactive Pantheia viewer has been helpful in experimenting with different markup strategies and algorithms, and can also aid end users in understanding when additional markers are needed. Fig. 5 shows a partially completed model of a bookcase, in which some regions have not yet been set to True because for the selected markup algorithm not enough markers have been placed.

4. DYNAMIC MARKUP HANDLER

Details of how the basic geometry of the model is constructed from the marker information are given in Section 5. This section describes how, given the basic geometry of the model, interactive elements such as doors, drawers, controls, and motion constraints are added.

4.1. Door

Doors are marked up using two or more markers to specify the bounding rectangle. For example, two markers can be placed in diagonally opposite corners of the door, or four markers can be placed, one for each edge of the door. A plane is fitted to these markers using eigen analysis, and the parameters of this plane are used to determine if the door is coincident with the plane of a wall, or intersects a wall near one of its edges.

The joint axis for a door hinge is determined in one of two ways. If markers do not explicitly specify an axis, the axis of rotation is determined from the geometry. To be able to make this determination, the door must be open. In this case, the system calculates the line of intersection between the door plane and the wall it intersects. The system places a hinge between the door and the wall with this line as the axis of rotation. An opening for the door is also created in the wall. A user may specify the axis explicitly. The axis of rotation may be specified by one or more *hinge* markers. These markers are a type of line marker, markers for which one of the sides has a special status and an offset is specified. Specification of a hinge does not necessarily require more markers; markers can take on multiple meanings so a hinge marker can serve as door definition marker as well. By default, a door is given a handle that is vertically centered and $4/5$ of the way across the door from the hinge.

4.2. Drawer

The front face of a drawer is specified in the same way as the face of a door. To handle closed drawers, we use a default thickness for drawer walls. We calculate the depth of a drawer by finding the closest marker-defined plane parallel to and behind its front face. In the real world, most drawers point up. Pantheia deduces the up direction from the location of ceiling and floor planes. Unless an orientation is specified by markers, Pantheia creates a door oriented to point up as much as possible. It is not safe, however, to assume that drawers point up; a piece of furniture may have been placed on its side. In this case, the bottom of the door can be specified by bottom markers which, like hinge markers, indicate a special edge. A drawer by default is given a handle in its center.

4.3. Color and Lighting Control

Our previous version of Pantheia generated models with fixed color and lighting properties, which we have now extended to allow interactive control. A color sampler marker defines a color by indicating the region from which color should be sampled in images showing that region. Color markers assign colors to various surfaces. These colors may be predefined, such as ‘red’, may be colors defined by color samplers, or may be ‘dynamic colors’ which a user may select by clicking on a buttons placed in the model. A color selector marker is placed to show where in the model such a button is placed, and to specify the color it will assign as the value of the dynamic color. A similar mechanism could be used to allow interactive selection of textures, or lighting conditions, although we have not yet implemented those capabilities.

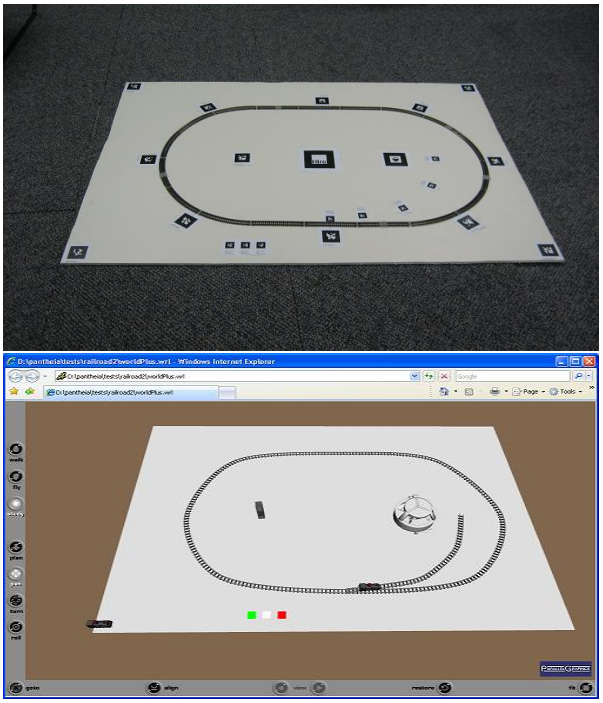

Fig. 6. System generated model of toy railroad

4.4. Constrained Motion along Tracks

Doors and drawers are modeled in Pantheia using hinge and slider joints respectively that constrain the motions of those parts. Another useful motion constraint, is one dimensional motion along a path. Path definition markers that define a curve which may be smooth or polygonal, and open or closed. For the smooth case, Pantheia fits Bezier splines to the path definition markers. Many systems could be defined this way, such as track lighting, conveyor systems, etc. Given such a path, for any sub-model inserted along the path, it is natural to define an animation corresponding to fixed speed motion along the path. Controls buttons or sliders may be added to the model that set or modify that speed.

A defined path may be used to specify animation paths, but may also contribute static geometry to the model. The path may not be shown at all (but still used to define animation trajectories), may be shown as a curve or polyline in the model, or may be shown as unit cells replicated along the path. For example to model a railroad, a unit cell consisting of one rail tie, and a short segment of rail may be replicated to generate the appearance of track along the path.

5. GEOMETRIC MARKUP

This section describes our algorithm for defining the model geometry from the markers.

The success of a marker-based approach depends on the class of models under consideration, the markup placement strategy, and the markup handling algorithm used to determine a model from a set of marker positions and orientations and other metadata. While we have considered more general classes of models, including, for example, cylinders and railroad track splines, an attractive class of models to consider is polyhedra. The entire class of polyhedral models can be reconstructed from a markup strategy in which the user partitions each face into convex polygons, and gives each convex polygon its own set of markers to mark its vertices. In general, however, polyhedra can be determined using fewer markers. We are still developing efficient markup strategies and corresponding markup handling algorithms under which attractive classes of models can be uniquely reconstructed. In the meantime, we have developed a heuristic markup handling algorithm that works for many of the marked up models, both real and synthetic, we have tried.

When we markup a model, we make sure that each face has at least one marker, and that there are no isolated sets of markers: between every pair of markers there is a chain of markers so that successive markers appear together in at least one image. To obtain a good fitting, we recommend each plane have at least two markers, preferably placed far from each other.

5.1. Pantheia's current markup handling algorithm

The algorithm finds a best fit plane for each group of markers with the same planar group ID using eigen analysis. Planes that are near coincident are merged. Each plane is then partitioned into regions according to lines formed by intersection with all the other planes. Only finite regions are kept; we assume that the models we are trying to create are of finite size. The algorithm then endeavors to determine which of these regions should be included in the model, and which should not.

The algorithm follows the general strategy of marking regions we know should be included as True, those we know should not be included as False, and then using geometric reasoning to propagate this knowledge to neighboring regions. A number of strategies are possible; their success depends on the markup placement strategy and on geometric assumptions about the class of models under consideration. Reasonable strategies converge, which is easy to guarantee if the strategy marks a region as True or False only when it is certain of its status. We allow strategies that converge but are not complete in that they do not always result in the full model, a model for which all regions have known status. The full model may not emerge for two reasons: the user may not have provided enough markup to uniquely determine the model, or the algorithm may not be smart enough to determine the model. In both cases, a partial model can be shown to the user who can provide information through the interface or by placing more markers. Section 3.1 describes interface options that aid with ambiguity resolution.

Pantheia currently uses the following algorithm:

- Label all regions that contain a marker as True.
- Iterate through the following three steps until no changes are made during an iteration:
  - **True to True propagation:** For each region, check if any neighboring coplanar region is True, and if so, whether all other regions that share the edge between those two regions have status False, in which case the region is set to True.
  - **No self-intersections:** For each edge, check if it already has two True adjacent regions. If so, set all other adjacent regions to False.
  - **Setting unsupported regions to False:** Sets regions to False if they cannot be connected to a True region.

A major assumption behind these heuristics is that the model is not self-intersecting: each edge in the model has only two adjacent faces. This assumption underlies the *No self-intersections* rule. Another assumption we make is that faces are only terminated by intersection with other faces or by marked lines. The *True to True propagation* rule uses this assumption to propagate truth from a True region along one of its edges to the neighboring coplanar region if all the non-coplanar regions that share the edge are False so there can be no intersecting face in the model. The *Setting unsupported regions to False* requires a little more explanation.

The idea behind *Setting unsupported regions to False* is that every face must contain at least one marker, and the regions that make up a face are all adjacent, so if we can determine that there is no sequence, obeying the no self-intersection property, of adjacent regions from the region to a coplanar True region, then the region can be set to False. We call a region *supported* if there is a coplanar path from a True region that is not blocked by an intersection. More precisely, a region is *supported* if there is a connected chain of coplanar regions between the region and a True region such that none of the edges between successive regions have a non-coplanar adjacent region marked as True. A region is said to be *unsupported* if it is not supported. To determine which regions are unsupported in each iteration of the algorithm, support is propagated starting from the current set of True regions. All True regions are set to *supported*. Coplanar regions adjacent to a supported region are considered in turn. If the edge between a supported region and its neighbor does not border any True non-coplanar adjacent regions, then the neighbor region is given the status of supported. Support is propagated in this way until no more regions can be marked as supported. All regions that have not been labeled as supported are then set to False.

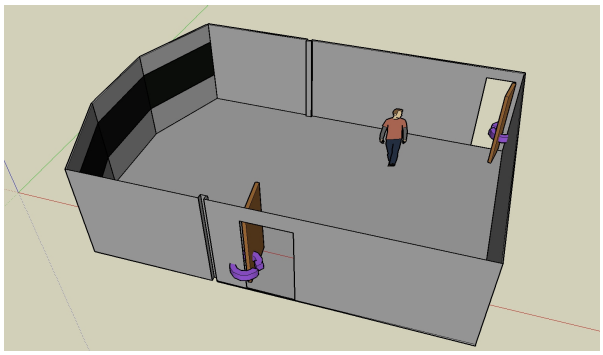

Fig. 7. Pantheia generated model of FXPAL's main conference room in SketchUp. The SketchyPhysics plugin to SketchUp supports door hinges, and enables end users to pull or push on the doors to make them open more or close.

6. EXPERIMENTS

We conducted three experiments: reconstructing the model of a cabinet, reconstructing a room, and defining a railroad track.

6.1. Cabinet

We placed 56 markers on an IKEA Bookcase cabinet (Fig. 1) in three sizes, 2, 4, and 10 cm, and took pictures with a Nikon D80 camera with a fixed focal length of 30 mm and a pixel resolution of 3,872x2,592. The camera was calibrated using the Matlab Calibration toolkit. Most markers were plane markers used to define the sides and shelves of the cabinet. The first drawer was defined by four drawer markers, one of which served as a reference marker. The second drawer was defined by a single replication marker. The door was defined by four markers, including two which also define the axis of rotation. All were given default handles. The model as rendered in the Pantheia viewer is seen in Fig. 1. Four markers on one of the shelves defined the color change buttons.

6.2. Conference room with doors

Pantheia produced an accurate model of FXPAL's main conference room. Quantitative accuracy results were reported in [1]. Pantheia now supports the creation of a model of the room with hinged doors. In addition to rendering it in VRML, we exported it to Google SketchUp, where the hinges and user interactions are supported by the SketchUp plugin SketchyPhysics [20]. Fig. 2 shows the Pantheia generated model in the Pantheia viewer, while Fig. 7 shows it in SketchUp.

6.3. Model Railroad

We used path definitions markers to capture the outline of a simple model railroad set up on a posterboard. Four markers

define the posterboard, and eight path definition markers define the path of the existing track. Three markers were used to insert speed control buttons that speed up, slow down, or stop the motion along the path. Five other markers were used to define a hypothetical track layout under consideration for being added to the physical model. Other markers are used to insert predefined models into the scene. Pantheia created a model of the scene from five 3648x2048 pixel resolution images taken by a Canon Powershot SD900 camera. One of the images, and a resulting VRML model are shown in Fig. 6. Clicking on the green button speeds up the train, the pink button decelerates it, and the red button halts it.

7. CONCLUSIONS

Pantheia takes a set of images, detects markers in the images, determines their identity and pose, and uses the associated meanings to determine a model. Interactive components of the model may also be specified using the markers. Pantheia renders the resulting model in such a way that end users can explore and interact with dynamic elements of the model.

Future work includes defining a richer marker-based markup language for interactive components so that a larger set of models can be created in this way. Furthermore we will be exploring efficient and easy to understand marker placement strategies that have an accompanying markup handling algorithm under which a model from a large class of models can be uniquely determined. To deal with other cases in which the model is not uniquely specified by the markers, or users make errors in marker placement, or the system makes errors in detection or estimation, we will continue to enhance the features of our viewer that support model creation. In [1], we used edge detection to improve Pantheia's image sampling and insertion capability. We will explore further ways in which the use of more sophisticated image processing techniques can enhance our system.

8. REFERENCES

[1] Don Kimber, Chong Chen, Eleanor Rieffel, Jun Shingu, and Jim Vaughan, "Marking up a world: Visual markup for creating and manipulating virtual models.," in *To appear in the Proceedings of Immerscom09*, 2009.

[2] Marc Pollefeys, "Visual 3D modeling from images," http://www.cs.unc.edu/ marc/tutorial.

[3] Marc Pollefeys and Luc Van Gool, "Visual modeling: from images to images," *Journal of Visualization and Computer Animation*, vol. 13, pp. 199–209, 2002.

[4] F. Dellaert, S. Seitz, C. Thorpe, and S. Thrun, "Structure from motion without correspondence," in *Proceedings of CVPR'00*, 2000.

[5] "ARToolkit," http://www.hitl.washington.edu/artoolkit/.

[6] Rafal Wojciechowski, Krzysztof Walczak, Martin White, and Wojciech Cellary, "Building virtual and augmented reality museum exhibitions," in *Proceedings of Web3D '04*, 2004, pp. 135 – 144.

[7] John Cosmas and *et al.*, "3D Murale: A multimedia system for archaeology," in *Proceedings of VAST 2001*, 2001, pp. 297–306.

[8] Jun Rekimoto and Yuji Ayatsuka, "Cybercode: Designing augmented reality environments with visual tags," in *Proceedings of DARE'00*, 2000, pp. 1 – 10.

[9] Michael Rohs and Beat Gfeller, "Using camera-equipped mobile phones for interacting with real-world objects," in *Advances in Pervasive Computing*, 2004, pp. 265 – 271.

[10] P. B Madden and *et al.*, "Computer assisted markup and parameterization for scene analysis," US patent 6249285, 2001.

[11] Antonio Chella, Massimo Cossentino, R. Pirrone, and A. Ruisi, "Modeling ontologies for robotic environments," in *Proceedings of SEKE'02*, 2002, pp. 15–19.

[12] Brendan Lucier, "Unfolding and reconstructing polyhedra," http://uwspace.uwaterloo.ca/handle/10012/1037, 2006.

[13] Therese Biedl, Stephane Durocher, and Jack Snoeyink, "Reconstructing polygons from scanner data," in *18th Fall Workshop on Computational Geometry*, 2008.

[14] Ke-Sen Huang, "Point-based graphics resources," http://kesen.huang.googlepages.com/PointBasedPaper.html, 2009.

[15] "Pointset3d," http://graphics.ethz.ch/pointshop3d/, 2009.

[16] Nina Amenta and Yong Kil, "Defining point-set surfaces," in *SIGGRAPH 2004*, 2004, pp. 264–270.

[17] Manolis Lourakis and Antonis Argyros, "sba: A generic sparse bundle adjustment C/C++ package based on the Levenberg-Marquardt algorithm," http://www.ics.forth.gr/ lourakis/sba/.

[18] "Open Dynamics Engine," http://www.ode.org/.

[19] "Newton Game Dynamics," http://www.newtondynamics.com/.

[20] "Physics plugin for Sketchup," http://code.google.com/p/sketchyphysics/.