Transitional Augmented Reality Navigation for Live Captured Scenes

Markus Tatzgern* Raphael Grasset Denis Kalkofen Dieter Schmalstieg

Graz University of Technology

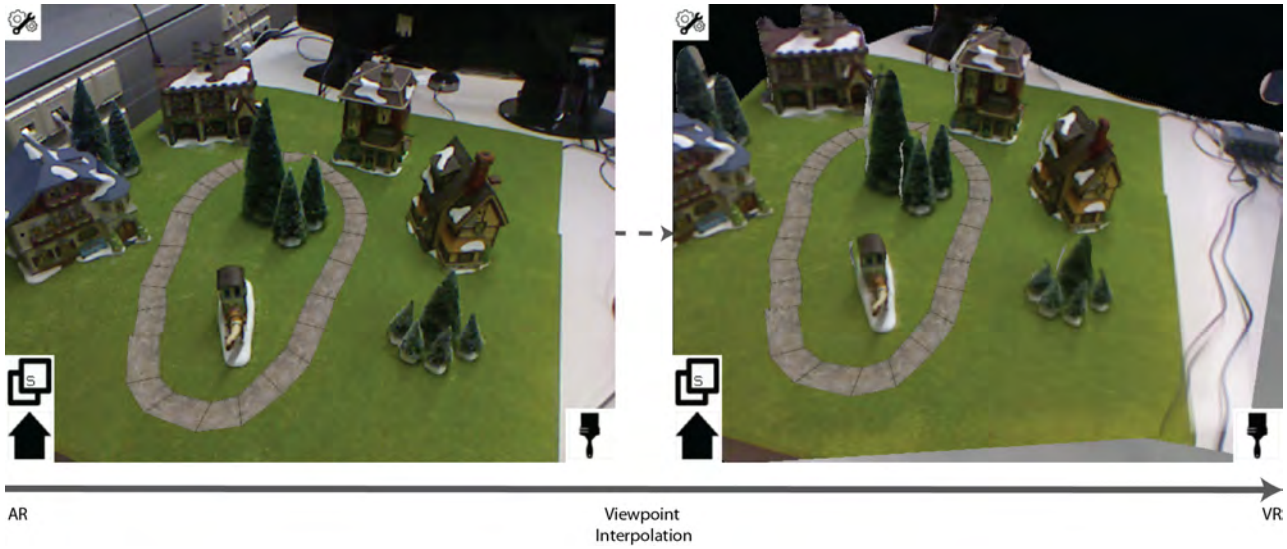


Figure 1: After placing physical objects on a table, our system offers the possibility to structurally navigate an unprepared scene with a set of new transitional navigation techniques in AR (Left) and VR (Right) modes. The techniques seamlessly switch from AR to VR modes.

Abstract

Augmented Reality (AR) applications require knowledge about the real world environment in which they are used. This knowledge is often gathered while developing the AR application and stored for future uses of the application. Consequently, changes to the real world lead to a mismatch between the previously recorded data and the real world. New capturing techniques based on dense Simultaneous Localization and Mapping (SLAM) not only allow users to capture real world scenes at run-time, but also enables them to capture changes of the world. However, instead of using previously recorded and prepared scenes, users must interact with an unprepared environment. In this paper, we present a set of new interaction techniques that support users in handling captured real world environments. The techniques present virtual viewpoints of the scene based on a scene analysis and provide natural transitions between the AR view and virtual viewpoints. We demonstrate our approach with a SLAM based prototype that allows us to capture a real world scene and describe example applications of our system.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction styles

*e-mail: tatzgern | grasset | kalkofen | schmalstieg@icg.tugraz.at

1 Introduction

Augmented Reality (AR) applications traditionally rely on predefined, rigidly modeled content that only applies to those conditions of the real world that were present when the application was developed. Hence, often an AR application can be deployed only in a single physical location, which limits its flexibility. This problem is aggravated when changes in the environment cause misalignments between the previously recorded data and the real world. Such a situation can impair the correct functioning of the application.

Fortunately, there has been tremendous progress in the area of real-time Simultaneous Localization and Mapping (SLAM) [12]. SLAM enables users to use AR in unprepared and unknown environments and allows them to capture the geometry and the visual appearance of the environment. Furthermore, changes to the real world scene can be captured and AR applications can react to these changes. This allows users to engage into AR applications anywhere and at anytime. However, real world scenes are not modeled beforehand anymore, and users need to interact with and manipulate unprepared environments that are unknown to the application.

In this paper, we combine rapid scene acquisition and extraction of basic semantic information from the captured scene with new transitional interface techniques that allow users to navigate and manipulate unknown real world environments. We complement the AR view with Virtual Reality (VR) views of the captured environment, which allows users to quickly switch between different views of the scene. This has been shown to facilitate AR interaction [15].

Our techniques provide natural transitions between AR and VR viewpoints and strategically place the virtual camera in locations that are better suited for the current task of the user. For this purpose, we analyze the captured scene and extract semantic information to determine the virtual camera viewpoint. We demonstrate our

approach with a SLAM-based prototype that allows us to capture a real world scene. However, the presented techniques can be used independently of the real-time capture system and can be extended by more sophisticated semantic information.

2 Related Work

Different approaches have been proposed, which complement the traditional egocentric viewpoint of AR. For example, Lee et al. [9] propose automatic zooming based on distance to the target. Other work [5, 10] implements a pause mode in AR that basically resembles a static VR mode. Hoang et al. [6] zoom distant objects by showing a second AR view of a remote camera in a window.

Other approaches allow the user to use both an egocentric AR view and VR view of the world. Höllerer et al. [7] combine the AR view through a head-mounted display with a handheld mobile computer that shows a virtual model of the environment. Similarly, Pierkaski et al. [14] allow users to switch between an egocentric AR view and a VR exocentric view. These approaches switch abruptly and do not provide transitions between AR and VR viewpoints.

Billinghurst et al. [1] introduce the concept of transitional user interfaces, where a user can seamlessly switch between an AR view and a VR view. They also present steering techniques for navigation in VR. Grasset et al. [3] evaluate collaborative transitional user interfaces and visual cues. Grubert et al. [4] allow users to take away content from a real world poster to explore multimedia content offline, but they do not consider 3D interaction. Tatzgern et al. [16] present a transitional AR interface which allows users to explore distant real world buildings using a virtual 3D copy metaphor. All of this work only considers environments that have been prepared beforehand. Furthermore, the interfaces do not allow the user much control over the target viewpoint of the transition.

Mulloni et al. [11] propose two interfaces for changing between egocentric AR and exocentric VR view in a mobile context: zooming out to a top-down 2D view and zooming out to a panoramic overview of the surroundings. However, the virtual views are bound to the user's location and can not be changed freely. Veas et al. [17] transition between remote static viewpoints in an outdoor AR scenario. Their work focuses on switching between a discrete set of cameras, which have been installed in a real environment. Similarly, Sukan et al. [15] propose an interface to transition from the current AR view to previously captured discrete viewpoints showing a recorded AR view. However, in contrast to our approach, their interface does not allow the user to reach physically impossible viewpoints, because the user must capture the viewpoints by reaching them at least once. Furthermore, both Veas et al. and Sukan et al. allow the user to switch between discrete viewpoints, and the techniques do not use scene geometry captured at run-time.

3 System Overview

Our system consists of four components: capturing a real world scene, analysis of the scene, retrieving knowledge about the user's task and camera navigation that fuses these components to create viewpoint transitions. Figure 2 provides a schematic overview of the components and their relation to the traditional AR pipeline.

In a traditional AR pipeline, the tracking component controls the viewpoint of the rendering. We expand this pipeline with our scene navigation component, which seamlessly switches between live tracking and virtual camera viewpoints. This component also selects camera manipulators to allow changes of the virtual viewpoints. The navigation component strategically selects viewpoint and manipulator based on the knowledge about the user's task and knowledge about the scene. In our system, we gather scene knowledge automatically by analyzing a virtual representation of the scene, which we capture and reconstruct at run-time. The update frequency of the capturing process can be adjusted to the reconstruction algorithm and its computational demands. The navigation runs in real-time and operates independently of the capturing.

In the following, we present the details of three components: scene capture, scene analysis and scene navigation. The task knowledge component depends on the application scenario (authoring, games, etc.). We do not discuss this component in detail.

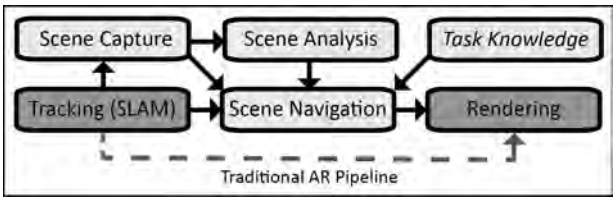


Figure 2: Overview. In a traditional AR pipeline (dashed line) the tracking system controls the viewpoint of the rendering. We expand this pipeline by a scene navigation component, which allows to switch from AR to VR views and chooses viewpoints based on scene and task knowledge. We gather scene knowledge from analyzing a scene captured at run-time, e.g. by using a SLAM system.

4 Capturing the Scene

Virtual representations of real world scenes can be captured in real-time using SLAM technology [12]. The combination of SLAM with depth sensors, (e.g., KinectFusion [12]) allows live capturing of detailed models, either as volumes, depth images or polygon meshes. Figure 3(b) shows the output of an open source KinectFusion implementation[1] applied to the scene shown in Figure 3(a).

Models can be constructed with monocular SLAM technology, stereo SLAM, or depth sensors. Alternatively, models can also be reconstructed using an online reconstruction service. While capturing the scene, the data is processed in the cloud and sent back to the user. For example, we created the model in Figure 3(c) by using a freely available online reconstruction service[2].

In our prototype system, we create a mesh approximation of the volume generated by an open source KinectFusion implementation. We reduce the number of voxels by using a grid filter that calculates the centroid of voxels within a cube having a side length of $1cm$. Then, we use a poisson mesh algorithm [8] using the filtered points to create the mesh representation. We chose the tree depth used for the poisson reconstruction to find a trade off between reconstruction performance and accuracy of the created geometry. A tree depth of six allows for more frequent mesh updates, but the details of the geometry are smoothed. However, the general shape of the object is approximated well. A tree depth of seven creates a good approximation of the objects at the cost of a lower update frequency. In our system, we allow the user to switch at runtime between these settings. Hence, a user may choose faster updates, when the real world scene is rearranged frequently, or a better mesh approximation with higher visual quality after the scene was arranged.

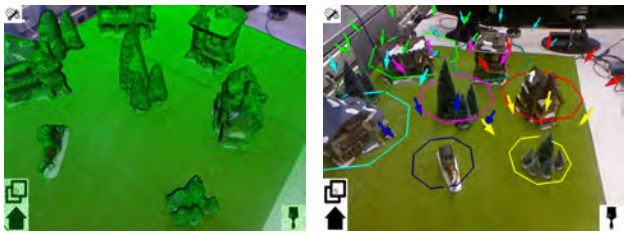
We create a separate thread for the meshing process to avoid a performance impact on the rendering thread. In our prototype system, a tree depth of seven took around eight seconds to reconstruct, while a depth of six took only around two seconds for a point cloud containing $25k$ points after filtering. This size corresponds to a complete scan of the scene. We deployed the prototype on a PC running Windows 7, equipped with an Intel i7 CPU quad-core 2.66GHz, 12 GB RAM and an Nvidia 780GTX graphics board.

To compensate for visible reconstruction inaccuracies, we use an image-based rendering (IBR) method to provide view dependent updates of the virtual representation (Figure 3(d)). This also improves visual fidelity and the visual similarity between a physical

[1] http://pointclouds.org
[2] http://www.123dapp.com/catch

(a) (b) (c) (d)

Figure 3: Scene Representation. Our system works independently of the scene capturing technology or rendering representation. The real scene shown in (a) can be reconstructed and rendered using different methods, such as (b) rendering colored voxels of a volumetric representation generated with KinectFusion, (c) rendering a polygonal mesh of a reconstruction or (d) image based rendering of a reconstructed proxy mesh.



(a) (b)

Figure 4: Reconstruction Feedback and Guidance. (a) We overlay the point cloud from the KinectFusion tracker in real-time to provide reconstruction feedback. (b) To support the user during scene capturing, we visualize viewpoints that have not yet been visited by rendering arrows around the objects segmented by the scene analysis.

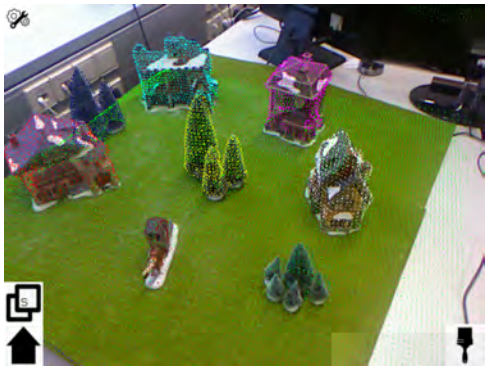


Figure 5: Scene Segmentation. We automatically segment the scene to identify the ground plane defining the world coordinate system and single objects on the ground plane. We use this knowledge to control the viewpoints of transitions. Colors represent the segmented object.

view of the scene (AR) and a virtual view of the scene (VR). Note the similarity of the IBR in Figure 1(Right) with the real world scene shown in Figure 1(Left). We record the images used for the IBR during the capturing process by sampling images at points that are located on a regular grid in 3D space. For each point we store a number of images with viewing directions offset by 45°. To be able to react to changes to the scene, we implemented a simple image aging algorithm, which expires stored images after a certain amount of time. Alternatively, images can also expire when the geometry shown in the image changes.

We also provide reconstruction feedback to the user by rendering the currently reconstructed point cloud in AR (Figure 4(a)). In contrast to the mesh representation, this visualization is updated almost instantaneously. Furthermore, the system can guide the user around objects in the scene to ensure that all sides of an object have been viewed with the camera (Figure 4(b)). Arrows around the object guide the user to viewpoints that have not been visited before. Arrows are removed when the viewpoint has been visited. To enable this feature, we use automatic scene segmentation, which is discussed in the next section.

5 Analyzing the Scene

Common camera controls, such as used in 3D games, rely on semantic knowledge about the scene to define camera manipulation and possible camera viewpoints. In a dynamically captured scene, this information is initially not present. Therefore, we extract semantics from the captured scene based on the reconstructed mesh.

Niederauer et al. [13] show how to generate semantic information from a 3D mesh. Additionally, the semantic information can be derived from others sources such as object recognition from captured video data [2]. Sensors in mobile devices facilitate the analysis by providing information about gravity or compass reading, which allows to identify a ground plane or a specific direction.

We need scene information for controlling the camera, but also for the transition between different AR and VR views. Depending on the user's current task, selecting a particular object can trigger a different viewpoint change. For instance, when selecting the ground plane during authoring, the virtual viewpoint might transition to a top-down view to allow the user to arrange virtual objects on the ground plane. However, during a game session, the camera may transition to a ego-perspective on the ground level.

In this paper, we focus on tabletop scenarios (e.g., Figure 3(a)). We identify the plane corresponding to the table and the single objects on the table by segmenting the reconstructed point cloud. The plane is identified using a sample consensus method that fits a plane model. After removing the plane points from the point cloud, we identify point clusters based on Euclidean point distances. Using this simple segmentation, the system is now able to differentiate between the ground plane and single scene objects (Figure 5). Additional information can be added to the analysis using domain knowledge. Because our example scene consists of houses, we can identify walls and roofs of the houses, either geometrically or by defining every point lying above a certain height to be a roof.

Aside from having knowledge about the objects in the scene, we also use the mesh normals to control the camera orientation. For instance, the plane normal defines the up vector of the world so that virtual geometry and virtual cameras can be placed with correct up orientation. We also use the normals of the convex hulls of objects to orient cameras towards the geometry. We use the normals of their convex hull to achieve a more homogenous normal distribution.

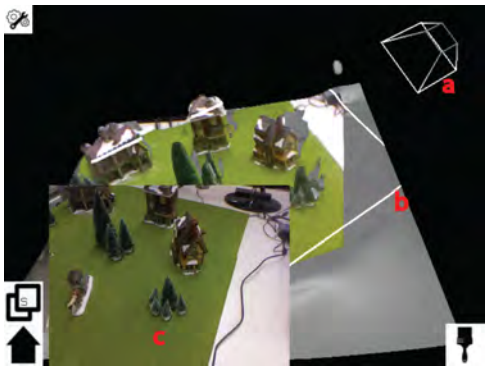


Figure 6: Spatial Cues. We provide spatial cues to facilitate orienting in the VR view. (a) A camera icon shows the pose of the AR view, that is still tracked when the user switches to the VR view. (b) The viewport of the AR view is also projected into the scene. (c) The user can optionally activate a small window showing the current AR view.

6 Transitional Navigation Techniques

We present four navigation techniques that can be used for dynamic scenes. Each technique allows a seamless transition between the AR view and VR view of the captured scene. We classify our techniques into two categories: context-aware transitions that use the knowledge gathered from the scene analysis and intermediate transitions that provide VR views but are still connected to the AR view.

The user can switch between AR and VR views at any time using one of the presented techniques. To quickly move from VR back to AR, we provide a home button located in the bottom left of the view. We provide spatial cues in the VR view to communicate the current position of the tracked AR camera and thus the position of the user relative to the virtual model. We visualize the tracked camera position in VR by rendering a camera frustum (Figure 6(a)). We also visualize the viewing direction of the tracked camera by projecting the borders of its viewport onto the scene geometry (Figure 6(b)). Furthermore, we allow the user to open an AR window inset in the VR view, which shows the AR view of the scene (Figure 6(c)). This also allows a user to see changes that were performed in the VR view in the current AR view without switching back to AR.

To demonstrate our techniques, we use the scene shown in Figure 3(a) and render the virtual scene using IBR. Semantic input is provided by classifying the scene into *ground*, *object* and *top*.

6.1 Context Aware Transitions

Context aware transitions make use of the data gathered from the scene analysis to control the viewpoint of the virtual camera.

Transitional Camera Control. The context aware transitional camera control changes from the AR view to an automatically calculated virtual viewpoint. The location of this viewpoint is determined by taking the semantic knowledge of the scene and the current task into account. To trigger the transition to the calculated viewpoint, a user can tap on any element of the scene. After reaching the viewpoint, the user can navigate the virtual scene with the provided camera manipulator. To provide feedback about the possible target viewpoints, we display icons representing the calculated viewpoints for these locations. These visual icons are activated by dragging the finger over the scene using a hover or swipe gesture.

The results for our test scene are shown in Figure 7. For the viewpoint calculation, we assume an authoring task using the reconstructed scene. Hence, the semantics are interpreted to reflect that task. When the user taps on the ground, the system switches to a map like top-down view and a panning camera manipulator. When the user taps on the top of an object, we provide a top-down view, but this time closer to the object. In both cases, the normal of the ground plane is used to orient the camera to look from top down. When the user clicks on the surface of an object, we automatically provide a close-up viewpoint, which uses the normal of the ground plane as up vector and is oriented towards the geometry

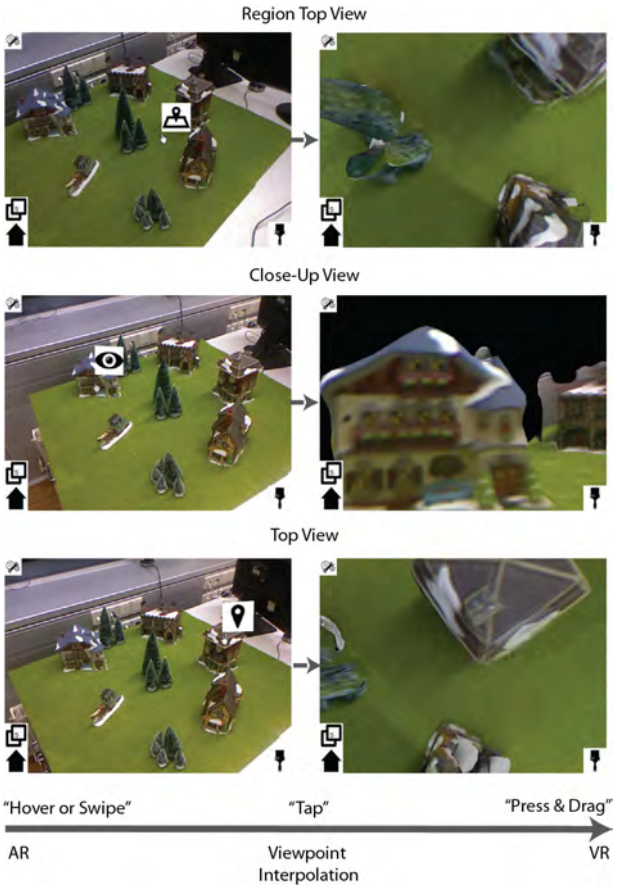


Figure 7: Context Aware Transitional Camera Control. Our system chooses different navigation modes and viewpoints in function of an area selected by a user. (Left) Visual icons define which semantic modes are available. By tapping on a visual icon, semantic navigation modes are triggered, such as (Top) top-down regional view, (Middle) front view of an object or (Bottom) top view of the object.

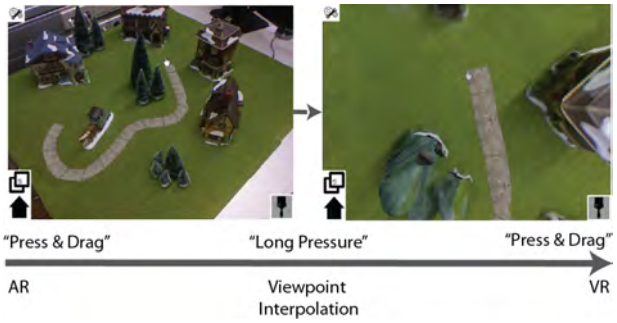


Figure 8: Context Aware Transitional Interaction. Authoring and manipulating 3D content requires constant camera manoeuvring for accomplishing the task. (Left) The user in the AR mode cannot continue the drawing task behind the trees because of limited visibility. (Right) Our technique allows the user to switch to a more adequate viewpoint without stopping the interaction.

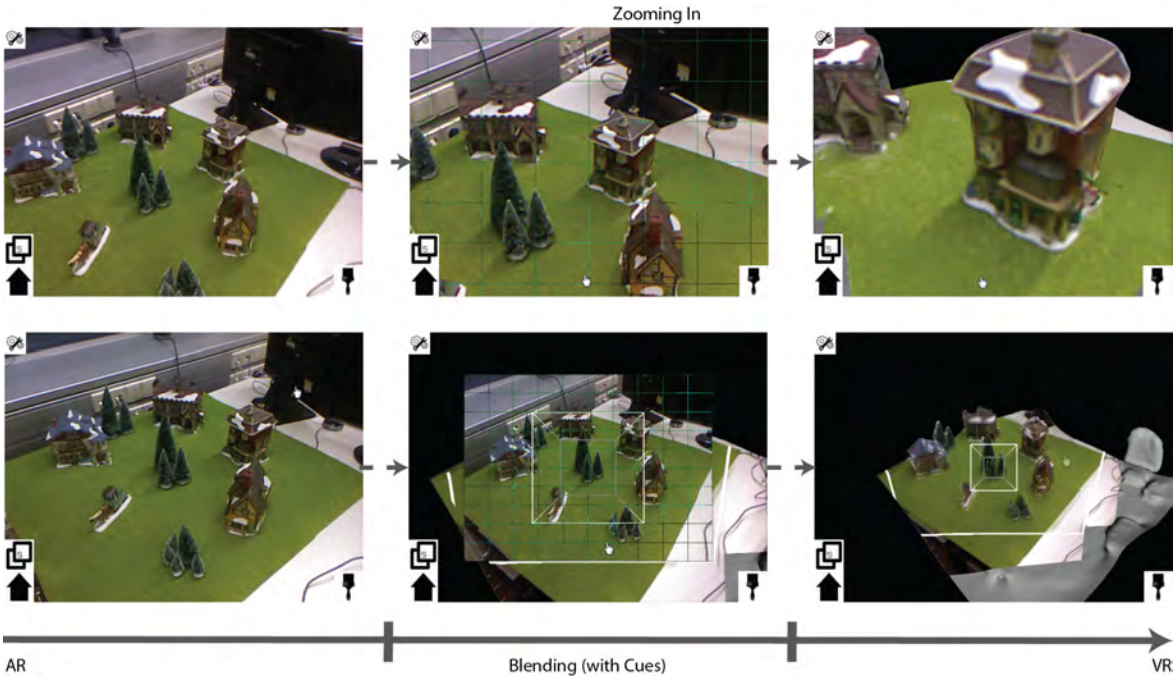


Figure 9: Transitional Zooming. (Top) To achieve a close-up view of a real model, a user can directly zoom in an AR view. (Bottom) A user can also zoom out of the AR view to get an overview of the scene. When zooming, the view gradually fades to the VR view when a certain distance threshold is reached. We use a virtual grid as a visual feedback to notify the user that the switch from AR to VR is imminent.

using the normal of its convex hull. In the close-up view, we switch to an orbit manipulator to rotate around the object. Once in VR, the user is free to choose other viewpoints.

Transitional Interaction. A context aware transitional interaction can be used to complement interactions that were started in the AR view. The user can indicate that an interaction that was started in AR should be continued in VR by pressing on the same location on the screen, without releasing it. The system then seamlessly switches to a viewpoint that is most appropriate for the current task. Hence, the interaction of the user is not interrupted by performing a gesture to switch to VR.

In Figure 8, the user starts drawing a path in AR and cannot continue drawing, because the view is blocked by scene geometry. By stopping and holding the interaction at the end of the path, the system recognizes the user's intention to continue drawing and switches to a virtual viewpoint. The viewpoint is top-down, because it is best suited for the task and scene semantics. The user can continue drawing in VR and pan the camera over the scene to extend the path in areas not reachable from the AR view.

6.2 Intermediate Transitions

Intermediate transitions switch to a VR view, but are still loosely connected to AR. From an intermediate transition, a user can always use a context aware transition to continue navigating in VR.

Zooming. When showing AR applications on mobile phones, we noticed that users regularly tried to zoom into the AR view to look at details of the real world scene or the augmentation. A naive implementation might digitally zoom the video image and adjust the augmentations accordingly. However, at higher zoom levels the quality of the video image degrades. To avoid this decrease in quality, we developed a transitional zooming technique that gradually replaces the video with the virtual representation, as the user zooms in on the model. The virtual geometry is able to provide more detailed close-up views than simple digital zooming (Figure 10). The virtual geometry also allows users to zoom out of the AR view to

(a) (b)

Figure 10: Scene Zooming Methods. (a) A pure digital zoom causes artifacts in the video image. (b) We zoom into a virtual representation, which provides close-up views with higher resolution.

get an overview of the scene from a larger distance. While being in a zoomed view, the system continues tracking as in the AR view. We use the tracked camera pose to directly control the viewpoint of the virtual camera so that the user can naturally explore the scene by moving the mobile device in the real world reference frame.

The zooming is situated in the continuum between the AR view and the VR view. When zooming, we gradually blend to the other view mode when a certain threshold is reached. We indicate this threshold by adding a fading virtual grid before starting to blend to the other view mode. When virtual objects are closer than the blending threshold, we switch to the virtual model before this threshold to avoid zooming through this object. Zooming can be controlled with a swipe gesture, or on a touch screen with a pinch gesture. The steps of the technique are shown in Figure 9.

Spring Loaded Navigation. The context aware transitional techniques allow a user to quickly navigate to virtual viewpoints of those parts of the scene that are visible from the current viewpoint. To investigate the occluded areas, the user either must change the physical AR viewpoint or manipulate the virtual camera after

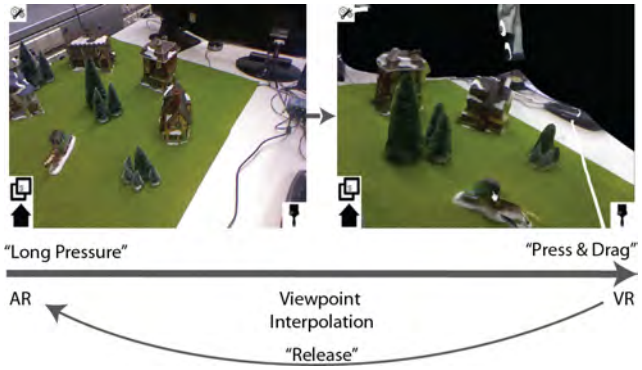


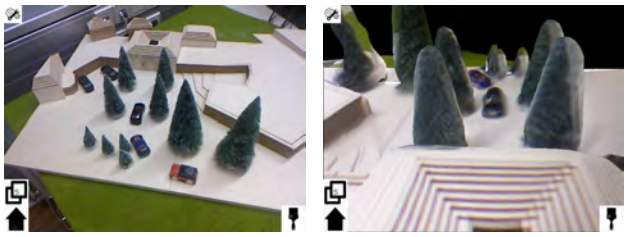
Figure 11: Spring Loaded Navigation. A user may want to switch to areas that are not visible from the current AR view. We provide a spring loaded navigation technique that allows a user to rotate around an object directly from the AR view. When the user stops interacting the technique switches back to AR.

switching to a VR viewpoint. To avoid these detours, we introduce a spring loaded navigation technique that enables users to quickly change to a virtual viewpoint of the occluded areas (Figure 11).

The spring loaded navigation allows the user to investigate invisible areas by rotating the virtual model, while still being in the AR view. To activate the spring loaded navigation, the user taps and holds a location on the screen, which triggers a transition to the VR view. The user can then drag the view to initiate a rotation around the pressed region. To be able to rotate around an object, we allow users to interrupt the dragging to reposition the input. When the user does not interact with the screen for a certain amount of time, the viewpoint transitions back to the current AR view.

7 Conclusion and Future Work

Real time capturing techniques allow a user to create representations of the real world environment. In this work, we presented transitional navigation techniques, which make use of the captured scene to provide virtual viewpoints. By extracting semantic information from the scene, we are able to provide seamless transitions from AR to strategically chosen VR viewpoints, which are relevant for the current task of the user.



(a) (b)

Figure 12: Example Urban Planning. (a) In an urban planning application users rearrange the scene to identify a good space design. (b) Using our system, users can achieve egocentric viewpoints of the scene and visit views that would otherwise be unreachable.

Our system enables a wide variety of new application scenarios in AR, such as more immersive AR games. A user can switch to an egocentric view of the game, or follow virtual characters that otherwise would disappear behind geometry. Furthermore, new game designs can make use of a combination of AR and VR views and the transitions between them. Other application areas are architecture and urban planning. Users can not only manipulate the scene using tangible objects, but also switch to a virtual view showing the planned areas from an egocentric virtual viewpoint. This allows users to verify the placement of buildings and detect undesired occlusions from certain viewpoints. As shown in Figure12(a), users might rearrange the trees in the architectural model of a public space to determine a good placement for enhancing the appeal of the space. Using our techniques, they can verify the result from an egocentric perspective. They can also move to viewpoints otherwise blocked by the real environment. In Figure 12(b), the user switched to the top of the roof looking down the alley from the direction opposite to the AR view, a viewpoint that would be blocked by the table. In future work, we will evaluate this type of techniques in the context of gaming and urban planning.

Acknowledgements

This work was funded by the Christian Doppler Laboratory for Handheld Augmented Reality, the Austrian Science Fund (FWF) under contract P-24021 and the FP7-ICT EU project Nr. 601139 CultAR.

References

[1] M. Billinghurst, H. Kato, and I. Poupyrev. The magicbook: a transitional ar interface. *C & G*, 25(5):745–753, 2001.

[2] A. Chia, S. Rahardja, D. Rajan, and K. Leung. Object recognition by discriminative combinations of line segments and ellipses. CVPR'10, pages 2225–2232, 2010.

[3] R. Grasset, A. Dünser, and M. Billinghurst. Moving between contexts-a user evaluation of a transitional interface. 2008.

[4] J. Grubert, R. Grasset, and G. Reitmayr. Exploring the design of hybrid interfaces for augmented posters in public spaces. NordiCHI,'12.

[5] S. Guven, S. Feiner, and O. Oda. Mobile augmented reality interaction techniques for authoring situated media on-site. ISMAR'06, pages 235–236, 2006.

[6] T. Hoang and B. Thomas. Augmented viewport: An action at a distance technique for outdoor ar using distant and zoom lens cameras. ISWC'10, pages 1–4, 2010.

[7] 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. *C & G*, 23(6):779–785, 1999.

[8] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. SGP '06, pages 61–70. Eurographics Association, 2006.

[9] G. A. Lee, H. Bai, and M. Billinghurst. Automatic zooming interface for tangible augmented reality applications. Applications SIGGRAPH'12, pages 9–12, 2012.

[10] G. A. Lee, U. Yang, Y. Kim, D. Jo, K.-H. Kim, J. H. Kim, and J. S. Choi. Freeze-set-go interaction method for handheld mobile augmented reality environments. VRST'09, pages 143–146, 2009.

[11] A. Mulloni, A. Dünser, and D. Schmalstieg. Zooming interfaces for augmented reality browsers. MobileHCI'10, pages 161–170, 2010.

[12] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. ISMAR'11, pages 127–136, 2011.

[13] C. Niederauer, M. Houston, M. Agrawala, and G. Humphreys. Noninvasive interactive visualization of dynamic architectural environments. I3D'03, pages 55–58, 2003.

[14] W. Piekarski and B. H. Thomas. Augmented reality user interfaces and techniques for outdoor modelling. I3D'03, pages 225–226, 2003.

[15] M. Sukan, S. Feiner, B. Tversky, and S. Energin. Quick viewpoint switching for manipulating virtual objects in hand-held augmented reality using stored snapshots. ISMAR'12, pages 217–226, 2012.

[16] M. Tatzgern, R. Grasset, E. Veas, D. Kalkofen, H. Seichter, and D. Schmalstieg. Exploring distant objects with augmented reality. JVRC '13, pages 49–56, 2013.

[17] E. Veas, A. Mulloni, E. Kruijff, H. Regenbrecht, and D. Schmalstieg. Techniques for view transition in multi-camera outdoor environments. GI'10, pages 193–200, 2010.