Ambient Point Clouds for View Interpolation

Michael Goesele Jens Ackermann Simon Fuhrmann Carsten Haubold Ronny Klowsky
TU Darmstadt

Drew Steedly Richard Szeliski
Microsoft

Figure 1: *View interpolation with ambient point clouds, representing uncertain geometry, shown at different time steps.*

Abstract

View interpolation and image-based rendering algorithms often produce visual artifacts in regions where the 3D scene geometry is erroneous, uncertain, or incomplete. We introduce ambient point clouds constructed from colored pixels with uncertain depth, which help reduce these artifacts while providing non-photorealistic background coloring and emphasizing reconstructed 3D geometry. Ambient point clouds are created by randomly sampling colored points along the viewing rays associated with uncertain pixels. Our real-time rendering system combines these with more traditional rigid 3D point clouds and colored surface meshes obtained using multi-view stereo. Our resulting system can handle larger-range view transitions with fewer visible artifacts than previous approaches.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Image-based Rendering

Keywords: ambient point cloud, uncertain geometry

1 Introduction

Although a lot of progress has been made since the introduction of view interpolation and image-based rendering in the mid-90s [Chen and Williams 1993; McMillan and Bishop 1995; Levoy and Hanrahan 1996; Gortler et al. 1996; Buehler et al. 2001], these techniques still often produce objectionable artifacts when applied to real-world data. For example, errors in reconstructing the true 3D geometry, due to stereo matching errors or failures of ranging systems, result in the implausible 3D motion of surfaces, as well as visible ghosting wherever inconsistent source images are cross-faded during transitions. Incompleteness in the reconstructed geometry can also lead to visible cracks and holes during view interpolation.

In this paper, we show how embracing the uncertainty in 3D reconstructions can help mitigate these visual artifacts. In our system, we render uncertain regions as translucent three-dimensional *segments* along each pixel's viewing ray. The effect of this rendering is to dissolve uncertain pixels into *streaks* aligned with the pixel's motion between the source cameras, which then resolve into the true image as the destination viewpoint is approached. In our current implementation, these segments and streaks are rendered as a collection of randomly sampled colored points, which we call the *ambient point cloud*. This rendering approach to uncertain geometry has two benefits. First, ghosting due to wrongly assigned depth values is reduced, since source and destination pixels map to overlapping streaks during the transition. Second, the ambient point clouds generated along uncertain rays fill in the visible cracks and holes in the rendering with a soft non-photorealistic colored wash, which disguises defects in the geometry while intensifying the perception of 3D structure and motion.

The ambient point cloud, however, is only meant to mitigate the artifacts generated by incomplete or inconsistent geometry. In regions where the 3D model is accurately reconstructed, we wish to retain crisp photorealistic view transitions. To this end, we combine the ambient point cloud with traditional 3D point clouds as well as texture-mapped 3D meshes produced by our multi-view stereo system. To optimize the visual quality of this latter component, we introduce a number of extensions to the 3D modeling pipeline, including the removal of small isolated regions and holes in the reconstruction.

Our paper therefore contains two main contributions:

- the concept of visualizing uncertainty in depth by distributing unknown geometry along a bounded segment of the viewing ray;
- a real-time rendering system for view interpolation which combines different geometric representations and rendering approaches.

The remainder of the paper is organized as follows: We first give an overview of previous work (Sect. 2) before introducing ambient point clouds (Sect. 3). We then describe the reconstruction pipeline (Sect. 4) and rendering approach (Sect. 5). Finally, we show results (Sect. 6) and conclude with an outlook on future work (Sect. 7).

2 Previous Work

Image-based rendering systems have their roots in the early work on view interpolation [Chen and Williams 1993] and plenoptic modeling [McMillan and Bishop 1995]. Seitz and Dyer [1996] present an improved view morphing algorithm that correctly models rigid camera motion and perspective projection. Their goal, like ours, is to create compelling transitions between pairs of images.

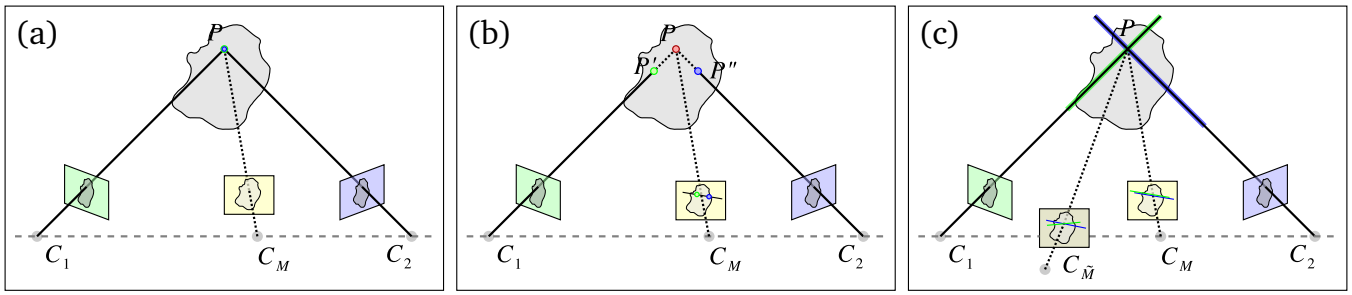

Figure 2: *Epipolar geometry with unknown depths. (a) A point P with consistent depth in both depth maps is projected consistently in arbitrary views C_M. (b) Given an only approximate geometry, the scene point P is in general modeled as distinct points P' and P'' in the two input views. The reconstructed point from C_1 will always overlap with the projection of the corresponding viewing ray from C_2 in any view C_M. For an intermediate camera position C_M located on $\overline{C_1C_2}$ the two points P' and P'' are projected onto two different locations on the corresponding epipolar line. (c) If P has an unknown depth in both depth maps, the corresponding viewing rays intersect at P. In general, their projections in an arbitrary view $C_{\tilde{M}}$ will result in two intersecting lines segments. However, if the camera center C_M is located on $\overline{C_1C_2}$, both lines are projected to a single line.*

Pulli *et al.* [1997] introduce view-based rendering, which represents the scene as textured depth maps acquired using active range scanning. An image from a novel viewpoint is rendered as a weighted blend of the three closest depth maps. They also introduce a soft z-buffer, which handles occlusions and blends pixels with similar z-values. Layered depth images [Shade et al. 1998] extend this idea by storing multiple depth and color values per pixel to avoid disocclusion artifacts. Zitnick *et al.* [2004] combine these ideas in their stereo-based system for video interpolation, which augments depth maps with a partial second layer near depth discontinuities.

Several other systems that render depth maps created using computer vision techniques have been proposed. Narayanan *et al.* [1998] capture images using a camera dome and compute global and per-view geometry models for rendering. Given a long video sequence of a scene, Heigl *et al.* [1999] hierarchically compute a piece-wise planar scene reconstruction from triples of camera views and use an image-based approach to render novel views. Due to the density of input images, they observe only minor ghosting artifacts. Lhuillier and Quan [2003] first reconstruct per-view depth maps and introduce a consistent triangulation of depth maps for pairs of views. Evers-Senne and Koch [2003] reconstruct dense but incomplete depth maps from a dense set of images captured under controlled conditions. At render time, they project a subset of these depth maps into the novel view and smoothly interpolate any remaining holes. Hornung and Kobbelt [2009] improve on this by using feature-aware particles and performing pixel-accurate color accumulation in a fully GPU-based pipeline. Of all these systems, the last two are probably the closest to our proposed approach, but they still rely on relatively complete geometric representations.

Photo tourism [Snavely et al. 2006] provides an interactive browsing tool for large photo collections. Global scene context is provided using NPR rendering techniques. Images are projected onto planes in space, yielding parallax artifacts when the user moves between views. These artifacts are reduced in Snavely *et al.* [2008a] who align the proxy planes to stabilized features in the scene. Recently, Furukawa *et al.* [2009] and Sinha *et al.* [2009] create piecewise planar proxies for interior and exterior scenes. We improve on those techniques by introducing a much more detailed geometric model that does not require planarity. A first step in this direction has been made by Shahrokni *et al.* [2008], who triangulate sparse scene points to create a global scene model. Their system is, however, only demonstrated on cases with small parallax. In addition, it cannot handle scene parts outside the convex hull of the sparse points illustrating the problems posed by complex scenes.

Finally, there are several lines of work that systematically handle uncertainty and incompleteness in view-based rendering. Hofsetz *et al.* [2004a; 2004b] reconstruct a depth map for each interpolated view using the range-space approach by Ng *et al.* [2002]. They also estimate the depth uncertainty for each pixel using photo-consistency and render the resulting model with ellipsoids whose size is adapted to the depth uncertainty. This approach is mainly applicable to textureless regions where an exact correspondence is hard to find but cannot handle the general case of incomplete data we are addressing. Alternatively, Fitzgibbon *et al.* [2005] avoid explicit geometry reconstruction and focus instead on reconstructing color in the presence of uncertainty using image-based priors. Xu and Chen [2004] capture 3D point clouds using a laser range finder, to which they apply various NPR rendering techniques. In a follow-up paper, Xu *et al.* [2004] focus on the following properties of scanned outdoor environment: *incompleteness* of the scene model, *complexity* of the objects, *inaccuracy* of parts of the scene, and the *large size* of the data sets. All of these also apply to our data sets, although often in a more extreme way (especially regarding incompleteness and inaccuracy). Xu *et al.* apply traditional NPR techniques such as rendering varying-sized strokes to visually mark uncertain areas and mask large holes using sparse rendering styles. In contrast, we propose a geometrically motivated NPR-like effect based on epipolar constraints.

3 Ambient Point Clouds

In this section, we introduce the idea of handling uncertain regions in view-based rendering and propose a point-based representation, which we call the *ambient point cloud*. Figure 2 illustrates the general idea of our approach for the case of a view transition. If a scene point P is correctly reconstructed in the depth maps corresponding to the two views between which we want to render a transition, we can render P from an arbitrary camera location C_M without introducing any artifacts (see Fig. 2a).

If the scene geometry is only reconstructed approximately, e.g., represented by some planar proxy, the viewing rays of the two original cameras C_1 and C_2 in general intersect the approximate geometry at two distinct points P' and P'', more or less far away from P (Fig. 2b). Naturally, this also holds if the geometry in the two depth maps is approximated independently but still inconsistently. For an intermediate camera position C_M located on $\overline{C_1C_2}$, the two points P' and P'' are projected onto two different locations on the corresponding epipolar line. If the error in the approximate geometry is small and the camera positions are sufficiently dense, minor ghosting effects will appear, e.g., as observed by Heigl *et al.* [1999]. If P is located in a homogeneous region, these artifacts may not be

visible at all as shown by Hofsetz *et al.* [2004a; 2004b]. In general, however, the resulting error can be arbitrarily large resulting in disturbing artifacts during rendering.

Spread out geometry. Let us focus for the moment on one of the cameras, e.g., C_1. The only reliable information we have if we cannot infer an accurate geometry for a given pixel, is that P was seen from C_1 as the first object along the corresponding viewing ray. We deal with this problem by not committing to a specific depth for P in C_1. Instead, we distribute the color of P along the viewing ray, i.e., the direction of uncertainty (Fig. 2c). This corresponds to rendering the point P in an intermediate frame C_{M} spread out along its epipolar line.

During a view transition from C_1 to C_2, P as rendered using the original view from C_1 therefore starts as a single point, which spreads out as a streak as the transition progresses. Likewise, P as rendered using the original view from C_2 starts as a streak that gradually collapses to a single point, at the true image of P in C_2. The epipolar geometry guarantees that both projected streaks coincide, i.e., are projected to a *single streak*, located on the projection of the corresponding epipolar plane in C_{M}, as long as C_{M} is located on $\overline{C_1 C_2}$. This results in a smooth visual transition.[1] Likewise, if an object is only visible in one of the two views, e.g., because it is occluded in the other view, it will dissolve into or appear from a streak during the transition.

Spreading range and density. One question remains: How far and with which density should P be spread out along a viewing ray? To avoid the possibility of ghosting, the spreading range should encompass all potentially visible scene content. This ensures that the two projected line segments are identical during the middle part of the transition. If a suitable probability distribution is available (e.g., derived from a photo-consistency volume created during scene reconstruction), the density could be adapted to this distribution. Otherwise, it could be modeled as homogeneous density.

Ambient point clouds. We implement the idea of spreading out geometry by creating a point cloud for each depth map. For each viewing ray corresponding to a non-reconstructed pixel, we create a number of points along that ray inside a range derived from the bounding box of the scene (see Section 4 for details). We found that a pointillist rendering nicely conveys uncertainty and complements the global point cloud, which provides overall context. During rendering, a point dissolves into a streak that spreads out in the direction along which the true scene point P moves.

4 Data Reconstruction

For data reconstruction, we use the following pipeline: The input images are first registered using a robust structure from motion algorithm [Snavely et al. 2008b]. We then compute depth maps for all successfully registered images using a multi-view stereo (MVS) system. Since we wish to apply our technique to a wide variety of scenes, we use the MVS technique of Goesele *et al.* [2007] (with 20 neighbors per image), which has been shown to perform well on a wide variety of input images. For each pixel in an image, MVS either returns a depth and confidence value or marks it as unreconstructed. Because of its conservative settings, it also mostly avoids reconstructing incorrect geometry. Note that in principle, we could use any MVS technique, as long as it can separate certain from uncertain geometry. This could even be achieved by computing a confidence measure in a separate step.

[1] Note that the projected streaks will cross for other camera locations $C_{\tilde{M}}$ yielding potentially disturbing artifacts (see Fig. 2c). In practice however, we observed that a moderate deviation from the ideal camera position results in renderings with negligible artifacts.

Figure 3: *One of the input images in the town square data set and the corresponding raw (middle) and final depth map including the background plane (right).*

Given this data, we create three representations for rendering: cleaned and filled *per-view depth maps* representing the certain geometry visible in each view; *per-view ambient point clouds* representing the areas with unknown depth distributed into the volume; and a *global point cloud* providing some overall scene context similar to Snavely *et al.* [2006].

Per-view depth map. While we could in principle use the raw reconstructed per-view depth maps, we achieve better visual results by removing outliers, i.e., small and isolated clusters of reconstructed pixels as well as moderately-sized holes. We therefore apply the following post-processing pipeline to each depth map. First, we remove small connected components of reconstructed pixels (up to 64 pixels) surrounded by unreconstructed pixels or pixels reconstructed at strongly different depth. Next, we create a smoothly interpolated version of each depth map using the 2D interpolation approach of Szeliski [2006], which places depth discontinuities across intensity edges detected using a Canny edge detector. We then use a graph cut based approach [Hammer et al. 1984] to select which parts of the filled depth map to transfer to the final depth map. Regarding each pixel as a node and establishing horizontal and vertical edges between neighboring pixels, we define costs for labeling each pixel with 0 (uncertain) or 1 (interpolated). The cost to label an unreconstructed pixel with 1 is 1 except for border pixels where we set the cost to 10^6 to avoid filling towards the image border. In general the penalty of having neighboring pixels with different labels is very high, i.e., 10^6. We define two exceptions: If one of the two pixels was originally reconstructed and the other was not, we relax the penalty to 100. If an intensity edge lies between the neighboring pixels the penalty is zero. This configuration encourages geometry boundaries to coincide with intensity edges (preferably) or multi-view stereo reconstruction boundaries. Finally, we again remove small isolated clusters of reconstructed pixels (up to 256 pixels) as explained above. All of the remaining pixels with label 1 are treated as known geometry.

The depth of all non-reconstructed pixels is set to an *impostor plane* orthogonal to the viewing direction at a distance corresponding to the 95^{th} percentile of the reconstructed and interpolated geometry. All pixels with a depth value beyond that plane are clamped to the plane. The impostor plane provides a consistent planar proxy geometry, which is only displayed at the very beginning and end of transitions. An example input image from the Town Square data set and the final depth map used for rendering can be seen in Fig. 3.

Ambient point cloud. For each pixel in the depth map that has not been reconstructed or interpolated, we create a set of N points along its viewing ray. To ensure interactive rendering, we typically set $N = 5$. The union of these points forms the ambient point cloud. Lacking exact information about the scene and the depth distribution of the visible geometry, we use the reconstructed points as lower bound for the extent of the scene. We therefore select depth values d within the interval $[d_{\text{near}} = 0.8 \cdot D_{\text{min}}, d_{\text{far}} = 1.5 \cdot D_{\text{max}}]$ where D_{min} and D_{max} are the minimum and maximum of all depth values in the depth map, respectively. In order to account for the effect that points close to the original camera positions will display larger parallax than points in the distance when projected into C_{M}, we randomly choose depth values such that d^{-1} is uniformly dis-

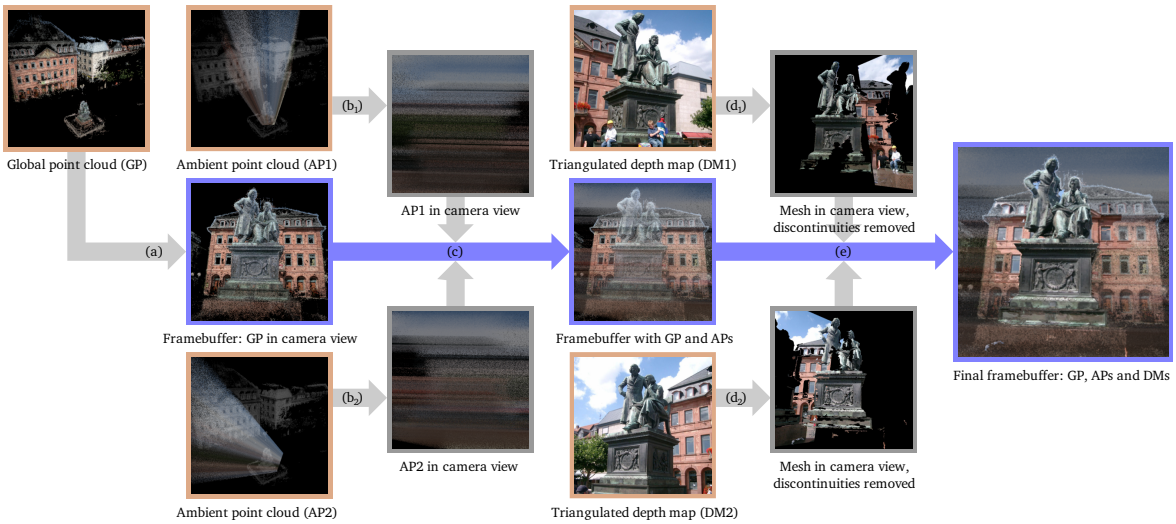

Figure 4: *The rendering pipeline. Input data are marked with beige borders, and intermediate renderings in separate render targets with gray borders. The blue arrows show the content of the framebuffer during various rendering stages. See Sect. 5 for detailed explanations.*

tributed between d_{far}^{-1} and d_{near}^{-1}. Rendering points strictly on their viewing rays creates aliasing artifacts if C_{M} is very close to C_1 or C_2. We avoid this by randomly jittering the direction of the viewing ray for each sample by up to half a pixel in each dimension of the depth map and place the point at distance d along this jittered ray.

Global point cloud. The union of all reconstructed 3D points in all depth maps forms a point-based global model. To create a moderate-sized global point cloud for efficient rendering, we randomly select a subset of these points (typically about two million points) and store their positions and color values as a global point cloud. This rigid point cloud is used during rendering to hint at geometry that lies outside the viewing frusta of the two source images.

5 Rendering

For rendering, we convert each depth map into a textured triangle mesh using naïve triangulation in image space. During a transition, the position, orientation (expressed as a quaternion), and field-of-view of the current camera C_{M} are interpolated linearly. To produce a fluid movement of C_{M}, we use a smoothly varying progress parameter $p(t)$ instead of linear time t as the interpolation parameter. $p(t) \in [0, 1]$ is calculated using univariate, one-dimensional Bézier curves of low degree. Novel views are rendered using the following pipeline (see Fig. 4):

Global point cloud. We first render the global point cloud directly into the framebuffer with depth comparison enabled (Fig. 4a). This yields a global background model and provides context during larger transitions.

Ambient point clouds. We turn off depth comparison and set the blend functions such that color values (including alpha, which is 1 for all points in the ambient point clouds) of all fragments at a location are summed up. We then render each of the two ambient point clouds into separate floating point precision render targets (Fig. 4b). After this, we render a screen-sized quad into the framebuffer (Fig. 4c). In the fragment shader we compute the normalized color of each ambient pixel by dividing its color value by its alpha channel. Pixels without contribution by the ambient point cloud receive a color and alpha value of 0. We blend the colors using the linear transition time t as weight and combine this with the framebuffer using alpha blending.

Per-view depth maps. We clear the depth buffer and re-enable the depth test and standard blending. We then render each of the depth maps into their own render target containing color and depth (Fig. 4d). We use the alpha channel to classify pixels as either belonging to the reconstructed geometry or to the impostor plane.

If a depth map is rendered from a novel viewpoint, elongated triangles spanning depth discontinuities create visible artifacts. We therefore use a custom geometry shader to compute the projected edge length of each triangle on the screen. Fragments belonging to triangles whose longest projected edge exceeds 3 pixels are discarded. This procedure corresponds roughly to the depth discontinuity detection in [Zitnick et al. 2004]. The two main differences are first that we detect depth discontinuities on the fly during rendering, while Zitnick *et al.* [2004] detect them in a preprocessing phase. Second, we do not create a separate boundary layer to cover the depth discontinuities, but use the ambient point clouds to fill in the resulting gaps, which can span an arbitrarily large area.

Finally, we render another screen-sized quad into the framebuffer to combine the two render targets from the previous step with each other and with the rendered point clouds (Fig. 4e). For each pixel, each of the two render targets used for depth map rendering contains either a fragment with finite depth representing real geometry, or the impostor plane, or nothing at all. During transition, we weight the contribution of both render targets with the progress parameter $p(t)$. We display the impostor plane during the first and last 5% of the transition. Geometry is combined using a soft z compare to compensate for noise in the geometry and lighting changes. If the geometry is only visible in one view, we display it with full alpha in order to fill holes caused by occluders, missing reconstruction, or out–of–view areas. In the case of strong lighting changes, holes may get filled with geometry with a different appearance, which can lead to some artifacts. Note that these artifacts can be avoided using appearance stabilization as described in Snavely *et al.* [2008a]. The blending result of the two render targets is combined with the current framebuffer content using alpha blending. This allows the ambient or global point cloud to appear inside gaps and holes in the rendered surface geometry.

6 Results

In the following, we present the results of our approach on the Pisa data set (1103 images), Town Square data set (289 images), and the

Figure 5: *Snapshot at 50% of the transition shown in Fig. 1.* Left: *Fitting a RANSAC plane to the known geometry in each depth map.* Middle: *Rendering the two depth maps together with the global point cloud.* Right: *Adding the ambient point cloud.*

Figure 6: *Town Square data set using just certain geometry (top-left), geometry with global point cloud (top-right), geometry with ambient point cloud (bottom-left), and all combined (bottom-right).*

Church data set (100 images downloaded from Flickr). All data sets were reconstructed fully automatically without user intervention. For more results and animated transitions, please see the accompanying video. Fig. 1 shows an interpolation between two images of the Duomo in Pisa. Note how unreconstructed geometry (e.g., parts of the vegetation and the sky) dissolve into the ambient point cloud, whereas reconstructed geometry such as the tower is always rendered crisply. Fig. 5 shows a snapshot in the middle of this transition. On the left, we blend between two per-view planar impostors created by fitting a plane to the reconstructed geometry using RANSAC. Ghosting artifacts are clearly visible. When rendering just reconstructed geometry and the global point cloud (middle), these artifacts are replaced by holes. On the right, our proposed rendering algorithm is shown. The ambient point cloud fills these holes with a soft non-photorealistic colored wash while retaining a crisp rendering of the reliable geometry. In Fig. 9, we show an extreme view transition, which breaks the limits of traditional view-based rendering.

In Fig. 6, we demonstrate the interplay between the ambient and global point clouds. The figure shows a transition at 30% with different configurations. One can clearly see how the sharp global point cloud complements the streaks in the ambient point cloud.

The ambient point cloud combined with alpha normalization handles occlusions and foreground clutter. Fig. 7 shows the beginning of a transition in the Church data set consisting of images downloaded from Flickr. Note how the tourists disappear and the geometry from the other depth map is shown with full alpha. Fig. 8 shows a transition between two photographs with very different lighting conditions. The soft z-buffer blends smoothly between the

Figure 7: *Beginning of a transition in the Church data set. The figure shows the original starting image (left) and renderings at 30% and 50% of the transition. Note how the occluding persons dissolve in the ambient point cloud. Holes in the church façade are filled using alpha normalization.*

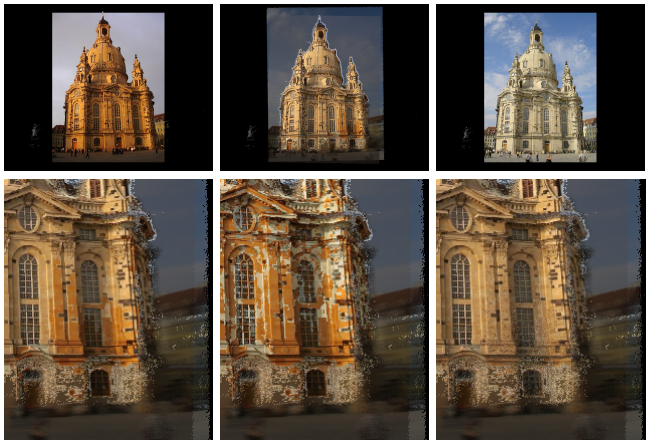

Figure 8: *Lighting changes in the Church data set.* Top row: *transition at 0%, 50%, and 100% transition time.* Bottom row: *close-up at 50% with soft z-buffer and alpha normalization (left), without soft z-buffer (middle), and without alpha normalization (right).*

views except for areas where geometry is only available in one of the depth maps. The close-up (bottom left) demonstrates the advantage of the soft z-buffer. Note however, that incomplete geometry in the second view yields visual artifacts. The remaining images in the bottom row demonstrate the result of turning either of the two effects off.

7 Conclusion

In this paper, we have developed a new image-based view interpolation algorithm, which uses ambient point clouds to represent uncertain portions of the scene. We render these clouds as point-sampled segments along uncertain viewing rays, distributed throughout the expected bounding volume of the scene. As a result, errors or omissions in the reconstruction are masked, as they appear to dissolve into and resolve from elongated streaks. When combined with more traditional rendering primitives such as sparse colored 3D point clouds and texture-mapped surface mesh geometry, these renderings produce three-dimensional transitions masking the visual artifacts such as ghosting and holes associated with previous image-based rendering methods.

As mentioned earlier in the paper, if the virtual camera center does not lie on the line between the original two camera centers, i.e., if we allow for more general camera motion, the streaks from the two cameras may intersect at visible angles, diluting the illusion of coherent 3D motion. One way to mitigate this would be to render the scene using a cross-slit camera [Zomet et al. 2003]. In future work, we would also like to investigate additional volumetric rendering primitives to represent visual uncertainty and ambient point clouds.

Acknowledgements This work was supported in part by the DFG Emmy Noether fellowship GO 1752/3-1 and Microsoft. We

Figure 9: *An extreme view transition breaking the limits of traditional view-based rendering. The global and ambient point clouds fill unknown regions providing a good 3D impression. Foreground occluders dissolve smoothly during the transition.*

thank Michael Cohen for discussions, Sebastian Koch for his help with an earlier version of this work, and the Flickr users Michael Henze and Markus Ellerbrock for the images in Fig. 7 and 8. We also thank SBAAAS (Superintendency for Cultural Heritage, Pisa, Italy) and the Visual Computing Group at CNR-ISTI, Pisa, for their support in acquiring the Duomo data set and Rolf Kruse and Cornelius Weidner for the Town Square data set.

References

BUEHLER, C., BOSSE, M., MCMILLAN, L., GORTLER, S. J., AND COHEN, M. F. 2001. Unstructured lumigraph rendering. *Proc. SIGGRAPH*, 425–432.

CHEN, S. E., AND WILLIAMS, L. 1993. View interpolation for image synthesis. In *Proc. SIGGRAPH*, 279–288.

EVERS-SENNE, J.-F., AND KOCH, R. 2003. Image based interactive rendering with view dependent geometry. In *Proc. EG*, 573–582.

FITZGIBBON, A. W., WEXLER, Y., AND ZISSERMAN, A. 2005. Image-based rendering using image-based priors. *IJCV 63*, 2, 141–151.

FURUKAWA, Y., CURLESS, B., SEITZ, S. M., AND SZELISKI, R. 2009. Reconstructing building interiors from images. In *Proc. ICCV*.

GOESELE, M., SNAVELY, N., CURLESS, B., HOPPE, H., AND SEITZ, S. M. 2007. Multi-view stereo for community photo collections. In *Proc. ICCV*.

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The lumigraph. In *Proc. SIGGRAPH*, 43–54.

HAMMER, P. L., HANSEN, P., AND SIMEONE, B. 1984. Roof duality, complementation and persistency in quadratic 0-1 optimization. *Mathematical Programming 28*, 121–155.

HEIGL, B., KOCH, R., POLLEFEYS, M., DENZLER, J., AND VAN GOOL, L. J. 1999. Plenoptic modeling and rendering from image sequences taken by hand-held camera. In *Proc. DAGM*, 94–101.

HOFSETZ, C., NG, K., CHEN, G., MCGUINNESS, P., MAX, N., AND LIU, Y. 2004. Image-based rendering of range data with estimated depth uncertainty. *CG&A 24*, 4, 34–41.

HOFSETZ, C., CHEN, G., MAX, N., NG, K. C., LIU, Y., HONG, L., AND MCGUINNESS, P. 2004. Light-field rendering using colored point clouds—a dual-space approach. *Presence: Teleoperators & Virtual Environments 13*, 6, 726–741.

HORNUNG, A., AND KOBBELT, L. 2009. Interactive pixel-accurate free viewpoint rendering from images with silhouette aware sampling. *Computer Graphics Forum 28*, 8, 2090–2103.

LEVOY, M., AND HANRAHAN, P. 1996. Light field rendering. In *Proc. SIGGRAPH*, 31–42.

LHUILLIER, M., AND QUAN, L. 2003. Image-based rendering by joint view triangulation. *TCSVT 13*, 11, 1051–1063.

MCMILLAN, L., AND BISHOP, G. 1995. Plenoptic modeling: an image-based rendering system. In *Proc. SIGGRAPH*, 39–46.

NARAYANAN, P., RANDER, P., AND KANADE, T. 1998. Constructing virtual worlds using dense stereo. In *Proc. ICCV*, 3–10.

NG, K. C., TRIVEDI, M. M., AND ISHIGURO, H. 2002. Generalized multiple baseline stereo and direct virtual view synthesis using range-space search, match, and render. *IJCV 47*, 1-3, 131–147.

PULLI, K., COHEN, M., DUCHAMP, T., HOPPE, H., SHAPIRO, L., AND STUETZLE, W. 1997. View-based rendering: Visualizing real objects from scanned range and color data. In *Proc. EGWR*, 23–34.

SEITZ, S. M., AND DYER, C. R. 1996. View morphing. In *Proc. SIGGRAPH*, 21–30.

SHADE, J., GORTLER, S., HE, L.-W., AND SZELISKI, R. 1998. Layered depth images. In *Proc. SIGGRAPH*, 231–242.

SHAHROKNI, A., MEI, C., TORR, P. H. S., AND REID, I. D. 2008. From visual query to visual portrayal. In *Proc. BMVC*.

SINHA, S. N., STEEDLY, D., AND SZELISKI, R. 2009. Piecewise planar stereo for image-based rendering. In *Proc. ICCV*.

SNAVELY, N., SEITZ, S. M., AND SZELISKI, R. 2006. Photo tourism: Exploring photo collections in 3D. *ACM TOG 25*, 3, 835–846.

SNAVELY, N., GARG, R., SEITZ, S. M., AND SZELISKI, R. 2008. Finding paths through the world's photos. *ACM TOG 27*, 3, 11–21.

SNAVELY, N., SEITZ, S. M., AND SZELISKI, R. 2008. Skeletal graphs for efficient structure from motion. In *Proc. CVPR*.

SZELISKI, R. 2006. Locally adapted hierarchical basis preconditioning. *ACM TOG 25*, 3, 1135–1143.

XU, H., AND CHEN, B. 2004. Stylized rendering of 3D scanned real world environments. In *Proc. NPAR*, 25–34.

XU, H., GOSSETT, N., AND CHEN, B. 2004. Pointworks: Abstraction and rendering of sparsely scanned outdoor environments. In *Proc. EGSR*, 45–52.

ZITNICK, C. L., KANG, S. B., UYTTENDAELE, M., WINDER, S., AND SZELISKI, R. 2004. High-quality video view interpolation using a layered representation. *ACM TOG 23*, 3, 600–608.

ZOMET, A., FELDMAN, D., PELEG, S., AND WEINSHALL, D. 2003. Mosaicing new views: The crossed-slits projection. *TPAMI*, 741–754.