On-the-fly Luminance Correction for Rendering of Inconsistently Lit Point Clouds

Thomas Kanzok
Chemnitz University of Technology
Department of Computer Science
Visual Computing Laboratory
Straße der Nationen 62
09111 Chemnitz, Germany
thomas.kanzok@informatik.tu-chemnitz.de

Lars Linsen
Jacobs University
School of Engineering & Science
Visualization and Computer Graphics Laboratory
Campus Ring 1
28759 Bremen, Germany
l.linsen@jacobs-university.de

Paul Rosenthal
Chemnitz University of Technology
Department of Computer Science
Visual Computing Laboratory
Straße der Nationen 62
09111 Chemnitz, Germany
paul.rosenthal@informatik.tu-chemnitz.de

ABSTRACT

Scanning 3D objects has become a valuable asset to many applications. For larger objects such as buildings or bridges, a scanner is positioned at several locations and the scans are merged to one representation. Nowadays, such scanners provide, beside geometry, also color information. The different lighting conditions present when taking the various scans lead to severe luminance artifacts, where scans come together. We present an approach to remove such luminance inconsistencies during rendering. Our approach is based on image-space operations for both luminance correction and point-cloud rendering. It produces smooth-looking surface renderings at interactive rates without any preprocessing steps. The quality of our results is similar to the results obtained with an object-space luminance correction. In contrast to such an object-space technique the presented image-space approach allows for instantaneous rendering of scans, e.g. for immediate on-site checks of scanning quality.

Keywords

Point-cloud Rendering, Image-space Methods, Luminance Correction, Color-space Registration

1. INTRODUCTION

In the field of civil engineering large structures like bridges have to be surveyed on a regular basis to document the present state and to deduct safety recommendations for repairs or closures. Typically this is done by measuring the structures manually or semiautomatically at predefined measuring points and adding detailed photographs or by using completely automatic 3D scanning techniques.

Nowadays, both dominant surface digitalization techniques, laser scanning [BR02] as well as photogrammetry [SSS06,TS08], produce colored point clouds where the color of each point matches the color of the respective surface part under the lighting conditions at the time of scanning. Many photographs become redundant with this additional information. However, when dealing with large structures one has to do several scans from different points of view in order to generate a complete building model of desired resolution. As in most cases only one 3D scanner is used and relocated for each scan, the time of day and therefore the lighting conditions may differ significantly between adjacent scans or, on a cloudy day, even within one scan.

When combining the different scans to one object representation and using a standard geometry-based registration approach, the resulting point cloud may have severe inconsistencies in the luminance values. Because of the scanning inaccuracies in the geometric measures, the renderings of registered scans exhibit disturbing patterns of almost randomly changing luminance assignment in regions where scans with different lighting conditions overlap. We present an approach to correct the luminance and create a consistent rendering. "Consistent" in this context

means that no local luminance artifacts occur; it does not mean that the global illumination in the rendering is consistent.

A simple, yet effective approach to adjust the luminance is to average the luminances locally within neighborhoods in object space, as outlined in Section 4. We use this approach as a reference to the image-space approach we propose in Section 5. The reason for introducing the image-space approach is that the luminance correction in object space is rather time-consuming and needs to be done in a preprocessing step. The engineers, however, would like to get an immediate feedback during their field trip whether the scans they have taken are capturing all important details and are of sufficient quality. Hence, an immediate rendering of the combined scans is required. Our image-space approach allows for such an instantaneous investigation of the scans since it is capable of producing high-quality renderings of inconsistently lit point clouds at interactive framerates.

2. RELATED WORK

Today both the amount and size of generated surface data are steadily increasing. Beginning with the Digital Michelangelo project [LPC+00], which was the first one generating massive point clouds, the scanning hardware was getting significantly cheaper while producing results of increasing resolution and quality. The datasets that are typically generated these days range from hundreds of million to several billion surface points [WBB+08].

In this setting it is obvious, that global reconstruction of the surface becomes infeasible and the use of local reconstruction techniques, like splatting [LMR07, PzvBG00, RL00] or implicit reconstruction [AA03, ABCO+03, GG07], has become state of the art. However, these approaches still need some preprocessing steps, constricting instant preview of generated data. With the advent of image-space point-cloud rendering techniques [DRL10, MKC07, RL08, SMK07] it became possible to interactively render and explore scanned datasets on the fly without any preprocessing.

These new possibilities open up a whole set of new applications, but also induce new challenges. The sampling of the generated point clouds can be highly varying, making the use of several rendering approaches difficult. This can be circumvented by using level-of-detail methods conveying a nearly uniform sampling in image space [BWK02, GZPG10, RD10].

Registration of color scans, produced under different light conditions, can result in luminance inconsistencies of the resulting colored point cloud. Consequently, renderings of such point clouds exhibit significant high-frequency noise. Removing such noise has always been an important topic in image processing. There exists a vast amount of approaches in this field [BC04, BJ10, KS10, WWPS10], which typically try to remove noise in an image by analyzing the spatial neighborhood of a pixel and adjusting the pixel value accordingly. Adams et al. [AGDL09] propose a *k*d-tree-based filtering which is also able to handle geometry if connectivity information is given. This is not the case for point clouds resulting from standard scanning techniques. The aforementioned approaches are specialized on denoising images and do not utilize the particular nature of point-cloud renderings. A notable example of denoising that was explicitly designed for point clouds was presented by Kawata and Kanai [KK05], but it suffers from the restriction to only two different points for denoising.

We will show how to effectively exploit the whole amount of surface points that project to a pixel for interactively generating smooth renderings of inconsistently lit point clouds.

3. GENERAL APPROACH

Let P be a colored point cloud, i.e. a finite set of points $\mathbf{p} \in \mathbb{R}^3$ where each point is enhanced with RGB color information. Furthermore, we assume that colors stored at the surface points approximate the respective surface color except for luminance correctly. Our goal is to produce an interactive rendering of the point cloud with smoothly varying luminance, following the assumption that neighboring points represent surface parts with similar luminance.

To adjust the luminance of the point cloud we decided to use the HSV color model, since it naturally describes the luminance of a point $\mathbf{p}$ in its V component. Thus, we are able to manipulate luminance without interfering with hue and saturation. The basic approach is to convert the colors of all points to the HSV model, average the V component between selected surface points and convert the colors back to the RGB format for final rendering.

As a first idea one could think of prefiltering the whole point cloud in object space to achieve this goal. We implement this idea by generating a space partition for the point cloud, enabling the efficient generation of neighborhood information. Luminance of neighboring surface points is smoothed to generate a point cloud with smoothly varying luminance, see Section 4. This approach can effectively eliminate the luminance noise in point clouds when choosing a sufficiently large neighborhood. However, it takes a significant amount of precomputation time, especially for massive point clouds with hundreds of million

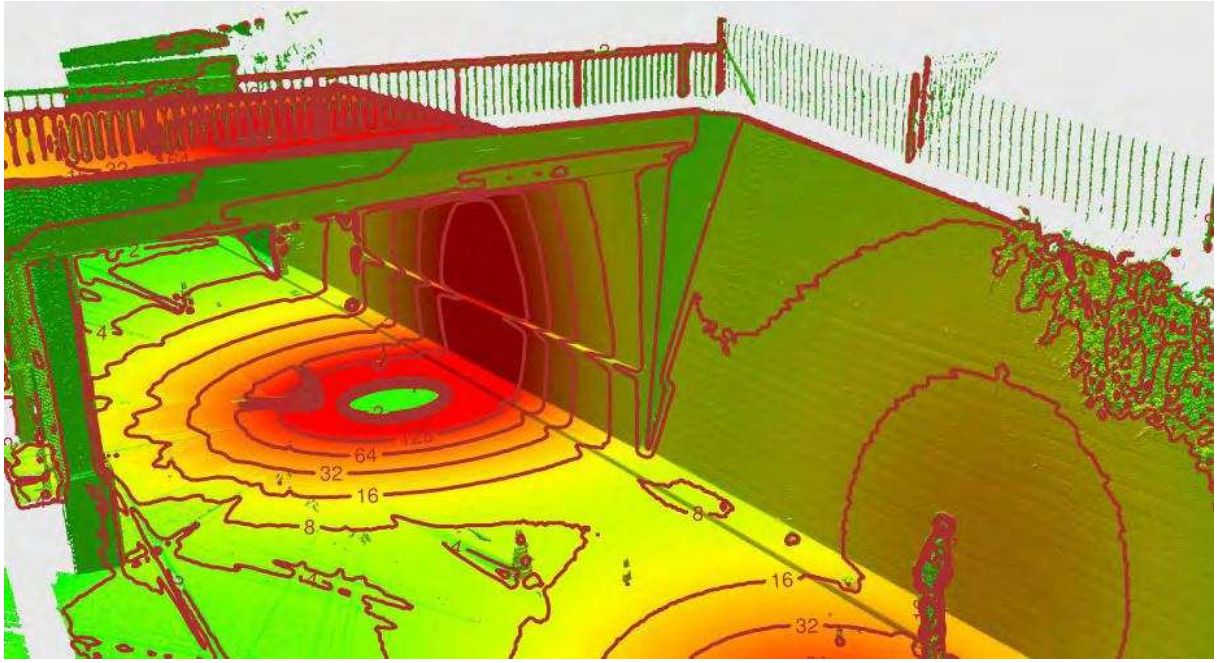


Figure 1. Image-space rendering of a real world point cloud. The number of projected points per pixel is color coded between 1 (green) and 150 (red), which is also emphasized by the isolines. The image was produced with applied depth thresholding and shading to enhance geometry perception.

points, which inhibits the instant rendering of generated point clouds.

To avoid this preprocessing step, we propose a GPU-assisted image-space luminance correction working on the fly. The approach utilizes the fact, that in most cases multiple surface points get projected to one pixel during rendering, as illustrated in Figure 1. When restricting the surface points to those representing non-occluded surface parts, a good approximation for the desired luminance can be obtained by averaging the luminance of the respective surface points. This is done in two rendering steps. In a first pass the scene is rendered to the depth buffer generating a depth mask. Following the idea of a soft z-buffer [PCD+97], an additional threshold ϵ is added to the depth mask, which defines the minimal distance between different consecutive surfaces. In a second render pass the depth mask is utilized to accumulate the luminance of all surface points, effectively contributing to a pixel. A detailed description of the method is given in Section 5.

After this step we apply image-space filters to fill pixels incorrectly displaying background color or occluded surface parts, as proposed by Rosenthal and Linsen [RL08]. The final rendering with associated depth buffer can be used to approximate surface normals per fragment, which opens up several possibilities for calculating postprocessing effects.

4. OFFLINE LUMINANCE CORRECTION

Luminance correction in object space requires the definition of a certain neighborhood for each surface point. We use the n nearest neighbors for each point as neighborhood. For fast detection of these neighborhoods, a three-dimensional kd-tree is built for the point cloud. Since luminance correction is done utilizing the HSV color space, all point colors are converted to this space. Then for each point we compute its neighbors and average their luminance. Finally the complete point cloud is converted back to RGB for rendering.

Note, that for weighted averaging also a kernel function can be used. However, several tests, e.g. with a Gaussian kernel, revealed no significant differences. Regarding the neighborhood size a value of $n=40$ has proven to produce appealing results. However, the precomputation time increases significantly with the number of points and number of neighbors. The luminance correction of a point cloud with 150 million surface points takes for example nearly six hours in an out-of-core implementation. Also when using an in-core implementation, the computation times are far too long for allowing instant views of generated point clouds.

5. IMAGE-SPACE LUMINANCE CORRECTION

Following the main idea of image-space point-cloud rendering, we propose an algorithm that facilitates high-quality point-cloud inspection without preprocessing, utilizing luminance correction in image space. The algorithm takes advantage of the fact that many points are projected to one pixel in models with high point densities, as already shown in Figure 1. Usually a large fraction of these points describe nearly the same geometric position on the surface. The other points belong to parts of the scene which are occluded by the surface closest to the viewer.

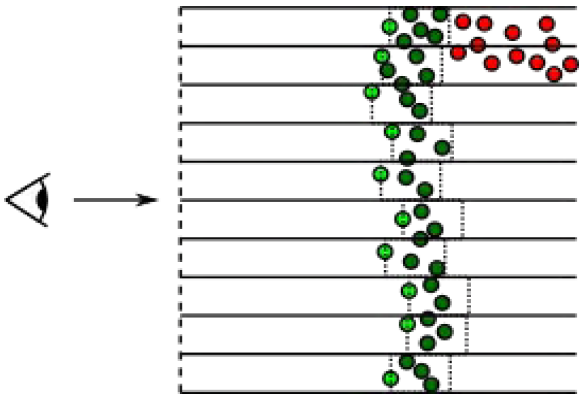
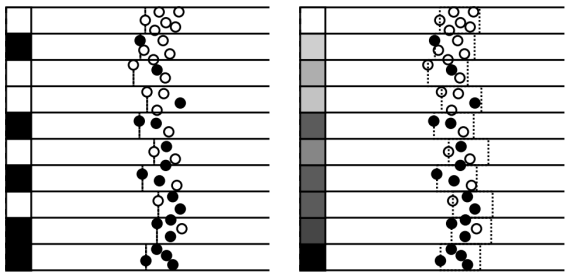


Figure 2. 2D illustration of point projection. The traditional z-buffer test renders only the nearest points (light green). Points representing occluded surface parts (red) but also redundant points on the same surface (dark green) are discarded. By increasing a fragment's z-value by a threshold ϵ (dotted lines) we still discard points from occluded surface parts but are able to blend the luminance of the remaining points for each pixel. (The view plane is the dashed line on the left-hand side.)

Our algorithm for correcting luminance in image space consists of two main rendering steps. In a first rendering pass a (linearized) depth map, selecting all points which represent visible surface parts, is generated (see Figure 2). In a second pass luminance is corrected for each pixel, taking all surface points into account which pass the depth-mask test.

For the first rendering pass all points are rendered resulting in a preliminary depth map and a texture T with the preliminary rendering result. The depth map is extended fragment-wise by the z-threshold, generating the desired, slightly displaced depth mask. Afterwards, we prohibit writing operations to the depth buffer such that every surface point that is closer than the value stored in the depth mask is drawn in the next render pass while farther points are discarded.



a) **b)**

Figure 3. Comparison of the rendering results with (a) normal z-buffering and (b) depth masked luminance correction. The change of predominant luminance at the surface points from top to bottom is rendered much smoother with luminance correction.

In the second render pass we accumulate the luminance of the rendered points using the OpenGL blending with a strictly additive blending function. All surface points are rendered to a non-clamped RGB floating point texture using the depth test. In the fragment shader we set the color of each fragment to (luminance,0,1), which produces a texture $\tilde{T}$ with the accumulated luminances in the R component and the number of blended points in the B component. Finally, we combine the blended texture $\tilde{T}$ with the preliminary rendering result T by converting T to the HSV color space, applying

$$T_{HSV} := \left(T_H, T_S, \frac{\tilde{T}_R}{\tilde{T}_B}\right)$$

and converting the result back to the RGB color space. The result per pixel is a color with averaged luminance over all surface points, which passed the depth test, i.e. which belong to the visible surface.

Note that one can also easily average colors in RGB space by using a four-channel texture for $\tilde{T}$ and blending (R,G,B,1) for each fragment. Then the resulting color would be given by

$$T_{RGB} := \left(\frac{\tilde{T}_R}{\tilde{T}_A}, \frac{\tilde{T}_G}{\tilde{T}_A}, \frac{\tilde{T}_B}{\tilde{T}_A}\right)$$

The difference between traditional z-buffering and our approach is depicted in Figure 3. Although our algorithm requires two rendering passes and therefore basically halves the framerate, we are able to produce smooth lighting much faster than with the preprocessing algorithm, making on-site preview feasible.

An enhancement to the base algorithm is to correct luminance not only in one pixel but to additionally use points from neighboring pixels. We do this by summing the luminance of a fragment in the final step over its 8-neighborhood and dividing by the total number of points. A Gaussian kernel can be used as additional weight for the points of the neighboring fragments to take their distances into account. This produces even smoother results than the simple

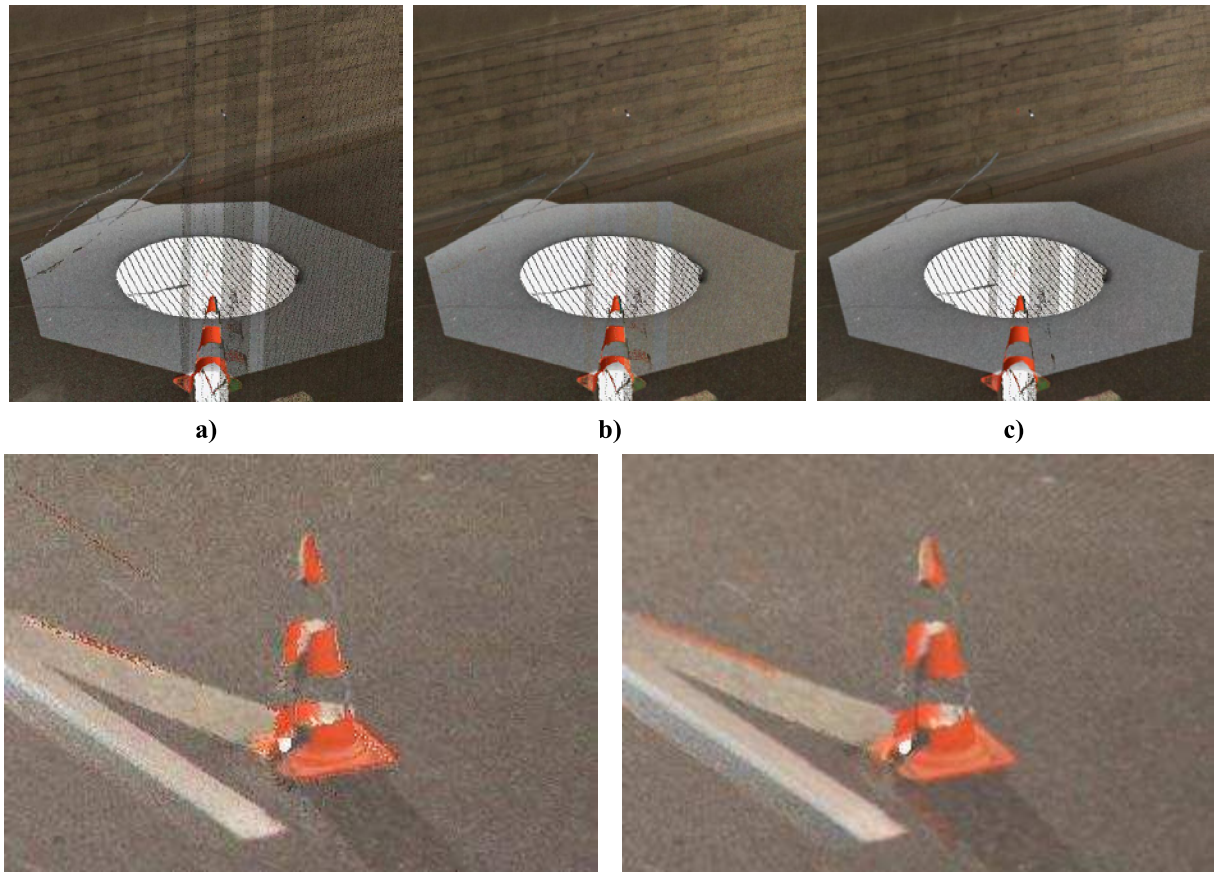


Figure 4. (a) - (c) Comparison of averaging in HSV colorspace without (left) and with Gaussian neighborhood (middle) and RGB space with Gaussian neighborhood (right). A part of the wall is visible between the viewer and the street. While in HSV space the wall is only brightened up but still visible, in RGB space it is smoothed away. (d) - (e) HSV smoothing (left) preserves sharp corners and the structure of the road in the rendering while using RGB (right) produces an antialiased image. In both cases the 3×3 neighborhood has been used.

averaging per fragment while not overblurring the image.

The advantage of using the HSV space lies in the error tolerance when applying the Gaussian. While RGB averaging produces smoother transitions in the image, it tends to blur away details in the rendering. HSV averaging in contrast preserves even small features and sharp edges if this is desired (Figure 4).

The amount of points necessary to successfully use this approach can be roughly approximated by the following calculation: Assume that a spherical volume element with a volume of $1\,\mathrm{cm}^3$ is projected to the center of a Full HD screen. Then the volume element gets projected to a spherical disc with the absolute (world-space) radius $r=\sqrt[3]{\frac{3}{4\pi}}$. Now let α be the vertical field of view of the virtual camera. The projected area of the volume element in distance d from the camera is then given by

$$A=\pi\cdot\left(\frac{1080\,r}{2d\,\tan(\alpha)}\right)^2 .$$

Our experiments have shown that a number of seven to ten points per pixel usually suffice to yield smooth transitions. Therefore, in order to view a model from a given distance d, the resolution k of the dataset in terms of points per cm^3 has to satisfy $k \geq 10\cdot A$, which corresponds to around 50 points per cm^3 for a view from a distance of 5 meters with $\alpha=35°$. If this requirement is met the angle from which the point cloud is viewed is not of great importance since the geometric measuring accuracy is usually very high, producing only minor inconsistencies in the point-to-pixel mapping in adjacent views. It can only pose problems in the immediate vicinity of edges, when the depth threshold is so high that occluded points are included during blending. This, however, can be mitigated by a sufficiently low threshold and did not induce visible errors in our experiments.

Having normalized the luminance of our points we can fill remaining small holes in the rendering using the image-space filters proposed by Rosenthal and Linsen [RL08]. Finally we apply a simple triangulation scheme over a pixel's 8-connected neighborhood in image space to achieve a satisfying rendering enhanced with lighting and ambient occlusion (See Figure 8).

6. RESULTS AND DISCUSSION

We have tested our luminance correction approach in terms of quality and speed with the help of two types of datasets: unaltered real world data (Hinwil) and real world data with artificial noise (Lucy, dataset courtesy of the Stanford 3D Scanning Repository). We implemented the method using C++ and GLSL on an Intel Core i7-970 machine with an NVIDIA Quadro 4000 graphics card. All results were obtained by rendering to a viewport of 1920 × 1080 pixels.

The Hinwil dataset was acquired by scanning a bridge with several stationary high-resolution laser scanners. Two directly adjacent scans, each one containing about 30 million points, were registered using a geometry-based approach to obtain a dataset with 59 million points. The two scans were taken at different times of the day, resulting in different lighting conditions. An image-space rendering of the data is shown in Figure 6. It exhibits three common problems in such data that are highlighted and magnified:

1. Scattering: A part of the object was scanned from different scanners, but the point cloud from one scanner is significantly more dense in that region than the other's. The result is a noisy appearance of the surface because single points are scattered over the surface.
2. Shadowing: A part of the object can be seen from only one scanner, resulting in aliased lighting borders.
3. Border regions: These are the regions, where the point density of two scanners is nearly equal, causing sharp borders when applying a median filter.

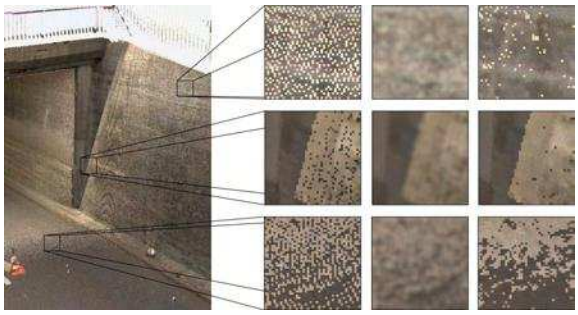
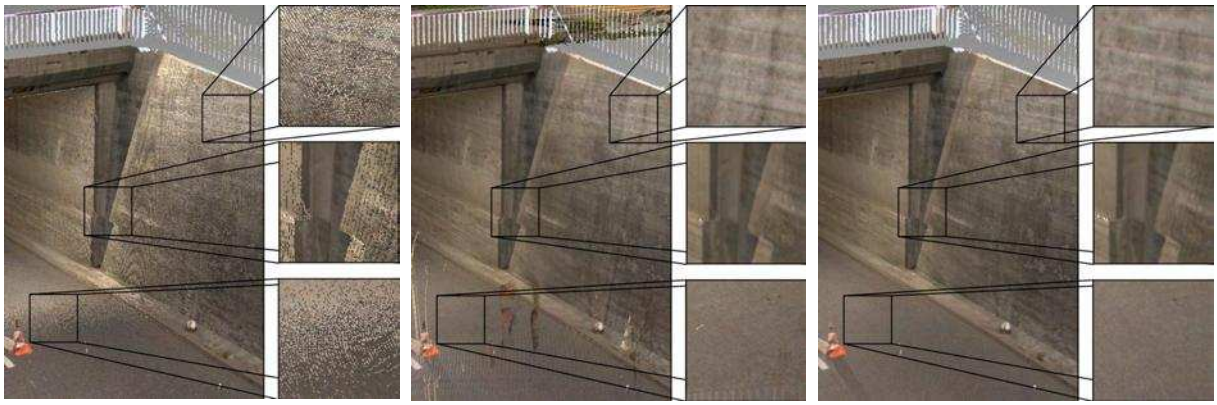


Figure 5. Image-space point-cloud rendering of the Hinwil dataset with closeup views of three problematic regions (left column). The application of standard image filters, like a Gaussian filter (middle column) or a median filter (right column) is not producing satisfying results.

The first problem can, in many cases, be satisfyingly handled by a median filter, which fails at the border regions since it produces a sharp border between the scans. Smoothing with a Gaussian filter yields slightly better results in border regions, but it keeps most of the scattered specks and leads to an overall blurry appearance of the rendered image, as illustrated in Figure 5. The shadowing problem is not easy to solve in image space, since we would have to correct the lighting over larger planar parts of the object.



a) **b)** **c)**

Figure 6. Detail view of the Hinwil dataset using different approaches. For each approach, the overview and three closeup views are presented. (a) Image-space point-cloud rendering, exhibiting typical artifacts in regions with inconsistent lighting. (b) Image-space point-cloud rendering of the complete dataset with offline luminance correction (n=50). The preprocessing effectively eliminates the artifacts. (c) Image-space point-cloud rendering with image-space luminance correction. The visual quality is comparable to the offline method.

a) b)

Figure 7. An image-space rendering of the Lucy model with artificial noise (a) without and (b) with luminance correction. The model was artificially up-sampled to 40 million points to achieve a high-enough point density. The normalization was again carried out over the weighted neighborhood. In this image the prevailing noise is less visible than in the torus rendering, since the surface shape of the model is not as homogeneous as the torus.

# Points	Rendering	Rendering + Correction
1M	140 fps	80 fps
4M	75 fps	40 fps
16M	19 fps	10 fps
64M	5 fps	2.5 fps

Table 1. Performance of luminance correction. For different numbers of points the performance for just image-space point-cloud rendering and for the combination with image-space luminance correction is given in frames per second.

Our image-space approach eliminates most of these problems and even weakens the sharp shadow borders slightly. In Figure 6(c) a smooth transition from one scan to the other was achieved without emphasizing borders or blurring the image. To judge our approach in terms of quality, we compare the result to the one obtained by offline luminance correction, shown in Figure 6(b). Both approaches achieve similar results in terms of quality. However, the image-space luminance correction is able to interactively display the dataset without time-consuming preprocessing (offline luminance correction took more than one hour of computation time).

To evaluate the performance of our image-space algorithm we used the full Hinwil dataset with 138 million surface points and downsampled it to different resolutions. As shown in Table 1 the average rendering performance decreases by around 45% when applying image-space luminance correction. This is due to the two-pass nature of our approach.

As a real world dataset with artificial luminance noise we used the Lucy dataset, which consists of 14 million surface points. To achieve sufficient point density in close up views we upsampled the model to 40 million points and, since we were only interested in point data, we discarded the given connectivity information and colored each point uniformly white. We simulated the effects of scans under different lighting conditions by shading the surfaces points using the Phong model with a light source randomly positioned on a 90° circular arc directly above the model. An image-space rendering without luminance correction as well as an image-space rendering with

Figure 8. Rendering of a scene with image-space luminance correction, lighting and screen space ambient occlusion.

image-space luminance correction are shown in Figure 7. Our algorithm was able to largely remove the noise and yielded a satisfying rendering.

7. CONCLUSIONS

We have presented an approach for rendering point clouds with corrected luminance value at interactive frame rates. Our luminance correction operates in image space and is applied on the fly. As such, we have achieved our goal to allow for instantaneous rendering of large point clouds taken from multiple 3D scans of architecture. No preprocessing is necessary and the quality of the results is pleasing.

In order to work properly our algorithm relies on a sufficient number of points per pixel. Moreover, we observed in our experiments that single very bright scanlines from far away scanners were blended with darker regions of small point density, still leading to noticeable artifacts. This can be solved by considering only point clouds from directly adjacent scanners for blending which, at the present time, was done manually but will hopefully be automated in a future version.

8. ACKNOWLEDGMENTS

The authors would like to thank the enertec engineering AG (Winterthur, Switzerland) for providing us with the real-world data and for their close collaboration. This work was partially funded by EUREKA Eurostars (Project E!7001 "enercloud").

9. REFERENCES

[AA03] Anders Adamson and Marc Alexa. Ray tracing point set surfaces. In SMI '03: Proceedings of the Shape Modeling International 2003, pp.272–279, Washington, DC, USA, 2003. IEEE Computer Society.

[ABCO+03] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. IEEE Transactions on Visualization and Computer Graphics, 9:3–15, 2003.

[AGDL09] Andrew Adams, Natasha Gelfand, Jennifer Dolson, and Marc Levoy. Gaussian kd-trees for fast high-dimensional filtering. ACM Transactions on Graphics, 28:21–33, 2009.

[BC04] Danny Barash and Dorin Comaniciu. A common framework for nonlinear diffusion, adaptive smoothing, bilateral filtering and mean shift. Image and Vision Computing, 22(1):73 – 81, 2004.

[BJ10] Jongmin Baek and David E. Jacobs. Accelerating spatially varying gaussian filters. ACM Transactions on Graphics, 29:169–179, 2010.

[BR02] Fausto Bernardini and Holly Rushmeier. The 3d model acquisition pipeline. Computer Graphics Forum, 21(2):149–172, 2002.

[BWK02] M. Botsch, A. Wiratanaya, and L. Kobbelt. Efficient high quality rendering of point sampled geometry. In Proceedings of the 13th Eurographics workshop on Rendering, pp.53–64, 2002.

[DRL10] Petar Dobrev, Paul Rosenthal, and Lars Linsen. Interactive image-space point cloud rendering with

transparency and shadows. In Vaclav Skala, editor, Communication Papers Proceedings of WSCG, The 18th International Conference on Computer Graphics, Visualization and Computer Vision, pp.101–108, Plzen, Czech Republic, 2010. UNION Agency – Science Press.

[GG07] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. ACM Transactions on Graphics, 26, 2007.

[GZPG10] P. Goswami, Y. Zhang, R. Pajarola, and E. Gobbetti. High quality interactive rendering of massive point models using multi-way kd-Trees. In Pacific Graphics Poster Papers, 2010.

[KK05] Hiroaki Kawata and Takashi Kanai. Direct point rendering on gpu. In George Bebis, Richard Boyle, Darko Koracin, and Bahram Parvin, editors, Advances in Visual Computing, volume 3804 of Lecture Notes in Computer Science, pp.587–594. Springer Berlin / Heidelberg, 2005.

[KS10] Michael Kass and Justin Solomon. Smoothed local histogram filters. ACM Transactions on Graphics, 29:100–110, 2010.

[LMR07] Lars Linsen, Karsten Müller, and Paul Rosenthal. Splat-based ray tracing of point clouds. Journal of WSCG, 15:51–58, 2007.

[LPC+00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3d scanning of large statues. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques, SIGGRAPH '00, pp.131–144, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[MKC07] Ricardo Marroquim, Martin Kraus, and Paulo Roma Cavalcanti. Efficient point-based rendering using image reconstruction. In Proceedings of the Symposium on Point-Based Graphics, pp.101–108, 2007.

[PZvBG00] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: surface elements as rendering primitives. In SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp.335–342, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[RD10] R. Richter and J. Döllner. Out-of-core real-time visualization of massive 3D point clouds. In Proceedings of the 7th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa, pp.121–128, 2010.

[RL00] S. Rusinkiewicz and M. Levoy. QSplat: a multiresolution point rendering system for large meshes. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp.343–352, 2000.

[RL08] Paul Rosenthal and Lars Linsen. Image-space point cloud rendering. In Proceedings of Computer Graphics International, pp.136–143, 2008.

[SMK07] R. Schnabel, S. Moeser, and R. Klein. A parallelly decodeable compression scheme for efficient point-cloud rendering. In Proceedings of Symposium on Point-Based Graphics, pp.214–226, 2007.

[SSS06] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3D. ACM Transactions on Graphics, 25:835–846, 2006.

[TS08] Thorsten Thormählen and Hans-Peter Seidel. 3D-modeling by ortho-image generation from image sequences. ACM Transactions on Graphics, 27:86:1–86:5, 2008.

[WBB+08] Michael Wand, Alexander Berner, Martin Bokeloh, Philipp Jenke, Arno Fleck, Mark Hoffmann, Benjamin Maier, Dirk Staneker, Andreas Schilling, and Hans-Peter Seidel. Processing and interactive editing of huge point clouds from 3D scanners. Computers & Graphics, 32(2):204–220, 2008.

[WWPS10] Z. Wang, L. Wang, Y. Peng, and I. . Shen. Edge-preserving based adaptive icc method for image diffusion. In Proceedings of the 3rd International Congress on Image and Signal Processing, CISP 2010, volume 4, pp.1638–1641, 2010.