Dynamic Level of Detail for Tiled Large High-Resolution Displays

Christian Scheel*, Falko Löffler†, Anke Lehmann⋆, Heidrun Schumann*, Oliver Staadt*

* Institute of Computer Science
University of Rostock
18051 Rostock
christian.scheel@uni-rostock.de
heidrun.schumann@uni-rostock.de
oliver.staadt@uni-rostock.de

† arivis AG
18055 Rostock
f.loeffler@arivis.com

⋆ Faculty of Civil Engineering
TU Dresden
01062 Dresden
anke.lehmann@tu-dresden.de

Abstract: Large high-resolution displays (LHRDs) combine high pixel density with a large display surface area. Level-of-Detail (LOD) methods can be used to accelerate view-dependent rendering of large data sets on such display systems. We present a new method for dynamic, view-dependent level-of-detail rendering for tiled display walls. Based on the user's tracked head position and orientation, we determine the visible display tiles. Subsequently, we calculate the suitable LOD for each visible tile based on its location in the user's field of view. We demonstrate the utility of our approach in a collaborative visualization setting for high-resolution digital terrain data.

Keywords: large high-resolution displays, level of detail, real-time rendering

1 Introduction

Large high-resolution Displays (LHRD) are becoming increasingly popular environments for visualizing complex data. LHRDs can be designed as CAVE systems [Kuh14, DDS+09], multi-monitor desktop setups or arrays of LCD panels or projectors [SFF+00, CSWL02]. For more details on LHRDs see Ni et al. [NSS+06]. The combination of high pixel density and large display surface area allows the user to explore high-resolution data sets without explicit zooming. The user can observe small details by simply stepping closer to the display and the larger context is visible by stepping further back. In remote collaboration sessions, where two LHRDs are connected over a network, it is often necessary to transfer big data sets between the participating sites. To reduce the amount of data that has to be transmitted, foveated rendering for reducing the amount of data can be used. Foveated rendering selectively renders and transfers the data that the viewer can perceive at any given time, which requires only a subset of the display surface if the viewer is located close to the display [AKS+08, GFD+12]. One possibility to reduce the data is the use of level-of-detail (LOD) rendering. Dynamic LOD rendering reduces the complexity of the data with respect to the point of view in a virtual 3D scene.

In LHRD settings, we cannot assume a fixed distance between the user and the display.

Therefore, a dynamic LOD approach has to consider the current position and viewing direction of the user with respect to the display surface. For example, if the user is located close to the display, we may need an LOD level with a sub-pixel screen-space error, while users will not perceive errors of multiple pixels if they are further away. Furthermore, users might not see information presented on peripheral tiles of a LHRD if they are located near the display.

For that reason, we propose to combine dynamic LOD approaches with foveated rendering. We introduce a novel approach that uses head-tracking data for dynamic LOD calculations to speed up rendering, to save computing capacity, and for data reduction during transmission. We divide the LHRD surface into non-overlapping tiles. For tiled LCD walls, these tiles are usually identical to the size of individual LCD panels, but we can consider tiles in a more general sense. For each of these tiles, a screen-space error will be calculated dynamically in relation to the tracking information from the user. The screen-space error metric has to ensure that there is no perceivable difference between the original high-resolution data set and the LOD rendered data and has to be evaluated in real time.

We demonstrate the utility of our approach using a terrain rendering application. It requires a multiresolution representation, because terrain data mostly consist of height maps and can have millions of data points. Large objects such as terrains have the problem that always a part of the object is near the viewer and another part is far away. So it is a problem to choose one LOD for the whole object. View-dependent LOD methods solve this problem by using different LOD levels for the same object [Paj98, Eri00, BGP09]. For interactive or real-time rendering, multiresolution data structures are needed. We refer to Pajarola et al. for more details [PG07].

2 Our Approach

If the viewer is located close to the display surface, only a subset of LHRD tiles is visible. Here one tile refers to a single LCD panel. The number of visible tiles increases with the distance between the LHRD and the viewer. Thus, it is not necessary to render data for all tiles when the viewer is close. Conversely, if the viewer is further away, the number of visible tiles increases, but the level of detail for each visible tile may decrease.

In our approach, a tile have only one a single LOD level. See Figure 1 for an illustration of our prototype architecture. This has the advantage that differences between LOD levels are masked by the display bezels. In a first step, the visibility of tiles with respect to the user position and viewing direction is calculated, and the tiles are then classified into (i) tiles in the user's central field of view, (ii) tiles in the periphery, and (iii) tiles outside of the user's field of view.

Then, the appropriate screen-space error is determined for each visible tile, so that there is no visible error for the user. The screen-space error for a tile depends on the position and view direction of the user in relation to the position of the tile. Based on this screen-space error the matching LOD level is chosen. Finally, the data will be rendered with the chosen

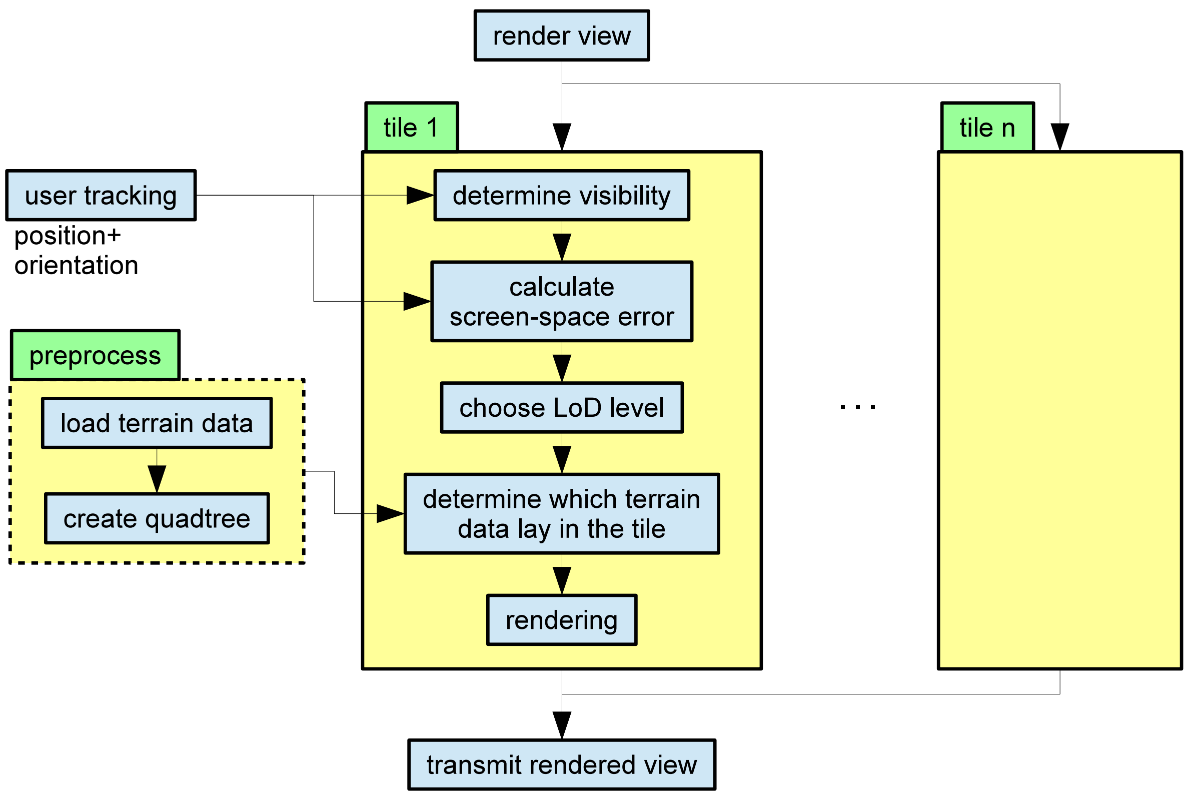


Figure 1: Architecture of our system.

LOD level and, depending on the communication scenario, the reduced rendered view, will be transmitted to the other communication side.

We consider two communication scenarios for data transmission between two communication sites A and B. One scenario is a presenter scenario where as example side A is a presenter and the other side B is the audience. In this case Side A only needs to see his own view, because the presenter doesn't need the views of the audience. But it's necessary to transmit the rendered view from side A to side B so that the audience can follow the presenter's explanations.

Another scenario is the collaborative work between side A and B. Here each side has to render their own view and each side has to know what the other side is seeing. So each side have to transmit his rendered view to the other side. See Radloff et al. [RLSS12] for more details on the two scenarios.

2.1 Tile Visibility

A tile can either be in focus (i.e., in the central view of the user), in the periphery or outside of the field of view. We track the user's head position and orientation to determine tile visibility. Based on the position P and the gaze vector $\vec{g}$ obtained from a tracking and with the physical characteristics of the LHRD (i.e., position, shape, size), the intersection point I between the gaze vector and the LHRD can be calculated, see Appendix A. I is the central focus of the user on the LHRD. It can be possible that the user's line of sight is not intersecting with the display surface.

Note that even if the focus of the user is outside of the display surface, it is necessary to render those tiles, which are in the periphery of the user's current field of view. Therefore, the angles α_i between the line of sight $\vec{s}$ and the vectors $\vec{c_i}$ between the middle points of the tiles to the user position P are determined.

If $\alpha > 0°$ and $\alpha \leq 90°$ then the corresponding tile is in the periphery of the user. If $\alpha = 0$ then it is the central focus point and if $\alpha > 90°$ than a tile is outside the field of view.

If we only use the middle point of a tile for visibility testing, we need to guarantee an error bound for the whole tile. So instead of testing for $\alpha > 90°$, a more conservative angle of 60° is chosen. For a more precise visibility calculation of a tile we would need to calculate the smallest possible angle between the line of sight and the lines from the user position P to all points inside the tile.

2.2 LOD Assignment

We calculate the screen-space error ρ using Eq. 1 based on Ware [War13]:

$$\theta = 2 \cdot \arctan\left(\frac{h}{2 \cdot d}\right), \tag{1}$$

where θ is the visual angle and is approximately one minute of arc for the foveal region of the human eyes with the highest acuity. h is the physical pixel height on the display and d is the distance between the eye and a pixel on a display. Rewriting Eq. 1 yields

$$1 = \frac{2 \cdot d \cdot \tan\left(\frac{\theta}{2}\right)}{h}. \tag{2}$$

Instead of θ in Eq. 2, we use a tolerance angle θ_t:

$$\theta_t = \frac{1 + (1 - \cos(\alpha_i)) \cdot 20}{60}. \tag{3}$$

Since we are using a threshold of $\alpha = 60°$, $0° \leq \alpha_i \leq 60°$, Eq. 3 implies than $\frac{1}{60} \leq \theta_t \leq \frac{11}{60}$. If a user looks straight ahead to the center of a tile, $\alpha = 0°$ and $\theta_t = \frac{1}{60}$ which corresponds to the limit of visual acuity. For a tile in peripheral vision, α becomes bigger and θ_t can be increased because visual acuity is also reduced in the periphery.

For the calculation of ρ, we change Eq. 2 by replacing θ with θ_t as follows:

$$\rho = \frac{2 \cdot d \cdot \tan\left(\frac{\theta_t}{2}\right)}{h} \tag{4}$$

We assume that the data used for LOD rendering is available in a multiresolution representation. Thus, we need to determine the correct multiresolution level for each visible tile. At the borders of the tiles the data can be cut off for this tile. For all parts of the object inside a tile, the LODs in the multiresolution data structure have to be determined based on the calculated screen-space error. This is a *cut* through the multiresolution data structure. This cut is done for each tile and than one cut for the whole multiresolution data structure over the LHRD is constructed that consist of all the cuts from the tiles. If more then one cut exists for the same part of the data the cut with the smaller error (higher details) will be chosen.

3 Implementation and Results

In this section we describe our prototype implementation and experimental results. The architecture of our prototype is shown in Section 2 in Figure 1. In a preprocess the terrain data will be load and the quadtree and tiles will be created. At runtime the following main steps are carried out for each tile.

1. Determine tile visibility
2. Calculate screen space error for the tile
3. Determine LOD level if tile is visible.

3.1 Implementation

Our terrain rendering method is based Löffler et al. [LSS10]. In our implementation, a quadtree is used as the central data structure. A quadtree node comprises its child modes and surface patches. A patch is a collection of triangles which is optimized for GPU rendering. We further use the *Chunked LOD* algorithm from Thatcher [Ulr02] for LOD management.

Additionally, we implemented a *feature selector* connected to a tracked input device. This feature selector corresponds to a fixed field of view of 20°, which allows the user to choose the highest LOD level for a tile inside of the field of view of the feature selector. This feature selector can be used to highlight parts of the terrain regardless of the current position of the user.

3.2 Prototype Setup

We implemented our prototype in C++, using the Vrui toolkit for rendering [Kre08]. Our rendering cluster consists of one master and six rendering nodes. Each rendering node is equipped with two NVIDIA GTX 260 graphics boards, and each graphics board is connected to two displays.

We use 12-camera OptiTrack tracking system from NaturalPoint. The tracked volume in front of the LHRD was approximately 3.4 m wide, 3.0 m deep, and 3.0 m tall.

The LHRD is a tile LCD wall comprising 24 with a resolution of 1920×1200 pixels for each panel, which are arranged is six columns and four rows (see Figure 2). The total resolution of the system is approximately 55 gigapixels.

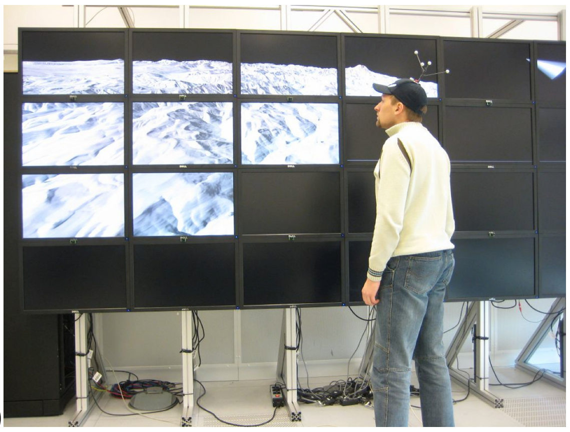
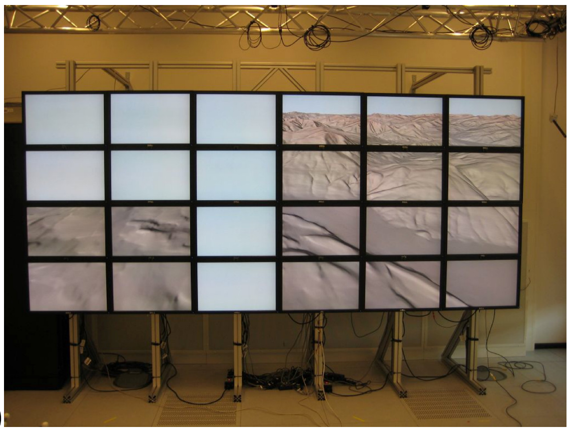
(a) (b)

Figure 2: a: One user is focusing on the upper left corner. b: Two users are focusing on different regions of the display (the users are not shown to avoid occlusion).

3.3 Results

Figure 2a shows the results for a single user focusing on the left side of the display. Figure 2b illustrates two simultaneous users. One user focuses on the right side of the display and the second user on the lower left corner. Note that, for illustration purposes, the users are not visible to avoid occlusion.

For our first experiment, we used a terrain data set with 8096×8096 points. Results are listed in Table 1. The table includes the distance of the user, the frame rate (fps) and the active tiles. The user is always looking at the center of the LHRD, except in the penultimate row. That the frame rates not change significant, except in the last line, is caused through the fact that the frame rate depends on the slowest node. And in our implementation the work balance in the cluster is that each node only cares about the four tiles that are connected to him. So at the moment we have no advantage, from the nodes that have less or nothing to do, for rendering.

The fact that in the last and in the penultimate line in Table 1 the same number of tiles are active but the frames per second change, is a result of our system architecture. In the last line two active tiles belongs to one node of our rendering cluster. In the penultimate line all four active tiles belong to one rendering node. So in the penultimate line a node in our rendering cluster has to render four active tiles and in the last line only two active tiles. This explains the different frame rates.

For the second experiment, we used a terrain model with a resolution of 4096×4096 data points. Results are shown in Table 2. The table includes additional to Table 1 the minimal and maximal screen space error of all tiles, which is calculated for the LOD selection. The frame rate is slowing down from the first row to the penultimate row because the user is stepping closer to the Powerwall. So the minimal screen space error has to decrease to achieve that it's look, in the users perception, as there is no error in the display. Note that in the last row the frame rate is increasing, because of the same reason as described for the first experiment above.

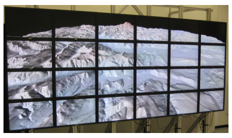
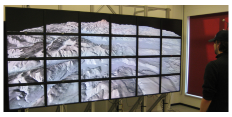
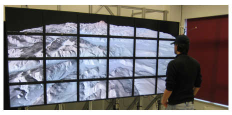
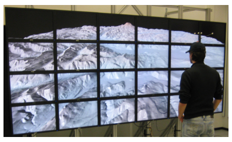
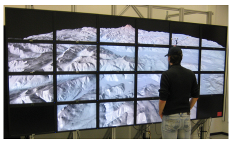
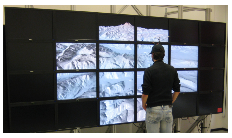
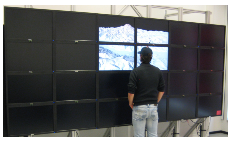
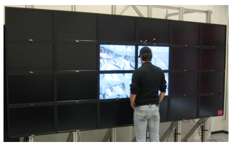
user distance	fps	active tiles
3m	14	
2,5m	12	
2m	11-13	
1,5m	11-15	
1m	11-16	
0,5m	12	
0,2m	12-20	
0,2m	30	

Table 1: Results with a data set with 8096 × 8096 data points. The user is always looking at the center of our Powerwall, except in the penultimate row.

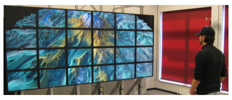
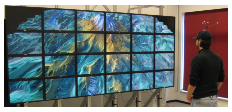
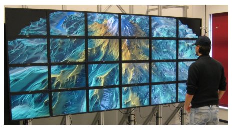
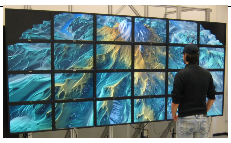
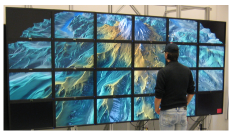
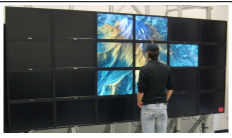
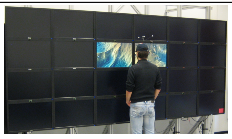
user distance minimal error maximal error	fps	active tiles
3,0m 3,01 37,06	30,0	
2,5m 2,45 33,23	28,7	
2,0m 2,16 17,09	27,5	
1,5m 1,59 16,81	20,0	
1,0m 1,38 11,9	20,8	
0,5m 1,16 6,12	20,0	
0,2m 0,49 –	37,7	

Table 2: Results with a data set with 4096 × 4096 data points.

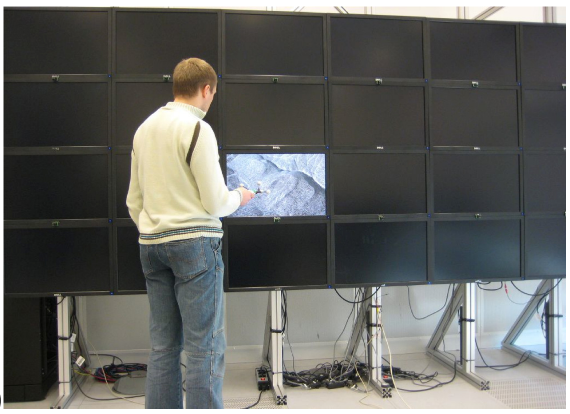
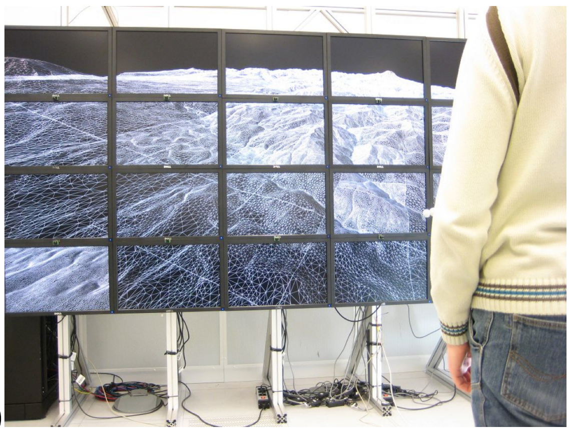
(a)  (b) 

Figure 3: a: Only the tile selected through the feature selector is activated. b: The user is looking in the above right corner and the feature selector points to the lower left corner.

The Perception of switching tiles on and off, for a user, depends on the assumed field of view. In our implementation it's possible to see the switching in the periphery, but because it's in the periphery our experiments have shown that it's not disturbing very much. To avoid a disturbance through the switching fully, a bigger field of view have to be assumed. But this would lead to higher computational costs and a higher data transmission rate.

Figure 3a depicts the use of the feature selector. Figure 3b shows the result when a user is looking at the LHRD and when the feature selector is used in addition.

4 Conclusions

We have presented a concept to do dynamic LOD in real-time on LHRDs for multiple user which depends on the distance and gaze vector of a user and on the position of the LHRD. This can be useful for exploring large data sets like terrains on LHRDs. With our approach we can reduce the computing power and the data transmission amount.

If displays with bigger physical sizes are used all four corners of a display could be checked instead of only the middle point. Otherwise it is possible that a tile is deactivated to early.

We have only used head tracking for determining the gaze vector of a user. So if the users head point in one direction and his eyes look in another direction the determined gaze vector from the head tracking is pointing to a wrong direction. This leads to a wrong rendering on the LHRD. For more accurate results for a user eye tracking is necessary.

A important point for our future work would be to evaluate our results. With the help of user study's we could evaluate if a user is aware of switching tiles on and off and that the display bezels conceal the change between different level of details. An also interesting point would be to measure the system latency, because a user can change his view direction very fast.

We also need to improve the work balance on our cluster to speed up the rendering. At the moment a node in the cluster is only doing the work for his four displays. So if a node have nothing to do because all of his displays are switched off, the node should be used for the rendering work of the other nodes.

5 Acknowledgments

This work was supported by the German Research Foundation (DFG) within the research training group GRK 1424 MuSAMA. We would also like to thank Steffen Girke for his implementation and testing work and the anonymous reviewers for their valuable comments and suggestions.

References

[AKS$^+$08] Benjamin A. Ahlborn, Oliver Kreylos, Sohail Shafii, Bernd Hamann, and Oliver G. Staadt. Design and implementation of a foveal projection display. *International Journal of Image and Graphics*, 8(2):243–261, 2008.

[BGP09] Jonas Bösch, Prashant Goswami, and Renato Pajarola. Raster: Simple and efficient terrain rendering on the GPU. *Eurographics Areas Papers*, pages 35–42, 2009.

[CSWL02] Han Chen, Rahul Sukthankar, Grant Wallace, and Kai Li. Scalable alignment of large-format multi-projector displays using camera homography trees. In *Proceedings of the conference on Visualization'02*, pages 339–346. IEEE Computer Society, 2002.

[DDS$^+$09] Thomas A DeFanti, Gregory Dawe, Daniel J Sandin, Jurgen P Schulze, Peter Otto, Javier Girado, Falko Kuester, Larry Smarr, and Ramesh Rao. The starcave, a third-generation cave and virtual reality optiportal. *Future Generation Computer Systems*, 25(2):169–178, 2009.

[Eri00] Carl M Erikson. *Hierarchical levels of detail to accelerate the rendering of large static and dynamic polygonal environments.* PhD thesis, University of North Carolina, 2000.

[GFD$^+$12] Brian Guenter, Mark Finch, Steven Drucker, Desney Tan, and John Snyder. Foveated 3D graphics. *ACM Trans. Graph.*, 31(6):164:1–164:10, November 2012.

[Kre08] Oliver Kreylos. Environment-independent vr development. In *Advances in Visual Computing*, pages 901–912. Springer, 2008.

[Kuh14] Torsten Kuhlen. Virtuelle Realität als Gegenstand und Werkzeug der Wissenschaft. In Sabina Jeschke, Leif Kobbelt, and Alicia Dröge, editors, *Exploring Virtuality*, pages 133–147. Springer Fachmedien Wiesbaden, January 2014.

[LSS10] Falko Löffler, Sebastian Schwanke, and Heidrun Schumann. A hybrid approach for high quality real-time terrain rendering and optimized a-priori error estimation. In *GRAPP*, pages 233–238, 2010.

[NSS+06] Tao Ni, G.S. Schmidt, O.G. Staadt, M.A. Livingston, R. Ball, and R. May. A survey of large high-resolution display technologies, techniques, and applications. In *IEEE Virtual Reality*, pages 223–236, 2006.

[Paj98] Renato Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. In *Visualization'98. Proceedings*, pages 19–26. IEEE, 1998.

[PG07] Renato Pajarola and Enrico Gobbetti. Survey of semi-regular multiresolution models for interactive terrain rendering. *The Visual Computer*, 23(8):583–605, 2007.

[RLSS12] Axel Radloff, Anke Lehmann, Oliver Staadt, and Heidrun Schumann. Smart interaction management: An interaction approach for smart meeting rooms. In *8th International Conference on Intelligent Environments (IE)*, pages 228 –235, June 2012.

[SFF+00] Daniel R Schikore, Richard A Fischer, Randall Frank, Ross Gaunt, John Hobson, and Brad Whitlock. High-resolution multiprojector display walls. *Computer Graphics and Applications, IEEE*, 20(4):38–44, 2000.

[Ulr02] Thatcher Ulrich. Rendering massive terrains using chunked level of detail control. *SIGGRAPH Course Notes*, 3(5), 2002.

[War13] Colin Ware. *Information visualization: perception for design.* Elsevier, 2013.

A Intersection Point

Based on the position P and the gaze vector $\vec{g}$ obtained from a tracking the user's line of sight $\vec{s}$ is calculated as

$$\vec{s} = P + \lambda \vec{g}. \tag{5}$$

If the LHRD is in a single plane and assuming that the center of the LHRD is the origin of our world coordinate system, the LHRD plane can be described parametrically with the parameters u and t as E:

$$E = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + u \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + t \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} u \\ 0 \\ t \end{pmatrix} \tag{6}$$

To obtain the intersection point between the line of sight and the LHRD we have to solve $\vec{s} = E$. This leads to the following system of equations:

$$\begin{aligned} P_x + \lambda \cdot \vec{g}_x &= u && (7) \\ P_y + \lambda \cdot \vec{g}_y &= 0 && (8) \\ P_z + \lambda \cdot \vec{g}_z &= t && (9) \end{aligned}$$

To determine λ we only have to solve Eq. 8:

$$\lambda = \frac{-P_y}{\vec{g}_y}. \tag{10}$$

Based on Eq. 5 and Eq. 10, we can calculate the intersection point I using

$$I = P + \frac{-P_y}{\vec{g}_y} \cdot \vec{g}. \tag{11}$$