PERGAMON

Computers & Graphics 0 (■■■■) ■■■–■■■

COMPUTERS & GRAPHICS

www.elsevier.com/locate/cag

Technical section

Real-time view-dependent image warping to correct non-linear distortion for curved virtual showcase displays

Oliver Bimber[a,*,1], Bernd Fröhlich[a], Dieter Schmalstieg[b], L. Miguel Encarnação[c]

[a] *Bauhaus University Weimar, Bauhausstraße 11, 99423 Weimar, Germany*
[b] *Vienna University of Technology, Favoritenstrasse 9-11/188, A-1040 Vienna, Austria*
[c] *Fraunhofer Center for Research in Computer Graphics, 321 South Main St., Providence, RI 02903, USA*

Abstract

We present a quadtree-based selective refinement algorithm for real-time, view-dependent image warping used to present images with mirror displays that require a non-linear predistortion. The algorithm applies two-pass rendering, and recursively generates and selectively refines the underlying image grid with locally adapted levels of detail. Each discrete level of detail is determined by computing and evaluating the displacement error on the image plane within the image space of the mirror optics. The algorithm generates and refines only those image portions that contain visible information by considering an oriented convex container of the rendered scene. While display specific details of the algorithm are explained based on the more complex case of a particular mirror display, the Virtual Showcase its general functionality is valid for other non-linear displays.

Keywords: Display algorithms; Viewing algorithms; Hierarchy and geometric transformations; Augmented and virtual realities

1. Introduction and contribution

Displays that require a non-linear predistortion to present undistorted images on non-planar surfaces (e.g. [1–3]) or to neutralize optical distortion (e.g. [4–6]) usually apply multi-pass rendering techniques. Projective textures [7] or uniform grids are used to deform the image generated during the first pass before it is displayed as a texture map during the final pass. However, these approaches do not consider the error generated from a piecewise linear texture interpolation to adapt the underlying geometry.

For instance, curved mirror displays that stereoscopically produce three-dimensional images generally do not predistort the graphics before they are displayed. Yet, some systems apply additional optics, such as lenses, to stretch or undistort the reflected image (e.g. [8,9]). However, these devices constrain the observer to a single point of view or to very restricted viewing zones. For the Virtual Showcase mirror display [4] a view-dependent rendering is required to support freely moving observers. The existing image warping algorithms for curved Virtual Showcases predistort a uniform image grid and consequently do not consider the local error that is generated from a piecewise linear texture interpolation. These algorithms can only produce an acceptable image quality within a significantly large amount of rendering time.

The algorithm presented in this article improves the uniform image warping techniques for curved Virtual Showcases by defining an appropriate error metric and implementing a quadtree-based selective refinement method that generates adapted local levels of detail. It attempts to approaches to produce a predefined

*Corresponding author. Tel.: +49-3643-58-3724; fax: +49-3643-58-3707.

E-mail addresses: oliver.bimber@medien.uni-weimar.de (O. Bimber), bernd.froehlich@medien.uni-weimar.de (B. Fröhlich), dieter@cg.tuwien.ac.at (D. Schmalstieg), me@crcg.edu (L.M. Encarnação).

[1] Formerly at: Fraunhofer Center for Research in Computer Graphics, USA.

0097-8493/02/$ - see front matter
doi:10.1016/S0097-8493(03)00087-6

image quality within a minimum amount of rendering time.

While display specific details of our algorithm are explained based on the Virtual Showcase display, its general functionality is valid for other non-linear displays, such as for curved projection-based systems.

2. Background

In [4] a new mirror display is presented—the Virtual Showcase. It consists of two main parts: the actual showcase covered with a half-silvered mirror coating and a stereoscopic graphics display underneath that is reflected towards the observer. By making use of optical see-through technology, it allows the three-dimensional graphical augmentation of real artifacts placed inside the Virtual Showcase and supports multiple tracked users looking at the Virtual Showcase from different sides. This functionality is supported by the fact that the Virtual Showcase has the same form factor as a real showcase traditionally used for museum exhibits.

Our cone-like prototype is particularly intriguing (cf. Fig. 1). It consists of a single convex curved mirror sheet, which provides an edge-free and seamless surround view onto the displayed artifact. Nevertheless, curved mirrors introduce a curvilinear optical deformation of the reflected image.

An image-based two-pass rendering method has been presented in [4], which warps a generated image by projecting it individually for each pixel (cf. Fig. 2): First, an image of the showcase's virtual content is generated from the observer's point of view using a traditional on-axis perspective projection (cf. Fig. 2-left). This image is geometrically approximated by a simple uniformly tessellated grid that is transformed into the current viewing frustum in such a way that it is positioned perpendicular to the optical axis. Next, the grid vertices are transformed with respect to the viewpoint and the mirror optics and projected onto the display surface (cf. Fig. 2-center). Finally, the image that was generated during the first pass is mapped onto the warped grid using texture mapping and bi- or tri-linear texture filtering during the second pass (cf. Fig. 2-right). This process is repeated for multiple individual viewpoints

Fig. 1. A head-tracked user (left) observes a warped image (right) inside the Virtual Showcase.

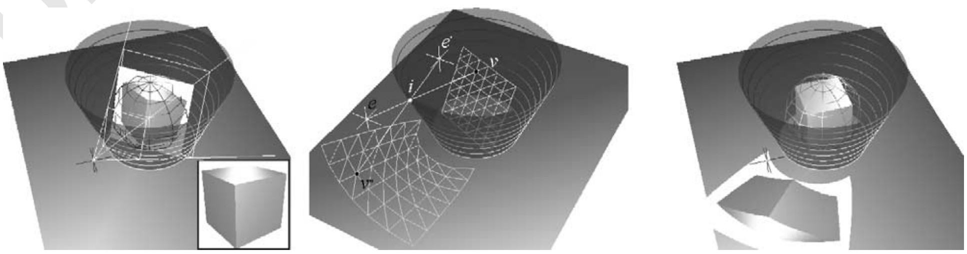

Fig. 2. The image-based two-pass rendering method that is used with a uniform tessellated image gird.

when stereoscopic viewing and/or multiple observers need to be supported.

3. Motivation

If image warping is applied for a non-linear predistortion, the required grid resolution of the underlying image geometry depends on the degree of curvilinearity that is introduced by the display (e.g. caused by the properties of a mirror, a lens, or a projection surface). Pixels within the triangles of the warped image mesh are linearly approximated during rasterization (i.e. after the second rendering pass). Thus, some image portions stretch the texture while others compress it. This results in different local image resolutions.

If, on the one hand, the geometric resolution of an applied uniform grid is too coarse, texture artifacts are generated during rasterization (cf. Fig. 11a). This happens because a piecewise bi- or tri-linear interpolation within large triangles is only a crude approximation to neutralize the curvilinear optical transformations. If, on the other hand, the grid is oversampled to make the interpolation sections smaller (cf. Fig. 11b), interactive frame rates cannot be achieved.

In addition to the speed of the first rendering pass, the performance of view-dependent multi-pass methods depends mainly on the complexity of the image geometry that influences geometric transformation times and on the image resolution that influences texture transfer and filter times.

This article focuses on the image geometry. We introduce a selective refinement method that generates image grids with appropriate local grid resolutions on the fly. This method avoids oversampling and the occurrence of artifacts within the final image. Note that we do not consider a level-of-detail refinement of the scene geometry that is rendered during the first pass. Although this would also reveal an additional speedup for the overall rendering process, it is out of the scope of this article. The interested reader may be referred to the extensive work that has been done in this area (e.g. [10–16]).

The main difficulty for a selective refinement method that supports mirror displays is that in contrast to simpler screens, a displacement error that defines the refinement criterion has to be computed within the image space of the mirror optics, rather than in the screen space of the display. For convexly curved mirrors, this requires fast and precise numerical methods. For displays that do not contain view-dependent optical components (e.g. curved screens), these computations are much simpler because analytical methods or look-up tables can be used to determine the error within the object space (e.g. the screen space).

While Section 5 first presents our image triangulation approach, Section 6 outlines the general recursive grid generation algorithm. Section 7 describes our main generation and refinement criteria, and how to efficiently compute them to achieve interactive framerates. Since the computations of the refinement criteria are partially display specific, some components are explained using the example of the cone-shaped Virtual Showcase. Finally, Section 8 presents the results that have been achieved for our example, and Section 9 outlines the display specific components of the algorithm and describes how they can be adapted for other displays.

4. Related work

The presented work is influenced by two major areas of computer graphics and visualization research and development. Developers of different types of displays need to predistort images in order to compensate for distortion caused by the particular display. On the other hand, the real-time visualization and transformation of complex geometry requires the introduction of areas of interest and corresponding different levels of detail in the rendering process. The following sub-sections outline the related work in both areas.

4.1. Warping for displays

Most traditional displays are designed as centered optical systems. The optics used for head-mounted displays (HMDs), for instance, are normally placed perpendicular on the optical axis (on-axis) and consequently allows for an efficient predistortion to correct geometrical aberrations during rendering. Rolland and Hopkins [17] describe a polygon-warping technique as one possible distortion correction method for HMDs. Since the optical distortion for HMDs is constant, a two-dimensional lookup table is precomputed that maps projected vertices of the virtual objects' polygons to their predistorted location on the image plane. This approach requires subdividing polygons that cover large areas on the image plane. Instead of predistorting the polygons of projected virtual objects, the projected image itself can be predistorted, as described by Watson and Hodges [5], to achieve a higher rendering performance.

Several projection-based displays apply multi-pass rendering and image warping to present undistorted images on non-planar and off-axis surfaces. Raskar et al. [3], for instance, apply projective textures [7] and three-pass rendering to seamlessly project images onto static real surfaces. Subsequently, Bandyopadhyay et al. [1] demonstrated the same approach for movable surfaces. Yang et al. [6] propose to warp uniform grids in combination with multipass rendering to support multiple roughly aligned projectors displaying a unified high-resolution image onto a planar surface. Van Belle et al. [2] integrate geometric predistortion methods that apply image warping into the projector hardware.

Projecting undistorted images with zero-parallax onto planar or non-trivial static (i.e. movable but not deformable) surfaces does not require a view-dependent warping however, even though the generation of the image content might be view dependent.

In contrast to the display approaches described above predistortion for non-centered off-axis displays that possibly integrate additional elements such as curved mirrors or lenses, into the optical path do require a view-dependent image generation and warping. This is also the case for curved Virtual Showcases.

4.2. Level of detail rendering

Recent advances in level-of-detail (LOD) rendering take advantage of temporal coherence to adaptively refine geometry between subsequent frames. Especially, terrain-rendering algorithms locally enhance terrain models by considering viewing parameters.

Hoppe introduced progressive meshes [11] and a later developed a view-dependent refinement algorithm for progressive meshes [12,13]. Given a complex triangle mesh, Hoppe first pregenerates a coarse representation called base mesh by applying a series of edge collapse operations. A sequence of precomputed vertex split operations that are inverse to the corresponding edge collapse operations can then be applied to the base mesh's regions of interest to successively refine them. The selection of the appropriate vertex split operations is based on his refinement criteria. Lindstrom [15] describes a method that generates a view-dependent and continuous LOD of height fields dynamically in real-time, instead of precomputing a coarse base mesh and a sequence of refinement steps. He hierarchically subdivides a regular height field into a quad-tree of discrete grid blocks with individual LODs. Beginning with the highest LOD, Lindstrom locally applies a two-step surface simplification method: he first determines which discrete LOD is needed for a particular region by applying a coarse block-based simplification, and then performs a fine-grained re-triangulation of each LOD model in which vertices can be removed. To satisfy continuity among the different LODs, Lindstrom considers vertex dependencies at the block boundaries.

The main difference between both methods is that Lindstrom performs a dynamic simplification of high-resolution height fields for domains in R^2 during rendering. Lindstrom's mesh definition provides an implicit hierarchical LOD structure. Hoppe applies refinement steps to low-resolution LODs of arbitrary meshes during rendering. His mesh definition is more general and does not require an implicit hierarchical LOD structure. Consequently, the refinement steps and the low-resolution base mesh have to be precomputed. In addition, he applies triangulated irregular networks (TINs) for triangulation, rather than regular grids. Note that these two types of refinement methods may be representative for related techniques.

Since our image grid can also be parameterized in R^2 and provides an implicit hierarchical LOD structure, multiple LODs or appropriate refinement steps do not need to be precomputed but can be efficiently determined on the fly. This is similar to Lindstrom's approach. However, simplifying a high-resolution mesh instead of refining a low-resolution mesh would require to re-transform all grid vertices of the highest LOD after a change of the viewpoint occurred. This is very inefficient, since for the type of displays that we consider, viewpoint changes normally happen at each frame.

In contrast to the static geometry that is assumed in Lindstrom's and Hoppe's case our image-grid geometry is not constant but dynamically deforms with a moving viewpoint. Consequently, the geometry within all LODs dynamically changes as well.

Therefore, we propose a method that dynamically deforms the image geometry within the required LODs while selectively refining the lowest-resolution base mesh during rendering. The method aims at minimizing the displacement error of the image portions, the complexity of the image geometry and consequently the number of vertex transformations and triangles to be rendered.

5. Image triangulation

Instead of transforming and rendering a uniform high-resolution mesh, we start from the coarsest geometry representation and successively refine it locally until certain refinement criteria are satisfied. Due to the well-defined structure of our image grid, all possible global or discrete LODs can be computed at runtime—including the highest, which is the uniform high-resolution representation of the mesh itself.

Fig. 3 illustrates our quadtree-based image triangulation approach, which is similar to Lindstrom's triangulation method for height fields [15]. While Fig. 3-left shows an unrefined patch at LOD n, Fig. 3-right shows the same patch at LOD $n+1$ with lower LOD neighbors. Given a highest LOD of m, we chose an indexed $(2^m+1)\times(2^m+1)$ matrix structure to store the grid vertices.

For illustration purposes we want to differentiate between the following types of patch vertices:

- *L-vertices* are vertices at the corners of a patch (e.g. at indices $[i,j], [i,l], [k,l]$ and $[k,j]$ in Fig. 3-left);
- *X-vertices* are vertices at the center of a patch (e.g. at index $[(k-i)/2, (l-j)/2]$ in Fig. 3-left);

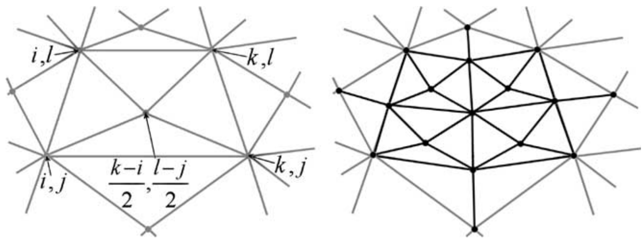

Fig. 3. Triangulation of unrefined patch at LOD n (left), and triangulation of refined patch at LOD $n+1$ with resolution transitions (right).

- *T-vertices* are vertices that split the patch edges after refinement (e.g. at indices $[i,(l-j)/2], [k,(l-j)/2]$, $[(k-i)/2, l]$, and $[(k-i)/2, j]$ in Fig. 3-right).

To refine a patch, it is divided into four sub-patches by computing the corresponding four T-vertices, as well as the four X-Vertices that lie inside the sub-patches. Note that the matrix structure is in our particular case equivalent to a quad-tree data structure. To ensure consistency during rasterization, the T-vertices have to be connected to their neighboring X-vertices wherever a LOD transition occurs, for example at all four neighbors of the refined patch, as shown in the example illustrated in Fig. 3-right.

Due to the well-defined matrix structure that contains the image grid, the following conditions are given:

(i) A clear relationship between the X-vertices and T-vertices exists: X-vertices can never be T-vertices, and vice versa.
(ii) Each patch has definite L-vertices, T-vertices and X-vertices, whose indices can always be computed.
(iii) Each X-vertex can be explicitly assigned to a single patch at a specific LOD.
(iv) Each T-Vertex can be explicitly assigned to exactly one or two adjacent patches at the same LOD.

The triangulation methods described above require *continuous* level-of-detail transitions [16]. This implies that neighboring patches do not differ by more than one LOD.

6. Recursive grid refinement

The objective of this step is to generate an image grid that provides a sufficient local grid resolution (i.e. appropriate discrete LODs) to avoid artifacts within the rasterized texture that would result from under-sampling, as well as oversampling.

The following pseudo-code illustrates our approach to recursively refine a grid patch, which initially is equivalent to the lowest LOD (i.e. the patch at the lowest LOD is outlined by the L-vertices at the four corners of the image geometry):

Algorithm 1. Recursive grid refinement.

RecursiveGridRefinement(i,j,k,l)
1: **begin**
2: $a=(k-i)/2,\ b=(l-i)/2$
3: **if** *GeneratePatch*$([i,j],[i,l],[k,l],[k,j],[a,b])$
4: **begin**
5: *TransformPatchVertices*$([i,j],[i,l],[k,l],[k,j],[a,b])$
6: $P=P\cup\{[i,j,k,l]\}$
7: **if** *RefineFurther*$([i,j],[i,l],[k,l],[k,j],[a,b])$
8: **begin**
9: *RecursiveGridRefinement*(i,j,a,b)
10: *RecursiveGridRefinement*(a,j,k,b)
11: *RecursiveGridRefinement*(i,b,a,l)
12: *RecursiveGridRefinement*(a,b,k,l)
13: **if** $j<2^m+1$ $\mathrm{TC}[a,j]+=1$
14: **if** $k<2^m+1$ $\mathrm{TC}[k,b]+=1$
15: **if** $l>1$ $\mathrm{TC}[a,l]+=1$
16: **if** $l>1$ $\mathrm{TC}[i,b]+=1$
17: **end**
18: **end**
19: **end**

The patch that has to be refined is stored at indices i,j,k,l within our matrix structure. Condition (ii) allows us to locate the position of the patch-individual X-vertex at indices a,b (line 2). First, we evaluate whether or not a patch has to be generated at all (line 3). The conditions that are implemented within the *GeneratePatch* function will be discussed in Section 7.1. The four L-vertices and the X-vertex are transformed from the image plane to the display surface (line 5)—as outlined in Section 2 and described in detail in [4]. For the Virtual Showcase the image plane is located within the image space of the mirror optics. In this case, these mappings composite the vertex individual model-view transformations to neutralize reflection and refraction, as well as the projection transformation that maps a vertex onto the display surface. Note that vertices are only transformed once—even if the recursive refinement function addresses them multiple times. This is realized by attaching a marker flag to each vertex. A reference to the transformed patch is stored in patch set *P* by adding the patch's indices to *P* (line 6). In line 7, we call a function that evaluates the transformed patch based on predefined refinement criteria and decides whether or not this patch has to be further refined. Our main refinement criterion is described in Sections 7.2–7.4. If this decision is positive, the patch is divided into four equal sub-patches and the refinement function is recursively called for all of these sub-patches (lines 9–12). Note that condition (ii) also allows us to determine the indices of the patch's four T-vertices, which become L-vertices of the sub-patches in the next LOD. Consequently, the *GeneratePatch* and the *RefineFurther* functions represent the exit conditions for the recursion.

In Section 5.1 we said that T-vertices have to be connected to their neighboring X-vertices whenever an LOD transition occurs to ensure consistency during rasterization. To detect LOD transitions, we attach a counter (TC) to each T-vertex. This counter is incremented by 1, each time the corresponding T-vertex is addressed during the recursive refinement (lines 13–16). Note that the if-conditions ensure a correct behavior of the counter at the image geometry's boundaries. Due to condition (iv) each counter can have one of the following three values:

- 0: indicates that the T-vertex is located at a boundary edge of the image geometry or it is contained by a discrete LOD that is higher than the required one for the corresponding region.
- 1: indicates a LOD transition between the two neighboring patches that—with respect to condition (iv)—belong to the T-vertex.
- 2: indicates no resolution transition between the two neighboring patches that belong to the T-vertex.

After the image grid has been completely generated, all patches that are referred to in *P* are rendered with appropriate texture coordinates during the second rendering pass. Thereby, the counters of the patch's four T-vertices are evaluated. Depending on their values, either one or two triangles are rendered for each counter. These triangles form the final patch. Counter values of 0 or 2 indicate no LOD transition between adjacent patches. Consequently, a single triangle can be rendered, which is spanned by the T-vertex's neighboring two L-vertices and the patch's X-vertex, illustrated in Fig. 3-right. A counter value of 1, however, indicates a LOD transition. According to Section 5.1, two triangles have to be rendered that are spanned by the T-vertex itself, the two neighboring L-vertices and the X-vertex of the adjacent patch, illustrated in Fig. 3-left.

7. Generation and refinement criteria

This section discusses the patch generation and refinement criteria that are implemented within the *GeneratePatch* and *RefineFurther* functions. The input for these functions is the four L-vertices, as well as the X-vertex of a patch. They deliver the Boolean value *true* if the patch has to be generated, transformed, rendered or further refined, or *false* if this is not the case.

The *GeneratePatch* function that supports an appropriate image clipping, and an evaluation criterion that considers the scene's convex container is described in Section 7.1.

In general, the *RefineFurther* function can represent a Boolean concatenation of multiple refinement criteria, such as maximal patch size, angular deformation, etc. An important refinement criterion for LOD methods is the screen space error. Since the computations of this displacement error are display specific, we describe an important variation of the screen space error that can be applied for convex mirror displays—the *image space error*. This error is explained in greater detail in Sections 7.2–7.4.

7.1. Spatial limits

The multi-pass rendering method that is described in Section 2 uses the scene's bounding sphere to determine the parameters of the symmetric viewing frustum and the image size (cf. Fig. 2-left). Since all image generation methods assume a rectangular image shape that is adapted to today's screen shapes, the bounding sphere provides enough information to determine the rectangular image size.

Bounding spheres, however, are only rough approximations of the scene's extensions and consequently cover a fairly large amount of void space. This void space results in grid patches on the image plane whose texture does not contain visible color information.

To speed up our method, we aim at avoiding these patches while creating the image grid. This implies that these patches are not transformed and refined during the *RecursiveGridRefinement* algorithm and that they are not rendered during the second rendering pass. As a result our algorithm generates and renders an image grid that is not rectangular but dynamically approximates the silhouette of the scene as perceived from the observer's perspective. A condition that causes the recursion to exit in these cases is implemented within the *GeneratePatch* function: We evaluate a tighter convex container, such as an oriented convex hull or a bounding box, of the scene that is generated either in a prepreprocess for static objects, or at runtime for animated scenes. For each untransformed patch that is passed recursively into the *RecursiveGridRefinement* algorithm, we have to determine whether the container is visible on that patch—partially or as a whole. Our approximation is twofold: First, we intersect the geometric lines of sight from e to all four L-vertices of the patch with the front-facing portion of the container. Second, we intersect the geometric lines of sight from e to front-facing container vertices with the patch. If at least one of the resulting rays causes an intersection, the patch might contain visible color information and it will be further processed. If, however, none of the rays cause intersections, the patch is not treated further (i.e. it will not be transformed nor refined or rendered). If the convex container is represented as a triangle mesh, a fast ray-triangle intersection method [18] is applied together with the front-facing container triangles. Note that as for vertex transformations, the intersection information are buffered and looked up in the memory, rather than recomputing them multiple times while evaluating adjacent patches.

We use oriented convex hulls as containers. It is obvious that the complexity of the container influences the performance of this method. Although a precise container can eliminate a maximum number of patches, the number of intersection tests increases with the container's number of vertices and polygons. Our experiments have shown that the highest speedups are reached if the container is as simple as an oriented bounding box or a very coarse but tighter convex hull (cf. Figs. 6, 7, 11). However, the complexity of the convex hull that maximizes the speedup and balances intersection tests with patch generations depends on the scene and the required rendering precision.

7.2. Image space error

The consideration of the screen space error that is determined relative to a display surface is a common measure for many computer graphics methods (e.g. [12,13,15]). In contrast to traditional displays, curved mirror displays create a view-dependent, non-planar image surface, which is located inside the mirror optics. Consequently, an appropriate error function has to consider this optical transformation. Our error is determined by first warping the curved image surface into an image plane, and then computing the screen space error on this plane. We call this error *image space error* (δ_{is}).

The image space error is a variation of a screen space error that can be computed for convex mirror displays that present the image plane within the image space of their optics, rather than on a display surface. We want to define the image space error as the geometric distance between the desired position (v_d) and the actual appearance (v_a) of a point on the image plane. Consequently, the image space error is given by $\delta_{is} = |v_d - v_a|$ and delivers results in image space coordinates (e.g. mm in our case).

In case of our Virtual Showcase mirror optics, the image space is the reflection of the object space (i.e. the physical display screen in front of the mirror optics) that optically overlays the physical space inside the Virtual Showcase. In addition, the optically deformed pixels do not maintain a uniform size within the image space. They are deformed in exactly the same way as the entire image after being reflected from the object space into the image space—although on a smaller scale. Consequently, we chose the Euclidean distance between geometric points as an error metric, rather than expressing the image space error with a uniform pixel size.

For any given pixel on the transformed patch with texture coordinates u, v, we can compute δ_{is} as follows (cf. Fig. 4):

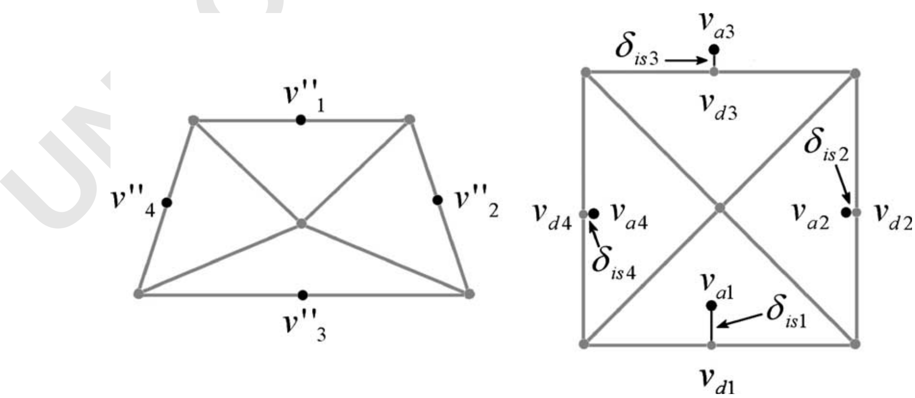

Fig. 4. Samples on transformed patch (left). The distance between desired and actual appearance of samples near the untransformed patch results in the image space error (right).

First, we determine the pixel's world coordinate v'' at u, v within the object space (i.e. on the display surface). Note that the pixels, which are mapped onto the patch's transformed vertices, optically appear at their correct locations on the image plane inside the image space. This is because their exact mappings have been determined during the patch's transformation. This transformation considers the laws of geometric optics, for example reflection and refraction laws. Note also that the pixels that are displayed anywhere else (i.e. inside one of a patch's triangle) do not necessarily appear at their correct locations on the image plane. This is because their positions on the display surface are approximated by a linear texture interpolation, rather than by optical laws.

The second step is to determine the position of the optical image (v') of v'' within the image space of the mirror optics. The projection of v' onto the image plane results in v_a. The transformation from v'' to v_a will be discussed in detail in Section 7.3.

In an ideal case, v_a is located at the position that also maps to the texture coordinates u, v within the untransformed patch. We can identify the location that does this as our desired image position v_d. However, if $v_a \neq v_d$, the image space error δ_{is} for this pixel is non-zero.

We chose to compute the image space errors for the four points on the transformed patch (4-left) that should map to the patch's T-vertices on the untransformed patch (4-right) as if the image space errors were zero for these positions. Obviously, this is not the case in our example, shown in Fig. 4-right.

Since the untransformed patch is a rectangular quad, small image space errors suggest that the optical mapping between the transformed and untransformed patch is linear. Furthermore, we can then conclude that a linear texture interpolation within the displayed transformed patch produces approximately correct results while being mapped (i.e. reflected and refracted) into image space. Consequently, we can say that the resulting image space errors describe a patch's curvilinearity at the representative pixel locations.

To decide whether or not a patch has to be further refined, we determine the largest of the four image space errors. If it is above a predefined threshold value $\bar{\delta}_{is}$ the patch has to be further refined and the *RefineFurther* returns *true*.

7.3. Computing object–image reflections

To compute v_a from a given viewpoint e, the object point v'' and the optic's geometry is equivalent to finding the extremal Fermat path from v'' to e via the optics. In general, this would be a difficult problem of variational calculus.

Beside ray- and beam-tracing approaches, several methods have been proposed that approximate reflection on curved mirrors to simulate global illumination phenomena within rendered 3D scenes. All of these methods face the above mentioned problem in one or the other way.

Mitchell and Hanrahan [19], for instance, solve a multidimensional non-linear optimization problem for explicitly defined mirror objects ($g(x) = 0$) with interval techniques. To compute reflected illumination from curved mirror surfaces, they seek the osculation ellipsoid that is tangent to the mirror surface, whereby its two focal points match the light source and the object point.

For a given viewpoint e, Ofek and Rappoport [20] spatially subdivide the object space into truncated tri-pyramid shaped cells. In contrast to solving an optimization problem, they apply accelerated search techniques to find the corresponding cell that contains the object v'' at interactive rates.

While Mitchell's multidimensional optimization approach is far from being applied at interactive rates, Ofek's search method offers a good approximation for rendering global illumination effects, such as reflections but does not provide the precision required by an optical display.

In the following, we present a numerical minimization method to compute the object–image reflection for specific mirror surfaces, such as cones and cylinders. For such surfaces, we can reduce the optimization problem to only one dimension. Consequently, our method provides an appropriate precision at interactive rates.

For the subsequent example we chose a cone-shaped mirror surface, since this surface type has also been used for our Virtual Showcase display (cf. Fig. 1).

Cones and similar bodies of revolution have the property that multiple surface points lie on a common plane. For example, all points on the straight line spanned by a cone's peak and an arbitrary point on its bottom circle lie on the same plane. Consequently, individual plane parameters can be determined for all angles around the cone's principle axis.

To determine an object's (v''_d) image for a given viewpoint e and mirror surface $g(x) = 0$, we first assume an arbitrary angle α around the cone's principle axis. We then determine the surface's tangent plane TP_1 at α by computing a surface point and the surface normal at α. Since $g(x)$ is an explicit function, we can compute the surface normal by using its first-order derivatives $g/(\partial x)$. Next, we reflect v''_d over TP_1 to its corresponding position within the image space and project it onto the image plane, as described in Section 7.2. In Fig. 5, the projected image point is outlined by v.

To verify the quality of our assumption, we reflect v back into the object space and project it onto the display surface. For a given v, $g(x)$ and e, a simple analytical

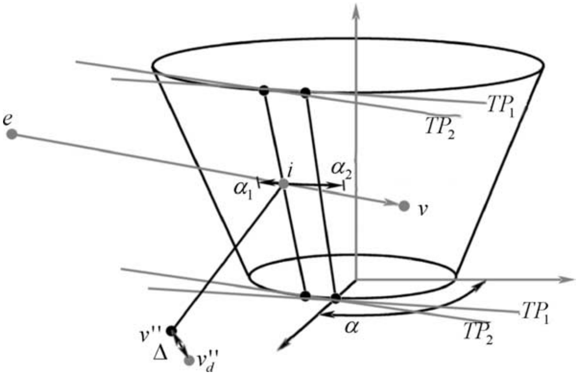

Fig. 5. Object–image reflection via numerical minimization.

solution exists to determine this image–object transformation: The ray spanned by e and v is intersected with $g(x)$ by solving a simple quadratic equation. In the case of our conical Virtual Showcase, the quadratic equation is given by inserting the linear ray equation into the quadratic equation of a cone, and solving for x. The surface intersection i, together with the normal vector at i determined using the surface's first-order derivatives gives the tangent plane TP_2 at i. Reflecting v and e over TP_2 and projecting the reflection of v onto the display surface using the reflection of e as center of projection, results in point v''. The image–object transformation is illustrated in Fig. 2-center. Note that for simplicity, the image–object transformation and the object–image transformation have been described as simple reflection/projection transformations. Normally, they incorporate refraction as well.

If the tangent plane at α produces the extremal Fermat path between v''_d and e, the geometric distance Δ between v''_d and v'' is zero, $TP_1 = TP_2$, and $v_a = v$. Otherwise Δ is non-zero.

To approximate the correct α, we minimize the above-described function for Δ. Since this function depends only on α, we can apply fast numerical optimizers for one dimension. However, because we cannot easily derive its first-order derivatives but it appears to be nicely parabolic near its minimum, we apply Brent's inverse parabolic interpolation [21] with bracketing.

To speed up the minimization process (i.e. to reduce the number of function iterations), we can constrain the function range for a particular v''_d. As illustrated in Fig. 5, the minimization is restricted to find α between α_1 and α_2. These boundaries can be determined as follows: given e and the untransformed patch that belongs to v''_d, we evaluate the projections on the mirror surface of the untransformed patch at the next lower LOD. The two angles α_1 and α_2 at the horizontal extrema of these projections are the corresponding boundary angles. Note that these projections are determined while transforming the patches (i.e. within *TransformPatchVertices*). Thus, for efficiency reasons, they are stored and looked up in the memory, rather than re-computing them again.

Our experiments showed that sufficiently precise solutions can be found after a small number of iterations. Typically, we achieve average errors of $\Delta = 0.002$ mm with an average of 3 iterations. This value is still two orders of magnitude smaller than the smallest image space error we experimented with ($\delta_{is} = 0.1$ mm).

7.4. Error direction propagation

Although the number of iterations is relatively small, the computational expenses of four minimization steps per patch result in a noticeable loss of performance. Especially, while evaluating the large number of higher LOD patches, such an approach might not produce a speedup.

We also noticed that the direction in which the highest image space error (i.e. the one of the four patch sides where δ_{is} is maximal) propagates is the same for higher LOD patches that are derived from the same parent patch.

However, from which level of detail on this is the case depends on the display's properties. If, for instance, the optics produce well-behaved image deformations, the propagation direction of the highest error is consistent for relatively high LODs. If, on the other hand, the optics produces noisy images, the error direction alternates as randomly.

To benefit from this situation, we specify a LOD Λ depending on the optic's and the display surface's properties. For patches that are below Λ, we determine the image space error as described above: we compute δ_{is} at all four edges and find the highest value. In addition, we record the error direction (i.e. the edge where δ_{is} is maximal) for each patch.

For patches that are above Λ we reuse the error direction of the corresponding parent patch and compute δ_{is} only for the edge in this direction, rather than at all four edges. By doing this, we assume that the largest error will occur in the same direction as for the parent patch. Consequently, we reduce the number of minimization steps from four to one for all patches that are above Λ—i.e. for the majority of all relevant grid patches.

Our experiments have shown that this heuristic leads to a significant speedup while producing the same visual final output.

Note that although we deal with a simple mirror shape (i.e. a cone), an algorithm that uses a precomputed look-up table (e.g. such as the one described by Ofek [20]) instead of dynamic numerical minimizations would either result in fairly large data-structures that cannot be efficiently searched, or in a precision that is not adequate for an optical display.

8. Results

Our experiments were concerned with the performance of the described approaches as well as with the resulting image quality that can be expressed by evaluating the average local displacement error on the image plane. Our Virtual Showcase mirror display that was driven by a Pentium IV with 2 GHz and FireGL II graphics acceleration was used to carry out these experiments.

Figs. 6 and 7 illustrate several outcomes of the *RecursiveGridRefinement* algorithm for different image space error thresholds while using the same point of view, scene, and optics (i.e. the Virtual Showcase, as illustrated in Fig. 1). Note that stretched image regions with a lower curvature are rendered with a lower discrete LOD, and compressed regions with a higher curvature are rendered with a higher discrete LOD. Note also that

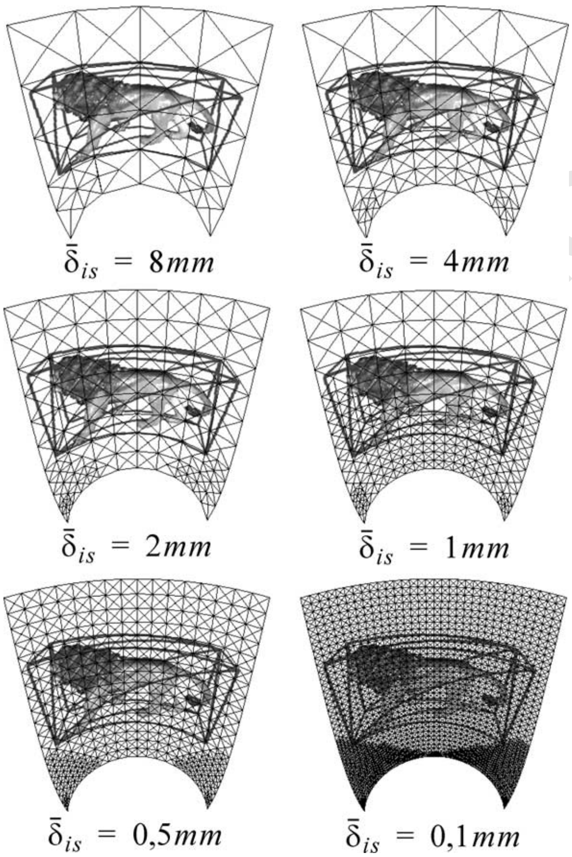

Fig. 6. Selective grid refinement for different image space error thresholds.

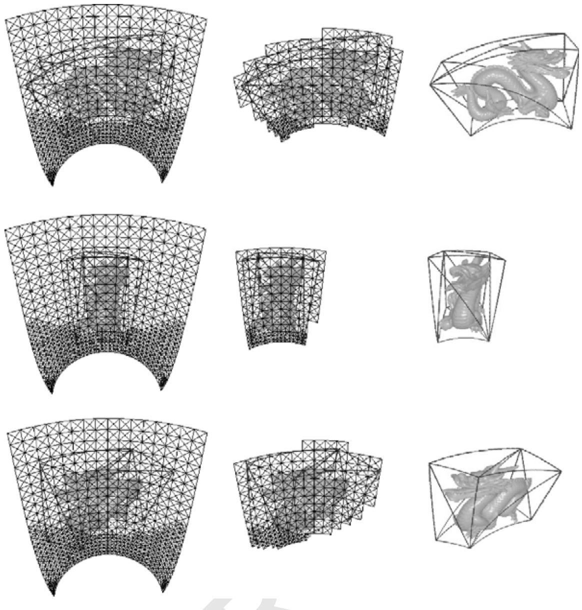

Fig. 7. Spatially limited grids for different perspectives: entire refined grid (left), grid portion limited by container (center), resulting image (right).

LOD transitions are continuous. Fig. 7 reveals how effectively the relevant image geometry can be identified with an oriented bounding box, rather than deforming the entire image grid. The image grids shown in the center row of Fig. 7 are the ones that are actually produced by our algorithm. The grids shown in the left row are only used to illustrate the effect of the selected image space error threshold.

We have chosen a Λ of 3 (i.e. error directions are only computed for the three lowest LODs), since our experiments have shown that this is a suitable value for our simple mirror geometry. The tessellation of the image grid varies with a moving viewpoint and is not necessarily as symmetric as shown in Figs. 6 and 7.

Figs. 8 and 9 plot the reduction of vertex transformations and number of triangles to be rendered for the examples shown in Fig. 6.

Diamonds indicate measurements for the uniform image grid with a resolution that is appropriate to compensate for the required image space error. While squares represent measurements for the entire selectively refined image grid, triangles outline measurements for the spatially limited image portion.

Fig. 10 illustrates the computation durations that were required for the image grid transformation and the rendering of the transformed grid.

Table 1 presents the speedup factors for different image space errors while comparing the uniform grid method with the unconstrained selective refinement

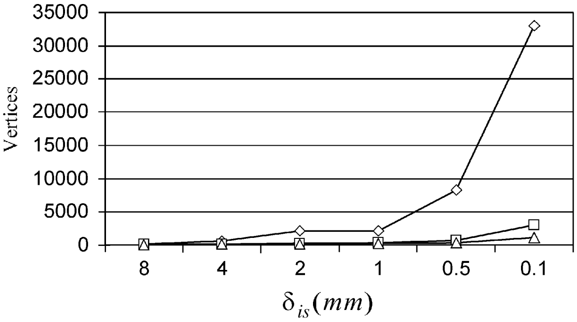

Fig. 8. Number of required vertex transformations (vertical axis) for a varying image space error $\bar{\delta}_{is}$ (horizontal axis). Diamonds indicate measurements for the uniform grid method, squares represent measurements for the entire selectively refined image grid, and triangles outline measurements for the generation of the spatially limited image portions.

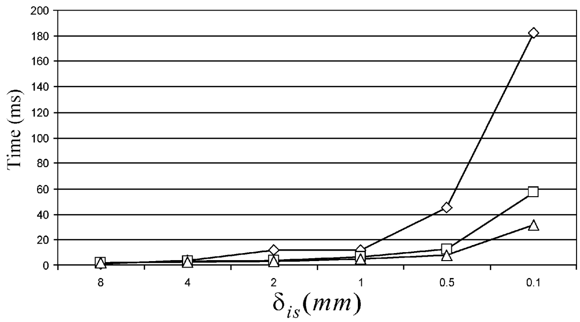

Fig. 10. Time required for vertex transformation and rendering (vertical axis) for a varying image space error $\bar{\delta}_{is}$ (horizontal axis). Diamonds indicate measurements for the uniform grid method, squares represent measurements for the entire selectively refined image grid, and triangles outline measurements for the generation of the spatially limited image portions.

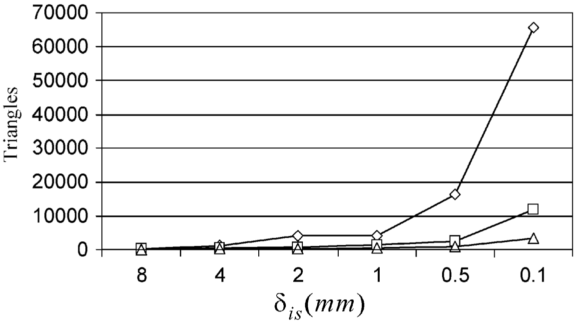

Fig. 9. Number of triangles that are required to be rendered (vertical axis) for a varying image space error $\bar{\delta}_{is}$ (horizontal axis). Diamonds indicate measurements for the uniform grid method, squares represent measurements for the entire selectively refined image grid, and triangles outline measurements for the generation of the spatially limited image portions.

Table 1
Speedup factor for different $\bar{\delta}_{is}$

$\bar{\delta}_{is}$	8 mm	4 mm	2 mm	1 mm	0.5 mm	0.1 mm
A	0.88	1.31	3.05	1.79	3.70	3.19
B	0.86	1.51	3.94	2.44	5.78	5.78

method (A) and with the spatially limited selective refinement method (B). One can observe that for large error thresholds (e.g. 8 mm in our examples) no speedup can be gained over the uniform grid method. For higher precision requirements, however, significant speedups are achieved. Note that the measurements include all required computations, such as recursions and minimizations.

Fig. 11 shows an example as it can be observed in the Virtual Showcase prototype that is illustrated in Fig. 1. Note that the photographs are not touched up. They are taken as seen from the observer's perspective. However, they have been rendered monoscopically. As in Fig. 7, the uniform (Fig. 11b) and the refined rectangular grid (Fig. 11c) are only shown to illustrate the effect of the selected image space error threshold. The image grid that approximates the bounding box's silhouette (Fig. 11d) is actually produced by our algorithm. It can be noticed in Fig. 11c that the algorithm generates higher LODs for those image regions, which are reflected by portions of the mirror that are less orthogonal to the observer. These regions generate a higher non-linearity within the warped image and would consequently produce a larger image space error. To prevent this, a higher geometric grid resolution is generated for these regions.

Note that the grid geometry shown in Figs. 11b–e appears to be slightly noisy. This results from small bumps and dents that are randomly scattered over the mirror surface. They were caused by the irregular surface of the Plexiglas foil, which was coated with a half-silvered mirror to build the optics. This distortion is too complex to be taken into account. Note also that the image in Figs. 11b and c is clipped by the boundaries of the mirror at its top and bottom. In the photographs, this clipping lets the image grid appear curved at these sides. However, it is rectangular.

9. Display specific components

While our algorithm is valid for other displays that require non-linear predistortion, its display specific components have been explained based on a particular

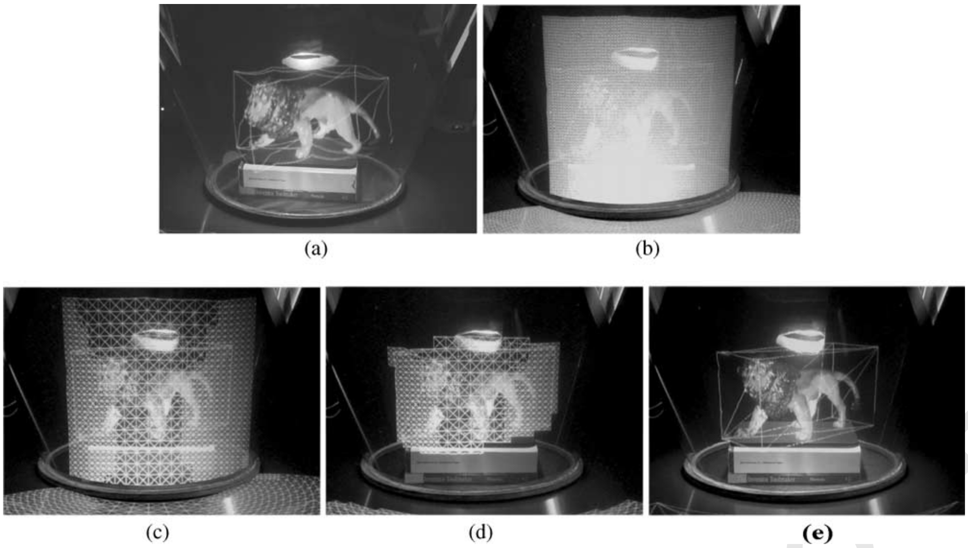

Fig. 11. Results observed in the Virtual Showcase: (a) distorted image created with an undersampled uniform grid, (b) oversampled uniform grid, (c) selectively refined grid for $\bar{\delta}_{is} = 0.25$ mm, (d) grid portion that projects onto container, and (d) final result.

mirror display, the cone-shaped Virtual Showcase. The nature of the additional mirror optics makes the transformation of the grid patches and the computation of the resulting displacement error fairly complex. In fact, the implementation of the *TransformPatchVertices* and the *RefineFurther* functions have been explained with an emphasis on cone-shaped mirror surfaces. These two functions represent the display specific components of our algorithm. While for a mirror display, *TransformPatchVertices* and *RefineFurther* have to consider laws of geometric optics, such as reflection and refraction transformations, to map grid vertices from the image plane onto the projection surface and vice versa, they can be generalized to do the same for other displays without modifying the general algorithm.

If, for instance, the algorithm is used to project images onto curved screens (e.g. a cylindrical or spherical projection device), *TransformPatchVertices* would incorporate only projection transformations (i.e. it would only determine intersections of vertex-individual geometric lines of sight with the display surface). The resulting displacement error that is computed by *RefineFurther* can then be determined within the object space (e.g. the screen space), rather than within an image space. Compared to our numerical approach for convex mirror displays, this would be less complex since it involves only simple intersection computations for that analytical solutions exist. If a view-dependence is not required, *TransformPatchVertices* and *RefineFurther* could also retrieve precomputed values from a look-up table.

10. Conclusion and future work

For displays such as the Virtual Showcase that correct non-linear distortion by applying multi-pass rendering, generating appropriate local levels of detail instead of applying uniform grids or constant image geometries allows to consider the error that is caused from a piecewise linear texture interpolation and to minimize it by adapting the underlying geometry.

For this purpose, we have presented an adaptive grid refinement algorithm for real-time view-dependent image warping. It can be applied to neutralize optical distortion or to display undistorted images onto non-planar surfaces. Although we have developed the algorithm mainly to enhance image quality and rendering performance of curved Virtual Showcases, it can be adapted to similar displays with only minor modifications. For instance, we have presented a method for object–image reflections (Section 7.3) that was optimized for particular second order mirror surfaces, such as cones or cylinders. For other mirror surfaces or projection screens, this method has to be replaced.

On the one hand, the algorithm prevents oversampling and texture artifacts that result from undersampling. On the other hand, it speeds up rendering for such displays significantly while guaranteeing a predefined maximal error on the image plane. Furthermore, the algorithm facilitates rendering on low-cost and off-the-shelf hardware, such as PCs and PC-based graphics acceleration boards.

Beside the displacement error, other criteria are evaluated and concatenated to define the final exit condition of the recursive refinement procedure. For .example, the area of the projected patches is computed to ensure that their size does not fall below the pixel size on the display surface.

The evaluation of upcoming graphics engines that provide enhanced user-programmable vertex operations is on our list of future work. We hope that they will allow for hardware acceleration of the required per-vertex transformations. The instruction sets of current architectures, such as the ones offered by nVidia, ATI or [14], however, are still too restricted and do not yet allow to implement our optical predistortion entirely.

Furthermore, a reactive progressive rendering [10], which dynamically adapts the screen space error threshold to reach and keep a desired frame rate, would represent a more intuitive user interface. Such a method could also consider the texture resolution to further accelerate our multi-pass rendering method. This speed-up, however, would be gained on the cost of the overall image quality.

Acknowledgements

This work was mainly funded by the Fraunhofer Center for Research in Computer Graphics. The Virtual Showcase project is supported by the European Union, IST-2001-28610.

References

[1] Bandyopadhyay D, Raskar R, Fuchs H. Dynamic shader lamps: painting on real objects. Proceedings of International Symposium on Augmented Reality (ISAR'01), 2001. p. 207–15.

[2] Van Belle R, Maximus B, Vandenbogaerde P, Clodfelter R. High quality geometry distortion tool for use with LCD and DLP projectors. Proceedings of IMAGE'00, 2000.

[3] Raskar R, Welch G, Fuchs H. Spatially augmented reality. In: Proceedings of First IEEE Workshop on Augmented Reality (IWAR'98), San Francisco, CA, 1998. p. 63–72.

[4] Bimber O, Fröhlich B, Schmalsteig D, Encarnação LM. The virtual showcase. IEEE Computer Graphics and Applications 2001;21(6):48–55.

[5] Watson B, Hodges L. Using texture maps to correct for optical distortion in head-mounted displays. Proceedings of IEEE VRAIS'95, 1995. p. 172–8.

[6] Yang R, Gotz D, Hensley J, Towles H, Brown MS. PixelFlex: a reconfigurable multi-projector display system. Proceedings of IEEE Visualization'01, 2001. p. 167–74.

[7] Segal M, Korobkin C, van Widenfelt R, Foran J, Haeberli P. Fast shaddows and lighting effects using texture mapping. Computer Graphics (Proceedings of SIGGRAPH'92), 1992. p. 249–52.

[8] McKay S, Mason S, Mair LS, Waddell P, Fraser M. Membrane mirror based display for viewing 2D and 3D images. Proceedings of SPIE, vol. 3634, 1999. p. 144–55.

[9] McKay S, Mason S, Mair LS, Waddell P, Fraser M. Stereoscopic display using a 1.2-M diameter stretchable membrane mirror. Proceedings of SPIE, vol. 3639, 1999. p. 122–31.

[10] Funkhouser T, Sequin C. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. Computer Graphics (Proceedings of SIGGRAPH'93), 1993. p. 247–54.

[11] Hoppe H. Progressive meshes. Computer Graphics (Proceedings of SIGGRAPH'96), 1996. p. 99–108.

[12] Hoppe H. View-dependent refinement of progressive meshes. Computer Graphics (Proceedings of SIGGRAPH'97), 1997. p. 189–97.

[13] Hoppe H. Smooth view-dependent level-of-detail control and its application to terrain rendering. Proceedings of IEEE Visualization'98, 1998. p. 35–42.

[14] Lindholm E, Kilgard MJ, Moreton H. A user-programmable vertex engine. Computer Graphics (Proceedings of SIGGRAPH'01), 2001. p. 149–58.

[15] Lindstrom P, Koller D, Ribarsky W, Hughes L, Faust N, Turner G. Realtime, continuous level of detail rendering for height fields. Computer Graphics (Proceedings of SIGGRAPH'96), 1996. p. 109–18.

[16] Taylor DC, Barret WA. An algorithm for continuous resolution polygonalizations of a discrete surface. Computer Graphics (Proceedings of SIGGRAPH'94), 1994. p. 33–42.

[17] Rolland JP, Hopkins T. A method of computational correction for optical distortion in head-mounted displays. Technical Report, No. TR93-045, UNC Chapel Hill, Department of Computer Science, 1993.

[18] Möller T, Trumbore B. Fast, minimum storage ray-triangle intersection. Journal of Graphics Tools 1997;2(1):21–8.

[19] Mitchell D, Hanrahan P. Illumination from curved reflectors. Computer Graphics (Proceedings of SIGGRAPH'92), 1992. p. 283–91.

[20] Ofek E, Rappoport A. Interactive reflections on curved objects. Computer Graphics (Proceedings of SIGGRAPH'98), 1998. p. 333-42.

[21] Brent RP. Algorithms for minimization without derivatives. Englewood Cliffs, NJ: Prentice-Hall; 1973.

Oliver Bimber is currently a scientist at the Bauhaus University Weimar, Germany. He received a Ph.D. in Engineering at the Technical University of Darmstadt, Germany under supervision of Prof. Dr. Encarnação (TU Darmtadt) and Prof. Dr. Fuchs (UNC at Chapel Hill). From 2001 to 2002 Bimber worked as a senior researcher at the Fraunhofer Center for Research in Computer Graphics in Providence, RI/USA, and from 1998 to 2001 he was a scientist at the Fraunhofer Institute for Computer Graphics in Rostock, Germany. In 1998 he received the degree of Dipl. Inform. (FH) in Scientific Computing from the University of Applied

Science Giessen and a B.Sc. degree in Commercial Computing from the Dundalk Institute of Technology. He initiated the Virtual Showcase project in Europe and the Augmented Paleontology project in the USA.

Bernd Fröhlich is a professor for Virtual Reality Systems with the media faculty at the Bauhaus University Weimar in Germany. His recent work focuses on input devices, interaction techniques, display systems, and support for tight collaboration in local and distributed virtual environments. He received his MS and Ph.D. in computer science in 1988 and 1992, respectively, from the Technical University of Braunschweig, Germany.

Dieter Schmalstieg is an assistant professor at the Interactive Media Systems Group at Vienna University of Technology, Austria, where he directs the Studierstube augmented reality research project. His research interests include virtual environments, augmented reality, 3D user interfaces, and real-time graphics. He holds a Ph.D. and Habilitation degree from Vienna University of Technology.

L. Miguel Encarnação is the head of the Human Media Technologies Department of the Fraunhofer Center for Research in Computer Graphics in Providence, Rhode Island, and is an adjunct professor in computer science at the University of Rhode Island. His research interests include next-generation user interfaces, mixed reality technologies, and advanced distributive learning environments. He received his Ph.D. in computer science in 1997 from the University of Tübingen, Germany.