Immersive Surface Interrogation

Manfred E. Brill, Robert J. Moorhead, Yanlin Guan
Engineering Research Center
Mississippi State University
Mississippi State, MS 39762
{brill, rjm, guanyl}@erc.msstate.edu

ABSTRACT
The geometrical quality of curves and surfaces is of central importance in the design and manufacturing process in many industries. There are well-established analysis techniques using curves on surface like reflection lines, highlight lines and isophotes. Such interrogation methods are state-of-the-art on desktop CAD systems. We describe the integration of these methods in a spatially-immersive virtual environment.

The computation and description of these interrogation lines is unified by using a contouring algorithm, either in parameter space for a NURBS geometry or in $3D$ space for polygonal nets. We introduce interrogation bands for light cylinders with a given radius instead of mathematically defined light lines. The interrogation bands can be rendered as geometry or by texture mapping techniques. With textures interactive visualization, even of view-dependent reflection bands, is possible. We describe an interactive system integrating these techniques in a spatially-immersive virtual environment.

Categories and Subject Descriptors
I.3.7 [**Computer Graphics**]: Three–Dimensional Graphics and Realism – color, shading, and texture; I.3.7 [**Computer Graphics**]: Three–Dimensional Graphics and Realism – Virtual Reality; I.3.8 [**Computer Graphics**]: Applications

Keywords
Surface interrogation, free–form surfaces, virtual reality, texture mapping

1. INTRODUCTION
One of the major tasks in the process of styling, especially in the automotive industry is the design of "visually pleasing" curves and surfaces, in a functional or aesthetic way. For example, the roof of a car should be a convex surface, without bumps or wiggles. These kind of features correspond to intrinsic geometric properties like continuity, curvature, or torsion. Using conventional rendering algorithms, these surface characteristics often are not visible, because of the smoothing in the interpolative shading algorithms.

In the automotive industry reflections are generated by fluorescent lightbulbs in long lines along a ceiling. The geometry and the aesthetic value of these reflections is used for the quality assesment of the car body. With modern graphics hardware and algorithms these reflections can be simulated on a computer. These surface interrogation algorithms [14] are used to visualize surface properties like convexity, geometric continuity, curvature analysis, and for the assesment of the aesthetic quality of a surface.

In a taxonomy of the interrogation methods, there are three different categories:

- visualizing scalar values describing quality measures using color maps or textures. The best example of these methods is a color map visualizing the gaussian or mean curvature on a given surface [10, 25]. In Figure 1 the mean curvature of a blending surface with tangent continuity is visualized by a scalar color map;

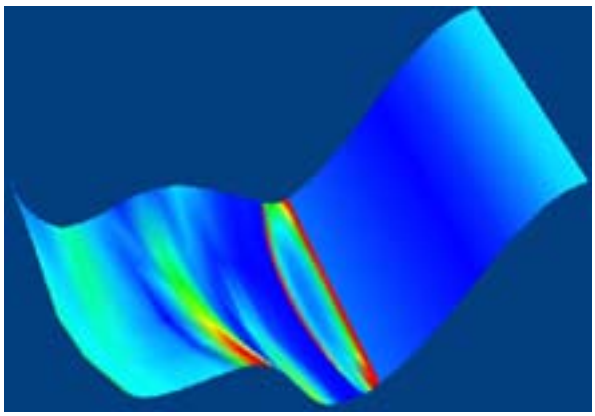

Figure 1: A color map visualizing mean curvature

- describing curvature properties of the surface by computing secondary geometry, like generalized focal surfaces [13]. In Figure 2 the discontinuity of the gener-

alized focal surface using the mean curvature as the offset function can be easily observed, indicating curvature discontinuity of the interrogated car body surface [12];

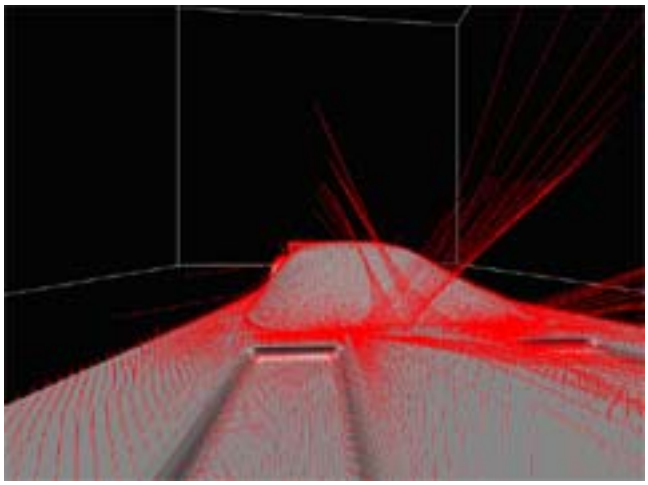

Figure 2: Continuity test using focal surfaces in a CAVE-like environment

- computing curves on surface, like the reflections generated by long parallel light lines on a real object. In Figure 3 the reflection lines generated by 50 parallel light lines on the body surface of a car model are visualized.

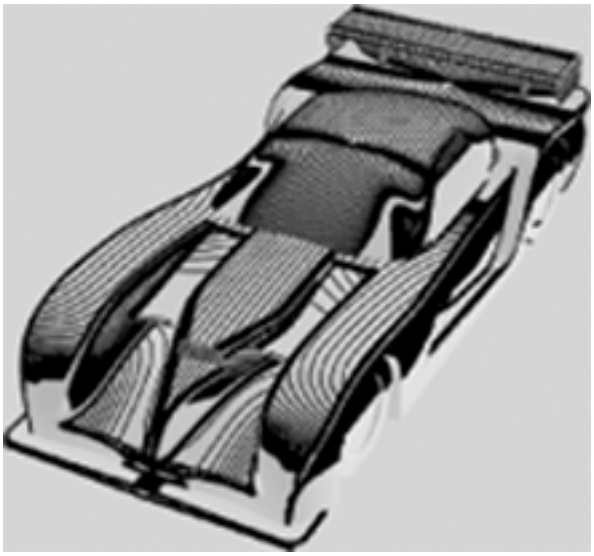

Figure 3: Reflection lines for 50 parallel light lines

1.1 Surface Interrogation using Curves on Surface

In [17, 18] reflection lines were first introduced as the projection of a light line on the surface, which can be seen from the current eye point, if the light line is reflected on the surface, like in Figure 4.

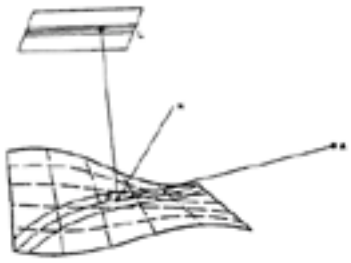

Figure 4: Reflection lines as reflections of parallel light lines observed from the current eye point

Isophotes [1, 23], the isolines of brightness on the surface are another surface interrogation method that computes curves on a surface. Like the reflection lines they are very sensitive to small shape deviations of the surface. In this way isophotes provide a fast and view-independent interrogation method.

The concept of highlight lines was introduced by [2, 3] as a view-independent variant of the reflection lines. Highlight lines can be described as reflection lines with the eye point at infinity. In Figure 5, 50 parallel light lines were used to compute highlight lines on the body of a car.

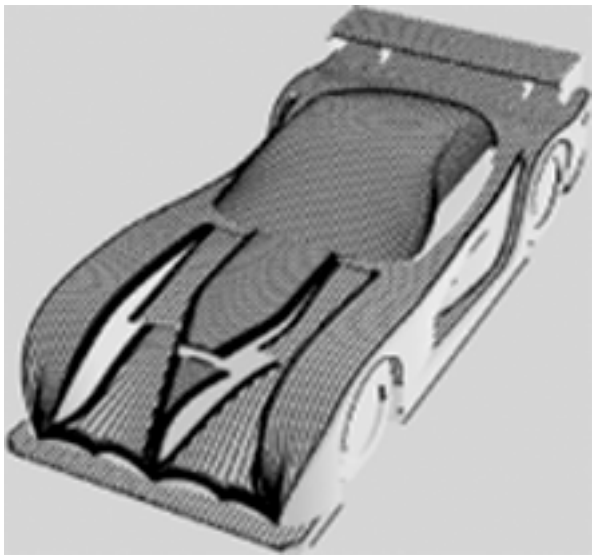

Figure 5: Highlight lines for 50 parallel light lines

All these lines are first-order methods. If the interrogated surface is C^k , the interrogation lines will be C^{k-1}. It is hard to visualize and detect the difference between C^2 and C^1 on a shaded surface. However, C^0 continuity for the

interrogation lines means we are looking for corners in the computed lines; something we can clearly observe. In Figure 6, 10 parallel light lines and the corresponding highlight lines are rendered on the same computed blending surface shown in Figure 1. The surface is tangent continous. As expected we easily see corners in the highlight lines in the center of Figure 6, where we expected C^1 behaviour of the interrogated surface.

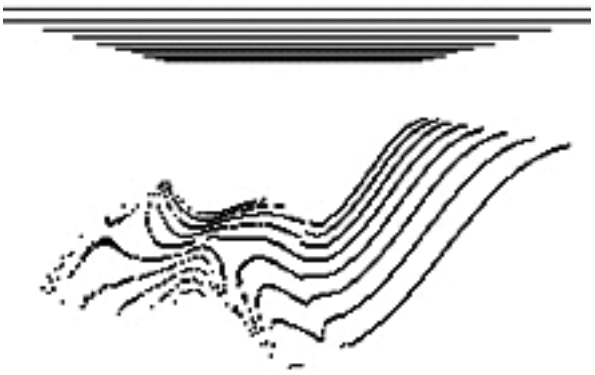

Figure 6: Highlight lines indicating a C^1 blend

1.2 Overview

In this paper we present a unified approach for the computation of the interrogation lines, using contouring. Our contouring approach is independent of the mathematical representation of the interrogated geometry. Light bulbs in reality are not mathematical defined lines but have a certain width. So all interrogation methods are described as interrogation bands, modeling light cylinders [3].

All interrogation lines were introduced for NURB surfaces. We simulate lighting properties of the interrogated geometry, so all we need are normal vectors. Thus we are able to implement a system which is independent of the mathematical representation of the interrogated surface. In styling, the interrogated objects are often represented by a set of trimmed free–form surfaces. These trimmed surfaces can be approximated by triangular meshes; see [19] for algorithms. If normals for the mesh vertices are given, all interrogation lines can be computed for these triangular meshes or polygonal nets.

Interrogation methods like the reflection lines and isophotes on a real car and with real lights are very popular, especially in the aesthetic assesment of car body surfaces. Although the methods are state–of–the–art in desktop CAD systems, the car stylists do not rely on simulated interrogation lines. Since they want to interact with the car model and the light, walking around the car and observing changes in the interrogation lines, we implemented an immersive visualization system using a CAVE–like environment. Such an environment improves the interpretability of the surface quality. The user is able to work like he is used to with the real object. The immersion and interaction with the light geometry and the interrogation lines on the surface increase the capability of decision making in design and manufacturing.

2. INTERROGATION BANDS

All interrogation lines use a *light line* L. We represent a line by a *light point* $\mathbf{L_0} \in \mathbb{R}^3$ and a *light direction* $\mathbf{s} \in \mathbb{R}^3$.

We always describe reflection properties of this line or an array of several lines, computing geometry on the surface S. Using the light line, the algorithms rely on intersections of lines in space. A fast criterion to decide if two lines $G_1(\mathbf{P_1}, \mathbf{d_1}), G_2(\mathbf{P_2}, \mathbf{d_2})$ in $\mathbb{R}^3$ intersect is the *perpendicular distance* of two lines. If the two lines are defined by the points $\mathbf{P_1}, \mathbf{P_2}$ and the directions $\mathbf{d_1}, \mathbf{d_2} \in \mathbb{R}^3$, they intersect if and only if their perpendicular distance is zero:

$$\frac{< \mathbf{P_1P_2}, \mathbf{d_1} \times \mathbf{d_2} >}{||\mathbf{d_1} \times \mathbf{d_2}||} = 0. \tag{1}$$

[3] were the first to compute highlight lines by solving a contouring problem, using the perpendicular distance. They compute scalar values for all vertices of a tesselation of the interrogated NURB surface. The perpendicular distance is computed as a function of the parameter values. The two lines plugged in the perpendicular distance are the light line L and the line given by the current vertex and the current lookup vector. Thus a contouring algorithm in the parameter space of the surface can be used to compute the isoline of zero distance.

This approach can be used to unify the computation of all interrogation lines. The difference is the direction of the line we plug in equation (1). For all vertices $\mathbf{P}$ on S with unit normal vector $\mathbf{n}$, we describe the interrogation lines as isolines of scalar functions defined by the perpendicular distance.

If $\mathbf{E}$ is the current eye-point, we can use the reflected vector

$$\mathbf{r} = 2 \langle \mathbf{n}, \mathbf{PE} \rangle \mathbf{n} - \mathbf{PE}$$

as the lookup-vector. With

$$f_R(\mathbf{P}, \mathbf{r}) = \frac{\langle \mathbf{r} \times \mathbf{s}, \mathbf{PL_0} \rangle}{\|\mathbf{r} \times \mathbf{s}\|} \tag{2}$$

the reflection lines are isolines with $f_R = 0$. Highlight lines are isolines $f_H = 0$ with

$$f_H(\mathbf{P}, \mathbf{n}) = \frac{\langle \mathbf{s} \times \mathbf{n}, \mathbf{PL_0} \rangle}{\|\mathbf{s} \times \mathbf{n}\|}. \tag{3}$$

Isophotes are the isolines $f_I = 0$ using

$$f_I(\mathbf{P}, \mathbf{n}) = \langle \mathbf{n}, \mathbf{s} \rangle . \tag{4}$$

If the surface S is given as a free–form surface, the vertices $\mathbf{P}$ are functions of the surface parameters. So contouring defines a polyline in the $2D$ parameter space and a curve on the surface. If we are only provided with a polygonal net with normals, contouring defines a polyline in 3D space.

Real lights on a ceiling are not mathematical lines but have a certain width. Using a value $R > 0$ for the radius of a *light cylinder* we introduce *reflection* and *isophote bands* like the highlight bands of Beier and Chen. A vertex $\mathbf{P}$ on the surface S with the lookup–vector $\mathbf{luv}$ is an *interrogation point* on an *interrogation band* with respect to the light cylinder given by the point L_0, the light direction $\mathbf{s}$, and light radius

R if and only if

$$f(\mathbf{P}, \mathbf{luv}) = |\frac{\langle \mathbf{luv} \times \mathbf{s}, \mathbf{PL_0} \rangle}{\|\mathbf{luv} \times \mathbf{s}\|}| \leq R. \quad (5)$$

For the visualization of the interrogation bands we compute the isolines representing the center of the interrogation band for $C = 0$ and the boundary with $C = R$. Any contouring algorithm [21] can be used to compute these isolines in parameter or $3D$ space.

For planar and parallel light bands a fast contouring algorithm can be developed. For simplicity of notation we assume that the bounding box of the interrogated object and all lookup vectors are normalized to $[0, 1]^3$. If the light bands are parallel to the x– or y–axis in a light plane parallel to $z = 1$, we have to compute the intersection point (x', y') of the lines given by the current vertex $\mathbf{P}$ and the direction $\mathbf{luv}$ with the light plane $z = z_{light}$:

$$x' = P_x + \frac{z_{light} - P_z}{luv_z} luv_x, \quad (6)$$

$$y' = P_y + \frac{z_{light} - P_z}{luv_z} luv_y. \quad (7)$$

For a light cylinder $x = \lambda$, a point P is an interrogation point, if

$$|x' - \lambda| \leq R;$$

for a light cylinder $y = \mu$ parallel to x, we get

$$|y' - \mu| \leq R.$$

Note that for reflection lines, the lookup–vector $\mathbf{luv} = \mathbf{r}$ has to be recomputed if the eye–point changes. If the light cylinders change their radius R or their position, the scalar values and the isolines have to be recomputed.

For the view–independent highlight lines, the scaled and projected vertex coordinates (x', y') can be processed in advance. The user can interrogate the surface in an interactive way by transforming the light cylinders, by translating or rotating them in the light plane. For the transformed light bands only the λ or μ values change.

3. TEXTURE MAPPING TECHNIQUES

With modern computer graphics hardware and texture mapping we can visualize intrinsic geometric and optical properties of an object [11]. Modern graphics APIs based on OpenGL provide built-in support for texture mapping. In general, texture mapping means projecting a $2D$ bitmap onto a $3D$ object. So, texture coordinates have to be assigned to the vertices of the textured object. An intermediate object [5] like a plane, a cylinder or a sphere is used to compute the texture coordinates. The bitmap is mapped on that intermediate object; then in a second step the texture coordinates for the rendered object are computed using a lookup vector and the intermediate object.

For planar light bands, a plane as the intermediate object is straightforward. We need a $2D$ bitmap representing the current light cylinders in the light plane. Figure 7 shows such a bitmap for a rake of 10 light cylinders, parallel to the $x-$coordinate axis. We model light attenuation in the cylinder. The light can be described as constant over the cylinder, or to drop from 1 in the center to 0 at the cylinder boundary in a linear or non–linear way. For Figure 7 a cubic attenuation was used. Light attenuation and prefiltering the bitmap prevents sampling artifacts in the rendered images.

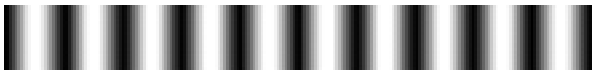

Figure 7: A bitmap representing 10 parallel light bands

Bier and Sloan [5] introduced several lookup–vectors for the computation of texture coordinates. These lookup–vectors correspond to the vectors we used to define the interrogation bands.

The "object normal" method for the texture coordinates uses the vertex normal as the lookup–vector and the light plane as the intermediate object. Obviously that results in the highlight bands, just as we described them. For parallel bands we generate a $1D$ gray–scale texture of parallel, planar light bands and compute the texture coordinates, like in Figure 7 where we rendered a $2D$ bitmap for illustration purposes. Equations (6) and (7) can be used like in the contouring algorithm. The luminance value lum for the current vertex is determined by the texture coordinates and is used as a blending factor. The fragment color C_f is determined by blending the object color C_o and the light color C_l:

$$C_f = lum \cdot C_o + (1 - lum) \cdot C_l.$$

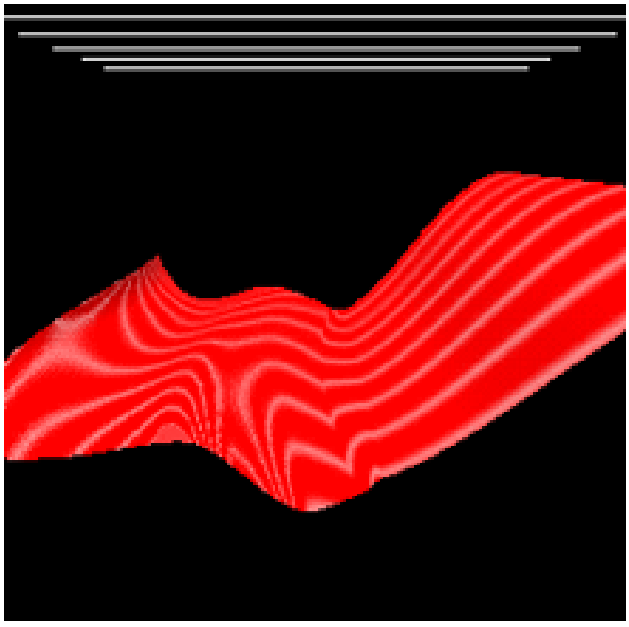

Figure 8: Texture mapped highlight bands

In Figure 8 the computed blending surface is rendered using the "object normal" and the $1D$ bitmap representing 10 parallel and planar light cylinders. Again the corners in the highlight bands, indicating the tangent continuity can be clearly observed. In Figure 9 highlight bands are ren-

dered on the body surface of a car. One centered planar light cylinder was used.

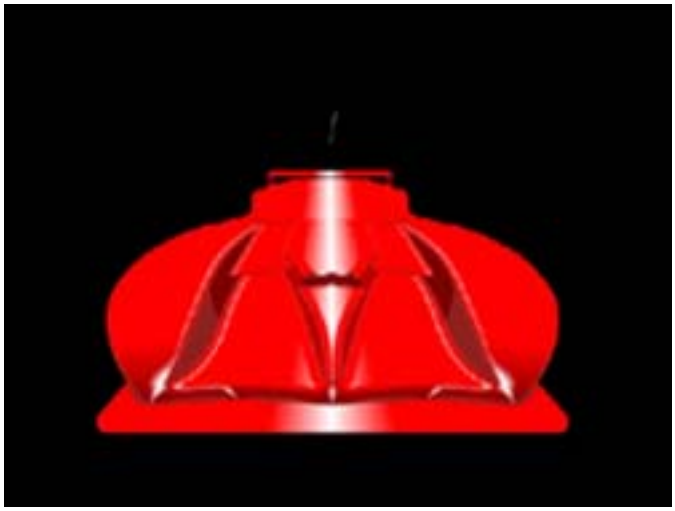

Figure 9: Texture mapped highlight bands on a car

For irregular distributions of the light bands a $2D$ bitmap has to be stored, and the perpendicular distance f_H can be used to compute the texture coordinates. The luminance value for the current vertex corresponding to the texture coordinates is again used as a blending factor.

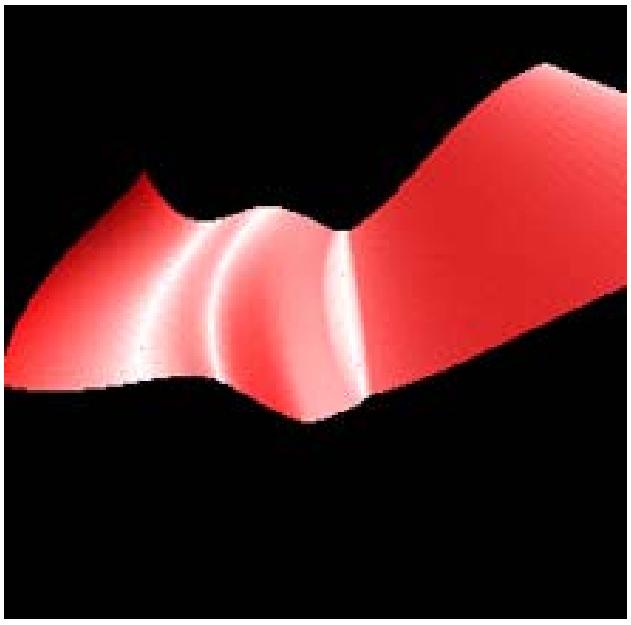

Figure 10: Texture mapped isophote bands

The isophote bands do not correspond to a special lookup-vector in two–pass texture mapping, but we can render isophote bands using texture mapping. First, we build a $1D$ texture map representing a ramp of luminance values increasing from 0 to 1. Then $|f_I|$ is used to compute the texture coordinates, and the fragment color is computed by a blend between object and light color. The texture is clamped, so values of $|f_I|$ outside the range $[0, 1]$ use a border luminance, which is defined as 0. Like the light band attenuation, the luminance ramp for the isophote band can be modeled in a linear or non–linear way. In Figure 10 we used a rake of parallel light cylinders to render isophote bands on the blending surface. In Figure 11 isophote bands are rendered on the car body. The planar regions of the body surface can be clearly observed.

Figure 11: Texture mapped isophote bands on a car

The "reflected ray" lookup–vector for two-pass texture mapping uses the reflection vector $\mathbf{r}$ to determine the texture coordinates. This corresponds to the scalar function f_R, so we finally get reflection bands. The reflected vector $\mathbf{r}$ is view–dependent, so everytime the eye–point changes, the texture coordinates have to be recomputed. For every change of view we have to recompute the texture coordinates.

Blinn and Newell [6] simulate reflection properties of an object using environment mapping – the so-called *sphere mapping*. An irradiance image equivalent to that which would be seen in a perfectly reflective hemisphere when viewed using an orthographic projection is used as a bitmap. Then, a sphere is used as the intermediate object to determine the texture coordinates. Modern 3D APIs, especially OpenGL, provide a texture coordinate generation mode to generate the texture coordinates of the rendered vertices of an object for such a sphere map.

The sphere map is computed in the viewing plane. Rays fired using the orthographic projection reflect in the center of the sphere, back to the viewer. There are several ways to generate a specular sphere map. Physical approaches, for example taking a photograph of a reflective sphere, can be employed. Six images of the environment, computed images or pictures taken in a real world scene, could be used. That cube is warped onto a unit circle within the square texture; see [7] for more details.

For given light cylinders in a plane the luminance values can be computed using the scalar function f_R. Then we warp the bitmap on a sphere. To avoid sampling artifacts a volume can be intersected with the light bands, prefilter-

ing the sphere map. If we assume a cone for that volume, the intersection with a light plane is defining a conic section. Figure 12 shows the luminance values on the sphere contained in the texture space $[0,1] \times [0,1]$, representing 10 parallel and planar light cylinders in the light plane $z = 1$. We used 128×128 pixels with a sampling rate of 16 bit for that image.

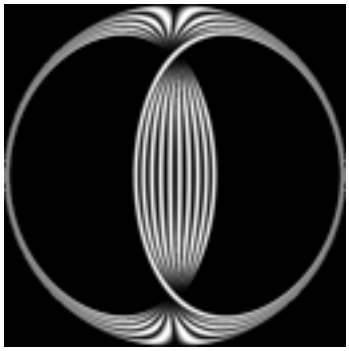

Figure 12: A sphere map for 10 parallel light cylinders

Using the texture mapping API of OpenGL we store the sphere map representing the light cylinders. The texture coordinates for the current eye point are computed by OpenGL. In this way we are able to achieve high frame–rates even for the view–dependent reflection bands. In Figure 13 we see reflection bands, rendered in OpenGL using the sphere map shown in Figure 12.

Figure 13: Sphere–mapped reflection bands

Figure 14 uses the same light cylinders that were used in Figure 9 and a OpenGL sphere map to render reflection bands on a car.

4. IMMERSIVE INTERROGATION

Interrogation methods are well-known to users of CAD–systems, but running the interrogation methods on a desktop system is like looking through a window into the real world. The stylists and designers in the automotive industry are used to walking around the interrogated object freely. They interact with the light and the real object, being able to look closely at details, moving quickly to another part of the object, or stepping back to gain an overall impression. There is no need to position an object on a $2D$ display with a pointing device. Thus there is a big difference between the simulation running on a desktop and the real process. For that reason, many users of interrogation methods in the automotive industry do not accept desktop driven interrogation software.

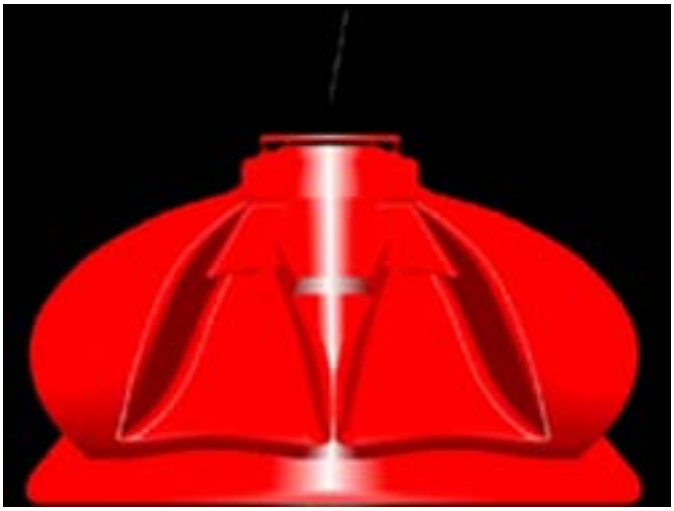

Figure 14: Sphere–mapped reflection bands on a car

Therefore we decided to build a immersive visualization environment for interactive surface interrogation. With precise tracking of the movement of the eyes and body, the observers can have better insight into the characteristics of the interrogated geometry than by running a program on a desktop system. An immersive visualization environment [8, 9, 26] gives the observers much more flexibility in interacting with the lights and the interrogated object, like they are used to in the real process. This technology gives the user the psychological experience of being surrounded by a virtual environment, which is very close to the environment they are used to in the real world.

In our CAVE–like virtual environment, the user assessing the quality of a geometry can walk around freely, analyzing the geometry from all viewpoints. In particular, the reflection bands visualized using hardware accelerated sphere mapping provide a high degree of immersion for a tracked user. Walking around the interrogated object the reflection bands on the surface change in realtime.

Using a wand, the users can change the lighting situation by transforming the parallel light cylinders. With the texture mapping techniques we implemented, high frame-rates can be achieved. The interrogation bands change interac-

tively, while the user is transforming the light. This gives the users a feeling of "being there." Whereas it takes on the order of 20 seconds to rotate an object on the desktop, in an immersive virtual environment, it takes around 2 seconds [27]. The user's hand and the light bands are collocated in a virtual space, so it is very intuitive to move the lights around. The user can change the radius and attenuation of the light definition. It is possible to switch between different styles of interrogation bands and rendering modes interactively. These features enhance the possibilities beyond reality, wherein the lights are fixed and changes in the setup are time consuming.

In the immersive interrogation environment the users can combine all the different categories of interrogation methods, like scalar maps visualizing the curvature, generalized focal surfaces [12], or the interrogation bands introduced in this paper. All these features together with the situational awareness of "being there" results in a new level in computer–based surface interrogation.

Figure 15: Step 1: Parallel light lines, an object and interrogation lines indicating the need for fairing

One major application of computer-simulated interrogation lines is the inverse modeling of geometry, based on prescribed reflection lines, highlight lines, or isophotes. [17] uses a partial differential equation to fit a free–form surface to a set of given reflection lines. [1] also proposes a PDE to fit a family of isophotes. [4, 28] propose a similar approach for the highlight lines. [20] presents an unified approach for the inverse modeling based on interrogation lines using a variational algorithm.

All these algorithms can be integrated in an immersive environment. There, an intuitive and haptic interface is possible. We start with a given object and a given set of light cylinders or light lines, like is illustrated in Figure 15. There one of the interrogation lines identifies a region on the surface which needs fairing.

The users in our immersive interrogation environment modify the light definition by transforming or bending the light lines. On the surface, the texture or isolines corresponding to the original surface and the changed lights are rendered. Figure 16 illustrates this step.

Figure 16: Step 2: Deformed light lines, the original object and fair interrogation lines

Figure 17: Step 3: The original light lines, the remodeled object and fair interrogation lines

If the interrogation lines are faired, the object is remodeled to fit the displayed interrogation lines in conjunction with the original light definition. We end up in a situation like Figure 17. This kind of interactive "modeling with light" is almost impossible to build for a desktop system, lacking the situational awareness and the haptic interfaces in a CAVE-like environment.

We use the CAVElibs [22] as the API for our CAVE-like environment. The rendering is implemented in SGI OpenGL Performer. Performer provides an object-oriented real-time API for scene graphs and supports the import of a wide range of $3D$ objects. In this way CAD models can easily be imported into the immersive system. NURB surfaces or tesselated polygonal nets with vertex normals can be used for the interrogation. We use the data structures and filters in VTK [24] to implement the contouring approach. On top of VTK we designed and implemented an object-oriented toolkit in C++ for the light cylinders and the different interrogation bands. This toolkit is designed independently of a rendering API, so an integration in a desktop system or in alternative software for immersive visualization systems like VRJuggler is easy.

5. CONCLUSION

Using a virtual visualization environment we are able to provide an immersive surface interrogation system. We believe that the immersion leads to improved interpretation of surface quality, increasing the capability in decision making in the design and manufacturing process. All categories of surface interrogation methods, scalar maps, secondary geometry, and interrogation bands are integrated in one object-oriented software toolkit.

The computation and rendering of interrogation bands is unified using a contouring approach. Texture mapping can be used to achieve high frame-rates for the user interface. The interrogation results can be visualized and the users can interact with the object and light like working with a real object and real lights. This improves the acceptance of computer-simulated surface interrogation. In an immersive environment we implement a very intuitive user interface for inverse modeling, fitting a surface to prescribed interrogation lines.

6. FUTURE WORK

We intend to extend the existing immersive environment to be a very intuitive modeling environment with haptic feedback and a user interface based on cognitive task analysis.

We will implement the view-independent environment mapping method proposed by [15, 16], using a paraboloid instead of a sphere as the intermediate object. This will result in even higher frame rates for the visualization of reflection bands. Also, we will implement different intermediate objects like a cylinder or a sphere, distributing the light cylinders on that geometry.

We also plan to link our immersive interrogation system with CAD systems, so the users can exchange their data between our system and their CAD system in a very convenient way. The users will be able to see their original CAD data. and the original modeling capabilities of their CAD system can be used for the tesselation and inverse modeling.

7. REFERENCES

[1] R. Andersson. Surface design based on brightness intensity or isophotes – theory and practice. In J. Hoschek and P. Kaklis, editors, *Advanced Course on FAIRSHAPE*, pages 131–143, 1996.

[2] K.-P. Beier and Y. Chen. The highlight–line: algorithms for real-time surface–quality assesment. *CAD*, 26:268–278, 1994.

[3] K.-P. Beier and Y. Chen. The highlight band, a simplified reflection model for iterative smoothness evaulation. In N. Spapidis, editor, *Designing Fair Curves and Surfaces*, 1994.

[4] K.-P. Beier, Y. Chen, and D. Papageorgiou. Direct highlight line modification on nurb surfaces. *CAD*, 14:583–601, 1998.

[5] E. A. Bier and K. R. Sloan. Tow–part texture mapping. *IEEE Computer Graphics and Applications*, 6(9):40–53, 1986.

[6] J. Blinn and M. E. Newell. Texture and reflection in computer generated images. *Communication of the ACM*, 19:362–327, 1976.

[7] D. Blythe. Advanced graphics programming techniques using OpenGL. In *SIGGRAPH Course Notes*. ACM SIGGRAPH, 1999.

[8] C. Cruz-Neira, D. Sandin, and T. DeFanti. Virtual reality: The design and implementation of the CAVE. In *SIGGRAPH Conference Proceedings*, pages 135–142, 1993.

[9] C. Cruz-Neira, D. Sandin, T. DeFanti, R. Kenyon, and J. Hart. The CAVE: Audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):65–72, 1992.

[10] J. C. Dill. An application of color graphics to the display of surface curvature. In *SIGGRAPH Conference Proceedings*, pages 153–161, 1981.

[11] P. Haeberli and M. Segal. Texture mapping as a fundamental drawing primitive. In *Eurographics Workshop on Rendering*, pages 259–266, 1993.

[12] H. Hagen, Y. Guan, and R. J. Moorhead. Interactive surface interrogation using IPT technology. In *Virtual Reality 2002 – Immersive Projection Technology Symposium*, 2002.

[13] H. Hagen and S. Hahmann. Generalized focal surfaces: a new method for surface interrogation. In *IEEE Visualization 92 Proceedings*, pages 70–76, 1992.

[14] H. Hagen, S. Hahmann, T. Schreiber, Y. Nakajima, B. Woerdenweber, and P. Hillemann-Grundstedt. Surface interrogation algorithms. *IEEE Computer Graphics and Applications*, 12(5):53–60, 1992.

[15] W. Heidrich and H.-P. Seidel. View-independent environment maps. In *Eurographics/SIGGRAPH Workshop on Graphics Hardware*, pages 39–45, 1998.

[16] W. Heidrich and H.-P. Seidel. Realistic, hardware–accelerated shading and lighting. In *SIGGRAPH Conference Proceedings*, pages 171–178, 1999.

[17] E. Kaufmann and R. Klass. Smoothing surfaces using reflection lines for families of splines. *CAD*, 20:312–316, 1988.

[18] R. Klass. Correction of local surface irregularities using reflection lines. *CAD*, 12:73–76, 1980.

[19] K. Lee and I. Choi. Evaluation of surfaces for automobile body styling. In *Computer Graphics International 96*, pages 202–211, 1996.

[20] J. Loos, G. Greiner, and H.-P. Seidel. Modeling of surfaces with fair reflection line pattern. In *Shape Modeling and Applications 99*, pages 256–263, 1999.

[21] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[22] D. Pape. A hardware–independent virtual reality development system. *IEEE Computer Graphics and Applications*, 16(4):4–47, 1996.

[23] T. Poeschl. Detecting surface irregularities using isophotes. *CAGD*, 1:163–168, 1984.

[24] W. Schroeder, K. Martin, and W. E. Lorensen. *The Visualization Toolkit: An Object–Oriented Approach to 3–D Graphics*. Prentice Hall, 1997.

[25] L. R. Seidenberg, R. B. Jerad, and J. Magewick. Surface curvature analysis using color. In *IEEE Visualization 92 Proceedings*, pages 260–267, 1992.

[26] A. van Dam, A. Forsberg, D. Laidlaw, J. LaViola, and R. Simpson. Immersive VR for scientific visualization: A progress report. *IEEE Computer Graphics and Applications*, 20(6):26–52, 2000.

[27] C. Ware and J. Rose. Rotating virtual objects with real handles. *ACM Transactions CHI*, 6(2):162–180, 1999.

[28] C. Zhang and F. Cheng. Removing local irregularities of nurb surfaces by modifying highlight lines. *CAD*, 30:923–930, 1998.

Note to reviewers: The images herein have been reduced in resolution for the sake of file size for the review process; the images for the final version will be much higher resolution; for example, Figure 1 is only 220x152 now, but will be 800x600 for final version.