Pacific Graphics 2008
T. Igarashi, N. Max, and F. Sillion
(Guest Editors)

Volume 27 (2008), Number 7

A Resolution Independent Approach for the Accurate Rendering of Grooved Surfaces

C. Bosch,[1] X. Pueyo,[1†] S. Mérillou[2] and D. Ghazanfarpour[2‡]

[1] Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain
[2] Institut de Recherche XLIM, Université de Limoges, France

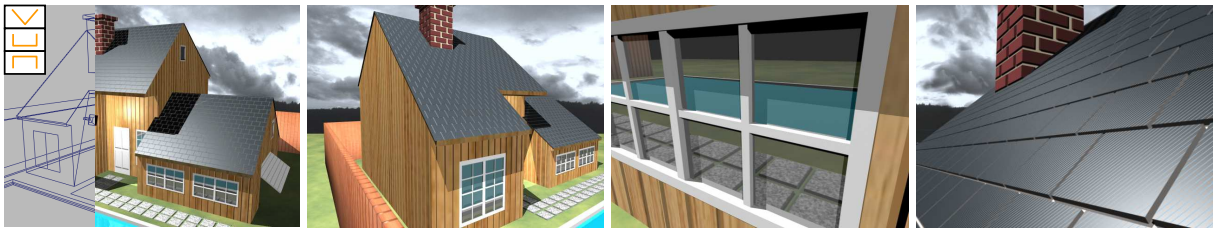

Figure 1: *Scene with several grooved surfaces simulated using our approach. Upper left: Cross-sections of the different grooves.*

Abstract

This paper presents a method for the accurate rendering of path-based surface details such as grooves, scratches and similar features. The method is based on a continuous representation of the features in texture space, and the rendering is performed by means of two approaches: one for isolated or non-intersecting grooves and another for special situations like intersections or ends. The proposed solutions perform correct antialiasing and take into account visibility and inter-reflections with little computational effort and memory requirements. Compared to anisotropic BRDFs and scratch models, we have no limitations on the distribution of grooves over the surface or their geometry, thus allowing more general patterns. Compared to displacement mapping techniques, we can efficiently simulate features of all sizes without requiring additional geometry or multiple representations.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism[color, shading, shadowing, and texture]

1. Introduction

Grooves and similar features like scratches and ridges, are common surface details that are found in many real world objects. They are especially present, for instance, on engraved objects, polished metals or assembled/tiled surfaces. Their main characteristic is given by the shape, which can be represented by a cross-section and a path over the surface.

In Computer Graphics, different techniques have been proposed for the simulation of grooves. Anisotropic BRDFs, for instance, allow the simulation of very small grooves that provide an anisotropic aspect to the surface [Kaj85, War92]. However, accurate models are limited to certain geometries and distributions [PF90, Sta99]. Scratch models deal with small individually visible grooves, but only consider isolated grooves and without special geometric situations like intersections or ends [BB90, MDG01]. Macro-geometric techniques such as bump mapping [Bli78] or displacement mapping [Coo84] are general methods that rarely pose restrictions on their geometry or distribution. Nevertheless, small or distant detail usually require good filtering techniques that can be very time consuming, especially for highly detailed surfaces. Some approaches have also proposed to perform smooth transitions between BRDF models and bump or displacement mapping [Fou92, BM93], but then masking and shadowing effects are rarely considered.

In this paper, we propose a method that overcomes these

† {carles.bosch|xavier.pueyo}@ima.udg.edu
‡ {merillou|ghazanfarpour}@unilim.fr

limitations by means of a resolution independent approach. Its purpose is to render grooves from any distance or size without using multiple representations or exhaustive sampling approaches, allowing accurate transitions between different geometric scales and an efficient rendering in terms of memory and time. For this, features are modeled using a representation based on paths and cross-sections. The local geometry of the grooves is then evaluated for every pixel and rendered taking into account masking and shadowing effects along with correct antialiasing. This results in a method that accurately simulates grooves and other path-based features with almost no restrictions on their geometry, size or distribution over the surface.

2. Previous work

2.1. Anisotropic BRDFs

Anisotropic reflection models are intended to simulate surfaces containing lots of micro-grooves. Most of these models are based on empirical approaches, which are used when the micro-geometry is not known [War92, Ban94]. Physically-based models are also available, but only for certain types of micro-geometry, such as parallel cylinders [PF90] or random Gaussian surfaces [Sta99]. For general detail, brute force methods precompute the reflection from a subset of directions, but this is mostly suitable for periodic patterns [WAT92]. Ashikhmin et al. [APS00] use arbitrary normal distributions to generate reflection models, but with a simple occlusion term. In our case, we are not restricted to a specific shape or distribution of grooves and we compute occlusion effects in an accurate way.

2.2. Scratches

Scratch models simulate isolated scratches first using a texture to specify their path over the surface. Such texture is either represented by means of a 2D image [BB90, MDG01] or a set of curves in texture space, which gives more accurate results [BPMG04]. The reflection at each scratch point is then specified by means of a BRDF, which is based on an empirical anisotropic model [KS00] or computed according to a derived scratch cross-section [BPMG04]. Scratch methods are restricted to isolated cases, where the reflection is given by a single entire cross-section. In our case, we model grooves using a similar representation. However, our rendering method can handle partially contained cross-sections (bigger or nearest features) and multiple cross-sections per pixel, including intersections and other situations. Furthermore, we take into account indirect illumination as well.

2.3. Macro-geometric techniques

Macro-geometric techniques, commonly known as displacement mapping techniques, allow the simulation of different kinds of surface details. Some of them are based on generating additional geometry [Coo84], while others simulate detail by modifying surface normals [Bli78] or performing ray tracing on height maps [POC05]. See [SKU08] for an in-depth survey of these methods for the GPU. These techniques are mostly suited for big surface features or low-frequency detail. Otherwise, they require high resolution maps and good antialiasing methods that can be very time consuming, especially as the distance to the viewer increases. Other approaches have modeled grooves by modifying the object geometry for interactive sculpting purposes [MOiT98], but these suffer from similar problems.

2.4. Multi-resolution methods

In order to efficiently simulate detail from any distance, a common solution is to perform smooth transitions between different representations. Becker and Max [BM93], for instance, address the transition among displacement mapping, bump mapping and BRDF. However, shadowing is neglected and transitions must be approximated due to inconsistencies between the representations. Other authors use multiple resolutions based on normal distributions or roughness maps [Fou92, Sch97, CL06]. Such methods perform a kind of efficient mip mapping of normal maps by storing distributions of normals, but since only normals are considered, occlusion effects can not be directly taken into account. Some displacement mapping techniques simply use mip mapping to handle distant detail [POC05], but directly pre-filtering heights or normals can not yield correct results [Fou92].

3. Representation overview

In our approach, we model details using a representation based on paths and cross-sections in texture space, similar to the one used in [BPMG04]. Such a representation is compact and can be easily applied to any surface having a texture parametrization.

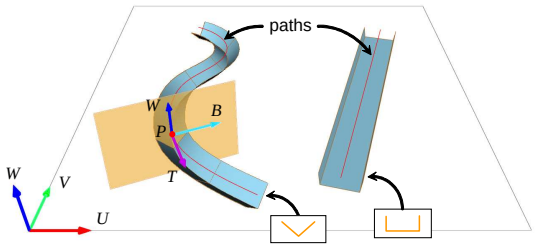

Figure 2: *Grooves are represented in texture space by means of paths and cross-sections.*

First, each feature is described by a path lying on the *UV* texture plane and a cross-section being perpendicular to it, following the path's tangent frame (see Figure 2). For a given point P on the path, such a frame is described by its tangent T, binormal B and texture vector W. Cross-sections are thus represented on the BW plane. Paths can be modeled as

2D curves or straight line segments, while cross-sections are modeled as 2D polylines. For cross-sections, polylines are preferred to curves because they allow faster computations of occlusion effects, thus curved profiles will be approximated by means of polylines. Such cross-sections can penetrate the surface, protrude from it or both.

The user can also assign specific material properties and a perturbation function to each feature. Perturbations are used to modify the cross-section shape or scale of the grooves along the path, so that non-regular shapes can be simulated.

4. Rendering grooves

This section introduces our method for rendering grooved surfaces, which is the main contribution of the paper. The method works like a common shader that is executed for every surface point that is visible and illuminated, and our purpose is to deal with the direct illumination at the local grooves. In Section 7, we then explain how to extend this in order to include inter-reflections and transmissions.

Given the current pixel projected onto the texture, commonly known as the pixel footprint, the algorithm consists of the following steps: first, we determine which grooves affect the footprint (Section 4.1), we detect if they are isolated or form a special situation (Section 4.2), and finally compute their visibility and shading according to the case (Sections 5 and 6). Since dealing exactly with all possible geometries can be very expensive, we here assume that the local geometry inside a pixel can be approximated by a set of flat facets in order to simplify our computations. This approximation will not introduce significant errors as long as the cross-section perturbation does not vary significantly in the footprint and the curvature of the paths and the surface is locally smooth. Concerning the footprint shape, we consider that it is defined by an oriented ellipse lying on the *UV* texture plane (see Figure 3), but other shapes could be considered as well.

4.1. Finding grooves

First of all, we need to determine which grooves affect the current pixel, that is, grooves contained in the footprint (visible grooves) and grooves casting shadows on it. For this, we simply evaluate the footprint against the different paths. In a previous stage, the *UV* plane is subdivided into a uniform grid, saving in each cell a list of all the paths crossing it. Then, the footprint's bounding box is computed onto this grid and the paths are retrieved from the cells covered by its boundary (see the solid cells in Figure 3). Since paths are rarely shorter than a pixel footprint, interior cells can be omitted without missing any path.

Only considering the bounding box of the footprint is not sufficient to find all necessary grooves, since bigger or nearer grooves may be partially contained without containing their paths. To solve it, we must also consider the dimensions of

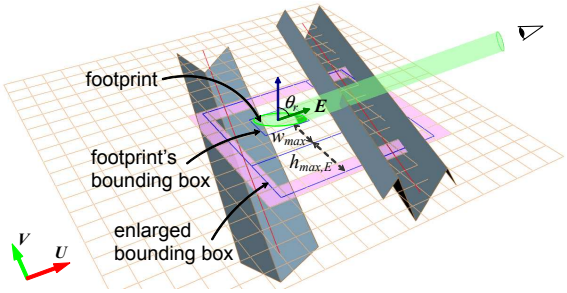

Figure 3: *Grooves affecting the pixel footprint are found by retrieving the paths from a uniform grid.*

the cross-sections and enlarge the bounding box accordingly before retrieving the cells. These dimensions may be precomputed as the maximum width (half-width), height and depth of all the cross-sections (w_{max}, h_{max} and d_{max}), taking into account any associated perturbation.

This procedure is depicted in Figure 3, where a footprint is affected by two grooves but their paths remain outside the bounding box. After enlarging this according to w_{max}, we can find the paths for all partially contained grooves (left groove in the figure). For grooves seen far from their bounds (right groove), we must consider h_{max} as well, specifically its projection according to the viewing angle θ_r, which is computed as $h_{max,E} = h_{max} \tan\theta_r$. Although not shown in the figure, the projection of d_{max} is also considered in order to find grooves that may be partially visible through an intersection or a similar case. The projection of h_{max} and d_{max} is then repeated using the light source direction, so that grooves casting shadows inside the footprint are also found.

For a certain pattern, the grid resolution is usually selected so that the cell size is similar to the average width of the grooves. The footprint average size could also be considered, i.e. according to the view distance or image resolution, but the former already gives good results in terms of efficiency.

4.2. Detection of special cases

The presence of an special case is detected by checking if any intersection between the paths or any of their endpoints remains inside the bounding box. In that case, their reflection is computed using the method of Section 6; otherwise, we use the approach from the following section. In case where no groove was found, no further processing is required and the common surface reflection is simply computed.

5. Isolated or non-intersecting grooves

In order to compute the reflection on the footprint, the actual contribution of the current grooves and the surface must be first evaluated by means of a set of clipping and occlusion operations. When no special case is present, the local geometry can be approximated using a 2D cross-section (see

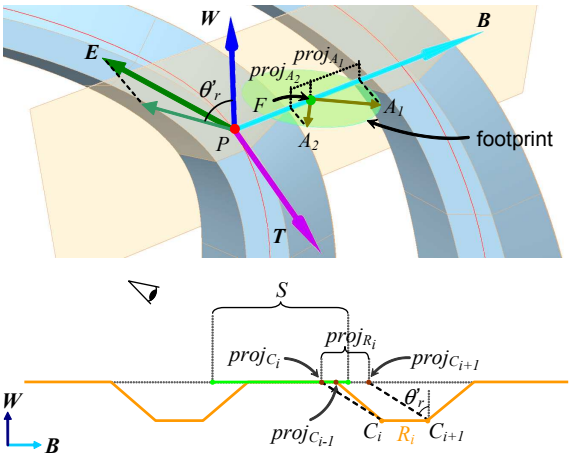

Figure 4: *For non-intersecting grooves, footprint is projected onto the BW plane (top) and visibility is computed in cross-section space (bottom).*

Figure 4). This greatly simplifies the computations by removing one dimension to the problem, and can be done due to the assumptions stated at Section 4. For such cases, an algorithm based on a fast line sampling approach is proposed. This approach consists in evaluating the visibility of the different facets against a 1D line segment, which is obtained after projecting the footprint onto the cross-section plane.

5.1. Footprint clipping

Clipping is used to remove portions of facets remaining outside the footprint. With this purpose, we first project the pixel footprint onto the cross-section plane, as stated above. Since the footprint is represented by an oriented ellipse (see Section 4) and its projection would be expensive, this is approximated using its two main axes A_1 and A_2 (see top of Figure 4). The axis with the largest projection is the one that better represents the original shape, and is computed as $proj_{max} = \max(|A_1 \cdot B|, |A_2 \cdot B|)$. The resulting footprint segment is then $S = [-proj_{max},\ proj_{max}]$, which is defined with respect to the footprint center F.

To clip the footprint segment against the different facets, each cross-section is also projected onto the binormal line where S lies (see bottom of Figure 4). The projection is performed according to the viewing angle θ'_r, which is defined by the vector E once projected onto the cross-section plane. Clipping is finally computed by means of simple 1D segment intersections between the facets and the footprint.

5.2. Occlusion

Occlusion is computed to determine the visible and illuminated parts of each facet. Using a similar approach than before, the idea is to first project the cross-sections onto the surface according to the view/light direction. The occluded parts can then be found by taking into account how the different points are successively projected onto it. According to the direction from which we must compute occlusion, two cases are considered. When the reference vector remains on the left side of a cross-section, i.e. $E \cdot B < 0$ for the viewing case, the cross-section points are sequentially projected from left to right (see bottom of Figure 4). Given a facet R_i defined by two points C_i and C_{i+1}, its projection onto the base line is represented by $proj_{R_i} = [proj_{C_i}, proj_{C_{i+1}}]$. Occlusion is then evaluated with the following expression:

$$proj_{R_i} = \begin{cases} \text{null} & \text{if } proj_{C_{i+1}} \leq proj_{C_j},\ j \leq i \\ [proj_{C_j}, proj_{C_{i+1}}] & \text{if } proj_{C_i} < proj_{C_j} < proj_{C_{i+1}},\ j < i \\ [proj_{C_i}, proj_{C_{i+1}}] & \text{otherwise} \end{cases}$$

This expression compares the order in which the projected points lie on the surface. The facet is completely occluded (first case) if its second point $proj_{C_{i+1}}$ lies before any previously projected point $proj_{C_j}$. The facet is partially occluded (second case), when $proj_{C_j}$ instead lies between the two facet points. In that case, the visible portion is defined by such point and the facet's second point. Finally, if none of the previous cases applies, the facet is completely visible and remains as is (third case). In Figure 4, for example, facet R_i is partially occluded and results in $[proj_{C_{i-1}}, proj_{C_{i+1}}]$, while previous facet is completely occluded ($proj_{C_i} < proj_{C_{i-1}}$).

When the current vector remains on the right side of the cross-section, the process is simply inverted. The cross-section points are projected from right to left and the occlusion expression is changed accordingly. Note that both masking and shadowing are computed in the same way. Masking, which is produced from the view direction, is computed during the clipping operation, so that clipping is done with only the visible parts of facets. Shadowing is computed after that, and partially shadowed facets are intersected with their corresponding clipped portions.

5.3. Reflection contribution

The total reflection on the footprint is finally computed as the sum of reflections of each obtained facet:

$$f_r = \sum_i f_{r,i} r_i\,, \qquad (1)$$

where $f_{r,i}$ is the BRDF associated to the facet and r_i its relative area. This area is simply computed as $|\ proj_{R_i}\ | / |\ S\ |$, which represents the ratio between the length of the current facet after projection, clipping and occlusion operations and the length of the original footprint segment.

6. Special cases

At points where the footprint contains groove intersections or ends, we have to deal with the different special cases

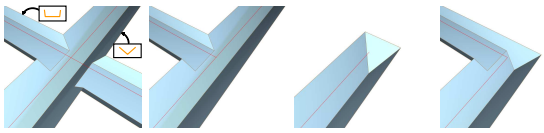

Figure 5: *From left to right: Intersection, intersected end, isolated end and corner.*

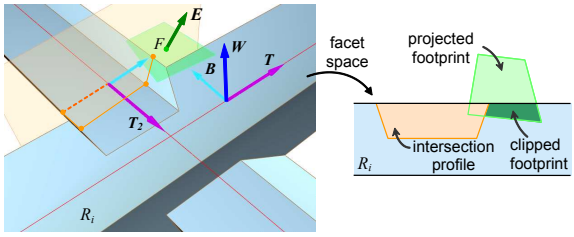

Figure 6: *Footprint is projected onto a facet R_i and clipped to its bounds. Then, the intersection profiles are subtracted.*

shown in Figure 5. For this kind of situations, the local geometry is significantly more complex than in the previous case, as it can be neither approximated with a single cross-section nor sampled using a simple segment. We rather need to consider the 3D geometry of the grooves as well as the entire footprint shape, i.e. by means of an area sampling approach. In this case, the footprint shape will be represented by means of a polygon, while the algorithm perform the operations in facet space rather than in cross-section space. Its different steps are described next for the case of common intersections, while the other cases are treated in Section 6.5.

6.1. Footprint clipping

For clipping, the polygonal footprint is first projected onto the current facet using common line-plane intersections. This is done in 3D groove space, which is described by the path's tangent frame TBW at the current point. The projection direction is then given by the view vector E (see Figure 6). Once projected onto a facet's plane, the footprint is transformed into 2D space by simply dropping the most representative coordinate of its projected points and the plane. Clipping is then easily performed with an axis-aligned facet bounded by two horizontal lines (see right part of Figure 6).

6.2. Intersection removal

This step removes the facet's portion that is lost due to the groove intersections. As shown in Figure 6, such a portion is represented by the cross-sections of the intersecting grooves, thus we simply need to project these onto the facet and remove them from the polygon obtained in the previous step. The projection is just performed as before but using each groove direction T_2 as the projection direction. The base surface around the grooves is treated as an extra ground facet,

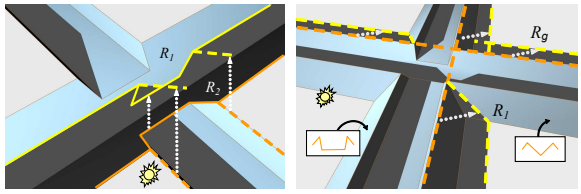

Figure 7: *Occlusions are evaluated after projecting the blocking facet (solid profile) and its prolongations (dashed segments) onto the current facet.*

where the portions to be removed are given by the bounds of each groove, i.e. the lines passing through their initial and final cross-section points and following the path tangent.

6.3. Occlusion

Occlusion can also be treated as a profile lying onto the current facet. This profile is obtained by projecting the blocking facet in the occlusion direction (see solid profile in left of Figure 7) along with its prolongations (see dashed segments). A blocking facet is a facet belonging to the same groove that cast occlusion to the current facet, where its profile is mainly described by the cross-sections of the intersecting grooves. Prolongations represent straight open-ended segments starting on this facet and following the highest points or peaks of such grooves. After their projection, the obtained profiles are unified in facet space and the final profile is subtracted from the footprint polygon.

Notice that occlusion is only computed if current facet is not already self-occluded. Also, concerning blocking facets, only self-occluded facets may cast occlusion onto the current facet and thus have to be considered. In left of Figure 7, R_2 may cast shadows on R_1 because it is self-shadowed ($N_2 \cdot L < 0$). For the ground facet, the occlusion profile is only given by the prolongations of the protruding grooves (see R_g in the right figure), which also applies for the external facets of such grooves (R_1 in the same figure).

6.4. Reflection contribution

The contribution of a facet to the total footprint reflection is computed using Equation (1). The main difference is that the area ratio r_i is here computed as A_{R_i}/A_F, where A_F is the area of the footprint polygon once projected onto the facet, and A_{R_i} the area after clipping, intersection and occlusion.

6.5. Ends and other similar cases

Groove ends and corners can be treated as special intersections by modifying the cross-sections that are projected during the previous steps. Concerning the intersection removal, if a facet belongs to an intersected end, the cross-section of the other groove must be extended following the ending direction (see top left of Figure 8). For isolated ends or corners,

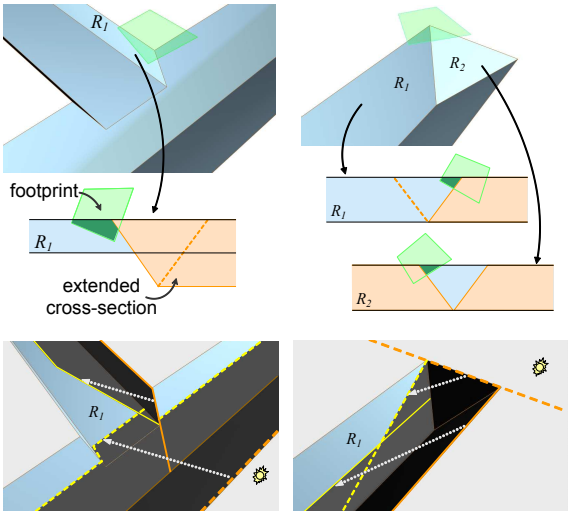

Figure 8: *Cases like intersected ends (left) and isolated ends (right) are handled by modifying the profiles that are projected during intersection (top) and occlusion (bottom).*

the cross-section is also extended in one or both directions, but without including the cross-section (see top right image).

During the occlusion step, the changes depend on the blocking facet. When the current facet belongs to an intersected end, its blocking facet is represented by half the cross-section of the intersecting groove (see $R1$ in bottom left image) and only one prolongation is projected. For facets belonging to isolated ends or corners, blocking facets are represented by a single straight line, since the occlusion does not depend on any cross-section (bottom right image).

7. Inter-reflections and transmission

This section describes how our method is extended to include indirect illumination due to the multiple scattering of light on the grooved surface. This extension is especially useful for the simulation of specular inter-reflections and transmissions, which can be correctly handled with some minor changes and a small increase in computational time.

The basic idea behind our method is to perform a kind of beam tracing on the grooved surface. For each facet contained in the footprint, this consists in recursively recomputing the algorithm using its visible portion as a new footprint and the reflection or transmission direction as a new view vector (see Figure 9). During the recursive execution of the algorithm, some considerations must be taken into account for this to work. For a certain bounce, we should only process facets lying on the visible part of the cross-section. If the entire cross-section is processed as in the direct pass, some facets from the non-visible part could mask the visible ones, which would be incorrect. On the other hand, view vectors may have negative heights ($E_w < 0$). When looking for the

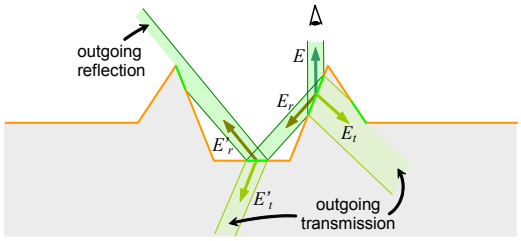

Figure 9: *Inter-reflections and transmissions on a groove.*

grooves affecting the new footprint (Section 4.1), this will require the footprint's bounding box to be enlarged in the opposite direction, since $h_{max,E}$ will result in a negative value. In the same way, the occlusion tests performed in Section 5.1 will need to be inverted accordingly.

Although at a higher computational cost, glossy and diffuse scattering can be computed using the same approach. This mainly requires using a set of random directions instead of the perfect specular ones. The indirect illumination coming from the rest of the scene or from other parts of the same object, such as side or back faces, can then be included using a global illumination technique such as ray tracing or similar. This should be considered when part of a new footprint does not project onto the grooves (see Figure 9), and be weighted according to the uncovered portion: $1 - \sum r_i$.

8. Results

	Our method		Geometry		Relief map	
Figure	time	mem	time	mem	time	mem
10 left	26	16	95	656	.043	256
10 right	27	19	114	53000	.029	4096
11 left	21	16	–	–	–	–
11 mid.	24	16	–	–	–	–
11 right	251	16	–	–	–	–
12 bot.l.	32	124	–	–	–	–
12 bot.r.	43	124	–	–	–	–
13	258	283	–	–	–	–
1 left	214	852	–	–	–	–
1 right	142	852	–	–	–	–

Table 1: *Performance of the different methods for each figure. Rendering time is in seconds and memory in kilobytes.*

Our method has been implemented as a Maya® shader, running on a Pentium 4 at 1.6 GHz. The performance for the following examples is shown in Table 1, where the memory corresponds to the size of the representations unless for the geometry case. First, in Figure 10, we compare the quality of our method with respect to ray traced geometry, which also offers high quality results, and with relief mapping [POC05], which also simulates detail without adding geometry.

In the left of the figure, the model corresponds to a surface containing lots of parallel grooves. Their cross-section

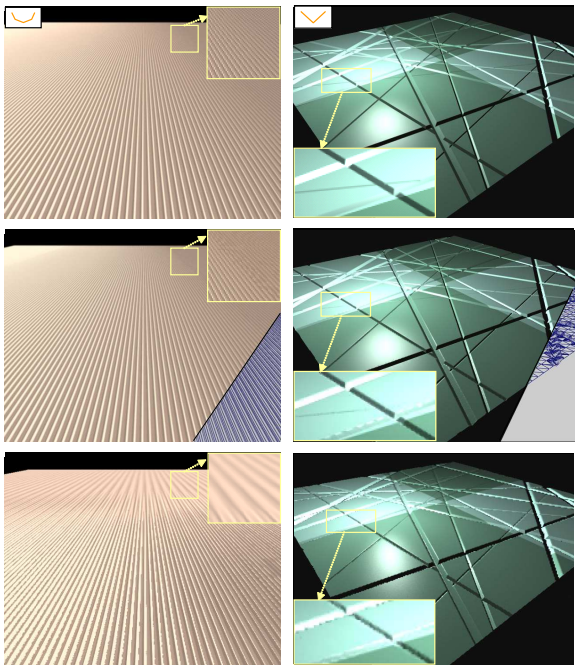

Figure 10: *Comparison between our method (top), ray traced geometry (middle) and relief mapping (bottom).*

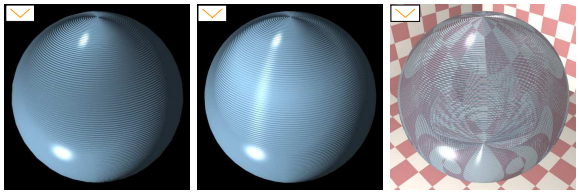

Figure 11: *Grooved sphere rendered with direct illumination (left), inter-reflections (middle) and refractions (right).*

is shown in the upper left. Our method and relief mapping use a plane as the base surface. In the ray traced version, the grooves have been added into the geometry and rendered using Mental Ray®, with 1 shadow ray and adaptive sampling of 4 to 64 samples per pixel. The images are nearly indistinguishable compared to our method, but our rendering is faster because a single sample is sufficient to accurately capture distant detail. For the bottom image, grooves have been encoded into a relief texture and rendered using a GeForce 6200. Relief mapping achieves interactive frame rates, but the obtained image presents several aliasing artifacts. For closer detail, the use of a single sample per pixel introduces aliasing. For distant detail, it is leveraged using mip mapping, but this simply smooths the detail according to the viewing distance. In the right of Figure 10, the model corresponds to a set of crossing scratches of different size. In this case, the geometry of the middle image is generated using displacement mapping. Even by using a feature-based approach, the number of obtained triangles is in the order of 500,000, which greatly increases the memory cost. For displacement and relief mapping, although a 1024x1024 texture is used, big grooves still show round shapes and some smallest grooves are missed, especially with relief mapping (see close views). This suggests that a higher resolution should be used in this case, which would suppose an increase in memory and rendering time. Note that if displaced geometry were rendered in hardware, its quality would similarly depend on the number of samples per pixel or the buffers resolution.

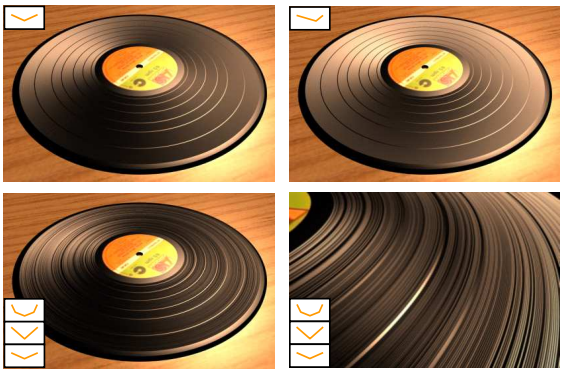

Figure 12: *Vinyls simulated using different cross-sections.*

In Figure 11, a grooved sphere is rendered with direct and indirect illumination, using our recursive approach. Middle image includes inter-reflections, computed with two bounces in a small increase in rendering time. Right image includes refractions due to a glass-like material, where ray tracing is used to capture the illumination from the surrounding box and the grooves in the back side. This increases the rendering time since multiple bounces are needed, although a single sample is sufficient to capture the refraction on the grooves.

Figure 12 shows three vinyls modeled using small concentric grooves. In the top, grooves share the same cross-section, while in the bottom, three different cross-sections are randomly applied. As can be seen, the anisotropic reflection produced by the grooves clearly depends on their cross-sections. First vinyl could be approximated using the anisotropic model of Poulin and Fournier [PF90] or even an empirical one, but these would fail for the others. Concerning the model by Stam [Sta99], it would require the derivation of an analytic function for each desired pattern, which would be impractical for non-regular patterns such as the one in the bottom images. Furthermore, BRDFs do not allow closer viewpoints like in the bottom right image.

In Figure 13, we can see a complex scene consisting of several grooved surfaces. This shows transitions from near to distant grooves and different groove situations. Some hieroglyphics are simulated using perturbations, and wider details such as circles are simulated using a contouring path and the second cross-section. Bump map is applied to simulate erosion on the columns, and shadows and reflections between objects are captured using ray tracing. Figure 1 shows another scene with many grooved surfaces simulated using our

Figure 13: *Complex scene with several grooved surfaces.*

approach. Roof consists of a set of macro-grooves simulating the tiles and small parallel micro-grooves that provide an anisotropic bluish effect to the tiles. Left image includes the underlying geometry, which only consists on 88 polygons.

9. Discussion and limitations

The memory requirements of our method mainly depend on the grid resolution. This resolution does not affect the quality of the results, but only our efficiency when finding the grooves. According to the rule stated in Section 4.1, common patterns generally require grids of 100×100. Complex patterns may result in bigger grids, such in the roof of Figure 1, where we use a grid of 300×300, but they are generally low. Concerning the performance, our area sampling approach is the one that requires most computations, but as special cases tend to be very localized, the increase in rendering time is usually low. With respect to the assumptions stated in Section 4, these may affect the simulation of highly curved paths or surfaces and produce visibility errors in certain situations. In such cases, their curvature should be considered during the computations. Finally, we believe that our method could be adapted for the GPU. For this, groove data could be stored into a texture and then be evaluated in a fragment shader. The area sampling approach could also be approximated using several line samples on the footprint to accelerate it.

10. Conclusions

In this paper, we have proposed an accurate method to simulate grooves and other path-based surface features. Small and large scale detail is taken into account using a resolution independent approach, which allows correct smooth transitions. It also considers occlusion effects and indirect illumination, and simulates general groove patterns including intersections and ends. All this solves several limitations existing on previous methods and offers high quality results with low memory and computational requirements.

References

[APS00] ASHIKHMIN M., PREMOZE S., SHIRLEY P. S.: A microfacet-based BRDF generator. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), pp. 65–74.

[Ban94] BANKS D. C.: Illumination in diverse codimensions. In *Proceedings of SIGGRAPH 94* (July 1994), pp. 327–334.

[BB90] BECKET W., BADLER N. I.: Imperfection for realistic image synthesis. *Journal of Visualization and Computer Animation 1*, 1 (Aug. 1990), 26–32.

[Bli78] BLINN J. F.: Simulation of wrinkled surfaces. In *Computer Graphics (Proceedings of SIGGRAPH 78)* (Aug. 1978), vol. 12, pp. 286–292.

[BM93] BECKER B. G., MAX N. L.: Smooth transitions between bump rendering algorithms. In *Proceedings of SIGGRAPH 93* (Aug. 1993), pp. 183–190.

[BPMG04] BOSCH C., PUEYO X., MÉRILLOU S., GHAZANFARPOUR D.: A physically-based model for rendering realistic scratches. *Computer Graphics Forum 23*, 3 (2004), 361–370.

[CL06] CANT R. J., LANGENSIEPEN C.: Efficient anti-aliased bump mapping. *Computers & Graphics 30*, 4 (2006), 561–580.

[Coo84] COOK R. L.: Shade trees. In *Computer Graphics (Proceedings of SIGGRAPH 84)* (July 1984), vol. 18, pp. 223–231.

[Fou92] FOURNIER A.: Normal distribution functions and multiple surfaces. In *Graphics Interface '92 Workshop on Local Illumination* (May 1992), pp. 45–52.

[Kaj85] KAJIYA J. T.: Anisotropic reflection models. In *Computer Graphics (Proceedings of SIGGRAPH 85)* (July 1985), vol. 19, pp. 15–21.

[KS00] KAUTZ J., SEIDEL H.-P.: Towards interactive bump mapping with anisotropic shift-variant BRDFs. In *2000 SIGGRAPH / Eurographics Workshop on Graphics Hardware* (Aug. 2000), pp. 51–58.

[MDG01] MÉRILLOU S., DISCHLER J.-M., GHAZANFARPOUR D.: Surface scratches: Measuring, modeling and rendering. *The Visual Computer 17*, 1 (2001), 30–45.

[MOiT98] MIZUNO S., OKADA M., ICHIRO TORIWAKI J.: Virtual sculpting and virtual woodcut printing. *The Visual Computer 14*, 2 (1998), 39–51.

[PF90] POULIN P., FOURNIER A.: A model for anisotropic reflection. In *Computer Graphics (Proceedings of SIGGRAPH 90)* (Aug. 1990), vol. 24, pp. 273–282.

[POC05] POLICARPO F., OLIVEIRA M. M., COMBA J. L. D.: Real-time relief mapping on arbitrary polygonal surfaces. In *Proceedings of I3D 2005* (2005), pp. 155–162.

[Sch97] SCHILLING A.: Toward real-time photorealistic rendering: Challenges and solutions. In *1997 SIGGRAPH / Eurographics Workshop on Graphics Hardware* (Aug. 1997), pp. 7–16.

[SKU08] SZIRMAY-KALOS L., UMENHOFFER T.: Displacement mapping on the GPU - State of the Art. *Computer Graphics Forum 27*, 1 (2008).

[Sta99] STAM J.: Diffraction shaders. In *Proceedings of SIGGRAPH 99* (Aug. 1999), pp. 101–110.

[War92] WARD G. J.: Measuring and modeling anisotropic reflection. In *Computer Graphics (Proceedings of SIGGRAPH 92)* (July 1992), vol. 26, pp. 265–272.

[WAT92] WESTIN S. H., ARVO J. R., TORRANCE K. E.: Predicting reflectance functions from complex surfaces. In *Proceedings of SIGGRAPH 92* (July 1992), vol. 26, pp. 255–264.