INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Local Illumination in Deformed Space

Fabrice Neyret

N° 2856

Avril 1995

THÈME 3

Rapport de recherche

ISSN 0249-6399

Local Illumination in Deformed Space

Fabrice Neyret *

Thème 3 — Interaction homme-machine,
images, données, connaissances
Projet Syntim

Rapport de recherche n° 2856 — Avril 1995 — 12 pages

Abstract: There are two ways of ray-tracing space-deformed objects: computing the resulting object and rendering it, or directly rendering the object in its initial space by curving rays. This last approach is advantageous for all non-explicit representations that often should otherwise be facetized (CSG, implicit functions), or for large databases for which too many components should be transformed (particle systems, volumes, complex geometry).

We deal here with ray casting in a structured space (voxels, complex geometry in volumetric textures, geometry in grids), focusing on the local illumination evaluation, especially in the anisotropic case. We apply it to the ellipsoidal reflectance model developed for volumetric textures in [Ney95].

Key-words: volumetric textures, space deformation, complex geometry

(Résumé : tsvp)

* Fabrice.Neyret@inria.fr http://www-rocq.inria.fr/syntim/research/neyret

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : (33 1) 39 63 55 11 – Télécopie : (33 1) 39 63 53 30

Illumination locale en Espace Courbe

Résumé : Il existe deux façons de calculer en lancer de rayon le rendu d'objets ayant subit une déformation spatiale : en construisant l'objet résultant avant d'en calculer le rendu, ou en rendant directement l'objet dans son espace propre et en courbant les rayons. Cette dernière approche est avantageuse pour toutes les représentations non-explicites qui devraient autrement être facetisées (CSG, fonctions implicites), ou pour les grosses bases de données pour lesquelles il faudrait transformer trop de composantes (systèmes de particules, volumes, géométrie complexe).

Nous traitons ici du lancer de rayon en espace structuré (voxels, géométrie complexe en texture volumique, géométrie triée en grille), en se concentrant sur l'évaluation de l'illumination locale, en particulier dans le cas de l'anisotropie. Nous l'appliquons au modèle de réflectance ellipsoïdal développé pour les textures volumiques dans [Ney95].

Mots-clé : textures volumiques, déformation spatiale, géométrie complexe

1 Introduction

Space deformation methods are used to model objets by deforming simple ones [SP86, Bec94], or to deform objects along the time [CJ91]. This is generaly done by transforming vertices for facets or control points for patches (patches can also be directly transformed into higher degree patches). So, this is not well adapted to other representations, often used in ray-tracing, such as CSG or implicit functions (however, these representations can be facetized). Furthermore, even with facets, some problems occur with high deformations, because transformed facets stay facets (nevertheless, some methods are able to subdivise the facets during the deformation). Moreover, this can be costly for huge representations, such as volumetric data, particle systems or complex repetitive geometry, for which the number of components can be higher than the number of visible pixels. And finally, reflectance specifications (bump mapping, anisotropy) are bypassed, as normals are recomputed from the resulting geometry.

A solution is to avoid effective geometric transformation, by transforming rays rather than space, thus doing the rendering in the initial object space. Alas, we have not found any implementation, as far as solving the geometry-ray intersection with non-linear rays is not easy. When data are structured in space (volume in voxels) or can be structured (e.g. with a grid which sorts the facets), the ray can be followed step by step as line drawing algorithms do in pixels. Concerning hudge databases, one has to note that complex repetitives geometries are still often hierarchized or sorted with grids, which are used for a long time to optimize ray-tracing, or can also be represented by volumetric textures. We propose in section 2 a simple ray-casting method for such a structured space, the general case being very difficult.

In section 3, we focus on the problem of local illumination, first with the Phong model, then with anisotropy. It is difficult to specify what can happen to a full BDRF (bi-directional reflectance function) after the deformation of an underlying surface when anisotropy is not caused by micro-geometry. Otherwise, the deformation of this micro-geometry can be computed. When caused by micro-geometry, BDRF can be represented by normals repartition function, or by a shape producing this normal distribution, acting like 'crystallization'. Thus, we supposed to have such a representation to encode the local reflectance [PF90, CMS87, Fou92].

In section 4, we apply these considerations to volumetric textures rendering: this method, introduced by Kajiya in [KK89], is extending in [Ney95] to the scope of repetitive complex geometry modeling, on a multiscale way which prevents aliasing and decreases cost. A volumetric sample stores a density and a local reflectance in voxels, structured in octree. The repeated and deformed copies, called *texels*, which are mapped on a surface, are thus a space deformation of the reference volume. In this section, we focus on the reflectance computation in the deformed space, with an ellipsoidal reflectance model.

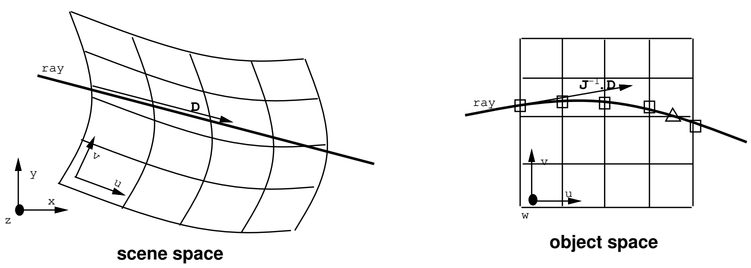
2 Curved Rays

A space deformation T transforms a point M_o into $M_s = T(M_o)$, the 'o' and 's' subscripts refering to the object space and to the scene space. The opposite transformation is generally difficult to obtain explicitly (e.g. inverse tricubic patch). A bounding box can be defined around the object and deformed at the same time, so that the transformation just need to be bijective inside this box, in order to be compatible with rendering in object space. Such a rendering is done by intersecting the deformed bounding box, converting these intersection points in the object space, then achieving the rendering with curved rays in the object space. We outline in this section how to ray-cast with curved rays.

The Jacobian matrix J_T of the direct transformation can be computed at any point, and the Jacobian $J_{T^{-1}}$ of the reverse transformation is the inverse of the former matrix. A vector V_o at the point M_o is transformed by T into $V_s = J_T(M_o).V_o$. The curved ray expression is hard to get and to use, while the intersections of the undeformed ray with the deformed faces of the bounding box is easier to get (it is a problem of intersection between straight rays and surfaces, see [Kaj82]), and motion along the ray can be done using the Jacobian. This suggests Newton-like numerical schemes to intersect the geometry, but one can predict an important cost added to the usual intersection cost. We propose here a simple numerical scheme adapted to structured space.

A first problem is to reverse the transformation T, i.e. to find M_o knowing M_s (respectively noted U and X in the following pseudo-code). Using the approximation $T^{-1}(M+dM) \simeq T^{-1}(M)+J_T^{-1}(M)dM$ then iterating gives a simple algorithm for *invT(M)*:

```
U = (.5 .5 .5)
loop
   X0 = T(U)
   dX = X-X0
   if |dX|<eps then exit
   invJ = inv(J(U))
   dU = invJ.dX
   U  = U+dU
```



The problem is now to cross the object space and to intersect the geometry. In structured space, the ray crosses cells where data is stored. A line drawing like method consists in parsing the cells, by finding the intersection between the ray and the grid which separates cells. This leads to the problem of determining the intersection between the ray and a slice parallel to an axis direction, defined by the index i of its constant component, and by the value c of this component. The ray is defined by the origin X0 and the direction D.

This gives the algorithm *reach_slice(i,c,X0,D)*:

```
X = X0
loop
   U = invT(X)
   if |U[i]-c|<eps then exit
   invJ = inv(J(U))
   dU = invJ.D
   d = (c-U[i])/dU[i]   ' or 1 if dU[i]=0
   X = X+d.D
   U = U+d.dU
```

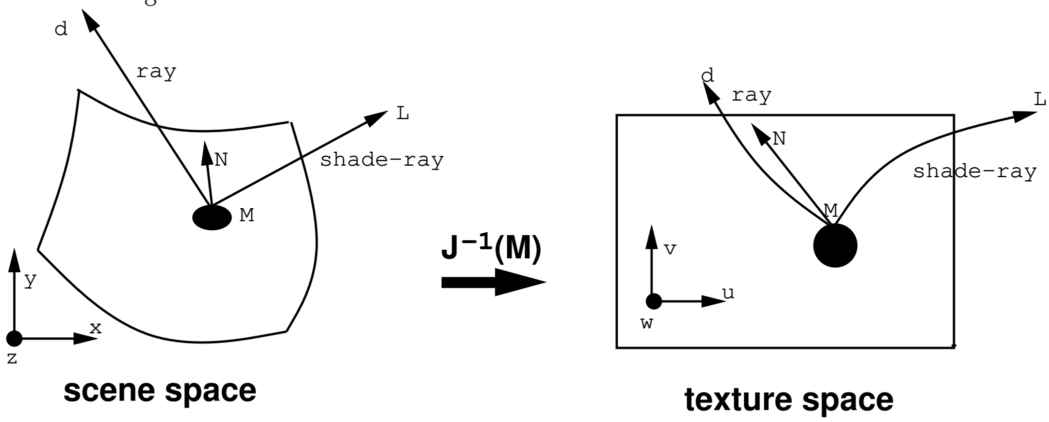
An improvement consists in initializing U=(.5 .5 .5) at the beginning of *reach_slice*, then not to initialize U in *invT*. Thus, the starting point is the last computed U, which improves the convergence rate. This can also be done for *reach_slice* initialisation, to keep the ray coherence. Another improvement is not to re-evaluate (and to inverse) J in *invT*, using the last value computed in *reach_slice*. To save the number of slice intersections, one can use hierarchical space structuration, in order to subdivide the space only where there is material, and to quickly cross empty areas. A simple one is octree representation [Sam90], which allowes to use rough cells in empty areas and to split them recursively in filled ones.

Convergence has to be guaranteed, at least by clipping U to keep it in the object bounding box. The complete algorithm which tracks successive grid intersections is a little more complicated than for line drawing. To go from a grid intersection to the next one along the ray, the problem is to choose the next slice to reach. An estimation can be got by assuming the ray to be straight inside a cell: if the step size between slices is 1, the current point is U, and the ray direction is $Du = invJ.D$, then the distances between U and the intersections of the straight ray with the bounding slices is given by $\frac{frac(U_i)}{Du_i}$ and $\frac{1-frac(U_i)}{Du_i}$. The smallest positive one gives an estimation of the real intersection point, and of the slice direction to use.

3 Local Illumination

After having reached a point on an object, the local illumination computation cannot be done in the object coordinate system instead of the scene coordinate system without modifications: illumination uses normal information, but the transformation of a normal vector N_o of the objet on a point M_o is no longer normal to the deformed object:

$normal_{T(obj)}(T(M_o)) \neq J_T.normal_{obj}(M_o)$. Consequently, the Phong illumination model which uses dot products between the local normal and vectors d and L in the scene space cannot be evaluated directly in the object space, using the local normal obtained in this space and transforming the other vectors. To compute the Phong model from object space data, one has either to deform the neightbourhood of M_o and re-evaluate the normal, either to modify the dot product so that it gives the same result as the canonical dot product in the scene space. Then the Phong model can be evaluated. Anisotropic reflectance will be evaluated as well after having dealt with the same problem for the normals repartition function integration.



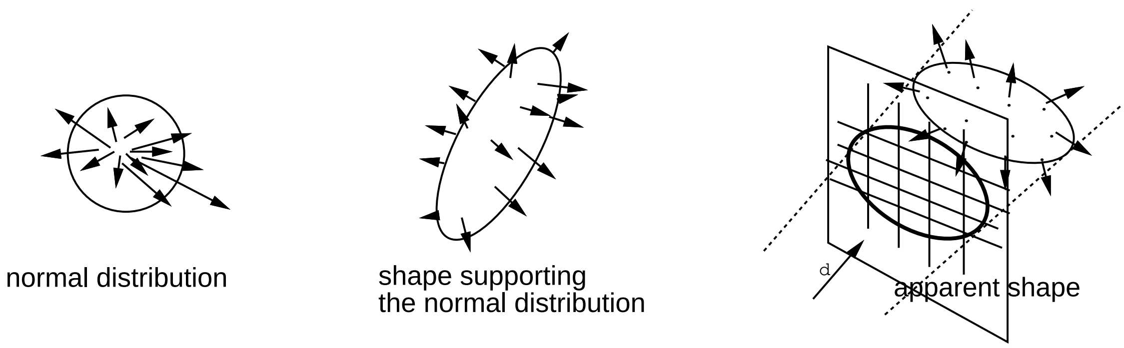
Let $< v_1, v_2 >$ the canonical dot product of v_1 and v_2, and $\|v\|$ the Euclidian norm of v. Given a vector V_o and the normal N_o on M_o in the object space, we have $< N_o, V_o > = < N_o, J^{-1}.J.V_o > = < J^{-1t}.N_o, V_s >$. This gives the normal in scene space [1] $N'_s = J^{-1t}.N_o / \|J^{-1t}.N_o\|$. So $< N'_s, V_s > = < N_o / \|J^{-1t}.N_o\|, V_o >$ where N_o and N'_s are the normal of the object before and after the deformation, which defines the modified dot product $< N_o, V_o >_T$. An other way to say the same thing is to convert the local normal N_o into N'_s instead of $J.N_o$ before doing the canonical dot product of N and a vector V in the scene space.

The Phong local illumination model [2] $C_{diff} < N, L > + C_{spec}(< N, H >)^r$ at a point M is now obtained from the visibility expressions $vis_d = < N, d >$ and $vis_L = < N, L >$ related to light and observer directions L and d, the dot product being the canonical or the modified one depending on the space where normal N is taken:
$Illu = C_{diff}.vis_L(N).1_{(vis>0)} + C_{spec}.(\frac{vis_d(N)+vis_L(N)}{\|d+L\|)})^r.1_{(vis>0)}$ (the function $1_{(cond)}$ value 1 if *cond* is true, else 0).

[1] If V_o is on the tangent plane in the object space, so that $< N_o, V_o > = 0$, then $V_s = J.V_o$ is on the tangent plane in the scene space. As $< J^{-1t}.N_o, V_s > = < N_o, V_o > = 0$, $J^{-1t}.N_o$ is orthogonal to V_s, being thus the normal in the scene space.

[2] $H = \frac{d+L}{\|d+L\|}$. r is the inverse of roughness. C_{diff} and C_{spec} are the diffuse and specular reflection coefficient.

For anisotropic reflectance, the local illumination at a point according to the given normal repartion function $\mathcal{N}$ is evaluated by integrating the local Phong model on the normal distribution on the Gaussian sphere: $Illu = \frac{\int_{\mathcal{G}auss} Phong(N,d,L).vis_d(N).1_{(vis>0)}.\mathcal{N}(N)}{\int_{\mathcal{G}auss} vis_d(N).1_{(vis>0)}.\mathcal{N}(N)}$
A normal is associated to a surface element ds having an apparent surface $ds.vis_d$ or 0 if the normal is not towards the observer, and has a distribution $\mathcal{N}(N)$, all this weightening the local Phong reflectance. The integral is normalized so that the underlying apparent surface is 1.



Whe assume that N is defined by a shape, the normal distribution being the normals of the shape which thus define a kind of 'crystallization'. Then illumination can be written
$Illu = \frac{\int_{shape} Phong(N(M),d,L).vis_d(N(M)).1_{(vis>0)}.dS}{\int_{shape} vis_d(N(M)).1_{(vis>0)}.dS}$
This integral can be written directly on the apparent 2D surface:
$Illu = \frac{\int_{2Dsurf} Phong(N(M_{(x,y)}),d,L).ds}{\int_{2Dsurf} ds}$,
which can be approximated by $1/n.\sum^n Phong(N(M_{(x,y)}), d, L)$

The anisotropic illumination computation scheme is thus:

- evaluating the apparent surface of the shape modeling the reflectance,
- at each sample on this surface, calculating N on the shape
- evaluating vis_d, vis_L, in object or scene space, which gives the Phong illumination, that is cumulated to finally gives $Illu$.

4 Ellipsoidal Reflectance

The reflectance model we described in [Ney95] encodes the normal repartion function $\mathcal{N}$ by an ellipsoid stored in voxels in the reference space, in addition to a density. This model is compact, but generic enough in the scope of volumetric textures, where it encodes sub-voxel geometric variations.

The initial space is deformed by the transformation T into the scene space. Assuming linear deformation at voxel size around the voxel center O_c (the local deformation being determined by the Jacobian $J_T(O_c)$), the reflectance can still be encoded by an ellipsoid in the scene space. So to compute the illumination, we chose to convert the local ellipsoid defined in the initial space, into the scene space. The problem is thus to evaluate the resulting ellipsoid in order to get its normals, and the apparent ellipse to be sampled. In the case of volumetric textures, the 'crystallization' also encodes the anisotropic occlusion (a voxel is not a surface element), so that an additional normalisation has to be done.

An ellipsoid is characterized by a basis R and three length r_i, so the associated quadratic form $M_o^t.Q.M_o = 1$ is $Q = R^t.D^{-2}.R$ in the initial space, with D diagonal such as $D_{i,i} = r_i$, R normal, and M_o a point on the ellipsoid (the origin being the ellipsoid center).

Converting the ellipsoid in the scene space is equivalent to back-convert a point M_s on the resulting ellipsoid into the initial space:

$M_s^t.Q.M_s = 1$ (with Q positive) $\Leftrightarrow \|D^{-1}.R.J_{T^{-1}}.M_s\| = 1 \rightarrow Q = J_{T^{-1}}^t.R^t.D^{-2}.R.J_{T^{-1}}$

According to the previous section, we have to find the apparent shape, and the normal corresponding to a point sampled on this surface. The apparent ellipse is the projection of the ellipsoid following the observer direction d. Its quadratic form is $Q' = Q - Qdd^tQ/d^tQd$. Eigenvalues (l, h) and eigenvectors determine a bounding box of the ellipse. Given (x, y) on the apparent ellipse surface, z so that $(x, y, z) = M$ is on the ellipsoid is obtained from the second order equation

$z^2.d^tQd + 2z.(x.d^tQl + y.d^tQh) + (x^2.l^tQl + y^2.h^tQh + 2xy.h^tQl - 1) = 0$

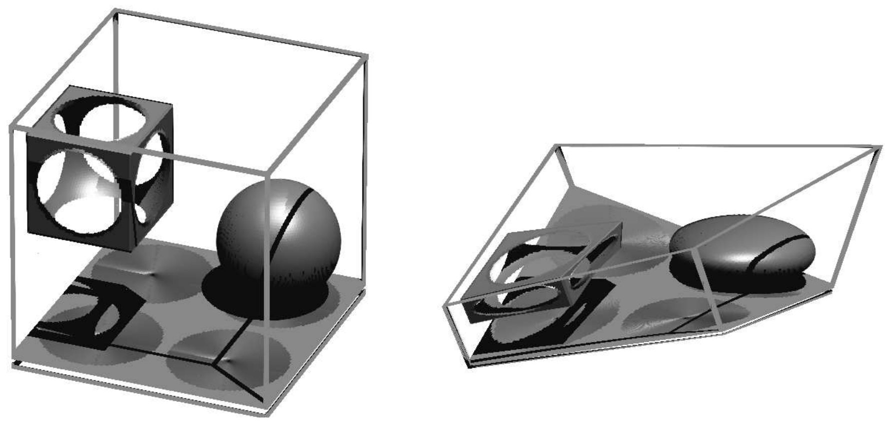
z(-x,-y) can be obtained at the same time, so half of the ellipse need to be sampled. All the terms $v_1^tQv_2$ are of course pre-computed.

A normal to the ellipsoid at M can them be obtained by $N = \frac{Q_s.M}{\|Q_s.M\|}$, which gives the illumination on this point according to the Phong model. QM can be directly extracted by $x.Ql + y.Qh + z.Qd$, avoiding matrix product. The illumination computation scheme detailled in the previous section can then be evaluated.

An additional normalisation has to be done in the scope of volumetric textures, in order to treat with the same weight the different shapes, which also encode anisotropic occultation. This normalisation is given by the mean apparent surface of the ellipsoid. We approximate it by using the apparent surface in the three axis directions:

$Smoy \simeq \pi/3.(r_1r_2 + r_2r_3 + r_1r_3)$. With $K = J_{T^{-1}}^t.R^t.D^{-1}.R.J_{T^{-1}}$, which has eigenvalues $1/r_i$ when $J_T = Id$, $Smoy \simeq \pi/3.|trace(K)/det(K)|$ (exact only if the deformation is a rigid motion).

5 Results



The figure shows a texel of volumetric texture before and after the trilinear distortion. The floor is slightly anisotropic. One can see on the sphere that the transformation of the illumination effects (specular, diffuse and shadows) do not simply follow the geometric deformation as a texture would do. The cube is here to illustrate CSG (although CSG and implicit constructions are voxelized, using volumetric textures representation).

6 Conclusion

We have proposed a guideline to ray-trace space-deformed objects in their initial space, which improves computation in the case of huge databases, and allows non-explicit representations often used in ray-tracing.

The reflectance computation in the simple and anisotropic cases has been presented. A simple anisotropic reflectance model, usable for volumetric textures, has been described.

A simple curved ray casting has also been proposed for structured space. It has to be fully described in the general case.

References

[Bec94] D. Bechmann. Space deformation models survey. *Computers & Graphics*, 18(4):571–586, 1994.

[CJ91] Sabine Coquillart and Pierre Jancène. Animated free-form deformation: An interactive animation technique. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 23–26, July 1991.

[CMS87] Brian Cabral, Nelson Max, and Rebecca Springmeyer. Bidirectional reflection functions from surface bump maps. In Maureen C. Stone, editor, *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21(4), pages 273–281, July 1987.

[Fou92] Alain Fournier. Normal distribution functions and multiple surfaces. In *Graphics Interface '92 Workshop on Local Illumination*, pages 45–52, May 1992.

[Kaj82] James T. Kajiya. Ray tracing parametric patches. In *Computer Graphics (SIGGRAPH '82 Proceedings)*, volume 16(3), pages 245–254, July 1982.

[KK89] James T. Kajiya and Timothy L. Kay. Rendering fur with three dimensional textures. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23(3), pages 271–280, July 1989.

[Ney95] Fabrice Neyret. A general and multiscale method for volumetric textures. In *Graphics Interface'95 Proceedings*, volume 00(0), pages 000–000, May 1995.

[PF90] Pierre Poulin and Alain Fournier. A model for anisotropic reflection. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24(4), pages 273–282, August 1990.

[Sam90] Hanan Samet. *Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, Massachusetts, 1990.

[SP86] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 151–160, August 1986.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399