影を考慮したレリーフテクスチャマッピングにおける効率的なレンダリングに関する研究

小磯 雄一　　西田 友是
東京大学大学院 新領域創成科学研究科
{ koiso, nis }@is.s.u-tokyo.ac.jp

レリーフテクスチャマッピングは、ソフトウェア的な前処理によってテクスチャを変形させておくことで、奥行きをもったテクスチャを高速にレンダリングすることができる手法である。しかし、従来法での影のレンダリングなどにおいては十分な最適化が議論されているとはいえなかった。そこで本稿では、レリーフテクスチャの影を効率的にレンダリングするための簡潔なアルゴリズムを提案するとともに、レリーフテクスチャを応用したウォークスルーシステムを提案し、そのレンダリングの実験を行ったので報告する。

An Efficient Rendering Method for Relief Texture Mapping Taking Into Account Shadows

Yuichi Koiso　　Tomoyuki Nishita
Graduate School of Frontier Sciences, The University of Tokyo

Relief texture mapping is a technique to render textures with depth information in high speed, by pre-warping the textures in the preprocessing step. The original paper of relief texture mapping did not provide an adequate discussion on the detection of shadows projected by a relief texture. In this paper, we propose a simple algorithm to render shadows of relief texture effectively. We also propose a walk-through system which uses the relief texture mapping technique and depict its experimental results.

1　はじめに

レリーフテクスチャマッピング[1]は、奥行きをもったテクスチャを高速にレンダリングするための画像ベースレンダリング手法である。レリーフテクスチャを用いることで、テクスチャ中の物体の立体感を表現することができるため、３Ｄシーンのレンダリングに利用することができる。レリーフテクスチャはバンプマッピング等の手法とは異なり、実際の位置の変位を表現することができる。このようなレリーフテクスチャを６面体の各面上に配置すると、オブジェクトを表現することもできる。このような３Ｄオブジェクトシーンのレンダリングにおける影づけは、物体の立体感を表現するにあたり大きな役割を果たす。しかし従来法については、文献[2]において、影はシャドウマップ法[3]を用いることで計算できると言及されていたに過ぎず、どのような実装が望ましいかについては、ほとんど議論がなされていなかった。そこで、本稿ではレリーフテクスチャマッピングにおける影の効率よい計算方法を提案する。

また、レリーフテクスチャを応用することで、複雑なジオメトリをもつ膨大なポリゴンからなるデータの表示を高速化することが可能である。そこで、本稿ではこのような表示の高速化の応用例として、レリーフテクスチャを利用したウォークスルーシステムについて提案する。

2　レリーフテクスチャマッピング

レリーフテクスチャはテクスチャの各要素（以下、テクセルとよぶ）に奥行き情報を保持しており、各テクセルがスクリーン上にレンダリングされる位置は、テクスチャと視点の位置関係と奥行き情報から求めることができる。レリーフテクスチャマッピングでは、グラフィクスハードウェアのテクスチャマッピング機能を利用できるように、前処理としてテクスチャの変形（プレワーピング）をソフトウェア的に行っておくことで、高速なレンダリングを実現している。

図１に示すように、ベクトル$\vec{a}$,$\vec{b}$をテクスチャ座標系における基底ベクトルとし、$\vec{c}$を視点位置から、テクスチャ座標の原点のワールド座標系での位置へ向かうベクトルとする。また、$\vec{f}$をテクスチャ面の単位法線ベクトルとする。このとき、テクセル(u,v)における$\vec{f}$方向への奥行き情報が、基準面からの変動量$displ(u,v)$によって

与えられるものとする。

もとのテクスチャにおける座標(u_s,v_s)のテクセルがプレワーピングによって(u_i,v_i)に写像されるとしたとき、透視投影モデルの場合は以下の式によって、プレワーピング後のテクスチャ座標を求めることができる。

$$u_i=\frac{u_s+k_1 displ(u_s,v_s)}{1+k_3 displ(u_s,v_s)},\ v_i=\frac{v_s+k_2 displ(u_s,v_s)}{1+k_3 displ(u_s,v_s)}, \quad (1)$$

$$k_1=\frac{\vec{f}\cdot(\vec{b}\times\vec{c})}{\vec{a}\cdot(\vec{b}\times\vec{c})},\ k_2=\frac{\vec{f}\cdot(\vec{c}\times\vec{a})}{\vec{a}\cdot(\vec{b}\times\vec{c})},\ k_3=\frac{1}{\vec{c}\cdot\vec{f}}. \quad (2)$$

また、平行投影モデルの場合は以下の式によって、プレワーピング後のテクスチャ座標を求めることができる。

$$u_i=u_s+k_4 displ(u_s,v_s),\ v_i=v_s+k_5 displ(u_s,v_s), \quad (3)$$

$$k_4=\frac{1}{|\vec{a}|^2}\cdot\frac{\vec{c}\cdot\vec{a}}{\vec{c}\cdot\vec{f}},\ k_5=\frac{1}{|\vec{b}|^2}\cdot\frac{\vec{c}\cdot\vec{b}}{\vec{c}\cdot\vec{f}}. \quad (4)$$

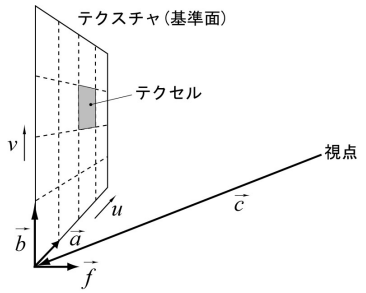

図1. レリーフテクスチャにおける変数の定義

3　レリーフテクスチャにおける影の考慮

3.1　平行光源モデル

レリーフテクスチャにおけるカメラモデルに関しては平行投影も透視投影もほぼ同様に扱うことができる。陰影計算においても平行投影（平行光源）や透視投影（点光源）を考慮でき得るが、簡単のために、以降では平行光源モデルのみを考え、ベクトル$\vec{l}$によって光源方向を表すものとする。

3.2　反射光計算に必要な法線の計算

拡散反射光や鏡面反射光の計算に必要となる、テクセル(u,v)における法線$\vec{n}(u,v)$は、以下の式によって計算することができる。

$$\vec{n}(u,v)=(2|\vec{a}|,0,\Delta displ_u)\times(0,2|\vec{b}|,\Delta displ_v), \quad (5)$$
$$=\left(|\vec{b}|\Delta displ_u,|\vec{a}|\Delta displ_v,|\vec{a}||\vec{b}|\right)$$

$$\Delta displ_u=displ(u+1,v)-displ(u-1,v), \quad (6)$$
$$\Delta displ_v=displ(u,v+1)-displ(u,v-1).$$

この法線$\vec{n}$と、光線ベクトル、視線ベクトルから反射光の強さを計算することができる。

3.3　影の計算

提案法は、シャドウマップ法[3]の考え方に基づいて影の計算を行う。その際に、セルフシャドウ（レリーフテクスチャがそれ自体に落とす影）と、レリーフテクスチャが他のレリーフテクスチャへ落とす影とを分けて考えることにする。

3.3.1　セルフシャドウ

シャドウマップ法は、光源方向から見たときに不可視領域が影領域である、という概念に基づいている。光線ベクトル$\vec{l}$をテクスチャ面上に投影し、そのテクスチャ座標系でのベクトル値を$\vec{l}_T$とする。光源方向から見た場合、平行投影モデルにおける*displ*の変化に伴うプレワーピング後の座標の移動は$\vec{l}_T$方向（図2の直線L上）のみに限られる。よって、各テクセルが影領域に含まれるかどうかの判定は、直線L上に沿って調べれば十分である。

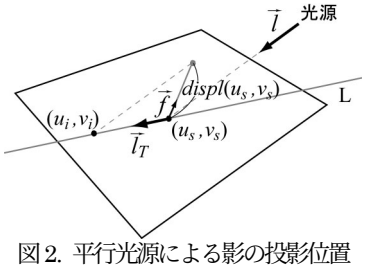

図2. 平行光源による影の投影位置

図3に影の判定の概念図を示す。判定アルゴリズムは以下のようになる。ただし、影の範囲を考える際には、座標値をu（またはv）軸に投影して考えるものとする。

(1) 影の範囲Sを空にする。

(2) L上にあるテクセル(u_s,v_s)をベクトル$\vec{l}_T$の方向にたどり、以下を適用する。

(i) プレワーピング後の座標(u_i,v_i)を計算する。

(ii) (u_i,v_i)がSに含まれるときは、テクセル(u_s,v_s)の位置は影であると判定する（図3a参照）。

(iii) (u_i,v_i)がSに含まれないときは、テクセル(u_s,v_s)の位置は影ではないと判定し、Sを(u_s,v_s)を含むように延長する（図3b参照）。

(3) (2)を繰り返す。

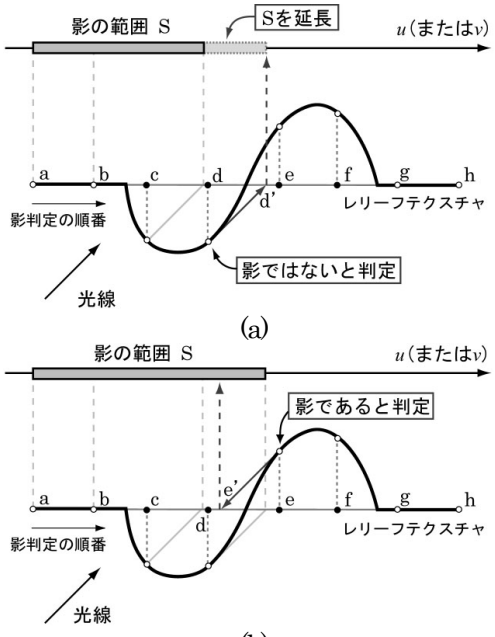

図3. セルフシャドウの判定の概念図

例えば図 3a では、テクセル d のプレワープ後の位置 d’は影の範囲に含まれないため、テクセル d の位置は影ではないと判定される。また、図 3b では、テクセル e のプレワープ後の位置 e’は影の範囲に含まれないため、テクセル e の位置は影であると判定される。

レリーフテクスチャにおけるセルフシャドウの計算では、次のような２つの特別なケースは例外として別処理を行う。１つ目のケースはテクスチャが光源方向に対して垂直である場合であり、その場合にはテクスチャ上に影ができないため計算を行う必要はない。２つ目のケースはテクスチャが光源方向と平行な場合で、この場合にはプレワーピング後のテクスチャ座標の計算をすることができないため、*displ* の値を用いて影の範囲を表現し、影の判定を行う（図 4 参照）。

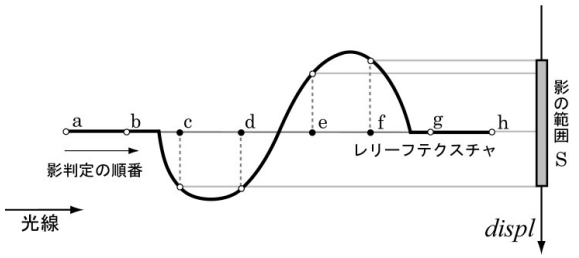

図 4. 光線とレリーフテクスチャが平行な場合の影計算

3.3.2 テクスチャが他のテクスチャに落とす影

レリーフテクスチャが他のレリーフテクスチャに落とす影を計算する手順は以下のアルゴリズムで示すことができる。

(1) 各レリーフテクスチャに対して優先順位を決定する（光源に近いものほど高い優先順位を持つ）。
(2) 各レリーフテクスチャで表現された物体のシルエットを考え、それに対して、光源方向を視点とみなしてプリワーピングを行い、それをシルエットテクスチャとして保存する。
(3) 各レリーフテクスチャのシルエットテクスチャを、そのテクスチャより優先順位の低い全てのレリーフテクスチャの基準面上へ投影し、それらの和をとることで影テクスチャを生成する（図 5 は、２枚のレリーフテクスチャ A,B が存在し、A の方が B よりも優先順位が高い場合の例）。
(4) 各レリーフテクスチャのセルフシャドウを計算する際に、影の範囲 S の初期範囲として影テクスチャを与え、影の判定を行う。

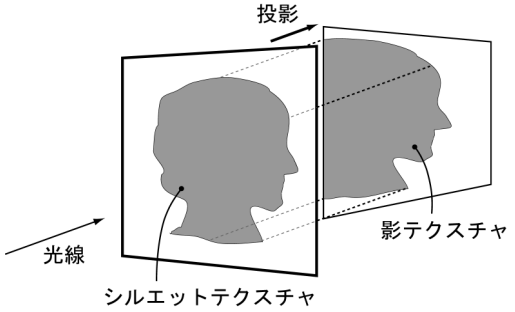

図 5. 影テクスチャの生成例

4 実験結果と考察

レリーフテクスチャの適用例として、レリーフテクスチャオブジェクトの表示実験を行った。図 6 に、陰影処理の有無によるレンダリング結果の違いを示す。反射光や影を考慮することで、より立体感が把握しやすく表示されている。また、図 6 の例をレンダリングするのにかかる時間を表 1 に示す。図 6 の例では、解像度が 128×128 のレリーフテクスチャ 6 枚を立方体面上に配置した、レリーフテクスチャオブジェクトを使用した。表示サイズは 512×512 であり、ここで使用した計算機は CPU が PentiumIII 800MHz の Windows 機である。

表 1 を見ると、レリーフテクスチャ自体のレンダリングと比べ、比較的高速に影をレンダリングできていることがわかる。

(a) 照明効果なしの場合　(b) 反射光を考慮した場合　(c) 影を考慮した場合

図 6. レリーフテクスチャと陰影処理の効果

表 1. 図 6 の例のレンダリング時間

	(a)	(b)	(c)
フレームレート[fps]	11.4	10.1	8.70

5 ウォークスルーシステムへの応用

レリーフテクスチャを応用することで、ウォークスルーシステムにおける表示を高速化することができる。複雑なジオメトリをもつ膨大なポリゴンデータからなる地形をリアルタイムに表示するには、十分なグラフィクスハードウェアの性能が必要である。これまでに、ビルボードを用いて仮想空間を高速に表示する方法[4][5]は提案されているが、それらの方法では、ビルボードの切り替え時に、各物体の表示される位置がずれてしまうという欠点があった。しかし、ビルボードとしてレリーフテクスチャを用いることで、この問題を解決することができる。

提案法では、視点位置からある距離以上離れた領域をレリーフテクスチャとして保存することで、レンダリングを高速化する。まず、図 7a に示すように、ある視点位置 V の周囲に、解像度が $n \times n$ の４枚のレリーフテクスチャ RT_i (i=0,1,2,3)を、V から *distance* の距離に、テクスチャの大きさが(4･*distance*)×(4･*distance*)の大きさになるように配置する。次に、セルの中心から見た領域 D_i の景色をレンダリングし、その結果画像をレリーフテクスチャ RT_i の色バッファに保存する（図 7b 参照；レリーフテクスチャの中心部のみにレンダリングされることに注意）。レリーフテクスチャでは *displ* の値も計算する必

要があるが、そのかわりに、Z-バッファの値を $z(u,v)$ としてレリーフテクスチャ RT_i に保存しておき、プレワーピング時に以下の式を用いて $z(u,v)$ を $displ(u,v)$ に変換し、テクスチャ座標の計算を行う（図 7b 参照）。

$$displ(u'_s, v'_s) = z(u_s, v_s), \tag{7}$$

$$u'_s = u_s - \frac{z(u_s, v_s)}{distance}\left(u_s - \frac{n}{2}\right), \tag{8}$$

$$v'_s = v_s - \frac{z(u_s, v_s)}{distance}\left(v_s - \frac{n}{2}\right). \tag{9}$$

ここで視点位置 V から移動する場合を考える。すると図 7b に示すように、領域 M_i 内に視点がある場合には、プレワーピング後にも領域 D_i 内の全てのオブジェクトはレリーフテクスチャ RT_i 内に表示することができるので、その限りにおいてはレリーフテクスチャを更新する必要がないといえる。各レリーフテクスチャについて同様のことがいえるので、領域

$$M = M_1 \bigcap M_2 \bigcap M_3 \bigcap M_4$$

から視点位置が外れた場合には、その位置を新たな視点位置 V として、再度レリーフテクスチャをレンダリングする必要がある。また、どの領域 D_i にも含まれていないオブジェクトについては、通常のポリゴンとしてレンダリングし、背景のレリーフテクスチャと合成して表示を行う。図 8 に、提案したウォークスルーシステムによる表示例を示す。

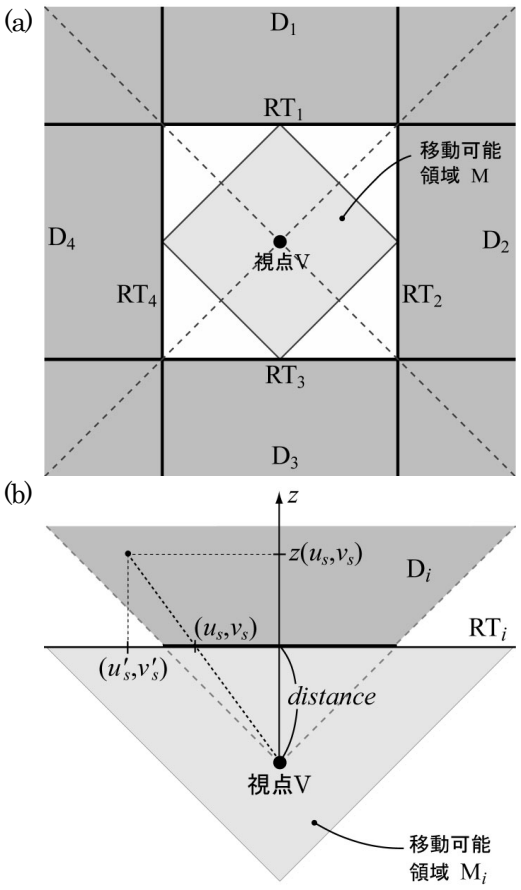

図 7. ウォークスルーシステムにおけるテクスチャ配置図

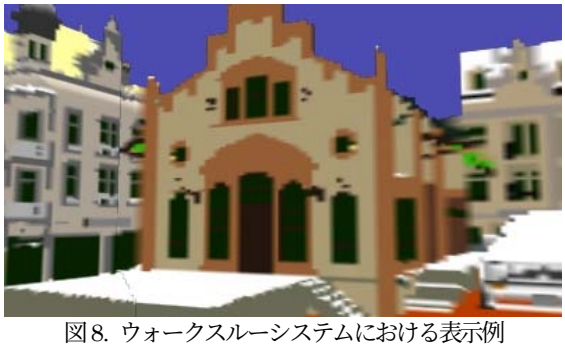

図 8. ウォークスルーシステムにおける表示例

6 まとめ

レリーフテクスチャマッピングにおける影の計算において、影の範囲の概念を用いた簡潔かつ効率的なアルゴリズムを提案した。提案法を用いることで、比較的高速に影を考慮したレリーフテクスチャをレンダリングすることができた。

本稿では光源モデルとして平行光源を仮定した。点光源は、レリーフテクスチャ毎に局所的な平行光源を仮定することで、近似的にシミュレートをすることが可能であるが、点光源や天空光などの光源によるレリーフテクスチャの影のレンダリングは、今後の課題のひとつである。

また、レリーフテクスチャを利用したウォークスルーシステムについても提案した。このシステムでは遠景をレリーフテクスチャとして保存しておくことで、遠景のレンダリングを高速化することで、ジオメトリが複雑な場合でも高速にウォークスルーを行うことができると期待される。しかし、今回の実験においては、レンダリング速度はリアルタイムの表示には、まだ十分ではないといえる。そこで、近年ハードウェア vertex shader が一般化されつつある現状を踏まえ、今後はハードウェアを用いたレリーフテクスチャの影計算や表示の実験を行っていく予定である。

参考文献

[1] M. M. Oliveira, G. Bishop, and D. McAllister, “Relief Texture Mapping”, *Proc. SIGGRAPH 2000*, pp.359-368, 2000.

[2] M. M. Oliveira, “Relief Texture Mapping, Ph.D. Dissertation”, *UNC Computer Science Technical Report TR00-009*, University of North Carolina, March 3, 2000.

[3] L. Williams. Casting Curved Shadows on Curved Surfaces, *Computer Graphics (Proc. SIGGRAPH'78)*, pp. 270-274, 1978.

[4] 斉藤 茂実, 西原 清一, 「ビルボードを用いた都市空間の高速表示」, 情報処理学会 第 58 回全国大会 講演論文集, pp.159-160, 1999.

[5] 平川 隆仁, 小堀 研一, 「円筒形ビルボードを用いた仮想空間の高速表示」, 情報処理学会 第 62 回全国大会 特別トラック(2) 講演論文集, pp.71-72, 2001.