グラフィクスとCAD 109－7
（2002. 11. 22）

レリーフテクスチャマッピングへのパーピクセルシェーダの適用

藤田将洋* 金井崇†
* 慶応義塾大学 政策・メディア研究科 † 慶応義塾大学 環境情報学部
E-mail: {syoyo/kanai}@sfc.keio.ac.jp

要旨 イメージベースドレンダリングの一手法に、デプス(奥行き)情報を持った画像をワーピング処理により変形して出力画像を生成する手法(Image-Based Rendering by Warping, IBRW)がある。イメージベースドレンダリングでは、既存のポリゴンのレンダリングに比べてシーンが複雑になっても一定時間でレンダリングできるという利点がある。レリーフテクスチャマッピング(relief texture mapping)はIBRWをプリワーピング(Pre-Warping)とテクスチャマッピングの2つの処理に分解してより効率的な計算を可能にする手法である。本論文ではレリーフテクスチャマッピングに近年のグラフィックスハードウェアが備えるパーピクセルシェーディング機能を適用することで反射マッピングなどのより高品質なシェーディング効果を行う方法を提案する。

Applying Per-pixel shading for Relief Textre Mapping

Masahiro Fujita* Takashi Kanai†
*Keio University, Graduate School of Media and Governance
†Keio University, Faculty of Environmental Information
E-mail: {syoyo/kanai}@sfc.keio.ac.jp

In this paper, we try to apply some hi-quality shading such as reflection mapping for relief texture mapping.

1 はじめに

レリーフテクスチャマッピング[8]はテクスチャマッピングを利用して比較的高速にイメージベースド画像のレンダリングを行う手法である。
一般的に、イメージベースドレンダリングはポリゴンのレンダリングに比べてシェーディング機能があまり豊かではなかった。ポリゴンのレンダリングの分野では、古くから画像を使用したシェーディング機能の向上を図ってきた。もっとも顕著な例がテクスチャマッピングである。また、法線情報やライティング情報をテクスチャ画像にエンコードすることにより、バンプマッピングやライティングを付加することができる。

これまでのグラフィックスAPIでは、ポリゴンの頂点を単位とした表現に合わせて設計されてたので、テクセル(テクスチャ画像の各ピクセル)単位での計算が必要になるイメージベースドレンダリング手法をハードウェア化することが困難であった。しかし最近のパーピクセルシェーディング機能の登場によりフラグメント(ピクセルになる前の状態で、テクスチャ色、アルファ値、頂点色などのピクセル色の計算に必要な情報を保持している)単位、テクセル単位での処理をハードウェア化することが可能になった。

本文ではこのパーピクセルシェーディング機能を利用し各種テクスチャマッピング技術を組み合わせることで、高品質なシェーディングを行う方法を提示する。

2 関連研究

2.1 画像を用いたレンダリング手法

画像を用いて曲面に3次元的な立体感を与える最もポピュラーな方法は、**バンプマッピング**[2]と**ディスプレースメントマッピング**[4]である。
バンプマッピングは、ピクセル単位で法線を摂動させることにより凹凸感のある曲面を表現する。バンプマッピングは曲面の幾何情報を変形させないため、視点と曲面とが平行に近づくと凹凸感がなくなってしまう。
ディスプレースメントマッピングは変位情報を持った画像を元に曲面の幾何情報を変位させる。ディスプ

レースメントマッピングはバンプマッピングよりもより自然な凹凸感を表現できるが、曲面の幾何情報が増大するためレンダリングコストが非常に高い。

2.2 イメージベースドレンダリング

この数年の間に、数々のイメージベースドレンダリング (image-based rendering, IBR) 手法が提示されてきた。
画像を用いてシーンを構築する方法として、Lippman[5] によるムービーマップ (movie-map) システムは最も初期のイメージベースドレンダリング手法である。ムービーマップは、各視点に対応するイメージあらかじめデータベースとして用意しておく方法である。視点は固定された範囲でのみ変えることができる。
また、Chen ら [3] は、シリンダ状のイメージからシーンを構築する QuickTimeVR システムを開発した。このシステムでは視点をインタラクティブに (20 フレーム以上) 変更することができる。
複数の画像とそれに関連づけられた奥行き情報を利用して画像のみでシーンを表現する手法として、McMillan[6] は 3D イメージワーピング法を提案した。
Oliveira[8] のレリーフテクスチャマッピングは、この 3D イメージワーピング法をプリワーピング変換とテクスチャマッピングの組み合わせで表現した方法であり、バンプマッピングとディスプレースメントマッピングの中間的な計算コストで立体感のある表現を可能にする。

2.3 パーピクセルシェーディング

米 NVIDIA 社により開発されたパーピクセルシェーディング機能 [7] は、テクスチャマッピングのテクセルフェッチ部およびフラグメントプロセッサの一部に対して、ユーザがプログラマブルに処理手順を与えることができる機能である。OpenGL グラフィックス API においてこのパーピクセルシェーディング機能を有効にすると、既存の OpenGL レンダリングパイプラインのテクスチャフェッチステージ、フォグ演算ステージ、およびカラーサムステージを、テクスチャシェーダステージおよびレジスタコンバイナステージで置き換える。

2.3.1 テクスチャシェーダステージ

テクスチャシェーダステージでは、21 種類のテクスチャフェッチプログラムが用意されている。
テクスチャシェーダでは、テクセル単位での内積計算、オフセット (テクスチャ座標の摂動)、クリッピングテストなどを実行することができる。

2.3.2 ハードウェアによるサポート

近年のグラフィックスカードがこのパーピクセルシェーディング機能をハードウェアで実装したことにより、バンプマッピングや反射マッピングなどのパーピクセル単位で内積計算が必要な処理をハードウェア化して高速に処理することができるようになった。
パーピクセルシェーディング機能は、広く普及している 2 種類のグラフィックス API の 1 つである DirectX のバージョン 8 においてすでにサポートされている。もう 1 つのグラフィックス API である OpenGL では、近い将来[1]に勧告されるバージョン 2.0 でこのパーピクセルシェーディング機能を正式にサポートする。

3 3D イメージワーピング法

ここでは、本研究の対象とするレリーフテクスチャマッピングとその背景にある 3 次元イメージワーピングについての詳細を説明する。

3.1 3 次元イメージワーピング方程式

3 次元イメージワーピング方程式 [6] は、参照元となるデプス値を関連付けたソース画像を利用して、任意の視点から見たパースペクティブ補正画像の構築をピクセル単位で計算する手法である。任意の視点位置に構築された画像はターゲット画像と呼ばれる。図 1において、$\dot{x}$ は $\dot{x} = \dot{C}_s + (\vec{c} + u_s\vec{a} + v_s\vec{b})t_s(u_s, v_s)$ と表現することができる。ここで $t_s(u_s, v_s) = |\dot{x} - \dot{C}_s|/|\vec{c} + u_s\vec{a} + v_s\vec{b}|$ である。任意のターゲット画像面への $\dot{x}$ の写像は以下のようにして求めることができる (図 2)。

$$\vec{x}_t \doteq P_t^{-1}P_s\vec{x}_s + P_t^{-1}(\dot{C}_s - \dot{C}_t)]\delta_s(u_s, v_s) \quad (1)$$

$\doteq$ は投影同一性を示す。ここで $\delta_s(u_s, v_s) = 1/t_s(u_s, v_s)$ である。式 1は **3D イメージワーピング方程式**と呼ばれる [6]。

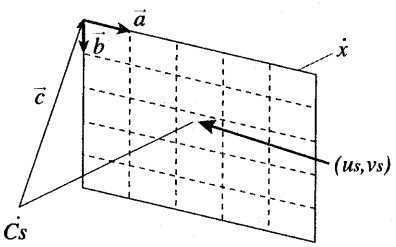

図 1: パースペクティブ投影カメラモデル

[1]2003 年 4 月頃

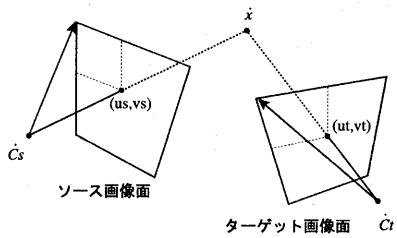

図 2: ソース画像面とターゲット画像面での $\dot{x}$ の投影位置

3.2 プリワーピング方程式

Oliveira[8] は上記の 3 次元ワーピング方程式をプリワーピング方程式部とテクスチャマッピング部の 2 つに分解した。
レリーフテクスチャマッピングはプリワーピング方程式によりソース画像面にプリワープ画像を生成し、それをテクスチャマッピングでターゲット画像面に写像することにより実現される (図 3,4)。各ピクセルに対応する奥行き値を平面からの距離と考えると、これは平面に垂直になるのでソース画像面を平行投影カメラモデルで表現することができる。(図 5)。

プリワープ画像は、図 3, 図 6よりソース画像面での移動先 (u_i, v_i) を求めることで生成することができる [8]。[8] より (u_i, v_i) と (u_s, v_s) には以下の関係が成り立つ。

$$u_i = \frac{u_s + k_1 displ(u_s, v_s)}{1 + k_3 displ(u_s, v_s)} \tag{2}$$

$$v_i = \frac{v_s + k_2 displ(u_s, v_s)}{1 + k_3 displ(u_s, v_s)} \tag{3}$$

ここで $k_1 = \frac{\vec{f}\cdot(\vec{b}\times\vec{c})}{\vec{a}\cdot(\vec{b}\times\vec{c})}$, $k_2 = \frac{\vec{f}\cdot(\vec{c}\times\vec{a})}{\vec{b}\cdot(\vec{c}\times\vec{a})}$, $k_3 = \frac{\vec{f}\cdot(\vec{a}\times\vec{b})}{\vec{c}\cdot(\vec{a}\times\vec{b})}$ である。式 2,3はレリーフテクスチャマッピングのための**プリワーピング方程式**と呼ばれる。

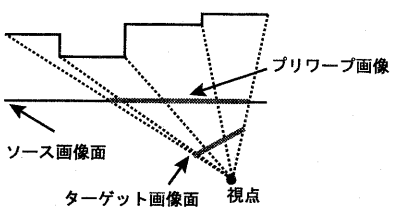

図 3: プリワーピング画像をソースイメージ面上で作成し、テクスチャマッピングによりターゲットイメージ面へ投影する

図 4: **左** ソース画像、**中央** プリワープ画像、**右** 矩形ポリゴンへのテクスチャマッピング

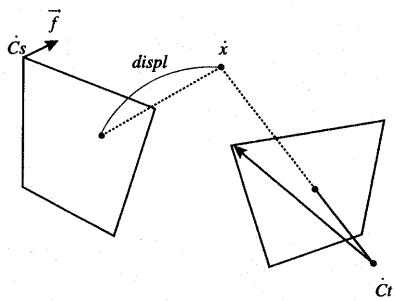

図 5: ソース画像面を平行投影カメラモデルで表現

3.3 レリーフテクスチャマッピングによるオブジェクトの表現

レリーフテクスチャを立方体で構成することによりオブジェクトを表現することができる (図 7,8)。

4 レリーフテクスチャマッピングへのパーピクセルシェーディングの応用

ここではパーピクセルシェーディングを機能を利用して、レリーフテクスチャマッピング処理の一部を置き換える方法、およびピクセル単位でのシェーディング計算を行う手法を説明する。

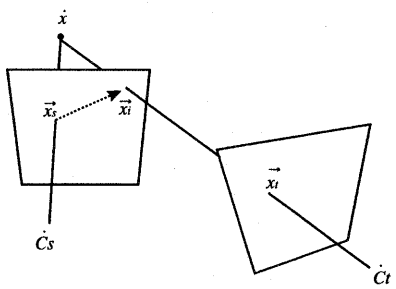

図 6: ソース画像面での $\vec{x}_s$ における $\dot{x}$ の移動先 $\vec{x}_i$

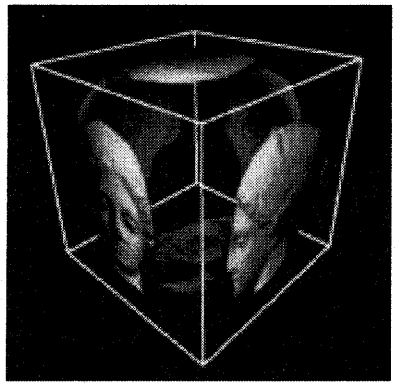

図 7: レリーフテクスチャ +6 面体

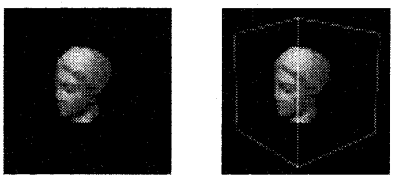

図 8: レリーフテクスチャマッピングにより表現されたオブジェクト

4.1 手順

本手法により、パーピクセルシェーディング機能を使用したレリーフテクスチャマッピングは以下の手順で行う。

1. 視点を設定し、定数テーブルを前計算する
2. ソース画像の各ピクセル位置 (u_s, v_s) 対して以下により**オフセットマップ**を生成する
   (a) プリワーピング先位置 (u_t, v_t) をプリワーピング方程式により計算する
   (b) プリワーピング先位置にソース画像の位置 (u_s, v_s) を **RGB 値**として書き込む
   (c) ターゲット画像の複数のピクセルを占める場合はサンプリング補間を行う
3. テクスチャシェーダに **OFFSET_TEXTURE_2D** を設定し、オフセットマップを参照先、ソース画像を適用先にして矩形ポリゴンをレンダリングする

4.2 オフセットマップの生成

テクスチャシェーダの機能の 1 つにオフセット・テクスチャ 2D(**OFFSET_TEXTURE_2D**) がある。これはテクスチャ座標を、テクセルレベルで摂動することができる機能である。
レリーフテクスチャマッピングのプリワープ画像はこのオフセットテクスチャへのオフセット値として表現することができる (図 9)。プリワープ画像を生成する際にソース画像のピクセルをコピーして動的にプリワープ画像テクスチャを再生成するのではなく、位置情報を RGB 値として書き込むことでオフセットマップを生成する。
テクスチャレンダリング時に **OFFSET_TEXTURE_2D** 機能を用いることでワーピング画像の生成 (図 10) とテクスチャマッピングを一度に行うことができる。
このオフセットマップはピクセル値が元のソース画像のどのテクセル位置に対応しているのかを表現しているので、一度オフセットマップが計算できれば後述の法線マップなどの他のソース画像にも適用させることができる (図 11)。

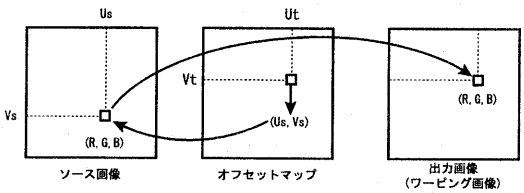

図 9: オフセット・テクスチャ 2D を使用したワーピング画像の生成

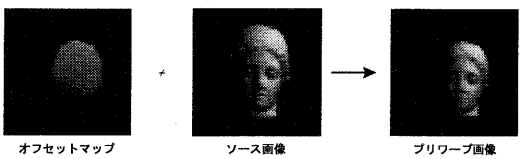

図 10: オフセットマップを使用したソース画像からのプリワープ画像の生成

4.3 法線マップの使用

レリーフテクスチャには曲面の表面色の情報だけしか含んでいないため、レリーフテクスチャのみではディフューズシェーディングまでしか表現することができない。スペキュラーやライティング、反射マッピングなどのより複雑なシェーディングを行うには曲面の法線情報が必要になる。
しかしレリーフテクスチャは画像であるため、レンダリング中に法線を算出することはできない。そこで、法線情報も画像であらかじめ用意することにより法

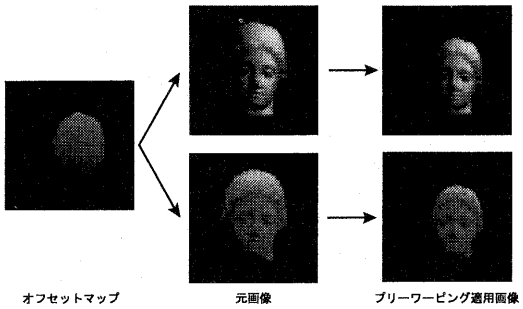

図 11: 異なるソース画像へのオフセットマップの適用

線を必要とするシェーディング計算を行うことができる。
法線マップは法線ベクトルの x, y, z 値を RGB 値として表現させたテクスチャである。(図 12)。レリーフテクスチャマッピングにこの法線マップを組み合わせることにより、以下で述べるパーピクセル単位での反射マッピングなどのシェーディング演算を行うことが可能になる。
法線マップは BMRT レンダリングツール [1] を用いて、サーフェスの法線ベクトルを RGB 値に変換するシェーダを作成して取得した。

図 12: 法線マップ

4.4 反射マッピング

テクスチャシェーダの機能の一つである固定視点内積キューブマップ **DOT_PRODUCT_CONSTANT_EYE_REFLECT_CUBE_MAP** を用いることで、ピクセル単位での反射マッピングをで行うことができる (図 13)。固定視点内積キューブマップは固定視点 (視点ベクトルが一定) でキューブマップテクスチャによる反射マッピングを行う機能である。キューブマップテクスチャは反射として映りこむ環境を立方体の 6 枚の画像として表現したテクスチャである (図 14)。

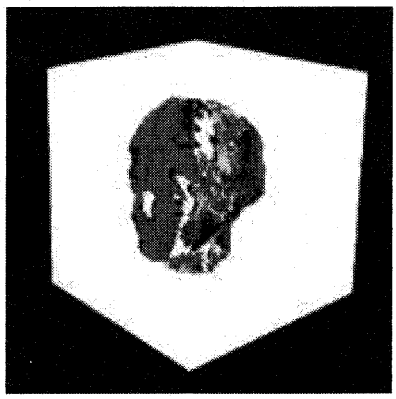

図 13: 固定視線によるパーピクセル反射マッピング

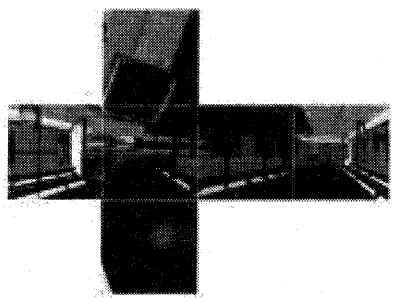

図 14: キューブマップテクスチャ (NVIDIA OpenGL SDK より引用)

5 結果・考察

本手法の適用時およびソフトウェア実装時のレンダリングパフォーマンスは表 (1) の通りである。数値は 256x256 ピクセルのレリーフテクスチャ 2 枚のレンダリングを、PentiumIII 1GHz、GeForce3 グラフィックスカードを搭載したマシンで計測したものである。
通常時ではソフトウェアでの実装とほぼ変わらないが、反射マップレンダリング使用時には、本手法がソフトウェアによる実装よりも約 25% 高速にレンダリングすることができた。

5.1 問題点

5.1.1 スクリーン空間でのデプス表現の問題点

レリーフテクスチャマッピングは最終的にはテクスチャマップ付き平面ポリゴンのレンダリングとなるのでデプスバッファには平面ポリゴンのデプスが記録される。そのため既存のポリゴンベースのシーンと合成するときに不整合が生じてしまう。レリーフテクスチャのスクリーン空間でのデプスを正確に表現するためにはピクセル単位でデプス値の変更を行わなければならない。
テクスチャシェーダの**GL_DOT_PRODUCT_DEPTH_OFFSET**プログラムでは、フラグメント単位での内積計算を元にしてデプス値を変移させることができる。この機能を使用することでスクリーン空間での正確なデプスの表現ができると思われる。

シェーディング	手法	フレーム / 秒
通常	ソフトゥエア	14.0
通常	本手法	13.0
反射マッピング	ソフトウェア	8.3
反射マッピング	本手法	10.4

表 1: レンダリングパフォーマンス (256x256 レリーフテクスチャ 2 枚のレンダリング)

5.1.2 レンダリング速度

現在の本手法の実装では、通常レンダリング時にはソフトウェアレンダリングとあまりレンダリング速度が変わらない。プリワーピング方程式およびリサンプリング処理をパーピクセルシェーディング機能で実装できればさらなる高速化を得ることができる。

6 結論・展望

レリーフテクスチャマッピングと各種イメージマッピング手法を組み合わせることで高品質なシェーディング画像が得られることを提示した。本手法を用いることによって、一度オフセットマップの計算ができればあとは非常に効率よくテクスチャベースのシェーディング計算を行うことができる。
マルチパスレンダリングで複数のテクスチャシェーダを実行すれば、複雑で精細なシェーディング処理をレリーフテクスチャマッピングに付加することができる。

またプリワーピング方程式をパーピクセルシェーディング機能を用いてハードウェア化することによりさらなる高速化が期待できる。

パーピクセルシェーディング機能の登場によりこれからのリアルタイムグラフィックスはさらに高品質なシェーディング処理が可能になって行くことが期待できる。一定時間で処理が可能で、かつ高品質なシェーディング結果が可能な本手法を用いることにより最適なポリゴン数で、最適なシェーディング画像を作成することが可能になると期待できる。

謝辞

本論文を作成するにあたり慶応義塾大学環境情報学部の金井先生に多大なる助言および文章の校正をしていただきました。また金井研究会の皆様からも貴重な助言を頂きました。ここに感謝の意を示します。

参考文献

[1] Bmrt: Blue moon rendering tools. http://www.exluna.com/.

[2] James F. Blinn. Simulation of wrinkled surfaces. In *Computer Graphics (SIGGRAPH '78 Proceedings)*, volume 12, pages 286–292, 1978.

[3] Shenchang Eric Chen. QuickTime VR — an image-based approach to virtual environment navigation. *Computer Graphics*, 29(Annual Conference Series):29–38, 1995.

[4] R. Cook. Shade trees. In *Computer Graphics (SIGGRAPH '78 Proceedings)*, volume 18, pages 223–231, 1984.

[5] A. Lippman. Movie-maps: An application of the optical videodisc to computer graphics. volume 14, pages 32–42, 1980.

[6] Leonard McMillan. An image-based approach to three-dimensional computer graphics. Technical Report TR97-013, 19, 1997.

[7] NVIDIA. Texture shaders , presentation slide. *Game Developers Conference*, 2001. http://www.nvidia.com/.

[8] Manuel M Oliveira. Relief texture mapping. Technical Report TR00-009, UNC Computer Science, March 2000.