2003－ＣＧ－113　(12)
2003／11／25

プログラマブルなグラフィクスハードウェアを用いたインテグラルフォトグラフィ画像のレンダリング

小池 崇文*
E-mail: koike@sdl.hitachi.co.jp

概 要

近年、裸眼立体視ディスプレイとして、インテグラルフォトグラフィ法が注目されている。インテグラルフォトグラフィ法は光線を忠実に再現する方式で、任意視点からの正しい立体像を再現できることで知られている。しかし、任意視点からの立体像を再現するために、非常に多視点からのレンダリングを行う必要がある。そこで、本研究では、従来の Z-Buffer 法を用いたインテグラルフォトグラフィ画像レンダリングにおける問題点を指摘し、レイトレーシングを用いてインテグラルフォトグラフィ画像レンダリングをハードウェアで実装する方法を検討した。

Integral Photography Rendering on Programmable Graphics Hardware

Takafumi Koike†
E-mail: koike@sdl.hitachi.co.jp

abstract

Today Integral Photography (IP) method receives much attention for an autostereoscopic display. IP method reproduces true rays and it is known that its method reproduces three-dimensional images from any points of view. But to realize autostereoscopic images, we need to render images from many other viewpoints. We point problems of a IP rendering method using Z-Buffer methods out and study a ray tracing IP rendering method on programmable graphics hardware.

*日立製作所システム開発研究所
†Hitachi, Ltd., Systems Development Laboratory

1　はじめに

近年、眼鏡等の特殊装置を身につけずに、立体が実現可能な、裸眼立体視ディスプレイの開発が盛んであり、試作品や製品が発表されている。しかし、こうしたディスプレイの多くでは、その方式として多眼方式を用いており、横方向の視差をある程度可能にするだけで、理想的な裸眼立体視ディスプレイと呼ぶには程遠い。理想時な裸眼立体視方式としてフォログラフィが長年、研究されているが、光波の忠実な再現の難しさのために、実用レベルまで達していないのが現状である。一方、もう一つの理想的な裸眼立体視方式としてインテグラルフォトグラフィ法 (以下、IP) がある [5]。IP は、90 年以上前に発表された方式であるが、高解像度のディスプレイや高解像度の記録素子が必要であり、長年、発展することがなかった方式である。

しかし、近年の画像表示装置の急速な発展により、IP を用いた裸眼立体視ディスプレイが急速に現実的な手法になりつつある。例えば、高解像度投影可能なプロジェクタを複数台用い、動画の立体像表示を可能にした IP ディスプレイも実現されている [3]。また、近年の液晶ディスプレイの高解像度化により、より簡易な装置で IP ディスプレイの実現が可能になりつつある。

撮影についても研究されている [7] が、高解像度撮影素子の開発に依存し、現状では、表示装置に比べて開発が進んでいない。そのため、コンピューターグラフィックス (CG) によって作成された IP 画像が用いられるが、IP は、その原理から超多眼方式とも呼ばれ、数十〜数百視点における画像のレンダリングが必要であり、従来のグラフィクスカードが得意とする、Z-Buffer を用いた実装は現実的ではない。

一方、分散コンピューティングを用いたレイトレーシングの高速化や、既存のプログラマブルグラフィクスハードウェア (または、グラフィクスプロセッシングユニット、GPU) を用いたレイトレーシングを実装する手法 [8] も提案されており、リアルタイムなレイトレーシングが現実的になりつつある。また、GPU 上では、浮動小数や条件分岐がサポートされつつあり、GPU の利用分野が模索されつつある。

そこで、本稿では、IP の原理と仕組みについて簡単に解説し、Z-Buffer を用いて IP 画像を生成する際の問題点と、レイトレーシングを適用した IP 画像の生成について考察する。

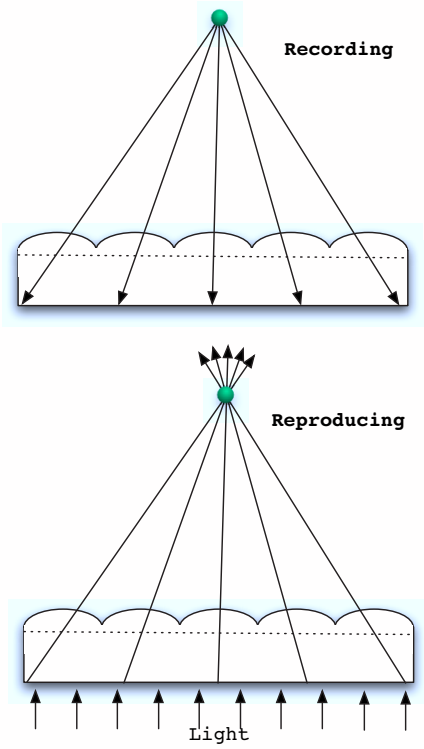

図 1: インテグラルフォトグラフィ法の原理

2　インテグラルフォトグラフィ法

IP は、数 100μm〜数 mm 程度のマイクロレンズをアレイ状に生成した、マイクロレンズアレイを通して実像を記録し、再生する方式である。通常、撮像部分は、CCD やフィルムで構成され、撮像部分の解像度が十分細かい場合には、非常に多くの方向からの光線を記録・撮影することが可能となり、実像を、マイクロレンズアレイを通して記録し、逆から光を当てることにより、立体像を再生する (図 1)。光線を記録再生する方式は、レイトレーシングそのものであり、CG を作成する場合も、レイトレーシングを用いることにより、容易に実現可能なことが理解できる。

図 2 に IP の元画像を示す。この IP 画像をマイクロレンズアレイを通して見ることにより、立体像の観察が可能となる。また、図 3 に複数視点から観察したときの画像を示す。写真の IP 装置は、解像度が低いため、少しわかりずらいが、文字 Test の e の穴の部分の奥に見える画像に差異があるのが確認できる。今、マイクロレンズをピンホールとして近似するか、レンズ性能を現す焦点距離を用いて正確にレ

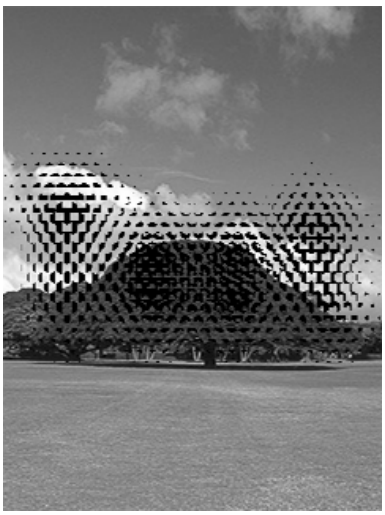

図 2: IP 元画像

図 3: 視点の移動による IP 立体像の変化

ンズの屈折を計算するかのどちらかが考えられるが、大きな違いは生じず、図 2 に示した画像はピンホールとして計算している。

以下に、IP を実現するために必要な構成要素について簡単に説明する。

1. 光線方向制御装置
   原理上、マイクロレンズアレイを用いるのが理想的であるが、ピンホールアレイが、その製作の容易さより、原理実験として用いられることが多い。ただし、ピンホールアレイは、輝度が落ち、ブラックマスク部分が多く感じられる欠点がある。一方、マイクロレンズアレイは製法が難しく、欠陥のないレンズアレイを作成するのが難しかったが、実用になりうるレンズアレイの製作が可能となってきている。ただし、その製法から、レンズはデルタ配列を持つものが多い。

2. 画像描画装置
   実用的な IP には、200〜300ppi 以上の解像度が必要であるが、300ppi 程度であればプロジェクタにより実現可能であることが確認されている。また、携帯電話や、ハイエンドに使用される液晶ディスプレイは 200ppi 前後であり、さらなる高解像度のディスプレイが開発されつつある。

3 Z-Buffer 法でのレンダリング

まずは、現在、リアルタイム描画でよく使われ、グラフィクスハードウェアとの相性も良い、Z-Buffer 法を用いての IP 画像のレンダリング方法について述べ、Z-Buffer 法には問題点があることを指摘する。以下では、簡単のために、ポリゴンのみをレンダリングオブジェクトとして扱う。

3.1 マイクロレンズ毎レンダリング

IP をその原理通りに解釈すると、各マイクロレンズの下にある画素をひとつの射影面とみなせ、そのマイクロレンズ毎にカメラパラメータを変更し、レンダリングする方法が考えられる。マイクロレンズ中心を光軸中心とし、中心位置にカメラを置くことにより、レンダリングが可能である。

この場合、1 フレームをレンダリングする際に、マイクロレンズ個数分のカメラパラメータ変更が生じる。マイクロレンズの個数を横 N_x、縦 N_y 個とした時に、マイクロレンズ総数 N は、$N = N_x \times N_y$ で、視点数を横 M_x、縦 M_y 個とした時に、総視点数 M は、$M = M_x \times M_y$ となる。よって、マイクロレンズ毎のレンダリングでは、画素数 M のレンダリングを N 回行う必要がある。ただし、GPU は大量のデータを処理する一種のストリームプロセッサであるため、定数にあたるレンダリングステートの変更を少なくする必要がある。しかし、カメラパラメータの変更はレンダリングステートの変更にあたり、通常 $N \gg M$ であるので、非常に多数回のレンダリングステートの変更がおこり、計算速度の低下が予想さ

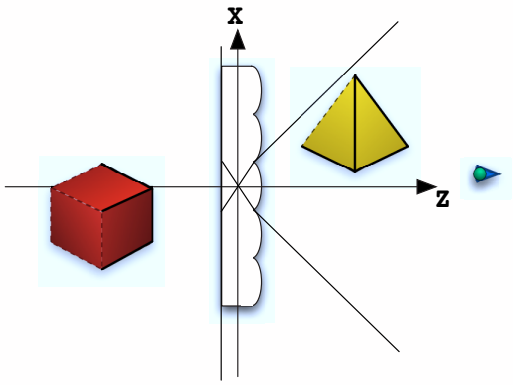

図 4:

れる。

図 4 には、2 つのオブジェクト (四面体と立方体) があり、この 2 つを描画する場合を考える。オブジェクトの実線部分をレンダリングする。視点毎のレンダリング結果を合成し、1 枚の IP 画像を作成するには、2 つの方法がある。1 つ目は、ステンシルバッファで各マイクロレンズをマスクしレンダリングする方法で、2 つ目は、複数のテクスチャにレンダリングし、テクスチャルックアップにより合成する方法である。今、ステンシルバッファを用いたレンダリング方法について実際のアルゴリズムを示す。

1. ステンシルバッファで描画領域をマスク
   デルタ配列のレンズアレイでは、各マイクロレンズ毎にステンシルバッファを書き込みマスクする。

2. カメラパラメータ設定
   カメラ位置の設定を行う。

3. 後面レンダリング
   Z-Buffer を用いてラスタライズを行う。図 4 では、立方体の実線部分をレンダリングすることとなる。

4. Z-Buffer のクリア
   いま、スクリーン前面は、スクリーン後面より必ず前にあるので、後面のレンダリングが終了した時点で Z-Buffer をクリアすることにより、前面のレンダリングを問題なく行うことが可能である。

5. カメラパラメータ再設定
   まず、カメラの向きの反転を行う。次に、Z テストを通常の小さい値なら真の代わりに、大きい値なら真として設定する。さらに、射影行列に変更を加える。通常の射影行列を P とすると、レンズ中心に対しての XY 面で反転するので、新しい射影行列 P' は、下記であらわせる。

$$P' = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times P$$

6. 前面レンダリング
   変更したカメラパラメータと射影行列により、スクリーン面前面にある部分をレンダリングする。図 4 における、四面体の実線部分をレンダリングが可能となる。

3.1.1 スクリーン面をまたぐレンダリング

IP ではスクリーン面の前後に像を結像することが可能であるが、IP 画像を生成するには、前記のように、スクリーン面の前後での二度のレンダリングが必要である。

しかし、Z-Buffer 方式等の射影を行うレンダリングでは、近平面でクリッピングを行う必要があるため、近平面より視平面に近いポリゴンのレンダリングが出来ない。さらに、視平面をまたぐポリゴンのレンダリングの際には、視平面による分割の問題が生じる [1]。視平面による分割の問題を回避するには、REYES で実装されているような sub-pixel dicing [2] を行う必要がある。ただし、sub-pixel dicing は、現状のグラフィクスハードウェア上で実装するのが困難であるので、実用的な方法ではない。

3.2 観察者視点毎レンダリング

次に、IP 画像を超多視点の立体画像と解釈し、各視点における画像をレンダリングし、それらの画像を各マイクロレンズの関連画素に配置する方法について考察する。

通常の裸眼立体ディスプレイは、2 眼式、4 眼式等と呼ばれるように、多視点の画素をレンダリングし、レンズやスリット等の配置に従うように、レンズ等の下に画素を再配置して多眼画像を生成する。同様に、視点毎にレンズ個数 N の画素数でレンダリングし、各マイクロレンズの対応する位置に画素を配置

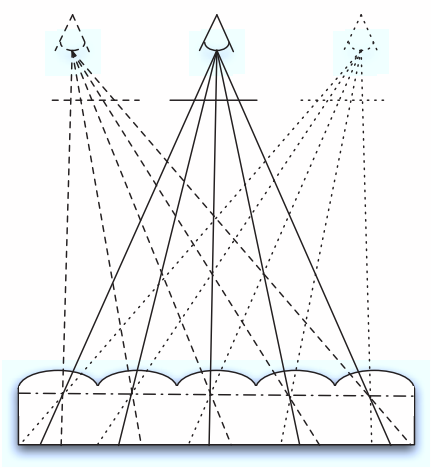

図 5: 複数視点画像からのサンプリング

(サンプリング) する (図 5)。よって、視点毎のレンダリングでは、画素数 N のレンダリングを M 回行う必要がある。しかし、今、デルタ配列のマイクロレンズアレイを用いた場合には、横方向に関して 2 倍、縦方向に関しては $\sqrt{3}/2$ 倍の解像度が必要であるので、最低でも、$2 \times \sqrt{3}/2 = \sqrt{3} \simeq 1.73$ 倍の解像度でレンダリングしサンプリングする必要がある。画素の配置には、ピクセルシェーダが適用可能に思われるが、現在の GPU では、同時に使用できるテクスチャ数は 16 個程度である。また、ピクセル座標を用いたサンプリングテクスチャの使い分けには、ピクセルシェーダ 3_0 以降を使用する必要があり、GPU だけで処理を行うには、まだ現実的な手法でないと考えられる。

4 レイトレーシング

前記議論により、レイトレーシングが IP 画像の生成にもっとも適している。レイトレーシングは、画素毎に光線を追跡し陰面消去を実施する手法であり、通常は、隣り合う画素間の光線には、図 6 上にしめすような相関がある。一方、IP 画像では、隣り合う画素間の光線には、図 6 下にしめすような相関を持つ光線を生成するだけでよい。

4.1 光線生成

まずは、画素毎の光線方向をあらわすためのレイバッファを作成する。各画素は、それを通るマイクロレンズを特定することが可能であるので、1) 各画素の担当するマイクロレンズを特定し、2) マイクロレンズを通った光線を生成しバッファに格納、することでレイバッファを生成する。

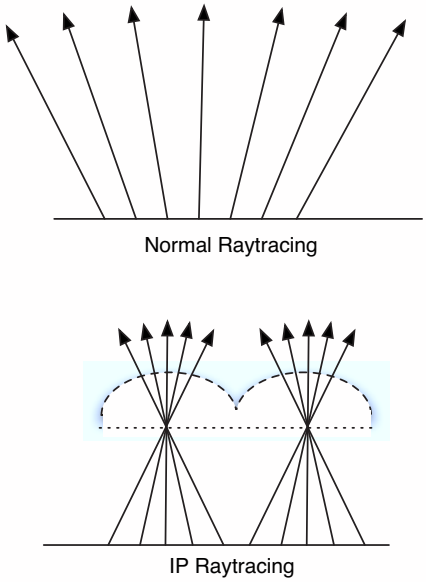

図 6: 光線相関

実際のレイトレーシングの際には、スクリーン面を挟んでの計算になるが、今、IP では、物理的制約により、立体描画可能領域がある程度制限される (図 7)。そこで、光線追跡の始点 $\mathbf{p}$ を立体描画領域の外に移動し、光線方向 $\mathbf{t}$(単位ベクトル) を反転する必要がある。スクリーン面から立体描画可能領域への距離を d とし、光線として下記変換を行う。

$$\mathbf{t}' = -\mathbf{t}$$

$$\mathbf{p}' = \mathbf{p} + d \cdot \mathbf{t}' = \mathbf{p} - d \cdot \mathbf{t}$$

以上の処理は、全てグラフィクスハードウェア上で実現することが出来る。GPU 上で動作するプログラムを記述するには、cg(c for graphics) [4] があるが、実装には、ほぼ同じ言語仕様である、HLSL(High-Level Shader Language) [6] を用いた。レイトレーシングは、1) 視点生成, 2) 光線追跡, 3) 衝突判定, 4) シェーディングの 4 つの部分で構成されるが、IP では 1) の視点生成部分にのみ変更を加えてやることで、他の処理は既存のアルゴリズムが容易に適用可能である。

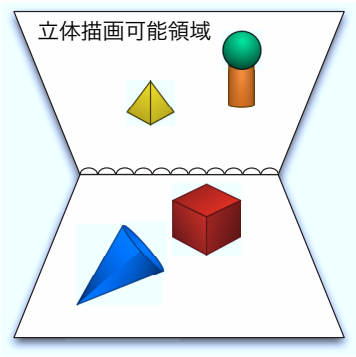

図 7: 立体描画可能領域

4.2 実装方法

光線の生成には各画素毎に頂点に割り当てて、頂点シェーダを適用する。それ以降の処理にはピクセルシェーダを使用し、Timothy 提案の手法 [8] で実装可能である。ポリゴンデータを全て三角形としてあらわし、頂点情報を 1 次元の三角形頂点テクスチャとして格納する。また、空間はボクセルに分割し、各ボクセルが含む全三角形をリスト化し、三次元テクスチャ内に、三角形リストへのポインタを格納する。三角形のリストは、1 次元の三角形リストテクスチャに格納する。三角形リストテクスチャには、三角形の識別番号が順に並んでおり、識別番号を用いて任意の三角形頂点テクスチャにアクセスする。

図 2 の IP 画像を作成し、十分実用的な方法であることを確認した。

5 まとめ

本稿では、Z-Buffer を用いて IP 画像を生成する際の各種問題点について検討し、現状では、レイトレーシングが一番適した方法であることを示した。Z-Buffer を用いてのレンダリングが望ましいが、いくつかの問題点をクリアする必要があることも分かった。今後は、プログラマブルグラフィクスハードウェア上で浮動小数の実装が進み、また、条件分岐がサポートされつつあるので、より容易で簡潔な実装が可能となると考えられる。

参考文献

[1] A. A. Apodaca and L. Gritz. *Advanced RenderMan*. Morgan Kaufmann, 2000.

[2] R. L. Cook, L. Carpenter, and E. Catmull. The reyes image rendering architecture. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, number 4, pages 95–102, July 1987.

[3] H. Liao, M. Iwahara, N. Hata, I. Sakuma, T. Dohi, T. Koike, Y. Momoi, T. Minakawa, M. Yamasaki, F. Tajima, and H. Takeda. “High-resolution integral videography autostereoscopic display using multi-projector”. In *Proceedings of the Ninth International Display Workshops, IDW’02*, pages 1229–1232, 2002.

[4] W. R. Mark, R. S. Glanville, K. Akeley, and M. J. Kilgard. “Cg: A System for Programming Graphics Hardware in a C-like Language”. *ACM Transactions on Graphics*, 22(3):896–907, July 2003.

[5] M.G.Lippmann. “Epreuves reversibles donnant la sensation du relief”. In *J. de Phys*, volume 7, pages 821–825, 1908.

[6] Microsoft. *“DirectX 9.0 Manual”*, 2002.

[7] F. Okano, H. Hoshino, J. Arai, and I. Yuyama. “Real-time pickup method for a three-dimensional image based on integral photography”. *Applied Optics*, 36(7):1598–1603, 1997.

[8] T. J. Purcell, I. Buck, W. R. Mark, and P. Hanrahan. “Ray Tracing on Programmable Graphics Hardware”. *ACM Transactions on Graphics*, 21(3):703–712, 2002.