グラフィックスハードウェアを用いた光跡の高精度・高速レンダリング法

An Accurate, Fast Method Using Graphics Hardware for Rendering Shafts of Light

正会員　土橋宜典†,　山本　強†,　西田友是††

Yoshinori Dobashi†, Tsuyoshi Yamamoto† and Tomoyuki Nishita††

Abstract　An important element in enhancing the reality of computer graphics is the effect of atmospheric scattering. This effect creates shafts of light when atmospheric particles are illuminated. We have developed an accurate and fast method for displaying shafts of light produced by studio spotlights. To calculate the intensity of the light reaching the viewpoint, multiple semi-transparent planes are placed in front of the viewpoint. The intensities of these planes are set to the intensities of the light reaching the viewpoint. The shafts of light are displayed based on the intensities of the planes. These processes are accelerated by graphics hardware, using, in particular, color blending, texture mapping, shadow mapping, and multi-texturing functions. The proposed method renders the shafts of light very quickly, making it useful for designing lighting effects in sudios, on stages, and so on.

キーワード：光跡，リアルタイムレンダリング，グラフィックスハードウェア，大気の散乱効果

1. まえがき

近年，コンピュータグラフィックスを用いて，室内やステージ照明などの事前評価が行われるようになってきた．リアルな映像を作成するためには，ホコリなど空気中に浮遊する微粒子による光の散乱・吸収による効果が重要な役割を果たす．これらの効果を考慮することで，スポットライトなどにより生じる光跡を表示することが可能となり，多くの研究が行なわれている[8)11)~13)16)]．しかし，従来法の多くは画像生成に多くの計算時間を必要とする点が問題となっている．

一方，近年，グラフィックスハードウェアの高性能化と低価格化が進んでおり，これを利用したリアルな画像生成手法の開発が盛んに行なわれている[2)3)5)7)15)18)]．しかし，これまで，グラフィックスハードウェアを利用して，前述の光の散乱効果を考慮した画像生成手法は提案されていない．本稿では，グラフィックスハードウェアを利用して，スポットライトなどの点光源によって生じる光跡を高精度かつ高速に表示する手法を提案する．提案手法では，計算量の削減を行って処理を高速化するのではなく，従来，レイトレーシング法により行われていた計算と等価な計算をグラフィックスハードウェアを利用して行う．グラフィックスハードウェアは極めて高性能化しており，提案手法により大幅な処理時間の短縮が実現できる．このとき，光の散乱によって生じる光跡だけでなく，光源の光が物体によって遮られることにより生じる物体表面上の影に加えて，空間中に生じる影も考慮したリアルな画像を極めて高速に作成できる．

2. 光の散乱効果に関連した従来法

リアルな画像を生成するためには，実際の物理現象を忠実にシミュレートする必要がある．光跡を表示する場合は，空気中の微粒子による散乱光を視線に沿って積分する必要があり，多くの手法が提案されている[8)11)~14)16)19)]．しかし，従来法では，レイトレーシング法やスキャンライン法を利用しており，画像生成に多くの計算時間を必要とする．そのため，近年，グラフィックスハードウェアを活用した手法が提案されている．Stam はグラフィックスハードウェアの 3D テクスチャマッピングの機能を利用して，煙などのガス状物体の映像をリアルタイムで表示する手法を提案した[18)]．しかし，3D テクスチャマッピング機能は未だハイエンドなグラフィックスワー

†北海道大学 大学院 工学研究科
（〒 060-8628 札幌市北区北 13 条西 8 丁目, TEL 011-706-6530）
††東京大学 大学院 新領域創成科学研究科
（〒 113-0033 東京都文京区本郷 7-3-1, TEL 03-5841-4106）
†Graduate School of Engineering, Hokkaido University
(Kita-ku, Kita 13, Nishi 8, Sapporo 060-8628, Japan)
††Graduate School of Frontier Science, University of Tokyo
(7-3-1, Hongo, Bunkyo-ku, Tokyo 113-0033, Japan)

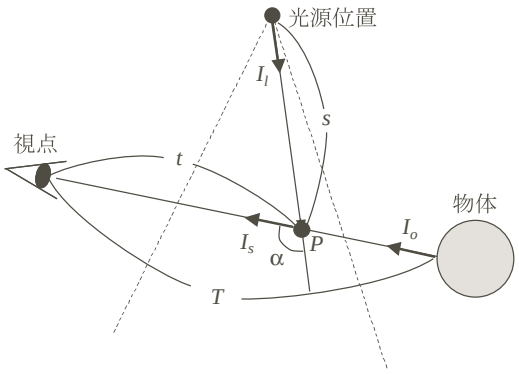


図 1 散乱光の計算.
Computation of atmospheric scattering.

クステーションのみでしか利用できず，また，この手法では煙の表示に主眼が置かれており，光跡の表示までは行っていない．また，筆者らは比較的安価なグラフィックスハードウェアを利用した光跡の表示手法を開発しているが，物体よって生じる影を精度よくサンプリングできず，エリアシング問題が生じ，表示精度の点で問題が残されている [3)4)]．本稿では，これらの問題を解決する手法を提案する．

3. 光跡表示のためのシェーディングモデル

本節では，大気中の微粒子による光の散乱を考慮した場合のシェーディングモデルについて説明する．

以下では，説明を簡単にするため，光源の数は 1 つとする．複数の光源が存在する場合，各々の光源について視点に届く散乱光を算出し，それらを足し合わせることで求めることができる．**図 1** に散乱光計算の考え方を示す．提案手法では，大気中の微粒子の密度は一定と仮定する．このとき，視点に到達する光の強さ I_v は次式で計算できる [13)]．

$$I_v = I_o \exp(-\rho\kappa T) + I_s \tag{1}$$

$$I_s = \int_0^T H(t) I_l(t) g(\alpha, s, t) dt \tag{2}$$

ここで，I_o は物体の輝度，ρ は微粒子の密度，κ は消散係数，T は物体と視点との距離，t は視点と視線上の一点 P との距離，s は光源と点 P の距離を表す．また，$H(t)$ は影の有無を表す関数で，点 P から光源が可視な場合は 1，そうでなければ 0 を返す．$I_l(t)$ は光源から点 P の方向へ発せられる光の強さ，g は光源の光 I_l が点 P に到達し，散乱して視点に届くまでの間に減衰する割合を表す関数である．いま，点光源を対象としているため，光源から発せられた光は微粒子によって減衰され，かつ，光源からの距離の 2 乗に比例して減衰する．すなわち，関数 g は次式で与えられる．

$$g(\alpha, s, t) = \rho F(\alpha) \exp(-\rho\kappa(s+t))/s^2 \tag{3}$$

ここで，$F(\alpha)$ は微粒子の位相関数を表し，α はその位相角である (図 1 参照)．位相関数は次式で与えられる [13)]．

$$F(\alpha) = K(1 + 9\cos^{16}(\alpha/2)) \tag{4}$$

ここで，K は散乱光強度を決定するための定数であり，ユーザにより指定される．

式 (1) における第一項は物体の色が減衰して視点に到達する成分であり，OpenGL などのグラフィックスライブラリの機能を利用することで計算できる [1)]．式 (1) の第二項，すなわち，式 (2) で表される I_s が散乱光成分であり，光跡の表示に直接関係する．以降，I_s のグラフィックスハードウェアを利用した計算方法について述べる．

4. グラフィックスハードウェアを用いた光跡の表示

まず，提案手法で利用するグラフィックスハードウェアの諸機能とそれらを利用した光跡の表示方法を述べる．

4.1 提案手法で利用するグラフィックスハードウェアの機能

主に利用する 4 つの機能，すなわち，アルファブレンディング，プロジェクティブテクスチャマッピング，シャドウマップ，マルチテクスチャリングについて簡単に解説する．これらの機能は標準的なグラフィックスライブラリの一つである OpenGL を用いて容易に利用できる．詳細については，参考文献 [1)] を参照していただきたい．

アルファブレンディング
フレームバッファ中に保存されている各画素の色と次に描画するポリゴンの色を混合する機能．混合の方法にいくつかの種類があるが，本稿では，単純に 2 つの色を足し合わせる機能を利用する．

プロジェクティブテクスチャマッピング
任意の点 (例えば，光源位置) からテクスチャを投影するようにマッピングする機能．投影するテクスチャに光源の配光特性に相当するものを用いれば，スポットライトなどによって照射された画像を生成できる．このとき，この配光特性に相当するテクスチャをライトマップと呼ぶ．

シャドウマップ
点光源 (または平行光源) による影を表示するための機能．まず，光源から見たときの奥行き画像を作成する．そして，視点から描画する際，各点の光源との距離と記憶しておいた奥行き画像の対応する画素値を比較して影かどうかの判定を行う．

マルチテクスチャリング
指定された複数のテクスチャに対して演算を施した結果生成されるテクスチャをマッピングする機能．演算方法にはいくつかあるが，本稿では，テクスチャ A の各画素とテクスチャ B の各画素を乗算して得られるテクスチャ $C(= A \times B)$ をマッピングする機能を利用する．

4.2 光跡のレンダリング

式 (2) で表される散乱光 I_s を解析的に計算すること

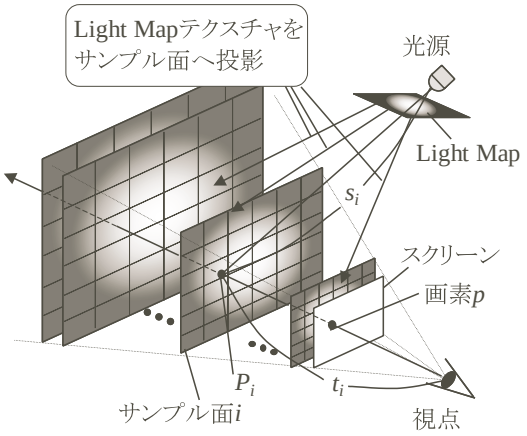


図 2 ハードウェアを利用した光跡のレンダリング.
Hardware-accelerated rendering of shafts of light.

は困難であるため，数値積分によって計算する．提案手法では，**図 2** に示すように，この数値積分を実行するための複数のサンプル面をスクリーンに平行に等間隔に配置する．そして，このサンプル面と視線との交点において，式 (2) の被積分関数の値を評価し，I_s を求める．いま，サンプル面の数を n とすると，画素 p に対する散乱光成分 $I_s(p)$ は次式により計算される．

$$I_s(p) = \sum_{i=1}^{n} H(t_i) I_l(t_i) g(\alpha_i, s_i, t_i) \Delta t \qquad (5)$$

ここで，図 2 に示すように，t_i は画素 p を通る視線とサンプル面 i との交点 P_i と視点との距離，α_i は点 P_i と光源とを結ぶベクトルと視線とのなす角 (点 P_i での位相角)，s_i は点 P_i と光源との距離，Δt は積分間隔を表す．上式から，光跡を表示するために，各サンプル面の視線との交点 P_i での H, I_l, g の値をグラフィックスハードウェアの機能を利用して求める．まず，影の有無を表す H の値については，各サンプル面に物体が落とす影を計算することで算出できる．これは，前節で述べたシャドウマップ法 [6] が利用できる．この方法では，リアルタイムで物体の影を計算でき，サンプル面上の任意の点での H の値を高速に求めることができる．次に，光源から発せられる光を表す I_l の値については，プロジェクティブテクスチャマッピング [17] の機能を利用する．まず，光源の配光特性をテクスチャとして記憶したライトマップ [1] を用意する．そして，ライトマップを光源の正面に配置してサンプル面に投影されるようにテクスチャマッピングを行う (図 2 参照)．これにより，サンプル面上の任意の点における I_l の値を算出できる．そして，減衰率を表す関数 g は，マルチテクスチャリングの機能を利用する．まず，関数 g を次式のように，2 つの関数 F および h の積で表現する．

$$g(\alpha, s, t) = F(\alpha) h(s, t) \qquad (6)$$
$$h(s, t) = \rho \exp(-\rho\kappa(s + t))/s^2$$

そして，これらの 2 つの関数について，前処理として，α および (s, t) を変化させてその関数値を計算したテーブルを作成する．このとき，F については，α を 0° から 180° までをサンプルして 1 次元のテーブルを作成する．また，h については，s および t の最大値 s_{max} および t_{max} をそれぞれユーザが指定して s, t に関する 2 次元のテーブルを作成する．そして，F についての 1 次元テーブルは 1 次元テクスチャとして，h についての 2 次元テーブルは 2 次元テクスチャとして使用し，マルチテクスチャリングの機能を用いてサンプル面へマッピングする．このとき，その関数パラメータである α および (s, t) をテクスチャ座標としてを指定する必要がある．そこで，各サンプル面をメッシュに分割し，そのメッシュの格子点において α および (s, t) を計算することでこれらのテクスチャ座標を与える．サンプル面上の任意の点のテクスチャ座標はその点を含むメッシュの格子点で与えられたテクスチャ座標からハードウェアにより自動的に補間して計算される．そして，補間されたテクスチャ座標をもとにテクスチャ要素 (すなわち，関数 F および h の値) がマッピングされる．このとき，マルチテクスチャリングの機能を利用し，これらの 2 つのテクスチャが乗算されるよう設定する．これにより，サンプル面上の任意の点の減衰項 g の値が計算される．

以上により，式 (5) における H, I_l, g のサンプル面上での値を計算できる．最後に，サンプル面を描画し，アルファ・ブレンディングの機能により，その色をフレームバッファに足しこむことで，式 (5) における総和 (Σ) の計算が行われ，光跡が表示できる．

4.3 サブサンプル面を用いたアンチエリアシング

前節の手法により光跡の表示が可能となるが，サンプル面数が少ない場合，物体の影や光源の配光特性を充分にサンプリングできず，エリアシング問題が生じる．本節では，この問題点とその解決手法について述べる．

（1） 問題点

一般に，配光特性や物体の影は急激な変化を伴うことが多い．したがって，エリアシングの少ない精度良い画像を作成するためには，物体の影を表す項 H および配光特性を表す項 I_l を精度良くサンプリングする必要がある．最も単純な方法はサンプル面を増加することであるが，この方法では，以下の 2 つの問題が生ずる．

・レンダリング時間の著しい増加．メッシュ数 × サンプル面数に比例して計算時間および各メッシュを描画するための時間が増加する．

・量子化誤差による画質の低下．減衰項 g の値をテクスチャとして記憶しているが，多くのグラフィックスハードウェアの性質上，その値は 8 ビットに量子化される．そのため，サンプル面数をあまり多くすると量子化誤差が蓄積され，画質が著しく低下する．

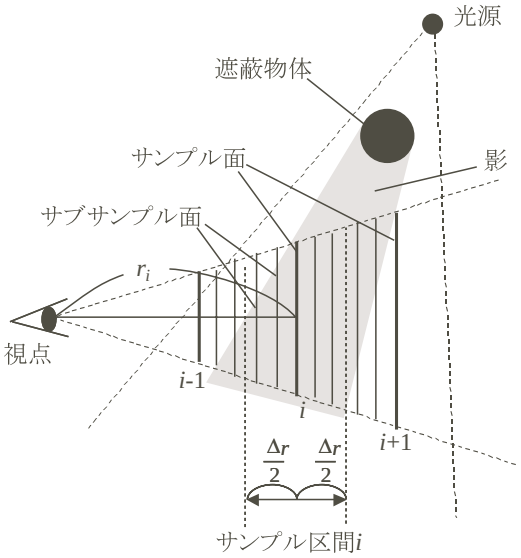


図 3 サブサンプル面を用いたアンチエリアシング.
Removing aliasing effects by using sub-sample planes.

（2） 解決法

配光特性 (I_l) および物体の影 (H) については，急激な変化を伴うため，多くのサンプル点数が必要である．一方，減衰項 (g) は，比較的滑らかな変化をするため，より少ないサンプル点数で充分である．このような性質を考慮し，H および I_l に対するサンプル点と g に対するサンプル点を異なった間隔でとる．

基本的な考え方は，**図 3** に示すように，サンプル面間にサブサンプル面を挿入し，これを用いて H と I_l をより精度よくサンプルするというものである．いま，サンプル面 i と視点との距離を r_i とし，サンプル面間の距離を Δr とする．このとき，サンプル面 i に対応するサンプル区間 i を $[r_i - \Delta r/2, r_i + \Delta r/2]$ と定義する (図 3 参照)．まず，サブサンプル面を用いて，サンプル面 i のサンプル区間内での，HI_l の値の平均値を計算し，その値と減衰項 g を掛け合わせる．例えば図 3 の場合，サンプル面 i のみでは，すべてが影，すなわち，サンプル区間 i 内の H の値は 0 と判定されてしまう．しかし，図 3 より，サンプル区間 i 内において，H の値が 1 である領域が存在する．そこで，サブサンプル面を用いて，H の値をより精度よく算出する．

サンプル面 i のサンプル区間内での HI_l の平均値を J_i で表すと，散乱光 I_s は次式により計算される．

$$I_s = \sum_{i=1}^{n} J_i g(t_i) \Delta t \tag{7}$$

ここで，サンプル面 i のサンプル区間内にサブサンプル面を m 枚発生させたとすると (図 3 では，$m = 5$)，J_i は次式で求められる．

$$J_i = \frac{1}{\Delta t} \int_{r_i - \Delta r/2}^{r_i + \Delta r/2} H(u) I_l(u) du \tag{8}$$

$$= \frac{1}{m} \sum_{j=1}^{m} H(u_j) I_l(u_j) \tag{9}$$

u_j は視点からサブサンプル面 j と視線との交点までの距離を表す．この平均値 J_i もグラフィックスハードウェアを利用して計算する．各サブサンプル面に対して，プロジェクティブテクスチャマッピング法とシャドウマッピング法を用いて，サブサンプル面上での $H(u_j)I_l(u_j)$ を計算する．そして，サブサンプル面を描画してアルファブレンディングの機能によりその色をフレームバッファに足しこめばよい．このとき，サブサンプル面は減衰項 g を表すテクスチャをマッピングする必要がないため，メッシュに分割する必要がない．そのため，高速に描画が行えるとともに，減衰項 g による量子化誤差を生じない．そして，その描画結果を新たにテクスチャとして利用し，減衰項 g を表すテクスチャと乗算したテクスチャをマルチテクスチャリングの機能を用いてサンプル面 i にマッピングする．この方法により，前節で述べた問題点による画質の低下を抑え，リアルな画像を作成できる．

5. 適 用 例

5.1 簡単な例による実験

簡単な例を用い，提案手法の妥当性の検討を行った．まず，サンプル面数と量子化誤差の関係を調べ，最適なサンプル面数の決定を行うため，単一のスポットライトのみが存在する場合について，サンプル面を 10 から 200 まで変化させ，画像生成を行った．4.2 節で述べたテクスチャ座標を指定するためのメッシュへの分割数は 15×10 である．計算機は CPU が PentiumIII 733 MHz であり，グラフィックスハードウェアとして NVIDIA 社の GeForce2GTS を搭載した PC を用いた．画像サイズは 720×480 である．作成した画像について，代表的な画素 10 点をサンプルし，その輝度値の真値との差の平均値を求めた結果を**図 4** に示す．ただし，視線上に 200 点のサンプル点を発生させて量子化を行わずに計算した値を真値とした．各サンプル面数にて画像を生成した際の計算時間も同じく図 4 に示す．図 4 において，横軸はサンプル面数である．実線で示すのが輝度差であり，縦軸 (左) は輝度レベル，また，点線は計算時間であり，縦軸 (右) は秒である．図 4 から，サンプル面数が 30 程度までは輝度差が減少し，それ以降は量子化誤差の影響により増加する．計算時間は，ほぼ線形に増加する．サンプル面数が 30 のときの計算時間は 0.05 秒と充分高速であるため，サンプル面数は 6 とした．

次に，サブサンプル面数を決定するため，単一のスポットライトに照らされた球の例について，サブサンプル面数を 1 から 25 まで変化させて画像を生成した．サブサンプル面数が 25 の場合を真の画像とし，サブサンプル面数が 1 から 24 の場合の画像との差分画像を求め，その 2 乗平均誤差を計算した．その結果を**図 5** に示す．

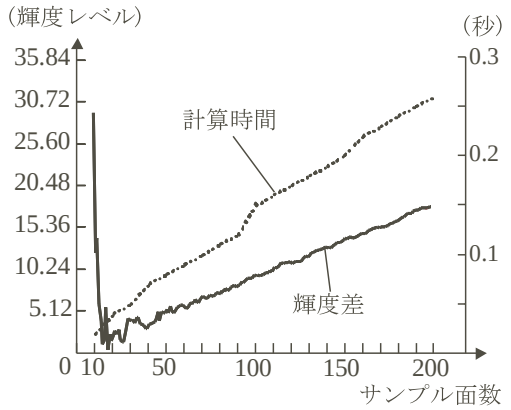


図 4 サブサンプル面数と量子化誤差・計算時間の関係.
Quantization errors and computation times of images rendered with various numbers of sample planes.

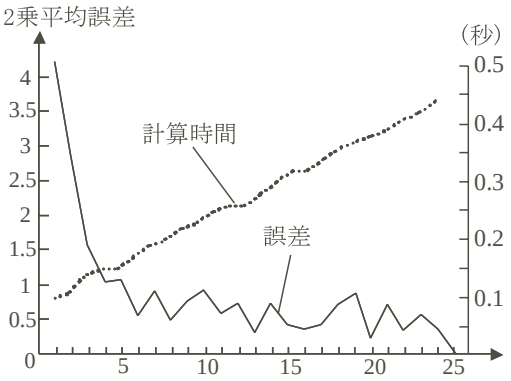
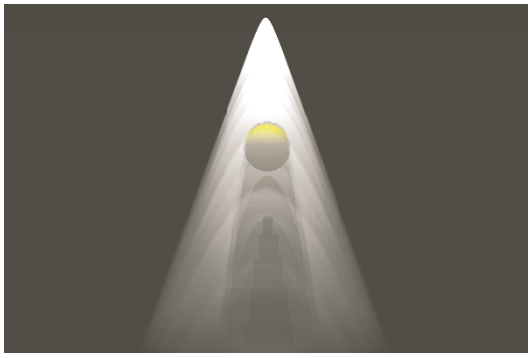
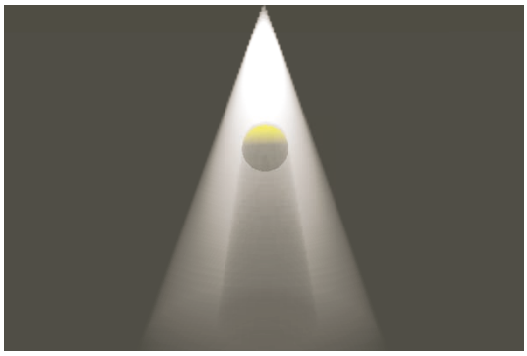


図 5 サブサンプル面数と表示精度・計算時間の関係.
RMS errors and computation times of images rendered with various numbers of sample planes.



a) サブサンプル面を用いない場合.



(b) サブサンプル面数を6に設定した場合.

図 6 サブサンプル面によりエリアシング除去した例.
Example of removing aliasing effects by using sub-sample planes.

図 5 では，各サブサンプル面数での画像生成にかかる時間も示している．図 5 から，サブサンプル面数が 6 以上では，誤差が 1.0 以下となっており，また，サブサンプル面数が 6 のときの計算時間は 0.17 秒と充分高速であるため，サブサンプル面数は 6 とした．**図 6**(a) は，サブサンプル面を用いない場合，図 6(b) はサブサンプル面数が 6 の場合の画像である．サブサンプル面により，エリアシングが軽減され，リアルな画像が生成できている．

次に，従来法との比較を行い，提案手法の有用性の検討を行う．従来法として，西田らの方法 [13] を実装し，図 6(b) と同じ画像を生成した．従来法では，レイトレーシング法を用い，視線上にサンプル点を発生させ，式 (1) を数値的に積分する [13]．計算量を同程度にするため，発生させるサンプル点数はサンプル面数とサブサンプル面数の和に等しくなるよう 180 点とした．各サンプル点における影の項 H は，サンプル点から光源方向にシャドウ・レイを発生させて計算した [13]．また，減衰項 g，光源の配光特性を表す項 I_l の値は，サンプル点と光源との位置関係より算出した．一方，提案手法では，影の項 H はシャドウ・マッピング法，減衰項 g および光源の配光特性 I_l はテクスチャマッピングの機能を用いている．従来法により生成した画像と提案手法により生成した画像の 2 乗平均誤差は 2.6%であった．ただし，256 レベルの誤差がある場合を 100%とした．提案手法では，シャドウマッピング法やテクスチャマッピングを利用しており，各マップの画素値は 8 ビットに量子化されて記憶されるために誤差が生じたと考えられる．しかし，視覚的にほとんど差が見られない程度の画質を得ることができた．計算時間は，従来法は 27 秒，提案手法は 0.17 秒であった．以上より，提案手法は従来法と同程度の画質の画像を約 1/60 の速度で生成できた．

次に，提案手法の CPU とグラフィックスハードウェアへの依存性を調査するため，異なるグラフィックスハードウェアや CPU 速度の異なる PC を用いて時間計測を行った．まず，CPU が PentiumIII 733MHz である PC を用いて，NVIDIA 社の GeForce256 および GeForce2GTS を用いた場合，また，グラフィックスハードウェアを利用せずソフトウェアにて実行した場合の計算時間を計測した．図 6(b) を生成するための計算時間は，GeForce256 の場合が 0.19 秒，GeForce2GTS の場合が 0.17 秒，ソフトウェアによる場合が 52 秒であった．GeForce256 と GeForce2GTS の間では大きな差はなかったが，CPU のみでは極めて多くの計算時間を必要とした．次に，グラフィックスハードウェアを GeForce2GTS とし，CPU を PentiumIII 600MHz に変更した場合を計測した．この

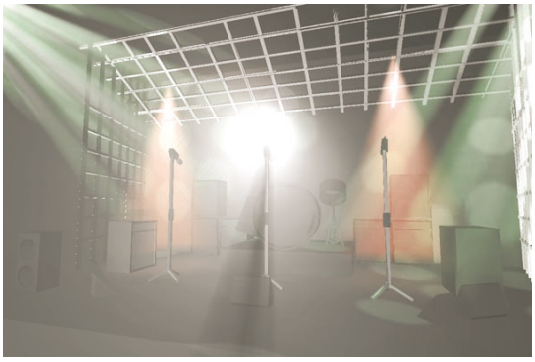


図 7　ステージ照明の表示例.
Example of displaying studio spotlights.

場合，0.17 秒であり，PentiumIII 733MHz の場合と同じ結果を得た．提案手法は，グラフィックスハードウェアの性能に大きく依存しており，グラフィックスハードウェアを活用することで大幅な高速化を実現できた．

5.2　ステージ照明の表示例

より現実的な例として，ステージ上の照明の表示例を**図 7** に示す．5 つのスポットライトを配置し，その配光特性には，水玉模様のパターンも使用している．サンプル面数およびサブサンプル面数は前節の議論より，それぞれ，30 および 6 とした．リアルな画像が生成されている．光源数が 5 つであるため，計算時間は 1.2 秒を要するが，インタラクティブなアプリケーションには充分高速であると考えられる．なお，提案手法では，光跡表示のための計算時間は光源数に比例する．この例では，物体の表示に 0.4 秒，光跡の表示に 0.8 秒要した．計算機および画像サイズは図 6(b) の場合と同様である．

6. ま と め

グラフィックスハードウェアを利用して，スポットライトによって生じる光跡を高精度かつ高速に表示する手法を提案した．本手法では，グラフィックスハードウェアを有効に活用しており，今後，グラフィックスハードウェアの進歩に伴い，その有効性はますます向上すると期待できる．提案手法は以下の利点を有する．

（1） 極めて高速に画像を生成することができる．
（2） 物体が光源の光を遮ることによって生じる空間中の影を考慮して光跡を表示することができる．
（3） サブサンプル面によって，影のサンプリング誤差によるエリアシングを低減したリアルな画像を作成できる．

今後の課題として，サンプル面数やサブサンプル面数の自動決定法の開発や大気中の微粒子の密度分布が一定でない場合での光跡の表示が挙げられる．

最後に，図 7 の物体の形状モデリングを行ってくれた情報科学芸術大学院大学の早乙女恵子さんに感謝します．

〔文　献〕

1）D. Blythe：“Advanced Graphics Programming Techniques Using OpenGL”，Course Note No. 29 of SIGGRAPH’99 (1999)
2）B. Cabral, M. Olano, P. Nemec：“Reflection Space Image Based Rendering”，Proc. SIGGRAPH’99, pp. 165-170 (Aug. 1999).
3）Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, T. Nishita：“A Simple, Efficient Method for Realistic Animation of Clouds”，Proc. SIGGRAPH2000, pp. 19-28 (Jul. 2000).
4）Y. Dobashi, T. Yamamoto, T. Nishita：“Interactive Rendering Method for Displaying Shafts of Light”，Proc. Pacific Graphics 2000, pp. 31-47 (Oct. 2000).
5）W. Heidrich, H. P. Seidel：“Realistic, Hardware-Accelerated Shading and Lighting”，Proc. SIGGRAPH’99, pp. 171-178 (1999).
6）W. Heidrich：“High-quality Shading and Lighting for Hardware-accelerated Rendering”，PhD thesis at University of Erlangen, pp. 51-54 (1999).
7）W. Heidrich, K. Daubert, J. Kautz, H. P. Seidel：“Illuminating Micro Geometry Based on Precomputed Visibility”，Proc. SIGGRAPH2000, pp. 455-464 (1999).
8）H. W. Jansen, P. H. Christensen：“Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps”，Proc. SIGGRAPH’98, pp. 311-320 (1998).
9）R. V. Klassen：“Modeling the Effect of the Atmosphere on Light”，ACM Trans. on Graphics, **6**, No.4, pp. 215-237 (1987).
10）K. Kaneda, T. Okamoto, E. Nakamae, T. Nishita：“Photorealistic Image Synthesis for Outdoor Scenery under Various Atmospheric Conditions”，The Visual Computer, **7**, 5&6, pp. 247-258 (1991).
11）N. Max：“Light Diffusion through Clouds and Haze”，Graphics and Image Processing, **13**, No. 3, pp. 280-292 (1986).
12）N. Max：“Atmospheric Illumination and Shadows”，Computer Graphics, **20**, No. 4, pp. 117-124 (1986).
13）T. Nishita, Y. Miyawaki, E. Nakamae：“A Shading Model for Atmospheric Scattering Considering Luminous Intensity Distribution of Light Sources”，Computer Graphics, **21**, No. 4, pp. 303-310 (1987).
14）T. Nishita, E. Nakamae：“Method of Displaying Optical Effects within Water using Accumulation Buffer”，Proc. SIGGRAPH’94, pp. 373-379 (1994).
15）E. Ofek, A. Rappoport：“Interactive Reflections on Curved Objects”，Proc. SIGGRAPH’98, pp. 333-342 (1998).
16）H. E. Rushmeier, K. E. Torrance：“The Zonal Method for Calculating Light Intensities in The Presence of a Participating Medium”，Computer Graphics, **21**, No. 4, pp. 293-302 (1987).
17）M. Segal, C. Korobkin, R. V. Widenfelt, J. Foran, P. E. Haeberli：“Fast Shadows and Lighting Effects Using Texture Mapping”，Computer Graphics, **26**, No. 2, pp. 249-252 (1992).
18）J. Stam, “Stable Fluids”，Proc. SIGGRAPH’99, pp. 121-128 (1999).
19）M. Watt：“Light-Water Interaction using Backward BeamTracing”，Computer Graphics, **24**, No. 4, pp. 377-376 (1990).

土橋　宜典（どばし　よしのり）1969 年生．1992 年，広島大学工学部卒業．1994 年，同大学大学院工学研究科博士課程前期修了．1997 年，同博士課程後期修了．同年，広島市立大学情報科学部助手．2000 年，北海道大学大学院工学研究科助教授．主として，コンピュータグラフィックスに関する研究に従事．博士 (工学).

山本　強（やまもと　つよし）1976 年，北海道大学工学部卒業．1978 年，同大学大学院工学研究科修士課程修了．1982 年 3 月同博士課程中退．1982 年同大学工学部講師．1986 年，同助教授．1999 年，同大学大学院工学研究科教授．工学博士．主として，データ放送，マルチメディア，コンピュータグラフィックス，並列処理に関する研究に従事．

西田　友是（にしだ　ともゆき）1971 年，広島大学工学部卒業．1973 年，同大学大学院工学研究科修了．同年，マツダ入社．1979 年，福山大学工学部講師．1984 年，同助教授．1990 年，同教授．1998 年，東京大学理学部教授．1999 年，同大学大学院新領域創成科学研究科教授．主として，コンピュータグラフィックスに関する研究に従事．工学博士．