グラフィクスとCAD　110−12
（２００３．２．１４）

# 天空光を考慮した都市景観画像の高速生成法

高田　信太郎†　土橋　宜典†　山本　強†

リアルな都市景観画像を作成するためには，太陽光だけではなく天空光によって照射された物体の輝度計算を行う必要がある．しかし，天空光による照度計算は非常に多くの計算時間を必要とする問題がある．本論文では，天空光による照度を高速に計算する手法を提案する．提案手法ではグラフィックスハードウェアを利用して画像生成を行い，LOD(Level of Detail) の考え方を利用することでポリゴンデータに対する計算コストの削減を図る．提案手法は特に都市景観画像のような大容量のポリゴンデータを扱う際に高速に画像を作成することができる．また，本手法のアニメーション作成への適用についても述べる．

## A Fast Method for Rendering Townscape Image with Skylight Illuminance

Shintaro Takada,† Yoshinori Dobashi†
and Tsuyoshi Yamamoto†

To create realistic townscape images, the skylight as well as the sunlight have to be taken into account to compute illuminance of objects. But it is extremely expensive to calculate illuminance by the skylight. We propose a fast method for rendering townscape images taking into account the skylight illuminance. In the proposed method, the rendering of images is accelerated by graphics hardware, and the calculation cost is reduced by using the idea of LOD(Level of Detail). The proposed method is efficient especially for rendering mass polygon data such as townscape. Furthermore, We extend our method to making animations.

## 1. はじめに

近年，コンピュータグラフィックス (CG) を用いた都市景観画像が作成されている．CG を用いることによって，天候や太陽位置に応じた建築物の見え方を事前に評価することができる．この際，リアルな都市景観画像を作成するためには，太陽光だけではなく天空光によって照射された物体の輝度計算をする必要がある．天空光とは，大気中の粒子に太陽光が当たって生じる周囲の天空からの散乱光であり (図 1 参照)，天空光による照度計算は非常に多くの計算時間を必要とする点が問題となっている．また，計算機性能の向上に伴い，複雑なシーンの描画をすることが多くなってきており，扱うポリゴンデータが爆発的に増加し，インタラクティブな表示が困難になってきている．

本論文では，天空光による照度を高速に計算する手法を提案する．近年，グラフィックスハードウェアは著しく高性能化しており，物体の輝度計算等を非常に高速に行うことが可能となっている．そこで，提案手法ではグラフィックスハードウェアを利用して画像生成を行う．また，提案手法では LOD(Level of Detail) の考え方を利用することで天空光による照度計算のコスト削減を

† 北海道大学大学院工学研究科
Graduate School of Engineering, Hokkaido University

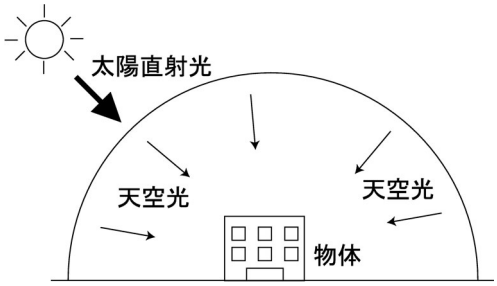


**図 1**　天空光照明モデル

図る．提案手法を用いることによって，都市景観画像のような大容量のポリゴンデータを扱う際にも天空光を考慮した画像を高速に作成することができる．

以下，2 節で天空光による照度計算のモデルを示し，3 節で提案手法による天空光を考慮した画像生成方法について述べる．4 節では本手法のアニメーションへの適用について述べ，その後 5 節で提案手法の適用例を示し，最後に 6 節で本論文のまとめを行う．

## 2. 天空光による照度計算のモデル

天空光は地上を覆う極めて大きな半径を持った半球状の光源と考えることができる[1)]．ただし，半球状光源の輝度分布は一様ではない．図 2 に示すように，天空を天空要素と呼ばれる小さな領域に分割する．それぞれの天空要素内では一定の輝度とする．光の逆二乗則を適用すると，天空要素 $P_e$ による計算点 $P$ の照度 $dE$ は次式で計算できる．

$$dE = H(\theta,\phi)L(\theta,\phi)\frac{\cos\gamma}{r^2}dA \qquad (1)$$

ここで，$L$ は天空要素の輝度，$\theta$ は天頂方向と線分 $PP_e$ とのなす角，$\phi$ は $x$ 軸から天空要素までの方位角，$\gamma$ は光源方向へのベクトルと計算点 $P$ の法線ベクトルがなす角，$r$ は $P$ と $P_e$ 間の距離，$dA$ は天空要素の面積である．また，$H(\theta,\phi)$ は影の有無を表す関数であり，点 $P$ から天空要素が可視であれば 1 を返し，そうでなければ 0 を返す．天空要素の面積 $dA$ は $dA = (rd\theta) \times (r\sin\theta d\phi)$ で表せる．よって半球状光源による点 $P$ の照度は $\theta$ と $\phi$ で積分することにより，

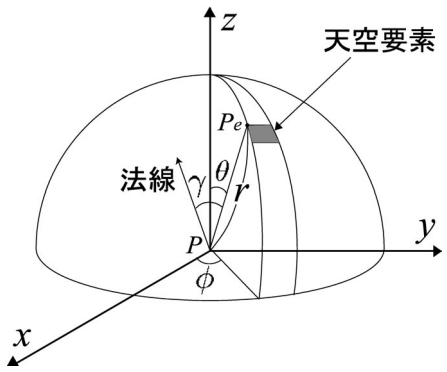


**図 2**　位置関係

$$E = \int_\theta\int_\phi H(\theta,\phi)L(\theta,\phi)\cos\gamma\sin\theta d\theta d\phi \qquad (2)$$

となる．

## 3. 天空光を考慮した画像生成

### 3.1 グラフィックスハードウェアを利用した画像生成

式 (2) で表される照度 $E$ を解析的に計算するのは困難であるため，数値積分によって計算する．すなわち，計算点 $P$ の照度 $E$ は次式で計算される．

$$E = \sum_i\sum_j H(\theta_i,\phi_j)L(\theta_i,\phi_j)\times \cos\gamma\sin\theta_i\Delta\theta\Delta\phi \qquad (3)$$

ここで $\Delta\theta$，$\Delta\phi$ はサンプリング間隔を表す．提案手法では，グラフィックスハードウェアを利用するために，天空要素を一つの点光源で近似する．こうすることでグラフィックスハードウェアを利用して物体の輝度計算を高速に行うことが可能となる．

$H$ の計算は，シャドウマッピング法を利用する．シャドウマッピング法とは，まず光源を視点とした画像 (シャドウマップ) を生成し，視点からレンダリングする際にシャドウマップとの奥行きを比較することで影を生成する方法である．グラフィックスハードウェアの機能であるテクスチャマッピングを利用したシャドウマッピング法[2)] によって，任意の点での $H$ の値を高速に求めることができる．

サンプリング間隔については，サンプリング間隔が大きい (天空要素への分割数が少ない) 場合

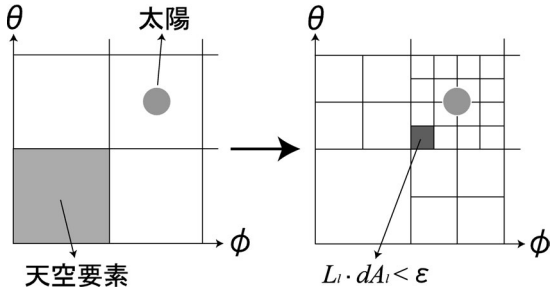


**図 3**　アダプティブサンプリング

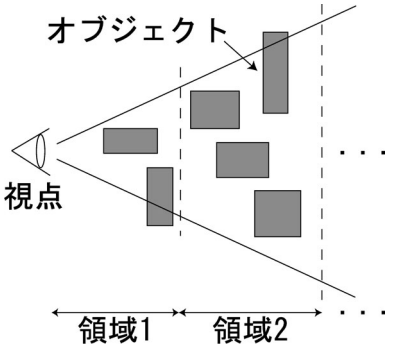


**図 4**　シーンの領域分割 (上面図)

は物体の影を十分にサンプリングできずエイリアシング問題が生じる．これを解決する最も単純な方法はサンプリング間隔を小さくすることであるが，単純に分割数を増加すると天空要素の数に比例して計算時間が著しく増加してしまう．そこで，以下のようにアダプティブにサンプリング間隔を決定する．

太陽の周辺は輝度が急激に変化するため，細かくサンプリングしなければ十分な精度が得られない．一方，輝度の小さい領域では少ないサンプル数で十分である．そこで，効率良くサンプリングするために，輝度分布に応じてアダプティブにサンプル点を配置する．すなわち，ある天空要素 $l$ の放射エネルギー $I_l(= L_l \cdot dA_l)$ がある閾値 $\epsilon$ よりも小さくなるようにアダプティブにサンプリングをする (図 3 参照)．これにより，輝度が高い太陽の周辺がより細かくサンプリングされ，単純に分割数を増加するよりも少ないサンプル数に抑えることが可能となる．

天空要素への分割が完了した後に，それぞれの天空要素の位置に点光源を配置して画像を描画し，グラフィックスハードウェアの機能であるアルファブレンディング (本手法では単純に二つの色を足し合わせる機能を利用する) によりフレームバッファにそれぞれの画像の色を足し込むことで画像を生成することができる．なお，量子化誤差を低減するために足し込みの際にはアキュムレーションバッファを利用している[3]．

### 3.2　アダプティブなシャドウマップの生成

都市景観画像を想定した場合は視界が広範囲になることが予想されるため，広範囲をカバーしたシャドウマップを作成する必要がある．しかし，テクスチャメモリに限界があるため，シャドウマップの解像度にも限界が生じる．そのため，1 枚のシャドウマップで広範囲をカバーすると一つ一つの物体に対するシャドウマップの解像度が低下するため影の精度が劣化するという問題が生じる．

そこで，本手法ではアダプティブシャドウマップ法[4] の考え方を利用して以下のようにこの問題の解決を図る．すなわち，描画シーンを複数の領域に分割しそれぞれにシャドウマップを作成することで影の精度の劣化を防ぐ．図 4 に示すように，視点から近い部分から順にシーンを分割する．そして，それぞれの領域に対してシャドウマップを作成し順に描画していく．これにより，視点に近い部分から複数枚のシャドウマップを作成することになり，1 枚のシャドウマップで広範囲をカバーすることがなくなるので，影の精度が劣化を防ぐことができる．

### 3.3　LOD の考え方を利用した画像生成

都市景観のような大量のポリゴンにより構成されるデータから画像生成をする場合は，天空光による照度計算は非常に計算コストがかかる．そこで，LOD の考え方を利用して計算コストの削減を図る．

透視投影で画像を作成する場合，基本的に遠方のポリゴンほどスクリーンへ投影される面積は小さくなり視覚的な影響は小さくなっていく．これを考慮して，視覚的な影響が小さいポリゴンを描画する際は，多少の照度計算の誤差を許容し，考慮する天空要素の数を少なくすることで計算コストを削減することを考える．

いま，あるポリゴン $p$ を考える．3.1 節で述べたアダプティブサンプリングによって，天空は $n$ 個の天空要素に分割されているとする．また，各天空要素は識別番号 $i(i = 1, 2, \cdots, n)$ が与えられているとする．識別番号が $i$ の天空要素により

計算した照度を $E_i$ とすると，$E_i$ の値が大きい順に天空要素に番号 $j$ を与える．また，照度の値が大きい方から $j$ 番目までの天空要素より計算した照度を $E_j$，すべての天空要素から計算した照度を $E_{all}$ とする．このとき，$j$ 番目までの天空要素で照度計算をした場合の誤差 $R_p(j)$ を次式で表す．

$$\begin{aligned} R_p(j) &= \sum_{i=1}^{n} E_i - \sum_{i=1}^{j} E_i \\ &= E_{all} - E_j \end{aligned} \tag{4}$$

このとき，$R_p(j) > R_p(j+1)$ を満たす．また，スクリーン上のポリゴンの面積は，視点とポリゴンの距離の 2 乗に反比例し，ポリゴンの大きさに比例する．そこで，照度の値が $j$ 番目までの天空要素を用いて照度計算をした場合のスクリーン上での見かけの誤差 $R_{sp}(j)$ を次式で評価する．

$$R_{sp}(j) = \alpha R_p(j) \cdot \frac{M_p}{d_p^2} \tag{5}$$

ここで，$\alpha$ は比例係数，$d_p$ はポリゴン $p$ の中心点と視点との距離とし，$M_p$ はポリゴン $p$ のスクリーン上での大きさであり，ポリゴンの中心から視点方向へのベクトルとポリゴンの法線ベクトルの内積とポリゴンの面積を乗算したものとする．提案手法では，閾値 $\mu$ を設定し $R_{sp}(j) < \mu$ を満たすサンプル数 $j_p$ を決定する．すなわち，ポリゴン $p$ に対しては照度の値が大きい方から $j_p$ 番目までの天空要素が計算に必要であるということになり，必要な天空要素の識別番号を記憶しておく．また，このときの誤差 $R_p(j)$ は，

$$\begin{aligned} R_p(j) &= E_{all} - E_j \\ &= \sum_{i=1}^{n} E_i - \sum_{i=1}^{j_p} E_i \\ &= \sum_{i=j_p+1}^{n} E_i \end{aligned} \tag{6}$$

となる．

レンダリングは前節で述べたグラフィックスハードウェアを利用した方法で行い，各天空要素で画像を描画しそれらを足し込むことで画像を生成する．そのため，識別番号が 1 番目の天空要素から順に処理を行い画像を生成していく必要がある．そこで，識別番号 $i$ 番目の天空要素に対する画像を生成する際，各ポリゴンについて記憶しておいた識別番号と照らし合わせ，必要なポリゴンのみを描画する．生じた誤差については環境光として描画する．

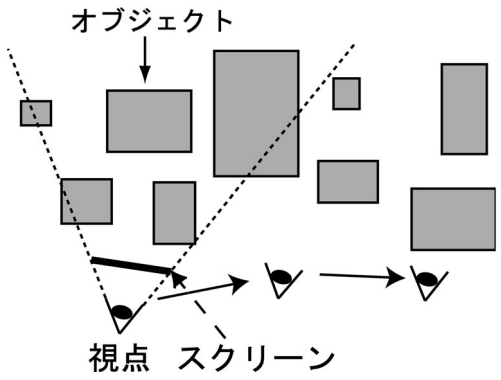


**図 5** 視点・スクリーン・オブジェクトの関係 (上面図)

以上の処理により，各天空要素に対してすべてのポリゴンを描画するのではなく一部のポリゴンを選んで描画するため，計算コストを削減することができる．

## 4. アニメーションへの応用

3 節で述べた方法を用いてアニメーションを作成する場合，繰り返し画像を生成する必要があり，計算コストが非常に高い．本節では，アニメーションを作成する場合の高速化手法について検討する．

図 5 に示すように，視点の動きは既知であるとする．まず，前処理で各フレームの表示に必要な点と色を取得してデータベースとして記憶し，その点をレンダリングすることで各フレームの表示をする．以下，詳細を述べる．

### 4.1 点の取得

視点・スクリーン・オブジェクトが図 6 のような関係にあるとする．まず，視点から画素 $(i,j)$ に向かうベクトル $\vec{s}_{(i,j)}$ を算出する．次に，視点を通り $\vec{s}_{(i,j)}$ の方向を持つ直線 $l_{(i,j)}$ と物体との交点 $p_{(i,j)}$ を求める．ここで，交点を計算するためには視線が物体のどのポリゴンと交差するかという情報が必要になる．これは図 7 に示すようにそれぞれのポリゴンに個別に色を与えて描画し，各画素の色を読み込むことで決定できる．以上の処理を全画素に対して行い，得られた点をデータベースとして保存する．2 フレーム目以降に関しては，まずデータベースにある点を特

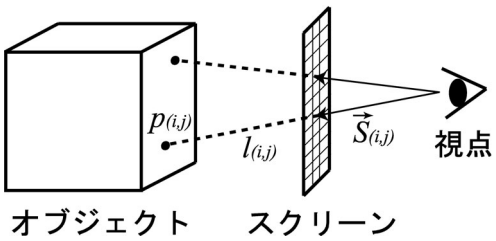


**図 6**　視点・スクリーン・オブジェクトの関係 (側面図)

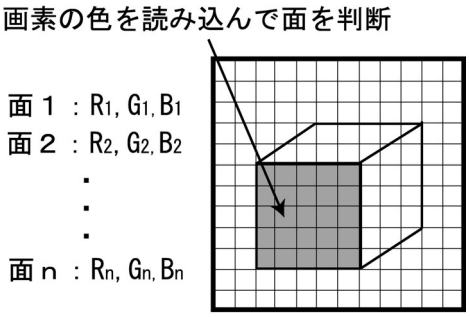


**図 7**　個別の色による面の判断

別色 (例えば黒) で描画した後に上記の処理を行う．ただし，特別色である画素については点の取得は行わない．これにより，新たに必要な点のみを取得しデータベースに追加する．以上の処理により，アニメーションの表示に必要な点をすべて取得できる．

### 4.2　点の色計算

色計算は 3 節で述べた方法を利用する．しかし，視点が移動するため LOD 判定の計算の際に用いるレンダリングする点と視点の距離が各フレーム間で変化する．データベース上のある 1 点を LOD を利用して輝度計算しアニメーションを作成する場合，その点の計算精度が最も要求されるのは点と視点の距離が最も近いフレームを描画する場合である．移動する視点の軌跡は既知であるため，すべての取得した点について視点の軌跡との距離が最も近いときの値を計算し，その値を用いて LOD 判定を行い色計算をする．得られた色はデータベースに追加する．

### 4.3　レンダリング

レンダリングは，データベースを参照し点の情報を読み込んでポイントレンダリングすることで画像を生成する．この方法でアニメーションを作成する場合は，ポリゴンデータ数よりもデータベースにある点の総数が少ない場合は特に高速な描画が期待できる．

## 5．適　用　例

簡単な例を用いて，提案手法による画像生成を行った．標準グラフィック API の 1 つである OpenGL を用いて実装した．画像サイズはすべて 512 × 512 である．実験に使用した計算機は CPU が AthlonXP1700 ＋，グラフィックスハードウェアとして NVIDIA 社の GeForce3 を搭載した PC を用いた．

まず，提案手法を用いて静止画を作成した (図 8 参照)．アダプティブサンプリングによって天空要素は 471 個に分割し，描画シーンは 2 つに分割してシャドウマップを 2 枚作成した．用いたデータはランダムに配置された直方体で構成されており，ポリゴン数は約 30 万ポリゴンである．すべての天空要素から光を照射して作成した画像を真の画像として，LOD を用いて作成した画像との比較実験を行った．計算時間は，真の画像が 107.2 秒であったのに対して，LOD を用いて作成した画像は 69.8 秒であった．差分画像を求め，その平均輝度値を求めたところ 7.2 であり，LOD の考え方を利用することによってほぼ同程度の画質の画像を約 1.5 倍の速度で作成できた．

次に，30 フレームのアニメーションの作成を行った (図 9 参照)．天空要素の数や描画シーンの分割条件は静止画作成の場合と同様に設定した．用いたデータも静止画と同様にランダムに配置された直方体で構成されており，ポリゴン数は約 30 万ポリゴンである．静止画を繰り返し生成しアニメーションを作成する方法では，1 フレームを描画するのに約 2 分かかり，30 フレームすべてを描画するのに 60 分ほどの時間を要した．前処理で必要な点を取得し色計算を行いポイントレンダリングする方法では，前処理で取得した点は約 72 万点であり，色計算にかかった時間は約 45 分であった．ポイントレンダリングでは 9〜10fps ほどの速度のアニメーションを作成できた．

これらのことから，提案手法は速度の面で非常に有用である．

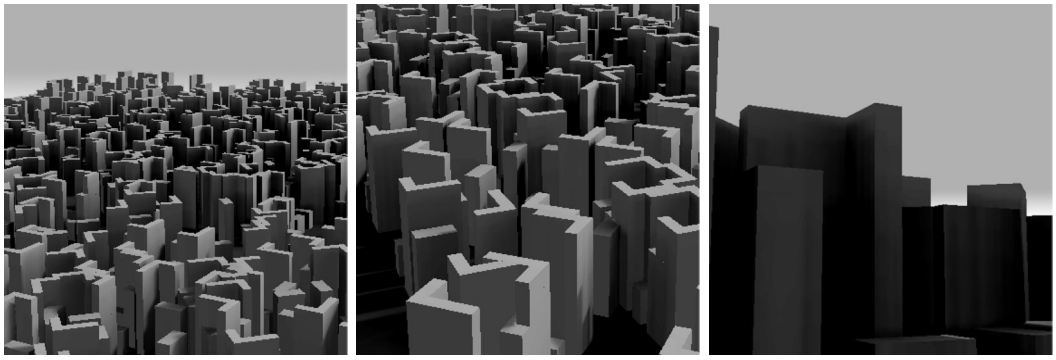

**図 8**　作成した画像

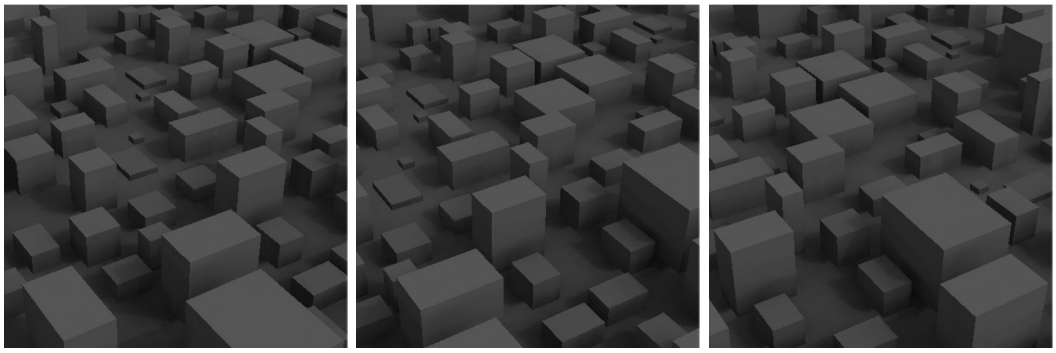

**図 9**　アニメーション作成例 (左:1 フレーム，中:15 フレーム，右:30 フレーム)

## 6. ま と め

高速に都市景観画像を作成する手法を提案した．提案手法では，グラフィックスハードウェアを利用して輝度計算を行い画像を作成し，LOD の考え方を利用して視覚的に影響の少ないポリゴンに対して天空のサンプル数を少なくすることで計算コストの削減を図った．また，視点の軌跡が既知である場合のアニメーション作成への本手法の適用について考察した．アニメーションへの適用は前処理で各フレームの表示に必要な点と色を取得してデータベースとして記憶し，データベースを参照してポイントレンダリングすることで各フレームの表示を行った．

今後の課題として，さらなる高速化や鏡面反射成分を考慮することが挙げられる．

## 参 考 文 献


1) T.Nishita, E.Nakamae, “Continuous Tone Representation of Three-Dimensional Objects Illuminated by Sky Light”, *Computer Graphics*, 20, No.4, pp.125-132 (1986)

2) 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)

3) 高田信太郎，土橋宜典，山本　強，“天空光に照射された 3 次元物体のグラフィックスハードウェアを利用した高速表示”，情報処理学会第 64 回全国大会講演論文集 (4)，pp.821-822 (2002)

4) R.Fernando, S.fernandez, K.Bala, D.P.Greenberg “Adaptive Shadow Maps”, *Proc.SIGGRAPH2001*, pp.387-390 (Aug.2001)