2004－CG－117　(19)
2004／11／27

反射分布を考慮した鏡面反射の高速レンダリング

山田友希[†]　土橋宜典[†]　山本強[†]

近年，コンピュータグラフィックス (CG) を用いたリアルな画像が広い分野で利用されるようになっている．CG を用いた画像生成において，鏡，車体，ガラス張りのビルなどに物体が映り込むといった鏡面反射による映り込みを表示することで，よりリアルな画像を生成することができる．そのため，映り込みを表示する研究は数多く行われているが，その複雑さゆえに計算コストが高いという欠点がある．一方，近年，グラフィックスハードウェアの高性能化が急速に進んでおり，これを利用した写実的な画像を高速に生成する研究が盛んに行われている．そこで，本研究ではグラフィックスハードウェアのテクスチャマッピング手法を用いて，反射分布を考慮した鏡面反射による映り込みを高速にレンダリングする手法を提案する．

A Fast Rendering of Specular Reflection Considering Reflectance Distribution

Yuki Yamada,[†] Yoshinori Dobashi[†] and Tsuyoshi Yamamoto[†]

Recently, realistic image synthesis using computer graphics has been used in various fields. The reality of the synthetic images can be improved by taking into account specular reflections such as mirrors, sufaces of cars, glass windows of buildings reflecting other objects. So, there are many methods to simulate the specular reflections. However, the simulation of this phenomenon is quite complex and therefore is very time-consuming. This is especially true for glossy specular reflections. On the other hand, the recent advancement of graphics hardware has made significant progress in the performance. This encourages researchers to develop methods for accelerating image generation processes using graphics hardware. We present a fast method for rendering glossy specular reflections considering reflectance distribution functions. Our method exploits hardware texture mapping functions to accelerate the computations.

1. はじめに

近年，コンピュータグラフィックス (CG) を用いた写実的な画像が映画やゲーム，景観評価など広い分野で利用されるようになっている．実世界には鏡，車体，ガラス張りのビルなどに物体が映り込む現象が数多く存在する．CG ではこれらのような鏡面反射による映り込みを表示することでより写実的な画像を生成することができる．鏡面反射は物体表面で起こる光の散乱による現象であり，正反射方向の近傍に反射光が分布する．映り込みはある物体表面で反射した光が，別の物体で鏡面反射したときに生じる．映り込みの表示に関する研究は，数多く行われているが，鏡面反射の反射分布を計算し，分布した光を追跡する必要があるため，非常に多くの計算時間を要する．

一方，近年，グラフィックスハードウェアの高性能化が急速に進んでおり，グラフィックスハードウェアを利用した高速化の研究が盛んになっている．岩崎らは水面への物体の映り込みを，グラフィックスハードウェアを利用して高速に表示する手法を提案した[1]．本研究ではこの手法を物体同士の映り込みに応用し，さらに反射分布を考慮することでより写実的な画像を高速に表示する手法を提案する．

提案手法では映り込ませる 3 次元物体を複数の仮想平面で表現する．すなわち，この平面をクリップ面として使用し，隣接する平面で挟まれた部分の画像をテクスチャとして保存する．次に，反射分布に応じて複数の反射ベクトルとこの仮想平面との交点を用いて，テクスチャをマッピングすることで，鏡面反射による映り込みを表示する．

2. 従来手法

鏡面反射による映り込みを表示する代表的手法として，レイトレーシング (ray-tracing)[2]，環境マッピング (environment-mapping)[3] がある．

レイトレーシングは物体に反射して視点に入射してくる光を追跡する手法である．この手法は視点からスクリーンの各画素を通過するレイを放射し，それぞれのレイについてすべてのポリゴンとの交差判定を行い，視点に最も近い交点 (可視点) を求めるというものである．可視点が鏡面反射物体である場合，反射方向に再度レイを放射する．光の反射を忠実に再現するため，映り込みや屈折，透過，影など各種光学現象を考慮した画像を生成できる．しかし，ポリゴンとレイの交点

† 北海道大学大学院情報科学研究科，札幌市

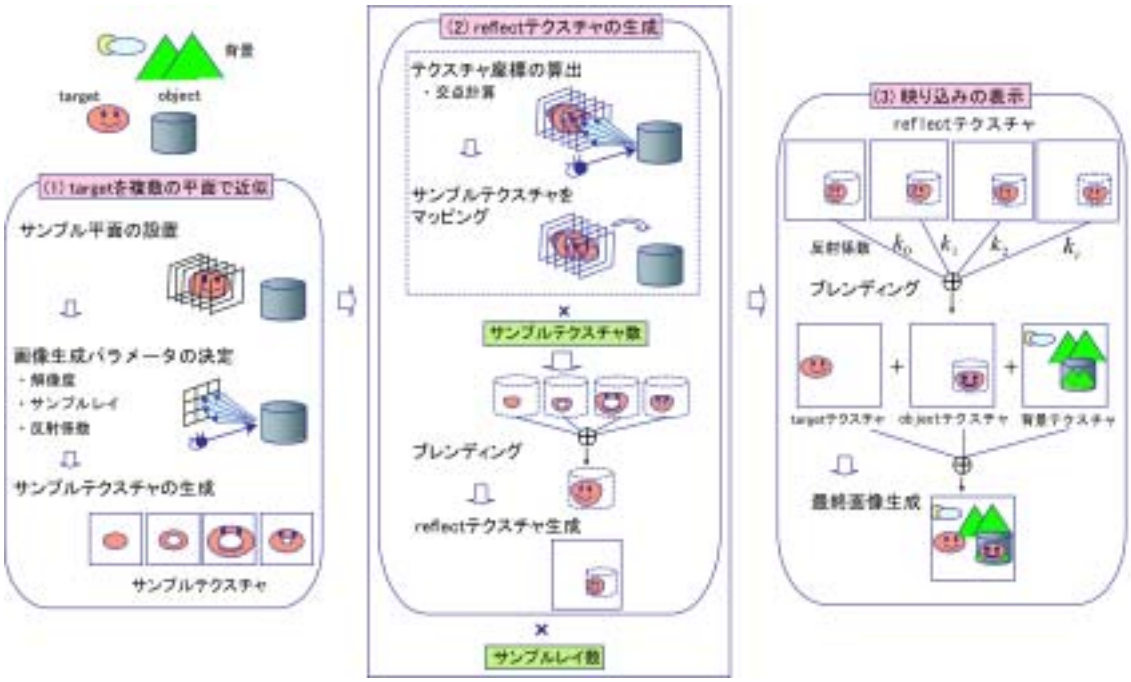

図 1 提案手法の流れ

計算に計算コストがかかり，非常に長い計算時間を必要とする点が問題となっている．鏡面反射における反射分布を考慮する場合，一本のレイにつき反射分布に応じて複数の反射ベクトルをランダムに放射し，その結果をブレンディングする (分散レイトレーシング法)．分散レイトレーシング法 (distributed ray-tracing)[4] では，さらにレンダリング時間が膨大になる．

一方，環境マッピングは無限に大きな球や立方体を考え，その内側にマッピングした環境のテクスチャが反射しているようにマッピングし，映り込みを表示する．環境マッピングはグラフィックスハードウェアを利用し，高速に映り込みを表示できるが，近くの物体の映り込みが不自然になるという問題がある．

提案手法では，これらの従来手法の問題点を解決し，高速に物体同士の距離に関係なく，反射分布を考慮した鏡面反射による映り込みを表示することを目指す．次章では，提案手法の概要について述べる．

3. 提案手法の概要

本論文では，説明簡略化のため，映り込ませる 3 次元物体を target，映り込みを表示する 3 次元物体を object と呼ぶ．

提案手法では，分散レイトレーシング法と同様に反射分布に応じて複数の反射ベクトルを放射する．提案手法では，target を平面で近似し，反射ベクトルとの交点計算を簡略化する．また，target をテクスチャとして保存することで，環境マッピングと同様にグラフィックスハードウェアのテクスチャマッピング機能を用いて，更なる高速化を図る．図 1 に提案手法の流れを示す．

提案手法では以下の 3 段階に分けて考える．

(1) target を複数の平面で近似
(2) reflect テクスチャの生成
(3) 映り込みの表示

まず (1) では，target の周囲に仮想的な複数の平面 (サンプル平面と呼ぶ) を設置する．次に，画像生成パラメータを決定する．画像生成パラメータはサンプル平面にマッピングするテクスチャの解像度，object 表面の反射分布に応じて生成する反射ベクトル (サンプルレイと呼ぶ)，さらに各サンプルレイに対する反射係数である．これらを決定した後，隣接するサンプル平面内の target 部分を平行投影し，テクスチャ画像 (サンプルテクスチャと呼ぶ) として保存する．このときのサンプルテクスチャの解像度には先ほど決定した解像度を使用する．

次に (2) では，各サンプルレイについて reflect テクスチャを生成する．ここで，reflect テクスチャは各サンプルレイの方向にレイトレースして映り込みを計算した場合の画像を表す．そのため，各サンプルレイについて以下の処理を施し，サンプルレイ数枚分の reflect テクスチャを生成する．まず，サンプルレイと各サンプル平面の交点を求める．この交点をテクスチャ座標として，object にサンプルテクスチャをマッピングした画像を生成する．各サンプルテクスチャに対する画像を合成し，reflect テクスチャとして保存する．

(3) では，reflect テクスチャを反射係数を考慮しブレンディングすることで反射分布を考慮した映り込みを表示する．その後，フレームバッファをテクスチャ

(object テクスチャと呼ぶ) として保存する．そして，target をテクスチャとして保存したもの (target テクスチャと呼ぶ)，背景と背景を映り込ませた object をテクスチャとして保存したもの (背景テクスチャと呼ぶ) を用意し，object テクスチャとブレンディングすることで最終画像を得る．

以降，4 節で target を平面で近似する手法，5 節で reflect テクスチャの生成法，6 節で反射分布を考慮した映り込みの表示法について詳しく述べる．

4. サンプル面による target の近似

本節では，target をサンプル平面で近似する手法について述べる．ここでは，生成画像の画質および生成時間を考慮してサンプル平面の大きさ，位置，枚数を決定する．

さらに，画像生成パラメータであるサンプルテクスチャの解像度，object 表面での反射分布に応じて発生するサンプルレイ，反射係数もここで決定する．

これらを決定した後，サンプルテクスチャを生成する．以下，サンプル平面の設置法，画像生成パラメータの決定法，サンプルテクスチャの生成法について述べる．

4.1 サンプル平面の設置

生成画像の画質はサンプル平面の大きさ，向き，枚数に依存する．また，画像生成時間は枚数に大きく依存する．本節では，最適なサンプル平面の大きさ，向きを決定し，一定の画質を保つための最小のサンプル平面数を決定する手法を提案する．

図 2(a) に示すように，target が内接するバウンディングボックスを設定し，バウンディングボックスの中心を target の中心とする．target の中心から最も離れたバウンディングボックスの頂点と target の中心との距離 λ を算出し，2λ を一辺とする正方形 (初期平面) を target の中心に配置する．初期平面は target の中心と object の中心を結んだ直線に垂直に設置する．この初期平面を用いて，最適なパラメータを決定する．まず，object を形成する各ポリゴンのうち可視であるポリゴンの頂点における正反射ベクトルを算出する．ポリゴンを構成する全頂点での各正反射ベクトルが初期平面と交点を持つポリゴンの個数 n，頂点の座標 $\boldsymbol{O_i}$，各反射ベクトル $\boldsymbol{R_i}$ を取得する．n が 0 の場合，以降の映り込みの計算を行わない．$\boldsymbol{O_i}$，$\boldsymbol{R_i}$ よりそれぞれ平均表面座標 $\bar{\boldsymbol{O}}$，平均反射ベクトル $\bar{\boldsymbol{R}}$ を求める．

サンプル平面は初期平面と同じ大きさとし，向きは $\bar{\boldsymbol{R}}$ に垂直に設定する．サンプル平面の大きさを target が納まる最小の正方形である初期平面と同じ大きさにすることで，サンプルテクスチャを生成した際の画質低下を抑えることができ，向きを $\bar{\boldsymbol{R}}$ に垂直に設定することで，なるべく多くの反射ベクトルと垂直に近い向きとなり，平面で近似した際の画質低下を抑えることができる．

次に，サンプル平面数を決定する．まず，最初の ($\bar{\boldsymbol{O}}$ に最も近い) サンプル平面と，最後の ($\bar{\boldsymbol{O}}$ に最も遠い) サンプル平面の位置を決定する (図 2(b) 参照)．その後，サンプル平面の間隔を決定し，最初のサンプル平面と最後のサンプル平面の間に配置していく．

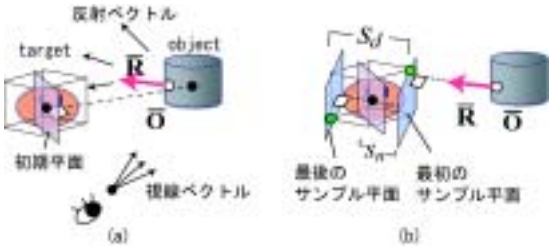

図 2 サンプル平面の設置

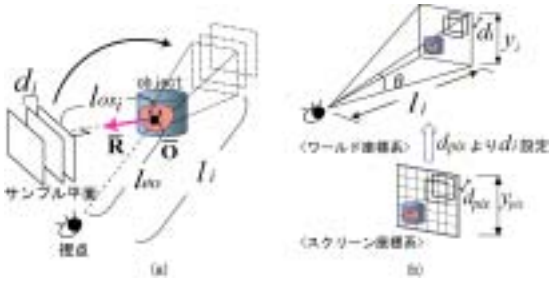

図 3 サンプル平面の間隔

図 2(b) のように，求めた $\bar{\boldsymbol{O}}$ に最も近いバウンディングボックスの頂点を通り，$\bar{\boldsymbol{R}}$ に垂直な平面を求める．この平面と target の中心との距離 s_n を算出し，最初のサンプル平面はこの平面上で，中心座標は target の中心より距離 s_n だけ離れた座標とする．同様に，最後のサンプル平面を決定し，target の中心との距離 s_f を求める．ここで，$s_d = s_n + s_f$ とする．

次に，サンプル平面の間隔 d_i の求め方について述べる．object に映り込んだ際のサンプル平面の間隔 d_i がスクリーン上でユーザが指定した画素数 d_{pix} になるよう d_i を決定する．しかし，d_i を正確に求めることは困難なため，本研究では次式により近似的に算出する (図 3 参照)．

$$y_i = 2\, l_i\, tan(\theta/2) \qquad (l_i = l_{eo} + l_{os_i}) \tag{1}$$

$$d_i = d_{pix}\, y_i / y_{pix} \tag{2}$$

$$\sum_{i=0}^{n} d_i > s_d \tag{3}$$

視点と $\bar{\boldsymbol{O}}$ 間の距離 l_{eo} と $\bar{\boldsymbol{O}}$ と i 番目のサンプル平面までの距離 l_{os_i} の和を l_i とする．図 3(a) のように，object に映り込んだ target のスクリーン上での大きさは視点より距離 l_i だけ離れた位置に存在した場合のスクリーン上での大きさにほぼ等しいと考えられる．そこで，式 (1) を用いて距離 l_i だけ離れた場所にスクリーンを配置した場合のスクリーンの高さ y_i を算出する (3(b) 参照)．ここでの θ は視野角である．次に，式 (2) における y_{pix} はスクリーンの画素数であり，y_i/y_{pix} により，スクリーン上の 1 画素には，target

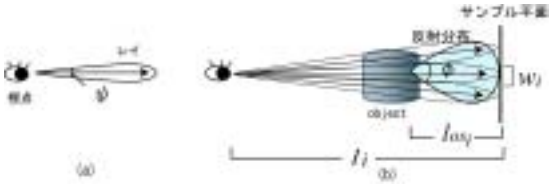

図 4 解像度の決定

位置でのどの程度の大きさが object に映り込んでいるかを見積もることができる．よって，式 (2) より，スクリーンにおいて d_i が占める画素数がユーザが入力する値 d_{pix} に設定され，d_i は視点，object，サンプル平面の位置によって変化する．d_i をサンプル平面間隔とし，式 (3) を満たす最小の自然数 n をサンプル平面の枚数とする．

以上のようにサンプル平面数を決定することで，生成画像の画質を一定に保ち，かつ効率的に画像を生成することができる．

4.2 画像生成パラメータの決定

次に，画像生成パラメータとなるサンプルテクスチャの解像度，サンプルレイ，反射係数を決定する．

サンプルレイ数が増加するほど，より正確に反射分布を考慮することができ，生成画像の画質が向上する．しかし，サンプルレイ数回映り込みを表示し，reflect テクスチャを生成するため，サンプルレイ数が増加するほど画像生成時間を要する．そこで，本手法ではユーザが指定する一定の画質を保つようサンプルレイ数を決定する．

図 4(a) のように，各レイは角度 ψ の範囲を代表していると考える．図 4(b) のように，視点とサンプル平面は距離 l_i だけ離れていると仮定できるため，サンプル平面で交点を持つ際のサンプルレイの広がりは $w_i = 2l_i \tan(\psi/2)$ となる．このサンプルレイを用いて object 表面で角度 ϕ の広がりを持つ反射分布を近似する．まず，サンプルテクスチャの解像度をサンプル平面において w_i が 1 画素の幅になるように決定する．これにより，1 本のレイで $w_i \times w_i$ の範囲の色を考慮することができる．さらに，サンプルレイの代表する範囲が重ならないよう放出し，反射分布の範囲を満たす数をサンプルレイ数とする．サンプルレイ数はすべてのサンプル平面において等しい必要があるため，最も多くのサンプルレイ数を必要とする最初のサンプル平面において算出した数を使用する．

各レイの広がりを示す角度 ψ の値が小さいほど，サンプルレイ数が増え生成画像の画質は高くなる．提案手法では，ψ をユーザが指定する値とし，ユーザが指定する一定の画質を保つ．

つづいて，各サンプルレイに対する反射係数 k_i を算出する．k_i はフォンのモデルを用い，正反射ベクトル $\boldsymbol{p}$ と各サンプルレイ $\boldsymbol{q_i}$ の内積で表される鏡面反射成分の減少の割合 s_i を用いて以下の式より算出する．

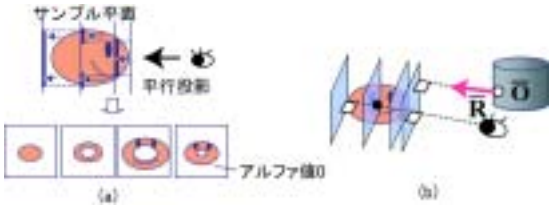

図 5 サンプルテクスチャ生成

$$s_i = \begin{cases} 1.0 & (i = 0) \\ (\boldsymbol{p} \cdot \boldsymbol{q_i})^{shine} & (i = 1, 2, \ldots, v-1) \end{cases}$$

$$k_i = s_i / \sum_{j=0}^{v} s_j \qquad (i = 0, 1, \ldots, v-1)$$

ここで，*shine* はハイライト特性を示す値であり，*shine* が大きいほどシャープなハイライトが得られる．$i = 0$ の場合，$\boldsymbol{q_0} = \boldsymbol{p}$ とし，s_0 は 1.0 とする．v はサンプルレイ数であり，反射係数 k_i は s_i をその総和で割ったものである．

4.3 サンプルテクスチャの生成

ここでは，target をテクスチャ(サンプルテクスチャ)として保存する手法について述べる．図 5(a) のように，サンプルテクスチャはサンプル平面をクリップ空間に利用し，その空間内に存在する target 部分を平行投影することで生成する．このときサンプル平面の中心を注視点とし，サンプル平面の法線方向に視点を設定する (図 5(b) 参照)．また，target が描画されていない画素のアルファ値を 0 に設定する．これはアルファブレンディング機能を利用することで target 部分のみ描画するためである．サンプルテクスチャの解像度は 4.2 節で求めた解像度を用いて生成する．

5. reflect テクスチャの生成

本節では，4.3 節で生成したサンプルテクスチャを object にマッピングして映り込みを表示し，reflect テクスチャを生成する手法について述べる．

サンプルテクスチャをマッピングするために，テクスチャ座標を算出する．そのために，各サンプルレイに対してすべてのサンプル平面との交点を計算し，テクスチャ座標として用いる．

ここで，object は三角形ポリゴンで構成されているため，図 6(a) に示すように，(1) ポリゴンを構成する全頂点での反射ベクトルがサンプル平面と交点を持たない場合，(2) 全頂点でサンプル平面と交点を持つ場合，(3) サンプル平面と交点を持つ頂点と持たない頂点が存在する場合が考えられる．図 6(b) に示すように，(1) の場合は考慮しない．(2) の場合，テクスチャ座標を算出する．(3) の場合，サンプル平面と交点を持たない頂点では，サンプル平面を含む平面との交点を平行移動し，近似的にテクスチャ座標を求める．つまり，テクスチャ座標を 0 か 1 に対応させる．

各サンプル平面ごとにサンプルテクスチャをマッピングした object の画像を生成する．さらに，その結

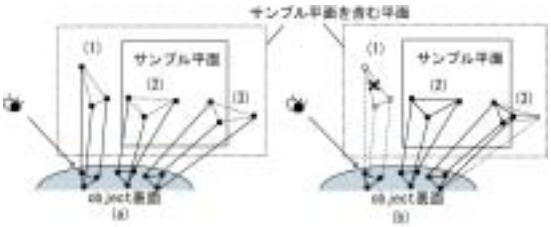

図 6 テクスチャ座標の算出

果をアルファブレンディングにより合成した画像を生成し，reflect テクスチャとして保存する．この作業をサンプルレイごとに行い，reflect テクスチャをサンプルレイ数だけ生成する．

6. 映り込みの表示

最後に 5 節で生成した reflect テクスチャをブレンディングし，反射分布を考慮した映り込みを表示する．

各 reflect テクスチャをスクリーンに対応する平面にマッピングして描画する．このとき，それぞれの RGBA 値に 4.2 節で算出した反射係数 k_i を乗算し，アルファブレンディングすることで反射分布を考慮した映り込みを表示する．得られた画像を object テクスチャとして保存する．次に，同様にして，reflect テクスチャの代わりに環境マッピングを使用して背景を映り込ませた object と背景をテクスチャとして保存し，背景テクスチャを生成する．最後に，target など，object より手前にある物体を透視投影して target テクスチャとして保存する．

以上の object テクスチャ，背景テクスチャ，target テクスチャをアルファブレンディングすることで最終画像を得る．

7. 適 用 例

提案手法を用いた適用例を図 7，8 に示す．計算機は CPU に Pentium4 3.2GHz，グラフィックスカードに NVIDIA 社の GeForce4 Ti 4800 を搭載した PC を用いた．

生成画像の解像度は 512×512 であり，サンプルレイの広がりを示す角度は $\psi = 0.167^\circ$ とする．適用例では，2 つの target に対する映り込みを表示している．図 9 に今回使用した 3 次元物体を示す．

図 7 に視点位置による画像の比較を示す．このときの各パラメータ値とレンダリング時間を表 1 に示す．表より，視点からの距離でサンプル平面数を変化させるため，視点が遠いほどサンプル平面数が減り，またサンプルレイの広がりは大きくなるためサンプルレイ数が減り，レンダリング時間は短縮されることがわかる．このことから視点位置に応じて効率的にレンダリングされているといえる．また，そのときの画質が一定に保たれていることが図 7 よりわかる．

図 8 は ϕ の値による画像の比較を示す．$\phi = 11.6$, 23.2，37.7° と設定した場合，それぞれのサンプルレイ数 (target1+target2) は 35(15+20)，52(23+29)，60(29+31) 本であり，レンダリング時間は 2.281 秒，3.172 秒，3.656 秒である．反射分布の広がりが大きいほどサンプルレイ数が増加するためレンダリング時間は長くなる．

図 9 使用した 3 次元物体

8. ま と め

本論文では，グラフィックスハードウェアを利用して，反射分布を考慮した鏡面反射による映り込みを高速に表示する手法を提案した．提案手法では，映り込む物体の周囲に複数の平面 (サンプル平面) を設定し，平行投影した画像をハードウェア処理に適したテクスチャ(サンプルテクスチャ) として保存した．映り込みの計算においては，反射分布内に散乱した反射ベクトルと，サンプル平面を使用して効率的に計算を行うことで高速化を図った．複数のテクスチャをハードウェア機能の 1 つであるアルファブレンディングを利用して合成し，映り込みを表示した．

今後の課題としてはさらなる画質の向上・高速化が挙げられる．

参 考 文 献

1) Kei Iwasaki, Yoshinori Dobashi, Tomoyuki Nishita, “A Fast Rendering Method for Refractive and Reflective Caustics Due to Water Surfaces”, *Proc. of EUROGRAPHICS2003(Computer Graphics Forum)*, Vol.22, No.3, pp. 601–609(2003).
2) T. Whitted, “An Improved Illumination model for shaded display”, *Communications of the ACM*,Vol.23, No.6,pp. 343–349(1980).
3) J. Blinn and M. Newell. “Texture and reflection in computer generated images”, *Communications of the ACM*,Vol.19, No.10,pp.542–546(1976).
4) R. L. Cook, T. Porter and L. Carpenter, “Distributed Ray Tracing”, *Computer Graphics*,Vol.18, No.3, pp.137–145(1984).

図 7　視点の変更 ($\phi = 11.6^\circ$)

視点位置	target	サンプル平面数	最初のサンプルテクスチャの解像度	サンプルレイ数	レンダリング時間 (秒)
(a)	1	4	64 × 64	9	1.203
	2	2	32 × 32	15	
(b)	1	8	64 × 64	15	2.281
	2	3	32 × 32	20	

表 1　図 7 での各パラメータとレンダリング時間

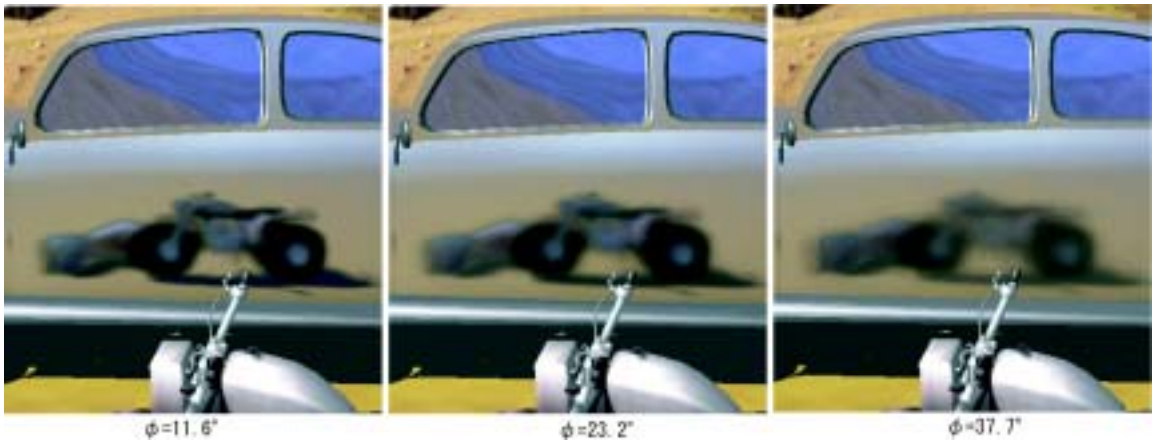

図 8　ϕ の値を変更