2008－CG－133 (20)
2008／11／8

曲面を持つ透明物体の映り込み表現手法

平野 智章　　向井 信彦　　小杉 信
武蔵工業大学大学院 工学研究科 電気工学専攻

コンピュータグラフィックス(CG)による映り込み表現には、通常、レイトレーシング法や環境マッピング法を用いるが、レイトレーシング法は画素単位の処理であるため多大な計算時間を要し、環境マッピング法を用いた場合でも、周囲環境が動的に変化する場合はマッピング用画像を再生成するために多大な処理時間を要する。また、映り込み面が曲面の場合、映り込み映像の特定は時間を要する処理となる。そこで本稿では、映り込み曲面を平面に分割し、分割された各映り込み平面に対して視点を鏡像反転した位置に仮想視点を設け、仮想視点からの映像を各映り込み平面にマッピングすることにより高速に映り込み処理を実現する手法について検討する。

A Rendering Method of Reflection on a Curved Transparent Object

Tomoaki HIRANO, Nobuhiko MUKAI and Makoto KOSUGI
Graduate School of Engineering, Musashi Institute of Technology

Environment mapping and ray tracing methods are usually used for the reflection by CG. However, they need huge amount of time for the processing and the rendering in case that the environment changes dynamically and the reflected objects have curved surfaces. Therefore, in this paper, we propose a rendering method of reflection on a curved object by dividing the curved surface into some planes and by using texture mapping images obtained from a virtual viewpoint set at the mirror-reversed position for the each divided plane.

1. はじめに

近年、建築分野等において、景観シミュレーションのためのリアルな表現が必要とされている。特に、ガラス張りのビルなどが数多く建築され、ビル建設前に周囲との調和を検証するためにはガラス面への映り込みを実現したリアリティの高い景観シミュレーションが必要である。

CG を用いて透明物体への映り込みを実現するためには、通常、レイトレーシング法や環境マッピング法を用いる。レイトレーシング法は、画素単位に反射や屈折を扱う過程で映り込みを実現するため、多大な処理時間を要する[1]。一方、環境マッピング法は予め生成した環境画像を用いて映り込みを実現する。そのため、表面の粗さが不均一な材質への映り込みを表現するためには、環境マップをフィルタ処理し、複数の粗さに応じた多重スケール環境マップを作成する必要がある[2]。しかしながら、この手法は、予め複数の粗さに対応した画像を用いるため高速な映り込みは可能であるが、周囲環境が動的に変化した場合、再度多重スケール環境マップを作成する必要があり、高速処理は困難である。そこで我々は、実環境データを基に映り込み面に対して鏡像反転した仮想環境データを作成し、この仮想環境データを用いることによって周囲環境が動的に変化した場合でも高速な映り込みが行える方式を提案した[3]。しかしながら、提案手法は、映り込み面を平面と仮定しているため、曲面を持つ透明

物体への映り込みには適用できなかった。

そこで本稿では、周囲環境が動的に変化した場合でも対応可能な、曲面を持つ透明物体への映り込み手法について提案する。

2. 曲面への映り込み手法

2.1. 平面分割と仮想視点の設定

図1に提案手法の概略を示し、そのアルゴリズムを以下に記述する。

〈 曲面への映り込みアルゴリズム 〉

(1)映り込み曲面を等間隔の平面に分割する。
(2)分割された各平面に対して、視点とは鏡像反転する位置に仮想視点を設ける。
(3)仮想視点からの映像を映り込み用テクチャとして取得する。
(4)平面を構成する各頂点のテクスチャ座標を計算し、テクスチャマッピングを行う。
(5)上記の処理を分割された全平面に対して行う。

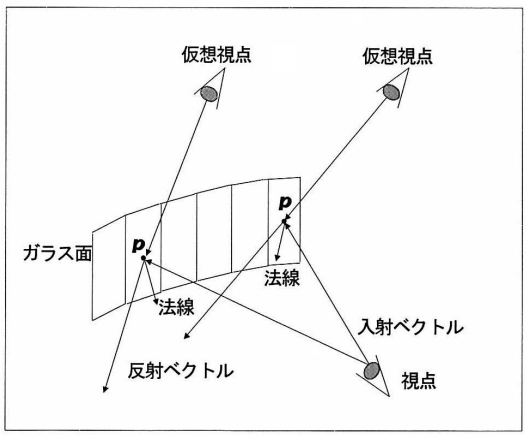


図1　平面分割と仮想視点の設置

2.2. テクスチャ座標の算出方法

図2に示すように仮想視点から見た投影面を設定し、視線ベクトルの各頂点における反射ベクトルより各頂点のテクスチャ座標を計算する。以下、テクスチャ座標の算出方法について詳しく述べる。

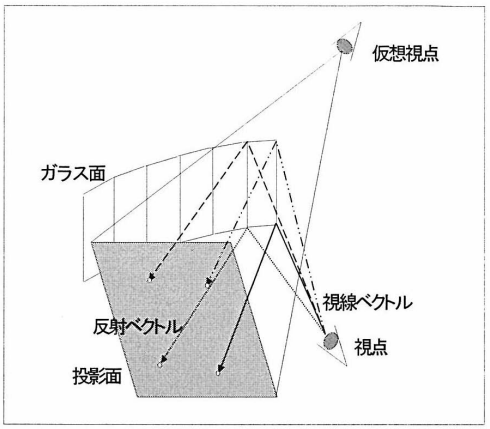


図2　テクスチャ座標の計算

図3に示すように、視点座標系上における投影面上の任意の点をP(α,β,0)、ワールド座標系上におけるワールド座標系の原点から視点座標系の原点へのベクトルを$\boldsymbol{V}_o$、ワールド座標系の原点から投影面上の任意の点Pへのベクトルを$\boldsymbol{V}_p$、視点座標系上における原点をO_v、$\overrightarrow{O_vP}$を$\boldsymbol{V}_e$、視点座標系からワールド座標系への変換行列を$\boldsymbol{M}$とすると

$$\boldsymbol{V}_p = \boldsymbol{M}\boldsymbol{V}_e + \boldsymbol{V}_o \qquad \cdots(1)$$

となり、

$$\boldsymbol{V}_e = \boldsymbol{M}^{-1}(\boldsymbol{V}_p - \boldsymbol{V}_o) \qquad \cdots(2)$$

となる。

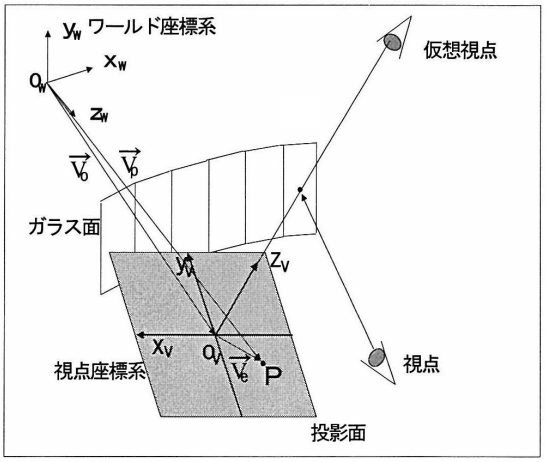


図3　ワールド座標系と投影面の位置関係

2.3. 変換行列の算出方法

視点座標系からワールド座標系への座標変換$\boldsymbol{M}$は一般に平行移動成分を伴うが、ここで求めるベクトル$\boldsymbol{V}_e=(\alpha,\beta,0)$は原点からのベクトルであるため、ここでは回転成分のみを考える。視点座標系の各軸$(\boldsymbol{X}_V,\boldsymbol{Y}_V,\boldsymbol{Z}_V)$の正方向を向く単位ベクトルを$\boldsymbol{i}=(1,0,0)^T$、$\boldsymbol{j}=(0,1,0)^T$、$\boldsymbol{k}=(0,0,1)^T$とし、これらのベクトルのワールド座標系におけるベクトルを$\hat{\boldsymbol{i}}=(\hat{i}_x,\hat{i}_y,\hat{i}_z)^T$、$\hat{\boldsymbol{j}}=(\hat{j}_x,\hat{j}_y,\hat{j}_z)^T$、$\hat{\boldsymbol{k}}=(\hat{k}_x,\hat{k}_y,\hat{k}_z)^T$とすると、視点座標系からワールド座標系への回転行列$\boldsymbol{R}$は$\boldsymbol{R}=(\hat{\boldsymbol{i}},\hat{\boldsymbol{j}},\hat{\boldsymbol{k}})$と表せる。ここで$\hat{\boldsymbol{k}}$は投影面の法線であるため仮想視点からの視線ベクトルより計算可能である。図4に示すようにワールド座標系のy軸正方向と視点座標系のupベクトルが平行になる様に視点座標系を定める。すると、ワールド座標系上における視点座標系のx軸$(\boldsymbol{X}_V)$はワールド座標系のxz平面に平行となるから$\hat{\boldsymbol{i}}=(\hat{i}_x\ \hat{i}_y\ \hat{i}_z)^T=(\hat{i}_x,0,\hat{i}_z)^T$となる。従って、

$$\hat{\boldsymbol{i}}\cdot\hat{\boldsymbol{k}}=\hat{i}_x\hat{k}_x+\hat{i}_z\hat{k}_z \qquad \cdots(3)$$

一方、$\hat{\boldsymbol{i}}$と$\hat{\boldsymbol{k}}$は直交しているから、

$$\hat{\boldsymbol{i}}\cdot\hat{\boldsymbol{k}}=0 \qquad \cdots(4)$$

故に、

$$\hat{i}_x\hat{k}_x+\hat{i}_z\hat{k}_z=0 \qquad \cdots(5)$$

つまり、

$$\hat{i}_x:\hat{i}_z=\hat{k}_z:-\hat{k}_x \qquad \cdots(6)$$

となる。従って、$a=1\Big/\sqrt{\hat{k}_z{}^2+\hat{k}_x{}^2}$とすると、$\hat{\boldsymbol{i}}=(\hat{i}_x,0,\hat{i}_z)^T=a(\hat{k}_z,0,-\hat{k}_x)^T$となる。また、$\hat{\boldsymbol{j}}$は$\hat{\boldsymbol{i}}$と$\hat{\boldsymbol{k}}$の外積により求められるから、$\hat{\boldsymbol{j}}=\hat{\boldsymbol{k}}\times\hat{\boldsymbol{i}}=a(-\hat{k}_x\hat{k}_y,\hat{k}_x{}^2+\hat{k}_z{}^2,-\hat{k}_y\hat{k}_z)^T$となり、$\boldsymbol{R}$は次式で求められる。

$$\boldsymbol{R}=(\hat{i},\hat{j},\hat{k})=\begin{pmatrix} a\hat{k}_z & -a\hat{k}_x\hat{k}_y & \hat{k}_x \\ 0 & a(\hat{k}_x{}^2+\hat{k}_z{}^2) & \hat{k}_y \\ -a\hat{k}_x & -a\hat{k}_y\hat{k}_z & \hat{k}_z \end{pmatrix} \cdots(7)$$

上記の通り、$\hat{\boldsymbol{k}}=(\hat{k}_x,\hat{k}_y,\hat{k}_z)^T$は仮想視点からの視線ベクトルより求めることができるため、式(7)で視点座標系からワールド座標系への回転行列$\boldsymbol{R}$が求められる。本計算において求めるベクトル$\boldsymbol{V}_e$は原点からのベクトルであるため平行移動成分は考慮する必要がなく、$\boldsymbol{R}$は$\boldsymbol{M}$に一致する。一方、平面分割された各平面の頂点座標と仮想視点を通過する直線が投影面と交わる点を点Pとすることで、ワールド座標系におけるベクトル$\boldsymbol{V}_p$は求められる。また、$\boldsymbol{V}_o$は仮想視点の注視点位置として求められる。このようにして、$\boldsymbol{V}_e$は式(2),及び式(7)を用いて計算できる。

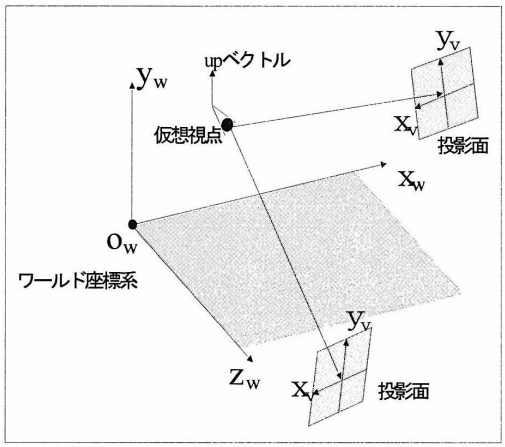


図4　視点座標系とワールド座標系の関係

3. シミュレーション結果

3.1. 提案手法による表示

提案手法を適用してシミュレーションを行った。なお、使用した計算機の仕様は表1の通りである。

表1　計算機の仕様

CPU	Intel Xeon 3.0GHz
メインメモリ	2.0GB
OS	Microsoft Windows XP
グラフィックスライブラリ	OpenGL

また、図5にシミュレーションで使用したモデル全体の鳥瞰図を示す。ポリゴンの総数は2,009であり、図5の窓枠に平面ガラス(曲率半径R＝∞)と、曲率半径R＝200、100、及び80の曲面ガラスを設置し、室内のりんごが自由落下して動的に変化する環境がガラス面に映り込む様子を、室内の中心から見たときの様子をシミュレーションした。このときのスナップショットを図6に示す。

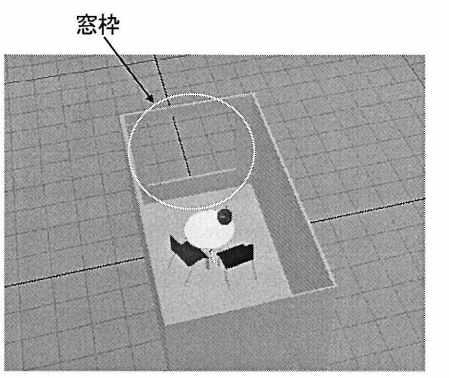
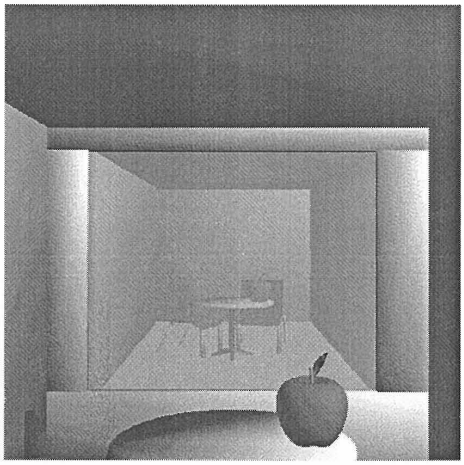
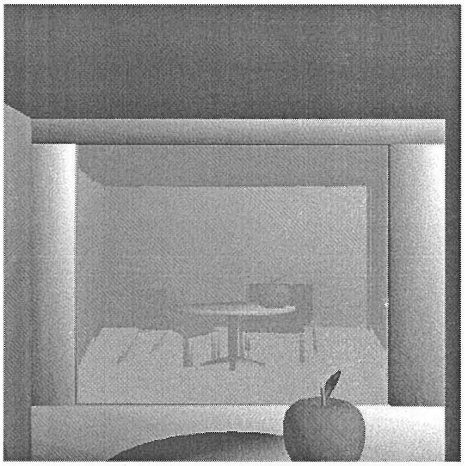


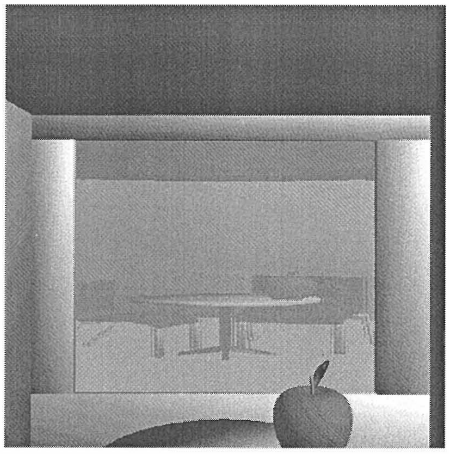
図5 シミュレーションモデルの鳥瞰図



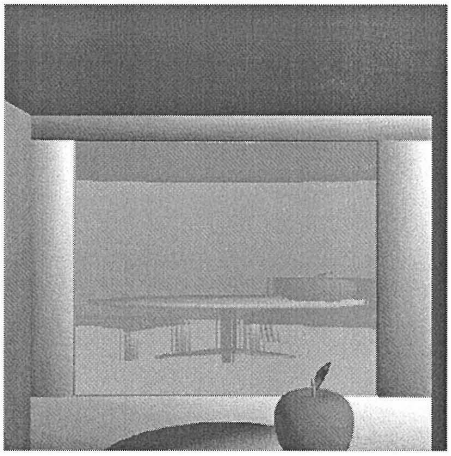
(a) 平面ガラス(R=∞)



(b) R=200



(c) R=100



(d) R=80

図6　本手法による映り込み

図6より、曲率半径が小さいほど映り込み映像は横に伸びて歪んでいることが判る。また、シミュレーションの処理性能を表2に示す。

表2　シミュレーションの処理性能

曲率半径 R	∞	200	100	80
分割平面数[枚]	1	20	36	40
処理時間[ms]	134	2,287	3,977	4,594
フレームレート [fps]	7.47	0.44	0.25	0.22

表2より分割平面数と処理時間の関係を図7に示す。

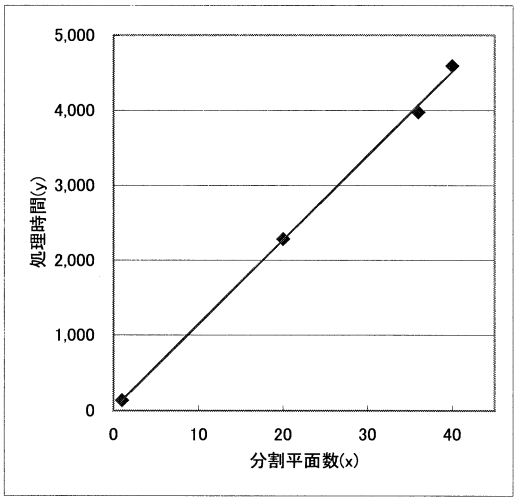


図7　分割平面数と処理時間の関係

曲率半径が小さいほどガラス面の歪みが大きくなるため、分割平面数が増加する。分割平面数に依存して映り込み用画像生成の時間が増加するため、処理時間がかかり、フレームレートは減少する。

3.2. 環境マッピング法との比較

次に、本手法と環境マッピング法の比較を行った。環境マッピング法は周囲環境を変化させずにシミュレーション行った。分割平面数 1 の映り込み面に対して、本手法と環境マッピング法で必要なテクスチャマッピング用画像生成時間と描画時間を表3に示す。

表3　テクスチャマッピング用画像生成時間と描画時間

	本手法	環境マッピング法
画像生成時間[ms]	64	2,212
描画時間[ms]	70	47

図 8 に本手法と環境マッピング法における分割平面数と処理時間の関係を示す。但し、直線の方程式は近似式である。

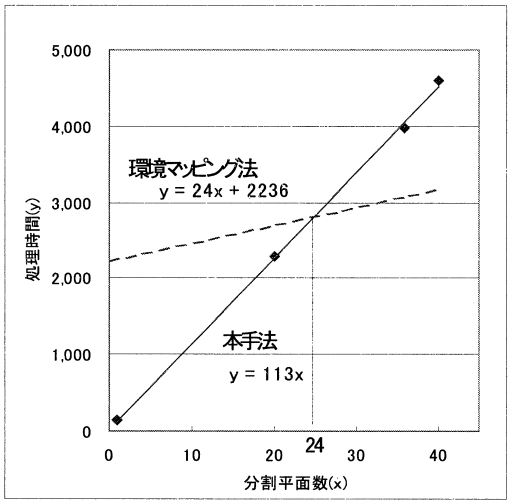


図8　本手法と環境マッピング法における処理時間の比較(周囲環境が変化しない場合)

図8より、周囲環境が変化しない場合、分割平面数が 24 枚以下であれば、本手法は環境マッピング法よりも高速である。また、フレーム数を f、分割平面数を x、処理時間を y とすると、本手法は $y=(113x)f$ であるのに対し、環境マッピング法では $y=(24x+2236)f$ となるので、本手法は、周囲環境が変化する場合、環境マッピング法に比べてかなり高速であることが判る。

4. まとめ

本報では、曲面を平面分割し、分割された各平面に対して鏡像反転した位置に仮想視点を設け、仮想視点からの映像をテクスチャマッピングすることにより、動的に変化する環境の映り込みを実現した。本手法の性能は分割平面数に依存するが、周囲環境が変化しない場合、分割平面数が 24 枚以下であれば、環境マッピング法よりも高速であり、周囲環境が変化する場合、環境マッピング法に比べてかなり高速であることが判った。今後は、GPU を用いた高速化について検討する予定である。

参考文献

[1]J.F.Blinn and M.E.Newell, “Texture and Reflection in Computer Generated Images,” Commun. ACM, pp.542-547 (1976)

[2]奥村文洋, 町田貴史, 横矢直和, “実写に基づく全周多重スケール環境マップを用いた不均一物体への写りこみの高速レンダリング,” 信学技報, PRMU2002-137, pp.49-54 (2002)

[3]Nobuhiko Mukai, Mariko Shida, Makoto Kosugi, “A real-time rendering method of dynamic reflection on a transparent object,” WMSCI2008, pp.130-135 (2008)