没入型ディスプレイでの人間の視野を利用したレイトレーシング法の詳細度制御

An Efficient Ray Tracing with LOD Control by Human Eyesight for Immersive Display System

望月 直樹† 牧野 光則†

†中央大学大学院 理工学研究科 情報工学専攻

概要

本研究では, 没入型立体視ディスプレイでのレイトレーシング法の計算の高速化を行うために, 没入型ディスプレイでの人間の視野を利用したレイトレーシング法の詳細度制御手法を提案する. 提案手法では, 視点から発せられるレイが視野のどの領域に属するかを判定し, レイごとに詳細度を設定する. この詳細度に従い, 計算量に強く影響を及ぼすレイの数, 物体のポリゴン数を制御することで計算量の削減を図る.

1 序論

コンピュータグラフィックス (CG : Computer Graphics) は, 現在様々な分野への応用も含めて研究されている. CG の利用技術の一つに, バーチャルリアリティ(VR : Virtual Reality) がある. VR の中でも CAVE(Cave Automatic Virtual Environment)[1] のような没入型立体視システムは, 臨場感や没入感を得ることができ, 3 次元映像や視覚的情報などの情報理解にも有効である. 今後も没入型立体視システムの需要の拡大が予想されるが, 文化財の可視化など忠実な質感再現が求められるコンテンツも増えており, 仮想空間を現実的に表現することが重要な課題の一つとなっている.

レイトレーシング法 [2] は, CG 技術の中でもよく知られているレンダリング手法である. レイトレーシング法は, 視点から視線 (レイ) を飛ばし, 人間の目に入る光線を順次逆追跡していくことで描画する手法であり, 光の反射 · 屈折や陰影を伴う CG 画像を生成できる. 光の反射 · 屈折などを忠実に再現できることから, 高質感な画像や写実的な画像を生成する際によく用いられる. このようにレイトレーシング法は写実的に表現することに適しているが, より写実的な画像を生成するには多くのレイを追跡する必要があるため, 計算量が多くなる. また, レイの数は主に解像度や画像の対象となる物体数に依存する. これにより, インタラクティブな操作が求められる VR システム, 特に複数のディスプレイで構成されることが多い没入型立体視ディスプレイでのレイトレーシング法の利用はあまり見られない.

一方で, 広範囲な没入型立体視ディスプレイの描画領域に対して, 人間の視野範囲は狭いため, 操作者の視野外, または注視していない領域が存在する. この領域を注視領域と同等の画質で描画せずとも, 操作者にとって違和感のない画像生成が可能である.

写実的な画像を生成できるレイトレーシング法を没入型立体視ディスプレイで利用することは, 質感の忠実な再現, 没入感や臨場感の向上に有効である. しかしながら, VR システムではインタラクティブな操作が求められるので, リアルタイムで描画することが望まれる. そのため, レイトレーシング法を利用するには, 操作者にとって違和感のない画像を表示しつつ, 計算の高速化を行う必要がある.

よって本研究では, 没入型立体視ディスプレイでのレイトレーシング法の計算の高速化を行うことを目的とする. この目的を達成するために, 没入型ディスプレイでの人間の視野を利用したレイトレーシング法の詳細度制御手法を提案する. 提案手法では, 視点から発せられるレイが視野のどの領域に属するかを判定し, レイごとに詳細度を設定する. この詳細度に従い, 計算量に強く影響を及ぼすレイの数, 物体のポリゴン数を制御することで計算量の削減を図る.

2 人間の視野 [3]

真正面の 1 点を注視した状態で, はっきりしなくても見える範囲を視野という. 人間の視野は視線方向との間の角度で例えられる.

人間の視野は, 視力特性などの機能面と情報探索特性から, 中心視野, 周辺視野に分類できる. 中心視野は眼球運動だけで注視, 瞬時に情報受容ができる領域である. また, 周辺視野は識別能力が低下するが, 視覚情報に基づく主観的な空間座標系に影響を及ぼし, 臨場感などの心理的な効果が引き起こされる領域である. 視野の広さは個人差はあるが, 日本人の平均では上方約 $\frac{1}{3}\pi$rad, 下方約 $\frac{7}{18}\pi$rad, 鼻側約 $\frac{1}{3}\pi$rad, 耳側約 $\frac{5}{9}\pi$rad の範囲となっている (図 1).

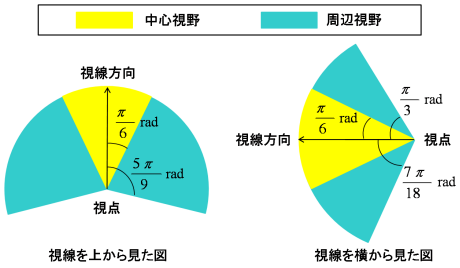

図 1: 中心視野 · 周辺視野

3　提案手法

3.1　概要

本研究では, 没入型ディスプレイでの人間の視野を利用したレイトレーシング法の詳細度制御手法を提案する. 先行研究として, 松本ら [4] は, 人間の視野を利用し, 物体の位置と距離によって詳細度を決めることで計算量の短縮を図っている. しかしながら, レイトレーシング法では反射 · 屈折の表現があり, 松本らの手法ではこれに対応できない. よって, 本手法では, 視点から発せられるレイが視野のどの領域に属するかを判定し, レイごとに詳細度を設定する. これにより, 反射 · 屈折の表現に対応する. また, レイトレーシング法ではレイの数, 物体のポリゴン数が計算量に強く影響を及ぼす. 両者を詳細度により制御することで計算量の削減を図る. さらにレイトレーシング法の高速化手法であるバウンディングボリューム階層を適用することでさらなる高速化を図る.

3.2　前提条件 · 前処理

3.2.1　プログレッシブメッシュ

本手法では, レイごとに詳細度を設定するので, 画素間で詳細度の違いが生じ, 画素ごとにポリゴン数が異なる物体を表示することになる. できるだけ画素間の物体の差を少なくするために, 本手法ではメッシュの連続的多重解像度表現であるプログレッシブメッシュ[5] を用いる.

3.2.2　バウンディングボリューム階層

既存のレイトレーシングの高速化手法は多く存在し, それらを導入できればより高速化を図れる. 本手法では物体ごとにポリゴン数を変化させるので, 物体ごとに構築可能であり, 計算量の大幅低減を期待できるバウンディングボリューム階層 (BVH : Bounding Volume Hierarchy)[6] を適用する. この BVH の構築を前処理として行う.

3.3　詳細度制御法

3.3.1　詳細度の決定

詳細度は視点から放たれるレイごとに設定する. また, 視点から放たれるレイから反射 · 屈折によって生成されるレイには, 視点から放たれるレイと同じ詳細度を設定する.

操作者の視点位置と視線方向の情報によって, 画素 i を通るレイ R_i の詳細度 $level_i$ を決定する. まず, 図 2 に示す二つの角度 $m_{\alpha,i}$ と $m_{\beta,i}$ を視線方向とレイの方向ベクトルから求める. 次に, R_i が中心視野, 周辺視野, 視野外のどの領域に属するかを判定する.

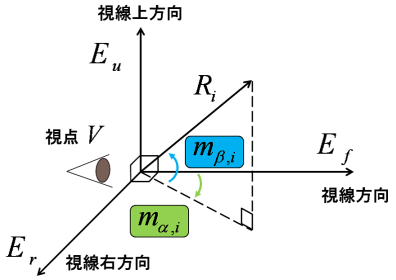

図 2: 視線方向とレイの間の角度

ワールド座標系における R_i の方向ベクトル D_i' を, 視点 V を原点, 視線方向, 視線右方向, 視線上方向を軸とするローカル直交座標系のベクトル $D_i(d_{i,f}, d_{i,r}, d_{i,u})$ に変換する. ここで, 視線方向を示す単位ベクトルを E_f, 視線右方向を示す単位ベクトルを E_r, 視線上方向を示す単位ベクトルを E_u とする. このとき, D_i は式 (1)〜(3) で求まる.

$$d_{i,f} = E_f \cdot D_i' \tag{1}$$

$$d_{i,r} = E_r \cdot D_i' \tag{2}$$

$$d_{i,u} = E_u \cdot D_i' \tag{3}$$

以上で求めた D_i から, R_i に対する左右の角度 $m_{\alpha,i}$ と上下の角度 $m_{\beta,i}$ はそれぞれ以下の式で求まる.

$$m_{\alpha,i} = \cos^{-1}\left(\frac{d_{i,f}}{\sqrt{d_{i,f}^2 + d_{i,r}^2}}\right) \tag{4}$$

$$m_{\beta,i} = \begin{cases} \cos^{-1}\left|\dfrac{d_{i,f}}{\sqrt{d_{i,f}^2 + d_{i,u}^2}}\right| & (d_{i,u} \geq 0) \\ -\cos^{-1}\left|\dfrac{d_{i,f}}{\sqrt{d_{i,f}^2 + d_{i,u}^2}}\right| & (d_{i,u} < 0) \end{cases} \tag{5}$$

2 節に基づき, 求めた $(m_{\alpha,i}, m_{\beta,i})$ から, R_i が視野のどの領域に属するかを決定する. まず, α, β を軸とした直交座標系における原点から点 $(m_{\alpha,i}, m_{\beta,i})$ までの距離 l_i を以下の式で求める (図 3).

$$l_i = \sqrt{m_{\alpha,i}^2 + m_{\beta,i}^2} \tag{6}$$

次に点 $(m_{\alpha,i}, m_{\beta,i})$ を通り, 原点, 周辺視野と視野外の境界を両端とする線分の長さ L_i を式 (7) で求める.

$$L_i = \begin{cases} \sqrt{\dfrac{\left(\frac{5\pi}{9}\right)^2 m_{\alpha,i}^2 + \left(\frac{\pi}{3}\right)^2 m_{\beta,i}^2}{m_{\alpha,i}^2 + m_{\beta,i}^2}} & (m_{\beta,i} \geq 0) \\ \sqrt{\dfrac{\left(\frac{5\pi}{9}\right)^2 m_{\alpha,i}^2 + \left(-\frac{7\pi}{18}\right)^2 m_{\beta,i}^2}{m_{\alpha,i}^2 + m_{\beta,i}^2}} & (m_{\beta,i} < 0) \end{cases} \tag{7}$$

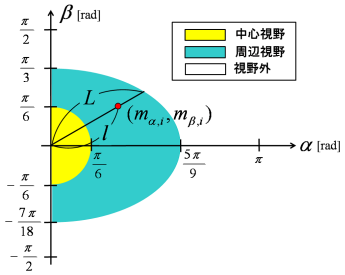

図 3: α, β を軸とした直交座標系

以上で求めた値から R_i は, $l_i \leq \frac{\pi}{6}$ の場合は中心視野内, $\frac{\pi}{6} < l_i < L_i$ の場合は周辺視野内, それ以外なら視野外とする. R_i が属する領域に従い, 詳細度 $level_i$ を以下の式で求める.

$$level_i = \begin{cases} 1.0 & (l_i \leq \frac{\pi}{6}) \\ \dfrac{L_i - l_i}{L_i - \frac{\pi}{6}} & (\frac{\pi}{6} < l_i \leq L_i) \\ 0.0 & (L_i < l_i) \end{cases} \tag{8}$$

3.3.2 レイの本数 · 物体のポリゴン数の制御

レイは交差する反射 · 屈折により最大 2 方向に分岐する. 反射 · 屈折の回数を n とすると, 1 本のレイから生成されるレイの数は最大 $2^{n+1} - 2$ となる. このことから, レイの本数は反射 · 屈折の回数に強く依存していることがわかる. よって, 本手法では反射 · 屈折回数を削減することでレイの本数を制御する. R_i の最大反射 · 屈折回数 $Reflect_i$ は以下の式で求める.

$$Reflect_i = level_i \times Reflect_{max} \tag{9}$$

このとき, $Reflect_{max}$ は全てのレイにおける反射 · 屈折回数の最大値である. また, 操作者にとって視野外の領域はほぼ見えないので, $level_i = 0$ の場合はレイの追跡を行わず, 画素 i には背景色を設定する.

同様に物体のポリゴン数によって交差判定の回数が大きく変わることから, 詳細度によってポリゴン数を制御する. 物体 $O_j (j = 1, 2, ..., n)$ のポリゴン数 N_j は以下の式で求める.

$$N_j = level_i(N_{j,max} - N_{j,min}) + N_{j,min} \tag{10}$$

このとき, $N_{j,max}$, $N_{j,min}$ はそれぞれ O_j の最大ポリゴン数, 最小ポリゴン数である.

3.3.3 アルゴリズム

本手法による詳細度制御付きレイトレーシング法のアルゴリズムを以下に示す.

Step 1: 操作者の視点位置と視線方向を取得する.

Step 2: 画素 i を通るレイ R_i の詳細度 $level_i$ を求める (3.3.1 節). このとき, $level_i = 0$ なら, 以降の Step を行わず画素 i に背景色を設定する.

Step 3: $level_i$ に従い, 式 (9) により R_i の最大反射 · 屈折回数 $Reflect_i$ を決定する.

Step 4: レイ R_i の追跡を開始する. このとき, レイと交差判定を行う物体は式 (10) によってポリゴン数を制御する.

Step 5: 交差する物体がある場合, 輝度値を求める. ない場合は追跡を終了する.

Step 6: 反射 · 屈折がある場合, 反射 · 屈折方向に詳細度を $level_i$ とするレイを生成し, Step 4〜6 を行う. 反射 · 屈折がない場合, または反射 · 屈折回数が $Reflect_i$ を超える場合は追跡を終了する.

以上の Step 2〜6 を, 全ての画素に対して行う.

3.4 特徴

本手法では, 人間の視野に従い詳細度を決定するため, 操作者の注視している領域では詳細に, 意識があまり向いてない領域では簡略に物体を表示している. この詳細度制御によって, レイトレーシング法の計算量に強く影響を及ぼす交差判定の回数を減らしている. また, レイごとに詳細度を設定しているため, 反射 · 屈折による表現に対しても適切な詳細度制御を行っている. 一方で, 本手法ではプログレッシブメッシュを前提としており, 事前の準備が必要である. さらに, レンダリング速度を上げるために BVH を構築しているので, 前処理時間を要する.

4 実装

4.1 実装環境

提案手法の実装には中央大学に導入されている CAVE システムを使用する. 中央大学の CAVE システムは, 前面, 右面, 下面の 3 面スクリーンで構成される没入型立体視ディスプレイである. ユーザは時分割液晶シャッタ立体視メガネを利用することで立体視が可能となる. また, トラッキングセンサによって操作者の視点位置や視線方向が測定される. さらに, スクリーン各面の計算, 表示に PC を 2 台ずつ, 合計 6 台使用する. PC の環境を表 1 に示す.

4.2 実装準備

本研究では, 頂点数 2904, ポリゴン数 5804 の 3 角形ポリゴンモデルである, cow モデルを配置した仮想空間を描画することで描画時間を測定する. また, 中心視野内に存在する物体の数や鏡面物体の有無が提案手法による処理時間に影響を及ぼすと考えられる. そこで, 鏡面物体を各スクリーンに均等に映るように配置したシーン (シーン A), 拡散物体を各スクリーンに均等に映るように配置したシーン (シーン B), 鏡面物体を正面スクリーンにのみ

表 1: 実装環境

PC	HP xw9300 Workstation(6 台)
CPU	2.59GHz (Dual Core AMD Opteron(tm) Processor 285)
Memory	4.00GB RAM
Graphics card	NVIDIA Quadro FX 4500
OS	Microsoft Windows XP Professional x64 Edition

映るように配置したシーン (シーン C), 拡散物体を正面スクリーンにのみ映るように配置したシーン (シーン D) の 4 つのシーンを用意する. また, 各シーンは, 物体数が 8, 光源が 1 つ (点光源), ピクセル数が 256×256pixels×6 枚, $Reflect_{max} = 3$, $N_{j,max}(j = 1, 2, ..., 8) = 5804$, $N_{j,min}(j = 1, 2, ..., 8) = 500$ である.

4.3 実行結果

視線を正面スクリーンに向けた状態 (状態 1) と下面スクリーンに向けた状態 (状態 2) の時にシーン A を描画した風景を図 4, 5 に示す.

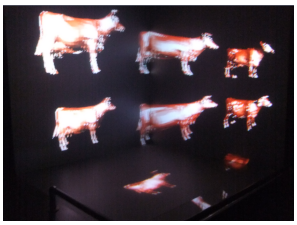

図 4: シーン A, 状態 1

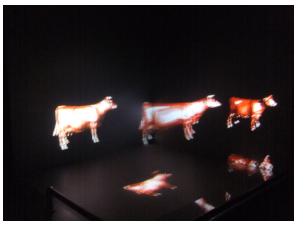

図 5: シーン A, 状態 2

4.4 評価 · 考察

図 4, 5 から視野外の領域は物体が表示されず, 背景色が表示されていることが分かる. 一方で, 中心視野内の物体は詳細に表示されていることが分かる.

提案手法を適用した場合と適用しない場合 (通常表示) の描画時間を表 2 に示す. 表 2 から状態 1 よりも状態 2 の時の方が描画時間は少ないことが分かる. このことにより, 中心視野外の領域における計算量を削減できたと言える. また, シーンごとに比較すると, 中心視野外に鏡面物体が存在する場合の方が描画時間の削減率が高いことが分かる. このことから, 提案手法を適用した場合, 鏡面物体が多いシーンの方が中心視野外での描画時間の削減率が高いと言える. さらに提案手法と通常表示を比較すると, 最小で通常表示の描画時間の約 27.8%で描画しており, 操作者の視線方向によっては提案手法を用いることで描画時間を短縮できると言える. 一方で物体が各スクリーンに均等に映っている場合の描画時間が通常表示よりも大きい. これは物体が各スクリーンに均等に映っている場合, ポリゴン数の変化が大きい物体が多く存在し, ポリゴン数の制御に時間を要したためであると考えられる. よって, 視野による物体のポリゴン数の制御に関しては, アルゴリズムの改良が必要であると言える.

表 2: 描画時間 [s/frame]

	A	B	C	D
提案手法 (状態 1)	2.078	1.109	0.859	0.484
提案手法 (状態 2)	0.751	0.547	0.243	0.203
通常表示 (平均)	1.641	0.969	0.875	0.532

5 結論

本研究では, 没入型立体視ディスプレイでのレイトレーシング法の計算の高速化を行うために, 没入型ディスプレイでの人間の視野を利用したレイトレーシング法の詳細度制御手法を提案した. 提案手法を CAVE で実装し, その有効性を検証した結果, 提案手法を適用することにより, 描画時間を適用しない場合の最大約 70%削減できることを示した. また, 提案手法は鏡面物体が多いシーンに対して特に有効であることを示した.

今後の課題として, 物体の数や性質, 光源の数が異なるシーンでの描画時間の追加検証, アンケート調査による見た目の違和感の評価, さらなる高速化を図るための物体のポリゴン数の制御法の改善や GPU の利用が挙げられる.

参考文献

[1] Cruz-Neira, C., Sandin, D.J., DeFanti, T.A., Kenyon, R.V., and Hart, J.C., “The CAVE: Audio Visual Experience Automatic Virtual Environment”, Communications of the ACM, Vol.35, No.6, pp.65-72, 1992.

[2] Whitted, T., “An Improved Illumination Model for Shaded Display”, Communications of the ACM, Vol.23, No.6, pp.343-349, 1980.

[3] 三橋 哲雄, 畑田 豊彦, 矢野 澄男, “画像と視覚情報科学”, コロナ社, 2009.

[4] 松本 哲也, 牧野 光則, “視野特性を考慮したポリゴンモデルの詳細度制御と高速表示”, 芸術科学会 第 21 回 NICOGRAPH 論文コンテスト 論文集, pp.31-36, 2005.

[5] Hoppe, H., “Progressive meshes”, Proc. of ACM SIGGRAPH ’96, pp.99-108, 1996.

[6] Rubin, S.M., Whitted, T., “A 3-Dimensional Representation for Fast Rendering of Complex Scenes”, ACM SIGGRAPH Computer Graphics, Vol.14, Issue 3, pp.110-116, 1980.