部分更新レイトレーシング†

村上公一†† 広田克彦†† 石井光雄†††

固定された視点の下での連続する映像に対するレイトレーシングの処理時間を，変化の起こった部分のみを再計算することによって高速化する方式を提案する．この方式は映像全体を再処理することなしに，モデルの幾何，質感，および光源変更によって変更された映像を生成することができる．本方式では，光線に着目した記述を履歴情報として保存する．この情報は再計算が必要な部分の決定や効率的な更新計算を行うために使われる．履歴情報として，光線追跡木に光線や可視物体の情報を付加した拡張光線追跡木を考案した．また，変化の影響を局所化するためにボクセルデータ構造を使う．ボクセルインデクスを用いて表現された光線の軌跡と変化ボクセル要素から再計算量を制限する．膨大な履歴データのアクセスが本方式の実現上の鍵となる．アクセス量を限定するために，ハッシング技法を導入して変化光線を高速に同定する．また選択的な履歴データのトラバースを行って，効率の良い部分更新処理を実現する．実験結果より映像全体を生成する場合に比較して，数～百倍ほどの高速化が達成された．本方式の実現によって，高品質な映像を対話的あるいは連続的に生成するような分野でのレイトレーシングの実用化が可能となると考えられる．

1. はじめに

反射・透過を含む高品質な映像を生成するレイトレーシングの問題点として生成時間の遅さが知られている．これを改善して高品質な映像が要求される分野での実用的な映像生成方式として利用するために，様々なコヒーレンスを利用する高速化方式が提案されている．近傍の画素から出た光線は似た振舞をする性質(レイコヒーレンス)をレイトレーシングに適用した例として，beam-tracing[1]，画素選択光線追跡法[2]がある．また，オブジェクト空間コヒーレンスを利用した手法として，空間を分割して光線追跡処理の大半を占める交差計算の回数を減少させる方式[3)~5)]もある．

これらの高速化を行って数倍から数十倍の高速化が達成されているが，対話的な映像生成に適用できるほどの性能には至ってはいない．これらの方式は一枚の映像を高速に生成することに主眼を置いているが，対話的，あるいは連続的に映像生成を行うような場合には別のアプローチが有効と考えられる．

映像生成処理を対話的に行う場合を考えよう．例えば，インテリアモデルの設計において，「ある部品の色を変えて，次に椅子の移動」という様に，デザイナはあるモデルを少しずつ修正して所望の映像を得ると考えられる．少しの修正を反復するような映像生成に対しては，フレーム間での変化部分を検出し，その部分のみを再計算する方式が有効であると考えられる．また，アニメーションにおいても連続するフレーム間での変化はわずかであるので，この方式によって大幅な高速化が達成される．

上で述べた性質はフレーム間コヒーレンスとして知られている[6)]．しかし，この性質をレイトレーシングに適用した例はない．レイトレーシングでは，物体変化の影響はその物体に留まらずに広い範囲に及ぶ．例えば，ある物体が別の物体に映っている時，物体の移動に伴って反射像や影も同時に変化しなければならない．

筆者らは，固定された視点の下で変化した部分のみの再計算を行う部分更新レイトレーシング方式を提案し，これを実現した．この方式は，(1)移動や回転等のモデル変更，サイズの変更，および物体の追加・削除等の幾何に関する変更，(2)色，反射，マッピング等の質感パラメータ値の変更，(3)光源の移動やモデルの移動に伴う影の変更等を統一された枠組みで処理することができる．

本方式では，フレームの光線に着目した記述を履歴情報として保存する．この情報を使って再計算が必要な部分を決定し，効率的な更新計算を行う．履歴情報を表現するデータ構造として，(1)光線追跡木に光線や可視物体の情報を付加した拡張光線追跡木，(2)ボクセルデータ構造を用いて表現された前フレームの光線経路の履歴，(3)前フレームの交点履歴，を考案した．これらの情報は光線に依存しているために固定された視点の下でしか利用することができないが，映像

† Incremental Ray Tracing by KOICHI MURAKAMI, KATSUHIKO HIROTA (Humaninterface Lab., Fujitsu Laboratories Ltd.) and MITSUO ISHI (Computer-Based Systems Lab., Fujitsu Laboratories Ltd.).

†† (株)富士通研究所ヒューマンインタフェイス研究部
††† (株)富士通研究所システム研究部

全体の再計算に比べて更新処理を大幅に高速化することができる．

2. 部分更新方式

2.1 概　　要

本方式での処理は図1に示すように，光線追跡処理で最初のフレームを生成する初期ステップと，その後の連続するフレームの変更を行う部分更新ステップに分けられる．初期ステップの結果，全体の映像と最初の履歴情報が生成される．部分更新ステップでは，モデルと質感の更新に対する映像を生成し，また次のフレームのための履歴データを管理する．部分更新ステップは視点が変わらない限り続けられる．部分更新ステップの処理時間は初期ステップのそれに比べて短いので，対話的なモデルの更新が可能となる．

本論文では，ある光線の輝度が変化した時に光線が変化したという．また，光線集合をある光線から分岐する全光線の集合と定義する．ある画素の輝度値は，そこを通過する一次光線の光線集合内の一つの光線に変化があった場合に更新される．議論を明確にするために，モデルと質感の変更による処理を分けて説明するが，これらは一つの枠組みで統合化される．

モデルの変更に対する部分更新処理のアルゴリズムの概略を示す．更新処理は，変化光線の検出から始まる．拡張光線追跡木をトップダウンにトラバースしながら，各々の光線の変化を調べる．注目光線が変化する場合には，可視点の再計算を行って新たな輝度値を決定する．分岐光線の変化の影響を画素に反映させるために，更新された輝度値をボトムアップに伝播して画素の輝度値を更新する．

変化光線を効率的に判定するために，ボクセル分割を用いて変化物体の影響を局所化する．前フレームの光線が通過したボクセル要素の集合と変化の起きたボクセル要素集合を調べることによって，変化の可能性のある光線を選択的に処理することができる．また，前フレームの交点を記録した履歴情報を用いて計算コストが大きい交差計算の回数を減少する．

2.2 データ構造

履歴データ構造として，拡張光線追跡木，通過ボクセル履歴，交差情報履歴を考案した．これらのデータは図2に示すように階層構造として構成されており，変化光線の検出と再計算量を制限するために使われる．

（1） 拡張光線追跡木

ある画素から始まる光線集合に対して一つの光線追跡木が生成される．従来の光線追跡木のノードには，分岐光に対するポインタやそのノードが表している可視点の輝度値等を記録していたが，拡張光線追跡木ではノードに

- 光線パラメータ（光線の方向，視点）
- 可視点とその法線ベクタ
- 他のデータ構造（後述）へのポインタ
- ハッシュインデクス（後述）

等の情報を付加する．光線の振舞は，この拡張光線追跡木を用いて表現することができる．

（2） 通過ボクセル履歴

筆者らが開発したレイトレーシングシステム[3]では，高速な処理を行うためにボクセル分割を用いて光線と物体の交差計算数を減少させた．そこでは物体の存在領域を局所化していたが，部分更新レイトレーシングでは物体変化の影響領域を局所化する．通過ボクセル履歴は一つの光線ごとに生成され，拡張光線追跡

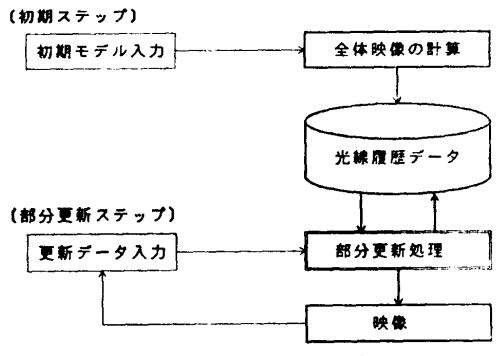


図1 二つの処理ステップ
Fig. 1 The two steps of image generation.

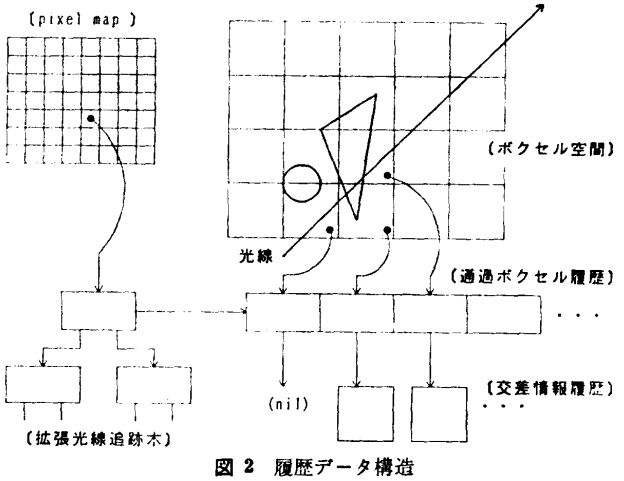


図2 履歴データ構造
Fig. 2 History data structure.

木の一つのノードからリンクされる．通過ボクセル履歴は，注目する光線の始点から可視点の存在するボクセルまでのボクセルインデクスを記録したものである．

（3） 交差情報履歴

交差情報履歴はモデル変更の際の再交差計算数を制限するために使われる．あるボクセル要素内に物体が存在する場合は交差情報履歴が生成されて，光線と交わる物体との交点，法線ベクタ，物体識別子が記録される．このデータ構造は通過ボクセル履歴からリンクされる．

（4） 他のデータ構造

ここでは，部分更新処理を実現するために従来のデータ構造に付加したデータ項目について述べる．

・従来のボクセルデータ構造の各要素に変化フラグを設けた．このフラグでその要素内に変化があることを示す．物体が移動する場合には，前フレームに存在していたボクセル要素と現フレームの要素に変化フラグを立てる．

・物体データベースに物体の変化を表す変化フラグを付加する．

2.3 変化光線の検出

可視点の再計算を行う基本的な方式として，すべての光線と変化物体の間で再交差計算を行うことが考えられる．しかし，この方式での計算コストは，N を変化した物体の数，R を全光線数とすると，$O(NR)$ となる．一般に，R はシーンの複雑度を表すため，この方式では複雑なシーンにおける部分変更の効率が悪い．使用形態を考慮すると，処理時間が変化物体の数に比例することが望ましい．

ある物体の変化がすべての光線に影響を与えるわけではないことに注意されたい．**図3**の例では，物体Bの変化の影響は光線 r1 に影響を与えていない．効率の良い部分更新方式として，物体変化の影響をボクセル分割を用いて局所化することを提案する．図3はボクセル分割を用いて物体Bの変化の影響を斜線で示す領域に局所化した様子を二次元的に説明したものである．フレームごとの前処理として，図中の斜線で示した変化ボクセルの算出を行う．同時に，物体データベース中の変化物体の変化フラグを立てる．現フレームの変化ボクセル要素が前フレームの光線の通過ボクセル経路に含まれる時，これらの積集合の部分で光線が変化する可能性がある．

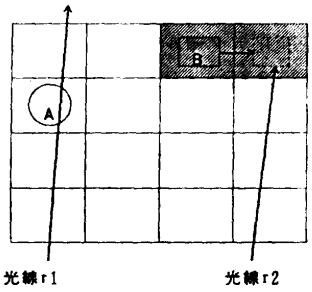


図 3 変化の影響の局所化
Fig. 3 Localization of the influence of change.

2.4 可視点の再計算

変化光線の新たな可視点を決定するために，注目光線の通過ボクセル履歴を用いて変化が起こったボクセル要素まで注目点を進める．履歴を用いるため，3 DDDA[3] アルゴリズムを用いた場合に比べて高速なトラバースができる．注目点が変化ボクセルに進んだ時に，そのボクセル要素内の物体と注目光線の間で再交差計算を行う．

交差履歴を用いることによって，変化しない物体に対する交差計算を省け，コストの大きい交差計算の回数を押えることができる．前フレームで可視点となっていた物体が移動し，それに隠されていた物体が新たな可視物体になるとしよう（**図 4**）．この場合，変化しない物体の交差情報は交差履歴に記録されているのでこの物体との交差計算は必要ない．新たな可視点は，交差履歴に記録されている変化しない物体に対する交点と変化物体に対して再交差計算を行って求めた交点とから，距離に基づいて決定される．

2.5 新規光線の生成

可視点が変った時は分岐光の始点や傾きが変るので，履歴情報を使って処理を進めることはできない．下に示す場合には履歴データを使った部分更新処理から分岐光の生成や追跡等のレイトレーシング処理を行う．

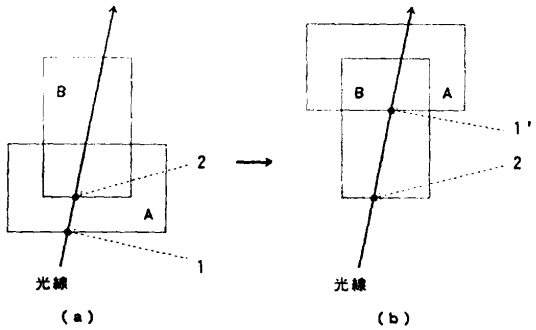


図 4 交差情報履歴の使用
Fig. 4 The use of the intersection history.

（1）可視点の位置が変る時.

（2）法線マッピングによって分岐光の傾きが変る時.

（3）屈折率の変更に伴って透過光の傾きが変る時.

（4）可視点からの分岐光がない状態から，質感パラメータの変更によって分岐光が発生する時.

3. 履歴データの効率的なアクセス方式

履歴データは膨大な量（数十MB）であるので，高速な更新処理を実現するためには効率的なアクセス方式が重要となる．ここでは，（1）効率的な変化光線の検出方式，（2）拡張光線追跡木の選択的トラバース方式を示す.

3.1 ハッシングによる変化光線の検出

部分更新ごとの変化物体の数は少なく，変化光線は少ないと仮定されるために，ほとんどの光線には変化が無いと考えられる．これを考慮すると，変化を検出するために全光線の通過ボクセル履歴を調べることは効率的でない．ここでは，ある光線集合内の光線がモデル変化の影響を受けるかどうかを即座に判定できる方式を与える.

ある光線が通過したボクセルインデクス集合をハッシングし，ハッシュ値（以下，ハッシュインデクス）で光線の通過したボクセル経路を表す方式を提案する．これは光線の簡潔な軌跡の表現と考えられるので，光線の通過経路の照合に使うことができる．このインデクスをすべての光線について計算し，拡張光線追跡木のノードに記録する.

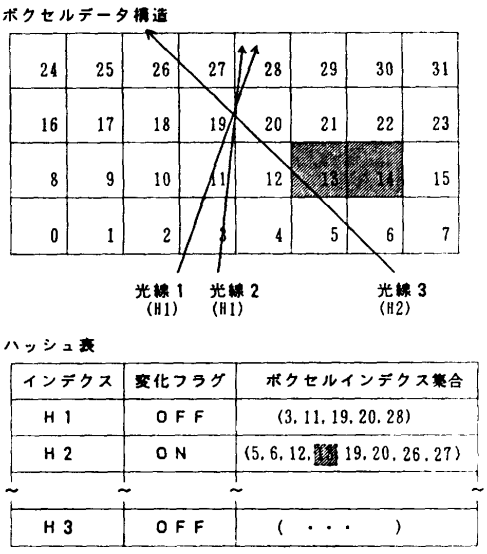


図 5 ボクセルデータ構造とハッシュ表
Fig. 5 Voxel data structure and Hash table.

図5に示すようにハッシュ表の一つの項目は，ハッシュインデクス，変化フラグ，ボクセルインデクス集合で構成される．更新ステップごとの前処理として，変化ボクセルインデクスを求める．次に，すべてのハッシュインデクスに対して，それが持つボクセルインデクス集合を走査する．変化ボクセルがボクセルインデクス集合に含まれる時には，対応するハッシュインデクスの変化フラグを ON に設定する．更新処理において，このフラグを調べることによって，光線の変化を即座に知ることができる.

レイコヒーレンスの性質によってハッシュインデクスの数は光線の数にくらべて縮退する．すなわち，同じボクセルインデクス集合を持つ幾つかの光線を一つのハッシュインデクスで表現できる．この性質を使うと変化ボクセルがボクセルインデクス集合に含まれるかを調べる判定計算の回数を制限できる．すべての光線に対する処理と比較すると，縮退した分だけ判定計算の回数は減少する.

図5で，ハッシュインデクスの生成と使い方を簡単に説明する．ここではボクセルインデクスを三次元表現から適当な変換によって一次元表現へ直している．光線3の通過したボクセルインデクスの列は（6 5 13 12 20 19 27 26）である．ここで，変化光線を検出するためには順序関係は重要でないために，ボクセルインデクス列をソートしてボクセルインデクス集合を作る．この操作によって組合せの数を少なくすることができるので，ハッシュ表の大きさを小さくする効果もある.

次に，ハッシュ関数を上で得られたボクセルインデクス集合に適用してハッシュインデクス H1 を得る．ここで，ハッシュ関数には線形検査法[7]を用い，表の大きさはモデルに依存して可変長とした．光線2は光線1と同一のボクセル経路を通過したので，光線1と同じハッシュインデクスを持つ．今，ボクセルインデクス "13" を変化ボクセルとすると，前述した前処理によってこれを含むハッシュインデクス H2 の変化フラグが ON になる.

3.2 光線追跡木のトラバース方式

前節で述べた方式によって効率的な変化光線の検出が可能となったが，変化光線を探索するために拡張光線追跡木のすべてのノードのトラバースが必要である．網羅的なトラバースでは効率的な部分更新処理は達成されない.

前節で述べた処理によって各々の光線に一つのハッシュインデクスが割り付けられる．ここで，分岐光線の積算ハッシュインデクスを分岐光線から始まるすべての光線に対するハッシュインデクスと定義する．この値はある光線から分岐したすべての光線の通過経路を表現している．積算ハッシュインデクスは分岐光線のボクセルインデクス集合の和集合に対してハッシュ関数を適用して求める．ここで，分岐光線の任意のレベルに対して積算ハッシュインデクスが定義されることに注意されたい．

積算ハッシュインデクスはトラバースの際に変化しそうな分岐光のノードを選択的に導くために使われる．ある光線集合の積算ハッシュインデクスの変化フラグは，この光線集合内の光線の変化の可能性を示唆している．例えば，ある画素を通過する一次光線に対する積算ハッシュインデクスの変化フラグを調べることによって，この画素の輝度値が変化するかを即座に知ることができる．同様に，任意の分岐光に対する積算ハッシュインデクスの変化フラグが OFF の場合には，この分岐光に対するノードをトラバースする必要はない．

可視点が変化した時には，引き続き再輝度計算が行われて，変化光線の輝度値が更新される．ある光線の積算輝度値を可視点の輝度値と二つの分岐光の積算輝度値の和と定義する．

積算ハッシュインデクスと積算輝度値を用いることによって，変化の起こらない光線に対する処理を完全に省くことができる．選択的トラバースをトップダウンに行って変化光線を探索する．トラバースされたパスに沿って再輝度計算を行い，結果をボトムアップに画素に伝播する．この過程でトラバースの行われたノードの積算輝度値と積算ハッシュインデクスが更新される．

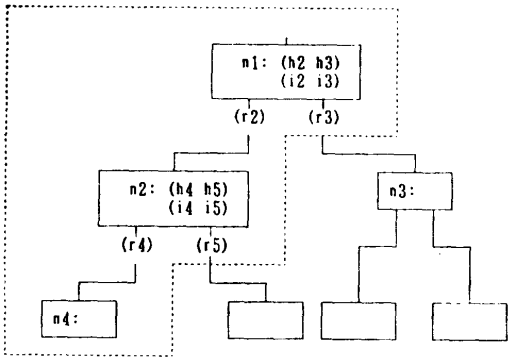


図 6 履歴データの最適トラバース方式
Fig. 6 Optimal tree traversal.

図 6 でこの方法を用いたトラバース方式を説明する．ここでは反射光，透過光をそれぞれ左と右のアークで示した．光線 r2, r3, r4 に対する積算ハッシュインデクスを h2, h3, h4, また光線追跡木のノードを n2, n3, n4 とする．ここで，光線 r4 のみが変化したと仮定すると，各ハッシュインデクスの変化フラグは，ON, OFF, ON となる．積算ハッシュインデクス h2 は光線 r4 の影響を受けることに注意されたい．拡張光線追跡木のノード n1 にトラバースの注目点がある時，積算ハッシュインデクス h2, h3 が調べられる．ハッシュインデクス h3 の変化フラグが OFF であるために，透過光に対するトラバースは必要ないことが分る．この例では，トラバースは図中に示した点線で囲んだ部分に対して行われる．

光線 r3 の積算輝度値を i3 とし，これをノード n1 での再輝度計算に使うことによってノード n3 での再輝度計算は不要となる．また，積算輝度値の定義からノード n2 の輝度値 i4 の変化はノード n1 の輝度値 i2 の再計算を引き起こす．

積算ハッシュインデクスを使ったトラバース方式によって，トラバースコストがどのくらい改善されるかを考察する．拡張光線追跡木がバランスされ，かつ変化光線は光線集合に一本しか含まれないと仮定する．この場合のトラバースコストはノード数の対数になり，複雑なシーンに対する性能は良くなる．また，積算ハッシュインデクスを計算するために，ノードの数だけの余分なハッシング処理が必要となる．しかし，変化しないインデクスは複数のフレームに渡って使うことができるため，このオーバヘッドは無視できる．

4. 付影，質感更新処理

4.1 付影処理の更新

可視点が決定された後は，可視点から光源へ向かって影光線を出射して付影処理を行う．付影処理の再計算は以下の変更が起こった時に行われる．

（1）他の物体が変化して影光線を遮る時．

（2）光源の移動によって影光線が変化する時．

（3）影光線と交差する透過物体の質感パラメータや，光源のエネルギーが変更された時．

上の(1)の理由から，ある物体の可視点が変化しない場合でもその物体についての付影計算が必要とな

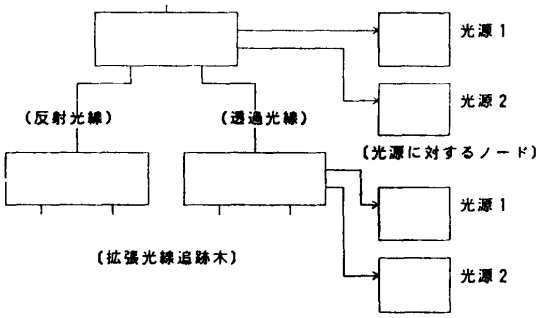


図 7 付影処理に対するデータ構造
Fig. 7 Data structure for shadowing.

る.

影光線に対する部分更新処理は分岐光に対する処理と同様に行われる. 図7に示すように, 拡張光線追跡木のノードから影光線に対応するノードを光源に対応してリンクし, ノードに各影光線に対するハッシュインデクスを記録する. (1)に対する処理では, このインデクスの変化フラグを見て影光線の変化を調べる. 変化物体がこの光線と交差しない場合には, 前フレームと同じエネルギーの光源からの寄与があると考えられる. さもなければ, 影光線との再交差計算を行う.

付影処理では以下の考察によって通過ボクセル履歴と交差情報履歴を持たないことにする. 光源に関する(2)の変更に対しては影光線の傾きが変るために履歴情報を用いることはできない. この変更は頻繁に起こると考えられる. また, 影光線は各拡張光線追跡木のノードごとに光線の数だけあり, これに対して通過ボクセル履歴や交差情報履歴を持つとデータ量が巨大となる.

4.2 質感パラメータの更新

質感の変更による再計算もモデルの変更と同様の処理を行うが, 変化光線の検出方法が異なる. 光線ごとにマスクを設け, 可視物体が持つ質感インデクス値の位置に対応するマスクのビットを ON にする表現を用いる. 変更された質感インデクスと各光線のマスクの論理積を調べることによって変化光線を検出できる.

質感インデクスマスクもハッシュインデクスと同様に積算化する. ある光線の積算質感インデクスマスクを, 注目光線と分岐光の質感インデクスマスクの論理和をとった値とする. 積算質感インデクスマスクと積算ハッシュインデクスを用いて質感変更とモデル変更に対する処理を統合することができる.

5. 実 現

本システムは並列計算機 CAP[3] 上に実現された. CAP は MIMD 型の並列計算機で, セルと呼ぶプロセッサを現在 64 台二次元格子上に構成したマシンである.

本論文で提案した部分更新処理を実現するためには膨大な量のデータを扱う必要がある. CAP では各セルに 2MB のメモリがあり, 全体で 128MB ある. このため, 全履歴データをメモリ上で管理することができ, 高速なデータアクセスが可能となる.

しかし, 処理時間の均一化と共に履歴データ量の均一化も重要な問題点となる. すなわち, ある特定のセルに膨大な履歴データが割り当てられる可能性もある. しかし, 以下の考察によってデータ量の均一化が予測される. 拡張光線追跡木を中心とする履歴データ量は, あるセルが担当する画素から始まった光線の総数にほぼ比例する. 処理時間は光線数に比例するので, 履歴データ量は処理時間に比例することになる. 筆者らの提案した静的負荷分散法[3] を用いると, 各セル間での処理時間はほぼ等しくなるので, 各セルが持つ履歴データ量も均一化されることになる. 以上の議論は実験を行って確認された.

6. 実験と評価

本方式の有効性を確認するために, 図 8,9 のモデル "CHESS" と "SPHERES を使って, 各種の変化に対する処理時間を測定した. この結果を表 1 に示す. 解像度は 512×384 で, アンチエリアジング処理は行っていない. 図8に示すようにモデルの変化に関しては, 物体移動1で駒を構成する物体の一つ (図中

表 1 映像生成処理時間 [sec] (スクリーンサイズ: 512*384)
Table 1 Timing test. [sec]

	CHESS (影なし)	CHESS (影あり)	SPHERES (影なし)
従来処理	70.4	101.4	113.3
物体移動 1	3.3(21.3)	5.8(17.5)	6.7(16.9)
物体移動 2	7.2(9.8)	13.8(7.4)	9.4(12.1)
サイズ変更	4.1(17.2)	6.9(14.7)	8.2(13.8)
光源移動	7.6(9.3)	33.8(3.0)	24.3(4.7)
色 変 更	0.8(88.0)	0.9(112.7)	1.5(75.5)

( ) 内: 従来処理に対する倍率
The relative performance is listed in parenthesis, which indicates the ratio with respect to the conventional process.

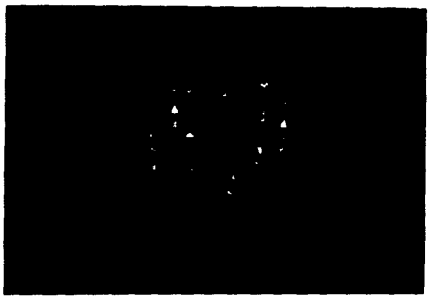
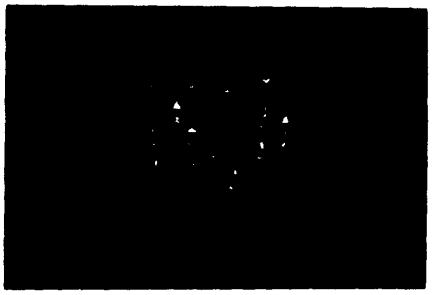
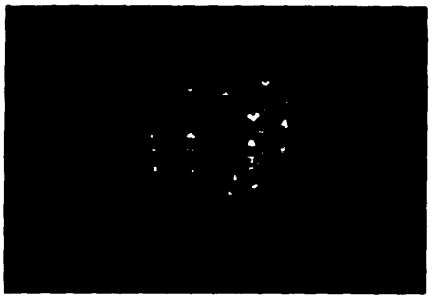


図 8 モデル "CHESS" (影あり)
物体数: 144 光源数: 1
平均拡張光線追跡木レベル: 2.25
Fig. 8 Model "CHESS" with shadow.
The number of objects: 144.
The number of lights: 1.
The average level of tree: 2.25.

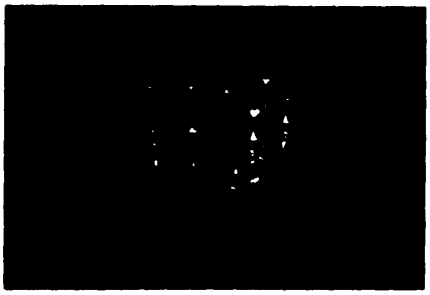


モデル "CHESS" の物体移動 1
駒 "pawn" の頭部を移動
One object of one pawn moves
in model "CHESS".

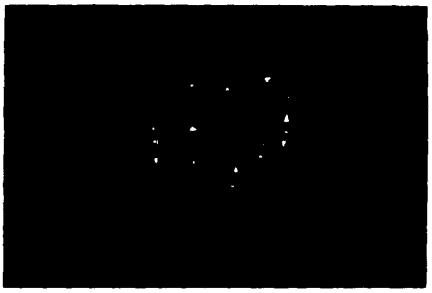
モデル "CHESS" の物体移動 2
駒 "pawn" (primitive 7 個) を移動
Entire pawn moves
in model "CHESS".



モデル "CHESS" のサイズ変更
駒 "pawn" の頭部を拡大
The size of one object of one pawn
is changed in model "CHESS".



モデル "CHESS" の光源移動
The light source moves
in model "CHESS".



モデル "CHESS" の色変更
駒 "pawn" の色変更
The color of entire one pawn
is changed in model "CHESS".

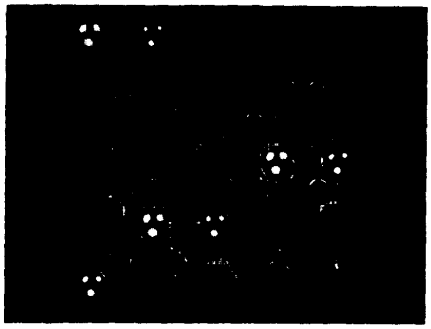


図 9 モデル "SPHERES" (影なし)
物体数: 126
光源数: 3
平均拡張光線追跡木レベル: 2.50
Fig. 9 Model "SPHERES" without shadow.
The number of objects: 126.
The number of lights : 3.
The average level of tree: 2.50.

に矢印で示す）を，移動2では駒の全体（7個の物体）を移動させた．サイズ変更では駒の頭を拡大した．質感パラメータの変更では一つの駒の色を変えた．表1から処理時間は変化物体がスクリーン画素に占める領域の大きさに依存していることが分る．より正確な表現をすれば，再交差計算の数に比例する．これは，ボクセル分割を用いた変化影響の局所化と交差情報履歴の効果があったことを示している．

この実験から，モデルの変化に対しては従来の方式である全体処理にくらべて十倍ほど，質感パラメータの変化に対しては数十〜百倍ほど，光源の変化に対しては数倍ほどの高速化が達成された．ここで，初期ステップ，すなわち光線追跡自体の処理時間は履歴情報の生成のために，筆者らの構築した従来のシステムと比較すると約40%ほど遅くなった．しかし，連続するフレーム数が多くなるに従って，この処理時間の増加は無視できる．

部分更新処理における各処理の割合を調べるためにモデル "CHESS" を用いて，可視点の決定，光線処理，付影処理等についての処理回数を測定した．

表2 から膨大なデータ量が必要な反面，各更新処理におけるデータのアクセスが少ないことが分る．例えば，移動1でアクセスされる拡張光線追跡木ノードの数は一つのセルでは131個であり，ノードの総数（1433個）に比べてかなり少ない．膨大な履歴データを用いて部分更新を行うのが本方式の本質ではあるが，提案した積算ハッシュインデクスを用いた選択的トラバース方式によって，実際にアクセスされるデータ量は変更される量に依存していることが分る．

表1と2から分るように，部分更新の処理時間は交差計算の数にほぼ比例している．物体移動1に対する交差計算数は全体処理の3%ほどであった．再交差計算数を限定することが本方式の実現上の問題点であったが，この結果から分るようにボクセルを用いた変化影響の局所化方式が有効に作用していることが分る．

次に光源の移動に関しての性能が他の変化にくらべて低い理由を考察する．影光線の数は，全可視点の数，すなわち全拡張光線追跡木のノードと光源数の積となるため，処理量が膨大になる．**表3**は，3個の光源で構成されるシーンにおいて，動かした光源の数を変えた場合の処理時間とその性能比である．これから分るように複数個の光源で構成されるシーンにおいて，一部の光源を移動する場合には性能が相対的に向上する．一般に，高品質な映像を生成するためには多くの光源が必要となるので，光源移動に関する性能の低さは実際にはそれほど問題とはならないと思われる．

表4に処理時間とデータの分散を示す．前述した考察のように，両者とも均一化されていることが分る．また，**表5**に各モデルに対する履歴データの量を種類ごとに示した．これらの統計から分るように，評価したモデルについての履歴データの総量は，数十MBほどであった．並列計算機では履歴データを分散して持

表2 処理内容の内訳〔回数/1プロセッサ〕（モデル：CHESS 影あり）
Table 2 Detailed statistic. The CHESS with shadow is used. These numbers are the numbers of items processed for each component.

	新規光線数	トラバース回数	交差計算回数	輝度計算回数
全体処理	1477	—	13765	807
物体移動1	5	131	419	31
物体移動2	27	231	1518	97
サイズ変更	28	81	620	45
光源移動	0	1433	4621	807
色変更	0	18	0	14

表3 光源移動に対する処理時間〔sec〕（モデル：CHESS 影あり 光源数3）
Table 3 The performance of light source movement. [sec] The CHESS with shadow is used. There are 3 lights in the environment.

全体処理	164.1

移動光源数	映像生成時間	倍率
3	102.4	1.60
2	75.4	2.18
1	48.8	3.36

表4 処理時間とデータ量のばらつき〔1プロセッサ〕
Table 4 Load and data balance statistics between the processors.

モデル	処理時間 min-max 〔sec〕 最大偏差 〔%〕 difference	履歴データ量 min-max 〔KB〕 最大偏差 〔%〕 difference
CHESS 影なし	68.5- 70.4 2.7%	642.3- 653.3 2.4%
CHESS 影あり	98.3-101.4 3.0%	720.9- 735.7 2.0%
SPHERES 影なし	106.0-113.3 6.4%	1046.0-1081.4 3.3%

最大偏差 =(max−min)/max*100
difference=(max−min)/max*100

表 5 履歴データの内訳
〔64 台プロセッサの総数，上段：個数，下段：データサイズ〕
Table 5 The number of data structures and amount of data structures (listed in parenthesis).

モデル	分岐光線に対するノード	通過ボクセル履歴	交差情報履歴	光源に対するノード	全履歴データ
CHESS 影なし	91780 (10.0MB)	1617664 (12.3MB)	295664 (5.6MB)	—	(39.8MB)
CHESS 影あり	91780 (10.0MB)	1617664 (12.3MB)	295664 (5.6MB)	91780 (0.7MB)	(44.5MB)
SPHERES 影あり	208192 (22.6MB)	1666943 (12.7MB)	522148 (10.0MB)	208192 (1.6MB)	(64.7MB)

つことができ，プロセッサ台数に比例して利用可能なメモリの量を増大することができる．大型計算機に関しては，光線集合ごとに履歴データベースを作成し，ハッシュインデクスを用いた選択的アクセス法を用いることで，本方式の効果が発揮されると考えられる．

7. むすび

レイトレーシングを高速化して実用的な映像生成方式として利用するために，連続するフレームにおける部分更新処理を提案し，システムの実現と評価を行った．各フレームの記述を光線の履歴情報として記録し，連続するフレームでこれを用いて最小の更新計算を行う．更新ステップでの計算量は減少されるために，レイトレーシングの処理時間が高速化され，高品質な映像が要求される分野に対して実用化できるようになった．

あるフレームでの投影面上で輝度値を評価し，別のフレームにおける可視点が同一画素であるかを判定することによって，連続するフレームで視点を移動する方式[8]が報告されているが，視点の移動量の制限や誤差等の問題がある．これはレイトレーシングは本質的に視点依存型の映像生成手法であることによる．本方式でも視点に依存した履歴データを用いるために視点を固定する必要がある．しかし，CAD システムでよく行われるように，視点の異なる複数の映像を生成し，各々を部分更新することによって頻繁に視点を変える必要はなくなる．また，アニメーションにおいてもカメラが固定するシーンはかなり多くの割合で存在すると考えられる[8]．これらの分野に対して提案した部分更新レイトレーシングを適用することで，膨大な映像制作時間が要求されていた映像生成を大幅に高速化できる．本方式によって，映像生成速度の点で不可能であった分野でのレイトレーシングの実用化が可能となると考えられる．

膨大な量のデータを扱わなければならない問題点は残るが，巨大なメモリ空間が実現されつつある現在，多くのマシンに容易に実現されると考えられる．特に，並列計算機では大量の実メモリが利用できるので，本方式の効果は大きい．この際に，データの均一分散が必要となる．これは，我々の提案した静的負荷分散方式で自動的に達成された．

今後の課題として，変化した画素に対するアンチエリアジング処理を行う必要があると考えられる．部分更新方式では光線が通過するボクセル経路の簡潔な表現としてハッシュインデクスを用いたが，ある光線集合内の可視物体集合の表現にもハッシング技法を適用することが可能であると考えられる．

謝辞 本研究に当り議論等で御援助して下さいました白石システム研究室長に感謝します．

参考文献

1) Heckbert, S. and Hanrahan, S.: Beam Tracing Polygonal Objects, *Computer Graphics*, Vol. 18, No. 3, pp. 119-127 (1984).
2) 秋元，間瀬：画素選択型光線追跡法，信学論，Vol. J 69-D, No. 12, pp. 1943-1952 (1986).
3) 村上，広田：セルラアレイプロセッサ CAP によるレイトレーシング，情報処理学会研究会報告，86-CAD-22-2, Vol. 86, No. 43 (1986).
4) Glassner, A.: Space Subdivision for Fast Ray Tracing, CC&A, Vol. 4, No. 10, pp. 15-22 (1984).
5) Fujimoto, A., Tanaka, T. and Iwata, K.: ARTS: Accelerated Ray Tracing, CG&A, Vol. 4, No. 4, pp. 16-26 (1986).
6) Sutherland, I., Sproull, R. and Shumacker, R.: A Characterization of Ten Hidden Surface Algorithms, *Comput. Surv.*, Vol. 6, No. 1 (1974).
7) Wirth, N.: アルゴリズム＋データ構造＝プログラム，日本コンピュータ協会, pp. 302-306 (1979).
8) Badt, S. Jr.: Two Algorithms for Taking Advantage of Temporal Coherence in Ray Tracing, *Visual Computer*, Vol. 4, No. 3 (1988).

(昭和 62 年 9 月 25 日受付)
(平成 元 年 4 月 11 日採録)

村上　公一（正会員）

昭和53年北海道大学工学部原子工学科卒業．昭和55年同大学大学院修士課程修了．同年(株)富士通研究所に入社．コンピュータ・グラフィックスにおけるモデリング技術と映像生成技術の研究に従事．昭和56年情報処理学会学術奨励賞受賞．ACM-SIGGRAPH 学会会員．

広田　克彦（正会員）

昭和56年豊橋技術科学大学情報工学科卒業．昭和58年同大学大学院修士課程修了．同年(株)富士通研究所に入社．コンピュータ・グラフィックス，特にレイトレーシングにおける映像生成の高速化と高品質化の研究に従事．

石井　光雄（正会員）

昭和41年名古屋工業大学工学部電気工学科卒業．昭和43年東京工業大学大学院修士課程修了．同年(株)富士通研究所に入社．CAD，画像処理，並列コンピュータ，コンピュータ・グラフィックスなどの研究に従事．現在，同社システム研究部長．工学博士．IEEE，電子情報通信学会，テレビジョン学会各会員．