グラフィクスとCAD 92−1
（1998. 10. 15）

実時間に変化するボリュームデータの当り判定アルゴリズム

平井 哲† 高井 昌彰†† 山本 強††
†北海道大学大学院 工学研究科
††北海道大学大型計算機センター

ボリュームデータは物体の表面と中身の両方を表現できるモデリング手法であり、様々な分野で応用されている。これに関連して、最近はボリュームデータ同士の当り判定の研究が行われている。しかしながら、これらの研究は実時間に変化しないボリュームデータ同士の当り判定を対象としている。今後、特に手術シミュレーション、リアルタイムゲームなどの分野では実時間に変化するボリュームデータの当り判定を行うことが重要になると予想される。そこで、本研究ではこのようなニーズに応えることを目標として、実時間に変化するボリュームデータの当り判定をできるだけ高速に行うことを目的とした第一段階の研究を行ったので、ここで報告する。

A Collision Detection Algorithm for Volume Data Which Varies in Real Time

Tetu Hirai† Yoshiaki Takai†† Tsuyoshi Yamamoto††
†Graduate School of Engineering, Hokkaido University
††Hokkaido University Computing Center

Volume rendering is a modeling method which can be used to describe the surface and insides of objects, and is being used in several different fields. In a related field, research for the detection of collisions between volume data is being conducted. However, research has been limited to volume data which is not time varying. It is predicted that the detection of collisions between time varying volumetric data will become important in the future in fields such as the simulation of operations, and real-time games. In order to meet the needs of these fields, we conducted initial research to detect collisions between time-varying volume data. We give the results of this research here.

1 はじめに

ボリュームデータは物体の表面と中身の両方を表現できるモデリング手法であり、様々な分野で応用されている [3]。それを可視化するボリュームレンダリング技術は大きく別けて二つの種類に分類できる。一つ目はレンダリングには多少時間がかかるが、高画質の画像をレンダリングする手法である。二つ目は画質はいくらか落ちるが、高速性を追求したレンダリング手法である。我々は後者のアルゴリズムを研究分野としている。

これに関連して、最近はボリュームデータ同士の当り判定の研究が行われている [1]。しかしながら、これらの研究は実時間に変化しないボリュームデータ同士の当り判定を対象としている。今後、特に手術シミュレーション、リアルタイムゲームなどの分野では実時間に変化するボリュームデータの当り判定を行うことが重要になると予想される。そこで、本研究ではこのようなニーズに応えることを目標として、実時間に変化するボリュームデータの当り判定をできるだけ高速に行うことを目的とした第一段階の研究を行ったので、ここで報告する。

2 用語の定義

1. ボクセル
   論文によっては「ボクセル」という用語を、格子状のボリュームデータにおける各格子上の点として定義している場合がある。しかし本報告では、「ボクセル」という用語を、ボリュームデータを構成する直方体の領域と定義する。各直方体の内部は全て同じ値を持つこととする（図1参照）。

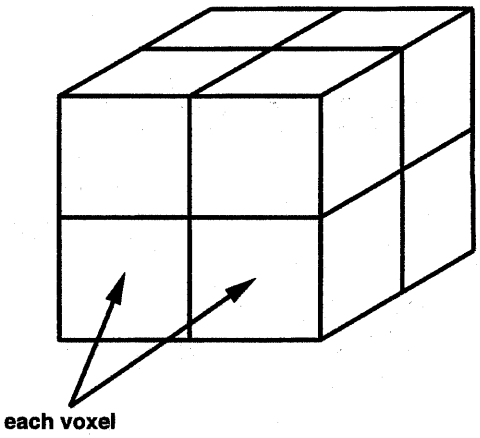

Fig.1 ボクセルの例

2. ボリュームオブジェクト
   ボリュームデータで表現する衝突可能な物体。ボリュームデータはそれぞれ独自のボリューム空間を持つ。例えば、図2では、2つのボリュームオブジェクトが存在していて、左側の物体が1つのボリュームオブジェクト、また右側の物体がもう1つのボリュームオブジェクトである。

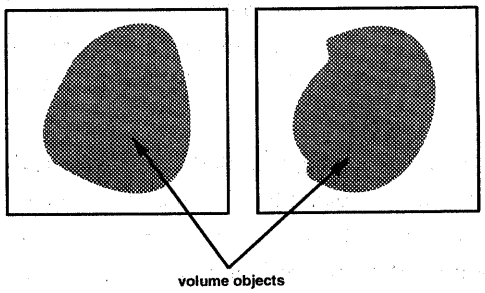

Fig.2 2つのボリュームオブジェクト

3 当り判定の基準

本研究では、ボクセルとボクセルが当っているかどうかの判定は各ボクセルの不透明度（アルファ値）で行っている。交差している両方のボクセルがあるしきい値以上なら、当っていると判定される（図3参照）。現在、そのしきい値は0（ゼロ）としている。

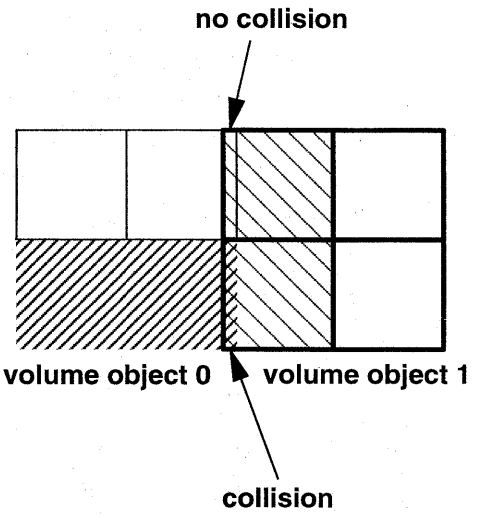

Fig.3 当り判定の例

4 提案手法の紹介

我々の当り判定手法は二つの方法により当り判定の計算量を減らし、高速化を行っている。一つ目は、三次元のバウンディングボックス (bounding box) を使用することにより、明らかにボリュームデータ同士が当らないケースの検出を行っている。二つ目は、現在ボリュームデータ同士が絶対に交わる可能性のないボリュームスライス領域を検出している。

5 具体例を使用した提案手法の説明

ここで二次元の簡単な例を用い、本手法を説明する。

1. バウンディングボックスによる衝突判定チェック
   図4にそれぞれ独自のボリュームデータを持つ二つのボリューム空間を示している。

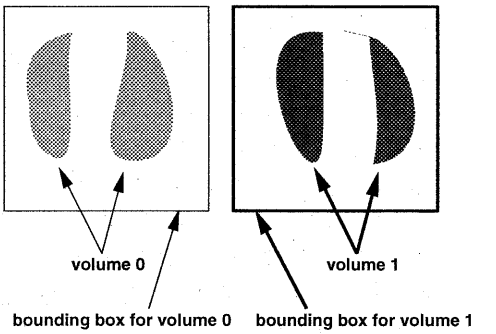

Fig.4 バウンディングボックスが交わっていない例

この図では、各ボリュームデータのバウンディングボックスが交わっていないので、ボリュームデータ同士の衝突は不可能である。しかし図5の場合、2つのバウンディングボックスが交わっているのでボリュームデータ同士も交わっている可能性がある。よって、

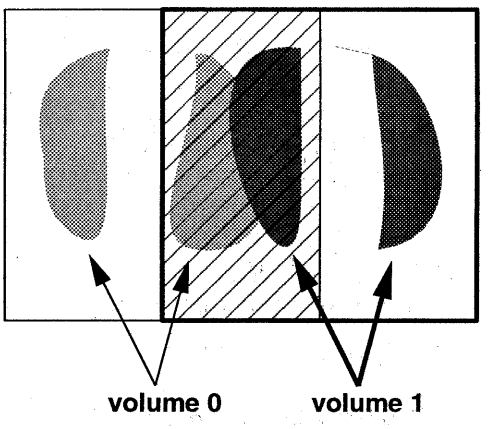

Fig.5 バウンディングボックスが交わっている例

されに詳しくボクセルデータを調べる。

2. ボクセル同士の衝突判定
   ボクセル同士の衝突判定を行うには、図5における網掛け部分の各ボクセルの不透明度を調べる。例えば、あるボリュームオブジェクトのボクセルがそのボリューム空間の (x_0, y_0, z_0) に存在していてそのボクセ

ルの不透明度はα_0だとする。その同じ位置に存在するもう一つのボリュームオブジェクトのボクセルの不透明度はα_1だとする。もし、α_0かα_1のどちらか（あるいはどちらも）衝突不可能な値であれば、その位置(x_0, y_0, z_0)では衝突不可能とされる。ここで本手法は、この衝突可能な領域内の全てのボクセルを調べなくても済むように、図6に示されているようなデータ構造を用いている（この図は2次元の場合の例を示している）。

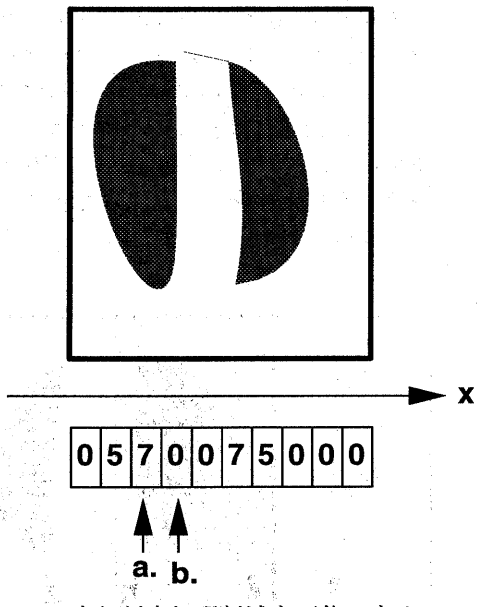

Fig.6 当り判定処理削減を可能にするデータ構造

このデータ構造は、各x座標に対して、それに相当するボリュームデータのスライスデータに現在存在する衝突不可能なボクセルの数を持っている。例えば、図6において$x = 2$の場合、7個のボクセルが衝突不可能なので、$x = 2$に相当するa.の場所に7を格納する。また、同図において、$x = 3$の場合、衝突可能なボクセルが一つもないので、$x = 3$に相当するb.の場所に0を格納する。実際に衝突判定を行う時、例えば、$x = 3$の場合、このボリュームオブジェクトにおいて、ボクセルの全てが衝突不可能なボクセルであるので衝突が起きることはない。したがって、そのスライスデータの衝突判定処理を行う必要はない。図7に両方のボリュームオブジェクトのデータ構造の例を示している。

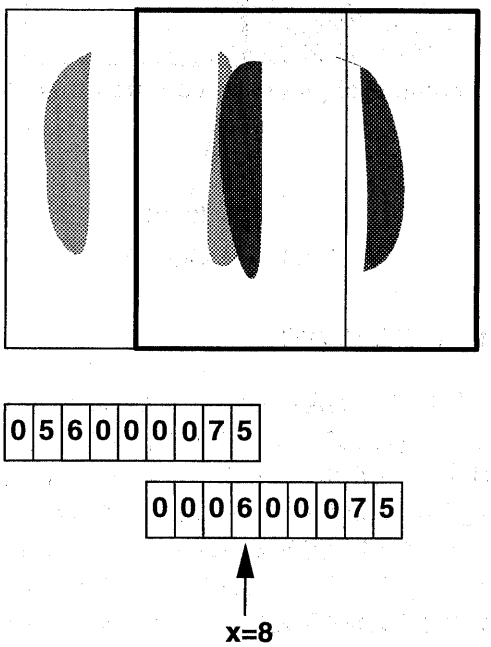

Fig.7 当り判定処理削減を可能にするデータ構造

$x = 8$の場合のみ、衝突不可能なボクセルが両方のボリュームオブジェクトに存在するので、この場合のみ、個々のボクセル同士の不透明度の比較を行う。こうすることにより、ボクセル同士の不透明度を比較する量を減らすことができる。ただし、ボリュームデータを変更するたびにこのデータ構造に入っている値を変更することが必要となる。なお、現在は、x座標に対してのみこのデータ構造を実装しているが、最終的にはx、y、z軸方向全てにおいて、このデータ構造が必要である。

表 1: 実行速度の比較

Data Set Number	Naive Algorithm	Proposed Algorithm	Ratio
1	9.7 fps	10.4 fps	1.07
2	9.7 fps	10.3 fps	1.06

6 実験

我々の当り判定アルゴリズムを評価するために実験を行った。この実験は以下の3つの処理の合計処理時間を評価したものである。

1. ボリュームデータを変更するのに要した時間
2. ボリュームデータの当り判定に要した時間
3. ボリュームデータを可視化するために必要とした時間

この実験は以下の設定で行われた。提案手法の実行速度を比較する対象として、上のデータ構造を使わない単純なアルゴリズムの実行速度を使用した（バウンディングボックスのみ使用）。レンダリングには [2] の手法を用いた。ボリュームデータの更新はテクスチャーデータを書き換えを行った。

1. 各ボリュームオブジェクトの解像度： 64^3
2. ボリュームオブジェクトの数： 2個
3. レンダリングされた画像の解像度： 256^2
4. 使用機種： Pentium II, 300 MHz（グラフィックアクセラレーターなし）
5. 使用グラフィックライブラリー： OpenGL

実験結果は表1に示されている。

上の表で分かるように提案手法は単純なボクセル同士の比較と比べわずか6パーセントのスピード向上となっている。しかしながら、グラフィックアクセラレーターなしのPCでボリュームデータのレンダリング、更新、そして当り判定がこれくらいのフレームレートでできることを実証した。

7 問題点と今後の課題

提案手法でわずか6パーセントのスピード向上となったのはまだ不十分なので、x座標に対してのみだけでなく、最終的にはx、y、z軸方向全てにおいて、このデータ構造を使用し、実験を行う予定でる。また、当り判定の部分だけではなく、レンダリングやボリュームデータの更新の部分の高速化も計る予定である。

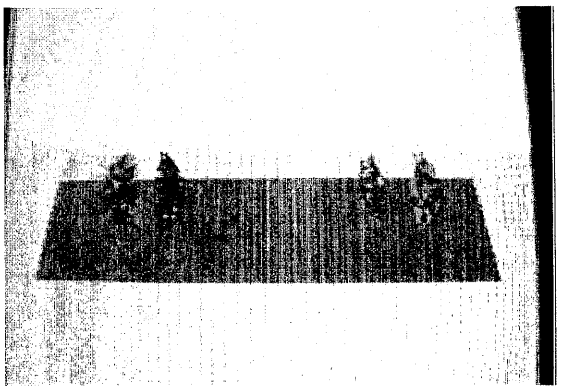

図 8: Data Set 1 の 1 枚の画像

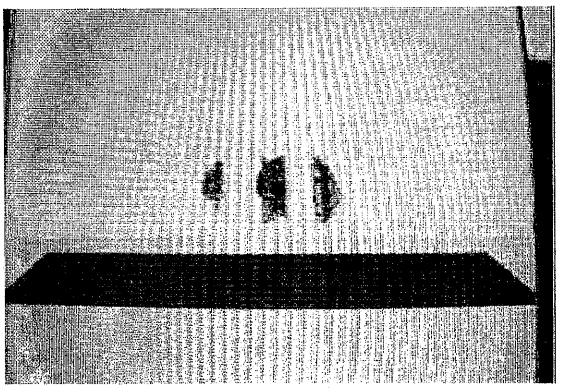

図 9: Data Set 2 の 1 枚の画像

参考文献

[1] T. He and A. Kaufman, "Collision Detection for Volumetric Objects," *IEEE Visualization '97*, URL: http://www.bell-labs.com/~taosong/taosong-papers.html

[2] T. Hirai and T. Yamamoto, "透明度の実時間変化を伴うボリュームデータの高速レンダリング手法,", 情報処理全国大会（春）１９９８年

[3] A. Kaufman, "Volume Visualization," *SIGGRAPH 1996 Course Notes #34 (Volume Visualization: Principles and Practice)*, 1996.