

2 値 VOXEL データにおけるモルフォロジー演算の高速化

1ZD-05

手島 裕詞 西尾 孝治 小堀 研一
大阪工業大学 情報科学部

1. はじめに

近年、3次元画像処理の需要が大きくなっている。これまで2次元の画像処理で使われてきたモルフォロジーも3次元分野で使われるようになってきている。モルフォロジーの応用で、代表的なものとして平滑化、形状認識などがあげられるが、3次元のボクセルモデルでモルフォロジー演算を行おうとすると高解像度になるほど処理時間が膨大になる。そこで、この問題を解決するためデータ構造にオクトツリーを用い2値Voxelデータのモルフォロジー演算を行う手法を提案する。

2. モルフォロジー演算

本研究では、モルフォロジー演算の基本演算であるミンコフスキー和とミンコフスキー差をとりあげる。入力画像を、構造要素の各々の位置ベクトルに従い平行移動させた全ての画像に対して論理和をとる処理がミンコフスキー和であり、論理積をとる処理がミンコフスキー差である。

3. 本手法

本手法では、モルフォロジー演算に用いる構造要素を、直方体の集合に分割することにより処理の高速化を行う。

3.1 長方形の構造要素

入力画像を A 、構造要素(長方形)を B とする時、式(1)の結合則^[1]が成り立つので、構造要素 B を x 軸方向の構造要素 B_1 と y 軸方向の構造要素 B_2 に分解する。ミンコフスキー和では、

$$A \oplus B = A \oplus (B_1 \oplus B_2)$$

$$= (A \oplus B_1) \oplus B_2 \quad (1)$$

すなわち入力画像 A を構造要素 B でミンコフスキー和を行った結果は、入力画像 A をまず x 軸方向の構造要素 B_1 でミンコフスキー和を行った結果に対してさらに y 軸方向の構造要素 B_2 でミンコフスキー和を行うことにより求めることができる。

ここで演算量を比較してみると、構造要素 B の x 軸方向、 y 軸方向の大きさを m 、 n とすると軸方向に分解して演算を行うことで $O(m \times n)$ の演算量を $O(m+n)$ の演算量に削減することができる。

ミンコフスキー差についても構造要素が長方形の場合、同様にして演算量の削減をすることができる。

3.2 直方体の集合への分割処理

本手法では、構造要素を直方体の集合に分割する処理にオクトツリーを応用している。まず2値のボクセルデータである構造要素をオクトツリーデータに変換する。

次に大きい黒オクタントから見つけるために木の根からオクトツリーを探索し、見つかった黒オクタントをシードオクタントとする。次に、その右隣のオクタントを検索してシードオクタントと同じ大きさの黒オクタントが存在していれば、マージを行いより大きな直方体を生成する。ここでマージとは、あるオクタントに対して空間的に見て隣接するオクタントを外包する直方体を生成することを言う。また、マージされた黒オクタントにはマージされたことを表すフラグを立てる。

さらに直方体の右隣のオクタントを検索して、オクタントと同じ大きさの黒オクタントが存在すれば、マージしてさらに大きい直方体を作る。右隣に、違う大きさの黒オクタントが存在する、あるいは黒オクタントが存在しない場合は、右方向のマージ処理

を終了する。

この処理を6方向について行う。次にマージする黒オクタントの大きさを一つ小さくして同様にマージ処理を行う。この処理を最小の黒オクタントまで繰り返して行う。

終了条件として対象となる全ての黒オクタントに対してフラグが立てば終了となる。

なお、事前に行った実験で最小の黒オクタント同士のマージは、数が多くマージ処理の負荷が大きくなることがわかったので、マージ処理を行わずにそのままミンコフスキー和やミンコフスキー差の演算を行うこととした。マージ処理の1例を図1に示す。

説明ではマージの順番として右左上下前後方向で行ったが、本手法では、マージの順番を上下前後右左方向、前後右左上下方向と3種類行ってその演算量を評価し、最も演算量の少ないものを採用して演算を行っている。

また、本手法では黒オクタントを高速に検索する方法として、ノードアドレス法⁴⁾を導入した。

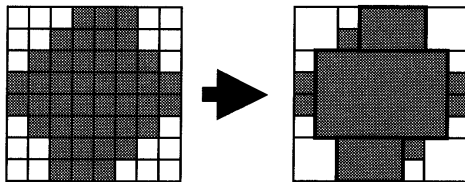


図1 マージ処理の結果

3.3 軸方向の構造要素での演算

本手法では、構造要素を3.2の処理により直方体に分割し、その直方体を軸方向に分解して演算を行う。軸方向に分解された構造要素で小さい方を e_1 、ボクセル数を n_1 とし、大きい方を e_2 、ボクセル数を n_2 とし、入力画像の黒ボクセル数を n_0 とする。構造要素 e_1 で演算し、次に構造要素 e_2 で演算した場合の演算量を式(2)に示す。また、構造要素 e_2 で演算し、次に構造要素 e_1 で演算した場合の演算量を式(3)に示す。

$$(n_0 \times n_1) + (n_0 \times n_1) \times n_2 = n_0(n_1 + n_1 n_2) \quad (2)$$

$$(n_0 \times n_2) + (n_0 \times n_2) \times n_1 = n_0(n_2 + n_1 n_2) \quad (3)$$

$n_1 < n_2$ より式(4)が成り立つ。

$$n_1 + n_1 n_2 < n_2 + n_1 n_2 \quad (4)$$

従って本手法では軸方向に分解した場合、小さい方の構造要素から演算を行った。

4. 実験

本手法の有効性を検証するために、構造要素の黒ボクセルに対して入力画像を平行移動して論理和、論理積をとる一般に行われている手法(従来法)との比較を行った。空間の一辺の解像度を2 levelとしたとき level を5, 6, 7と変化させて4種類の入力画像に対し、構造要素を球、直方体、円柱としてミンコフスキー和を行った場合の処理時間を計測した。なお、実験には、PC/AT互換機(Pentium II 300MHz, Windows NT)を用いた。図2にミンコフスキー和の結果を示す。

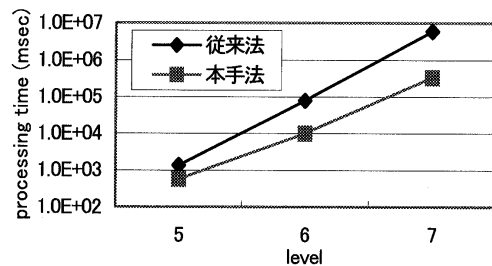


図2 ミンコフスキー和

5. 考察

従来法では、levelを1つ高くすると入力画像と構造要素の体積がそれぞれ8倍になるので、処理時間が64倍の増加になるのに対して、本手法では25倍程度の増加になった。高解像度になるに従い、本手法は処理速度の面で有効であることを確認した。

6. おわりに

本手法では、オクトツリーのデータ構造を用い、黒オクタントをマージすることで直方体の集合に分割したが、さらに最適に分割する手法を開発していきたい。

7. 参考文献

- [1] 小畑 秀文: “モルフォロジー”, コロナ社
- [2] 國井, 藤代: “ソフトウェア工学ハンドブック”, オーム社