三次元2値画像における情報源符号化の一手法

5U－02

西尾孝治[1]　金谷孝之[2]　藤村真生　小堀研一[3]
豊橋技術科学大学[4]　広島国際大学[5]　大阪工業大学[6]

1 はじめに

従来，三次元形状モデルには境界表現が用いられてきた．しかし，境界表現は形状操作の際に，形状の構成要素である幾何要素を意識した操作を必要とするという問題点があった．これに対し，三次元形状モデルの定義に三次元2値画像を用いて，直感的な操作を実現する手法が提案されている．今後も，三次元形状の定義に空間分割モデルを用いることが予想される．このため，空間分割モデルの効率的な保存形式が必要になると考えられる．そこで，本研究では，空間分割モデルである三次元2値画像の情報量を削減する手法を提案する．

2 オクトリー

一般に，三次元2値画像はボクセルと呼ばれる単位立方体の集合を用いて，これらに2値情報を保持することで形状を表現する．しかし，ボクセルはモデリング空間の一辺の解像度の3乗に比例した記憶容量を必要とする．このため，ボクセルを圧縮した表現であるオクトリーが広く用いられている．オクトリーは図1(a)の網掛けに示す形状を，同図(b)に示す木で表現する．なお，簡単のため図は二次元で表す．また，親ノードと子ノードの対応を同図(c)に示す．

このように，木を用いて形状を表現することで，必要とする記憶容量を削減することができる．

3 DF表現

通常，モデリングを行った結果はファイルに保存されるが，オクトリーのファイルフォーマットとして DF(Depth

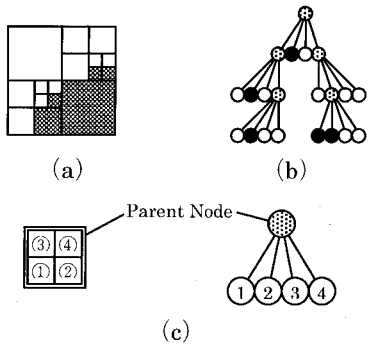


図1　オクトリーによる形状表現

First)表現[1]が一般に知られている．このフォーマットは，オクトリーの木を深さ優先探索で探索し，子ノードを持つノードを‘(’で表現するものである．図1(b)の木をDF表現を用いて白いノードを‘0’，黒いノードを‘1’として表すと“((010(010010(0(110000”となる．

4 DCC(Differential Chain Code)

これまでに，二次元2値画像の符号として RLE(Run Length Encoding)やチェーンコードが提案されている．このうちRLEは容易に三次元2値画像へ適用することができる．一方，チェーンコードは図形の輪郭を一筆書きでなぞることで形状を表現するが，一般に三次元形状の輪郭(形状の境界)を一筆書きでなぞることはできない．そこで，本手法では，チェーンコードを拡張し，木構造を用いて，三次元形状にチェーンコードを適用した．この木を符号木と呼ぶ．

符号化は，形状の境界に位置する任意の黒ボクセルから始め，白ボクセル，および符号化済みのボクセルに囲まれて，符号化途中のボクセルの隣接ボクセルに，次の符号化対象となるボクセルがなくなるまで続ける．ただし，符号化対象となる全てのボクセルを一筆書きでなぞることができないため，符号化されていないボクセ

1 Koji Nishio
2 Takayuki Kanaya
3 Masao Fujimura, Ken-ichi Kobori
4 Toyohashi University of Technology
5 Hiroshima International University
6 Osaka Institute of Technology

ルが残っている可能性がある．そこで，符号木のノードに対応するボクセルで，隣接ボクセルにこのようなボクセルがないかを探索し，このボクセルを始点に同様の処理を繰り返す．以上の処理を符号化対象のボクセルがなくなるまで続ける．

チェーンコードは進路の方向を表すラベルそのものを符号とするよりも，ラベルの差分を符号化する方が情報量を小さくすることができる．そこで，Bribiesca の手法[2]による相対的なラベルを符号として用いた．この符号を DCC(Differential Chain Code)と呼ぶ．

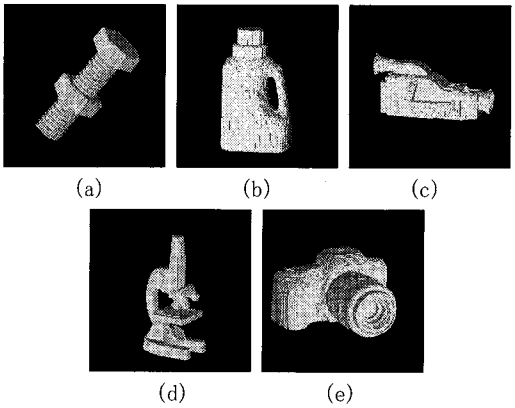
また，符号はシンボルの出現確率に偏りが大きいほど，情報量が少なくなる．本手法では，この偏りが大きくなるように符号化を行なう．二次元 2 値画像を対象とするチェーンコードでは符号化途中で符号化候補となるピクセルが一意に決まるが，三次元の DCC では図2に示すように符号化候補となるボクセルが複数存在する．そこで，すでに符号として得られたシンボルの出現確率を求めておき，符号化後に得られるシンボルの出現確率が最も高くなるボクセルを符号化対象とすることとした．これにより，情報量の削減を見込むことができる．

図２　符号化途中における符号化候補ボクセル

5 実験

DCC の有効性を検証するために，三次元2値画像の保存形式として一般に用いられるオクトリーを比較対象として情報量を算出した．いずれも DF 表現でファイルに保存し，ファイルに含まれるシンボルの出現回数から情報量を求めた．実験に用いた形状を図3(a)～(e)に示す．いずれの形状も空間の解像度を 128×128×128 とした．シンボルの出現回数と情報量を表1に示す．

また，DCC 形式および DF 形式で保存したファイルに対してハフマン符号を適用して，圧縮前と圧縮後のファイルサイズを求めた．この結果を表2に示す．



(a)　(b)　(c)

(d)　(e)

図3 実験形状

表1 シンボル数と情報量の比較

			a	b	c	d	e
Octree	シンボル	0	21,329	22,257	17,771	18,052	35,975
		1	17,774	19,877	15,550	13,988	32,549
		(	5,586	6,019	4,760	4,577	9,789
	情報量(KB)		7.7	8.3	6.6	6.3	13.5
DCC	シンボル	0	17,245	21,775	19,520	15,763	36,553
		1	1,238	2,850	1,718	1,333	4,327
		2	3,971	1,890	763	828	3,393
		3	1,376	2,123	1,097	548	1,699
		4	2,152	1,467	1,947	1,691	2,618
		(	2,027	1,859	986	835	2,608
		)	2,027	1,859	986	835	2,608
	情報量(KB)		5.8	5.9	4.0	3.2	8.8

表2 ファイルサイズの比較

		a	b	c	d	e
Octree	Uncompressed	43.6	47.0	37.1	35.7	76.4
	Huffman encoded	9.0	9.8	7.7	7.3	15.9
DCC	Uncompressed	29.3	33.0	26.3	21.3	52.5
	Huffman encoded	7.8	8.1	5.7	4.6	12.1

(KB)

表1よりオクトリーに比べて DCC では情報量が 51～75%に削減されていることがわかる．また，表2よりハフマン符号化前で 60～71%に，符号化後で 63～86%にファイルサイズが削減されていることがわかる．

6 まとめ

三次元 2 値画像の符号化手法として，提案手法はオクトリーよりも符号化効率が高いことを示した．今後は，他の符号化手法との比較を行う予定である．

文　献

[1] E.Kawaguchi and T.Endo, "On a Method of Binary-Picture Representation and Its Application to Data Compression," IEEE Trans. PAMI, 2, 1, pp.27-35, 1980.

[2] E.Bribiesca, "A chain code for representing 3D curves," Pattern Recognition, 33, pp.755-765, 2000.