CA-005

8*k* 次格子グラフモデルによる物体の重なり検知

夜久竹夫† Takeo Yaku
安斎公士†† Koushi Anzai
横田健††† Ken Yokota
横山隆介†††† Ryusuke Yokoyama

要旨

2D, 3D, 4D 物体の重なりをそれぞれ 16 次、40 次、96 次（疑似）格子グラフにより定式化する．次に、それらを用いて、$O(n)$と $O(n^2)$時間の重なり検知アルゴリズムを示す．

1. はじめに

移動する物体の衝突を含む物体の重なり検知は CAD や交通などの分野で重要である．

物体を矩形の集まりで表すことは処理の自由度が高いため、本論文は 2 次元、3 次元、4 次元の物体を対象として、矩形分割のためのデータ構造と効果的な重なり検知手法を扱う．

我々のゴールは矩形の重なり検知に適した、矩形のモデルと効率的なアルゴリズムである．

矩形分割のグラフモデルとして 19 世紀以来矩形双対モデルが知られている．その後、探索に適したグラフモデルとして 4 分木モデルが提案されて[1]、さらに 4 分木由来の 8 分木[2]、16 分木[3](本論文では総称して"4*k* 分木"という)が提案されている．矩形双対グラフや 4*k* 分木モデル上で重なり検知アルゴリズムや合併を通常の方法で構成すると計算時間は $O(n^2)$になることに注意する．矩形分割に関しては他に、罫線指向型処理に適した 8 次格子モデルが提案されて[4]、その後一般的な 24 次[8]、64 次格子グラフ[9]が提案されている．さらに一般化されて、複数物体のグラフモデルとして、16 次[8]、40 次[10]、96 次格子グラフ[11]（本論文ではそれらを総称して"8*k* 次格子グラフ"という）が提案された．8 次格子グラフでは合併などいくつかの処理が $O(n)$時間で解ける[5, 6, 7].

そのため、8*k* 次格子グラフでは重なり検知が低い計算量解ける可能性がある．

そこで、本論文は次の事を目的とする：(1) 8*k* 次格子における重なりの定式化、(2) 2 次元物体の重なり検知手法、(3) 3 次元、4 次元物体の重なり検知．結果は以下の通りである：(1) 8*k* 次格子による重なりの定義、(2) 2 次元物体の 16 次格子による $O(n)$時間と $O(n^2)$時間の重なり検知アルゴリズム．（3）3 次元、4 次元物体の 40 次、96 次格子を用いた $O(n)$時間と $O(n^2)$時間の重なり検知アルゴリズム．

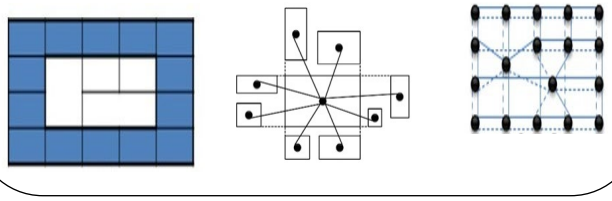
結果は静止物体の重なり検知や移動物体の衝突検知に応用される．

† 日本大学　Nihon University
†† 関東学園大学　Kanto Gakuen University
††† 日本大学第二中学校・高等学校　Nihon University 2nd Highschool
†††† 応用オートマトン研究会　Workshop on Automata and Their Applications

2. 準備

2.1 8 次格子:矩形分割のためのデータ構造[4]

定義 2.1 格子グラフ *G* が**矩形分割 *D* に対する 8 次格子** <=def=> *G* の 2 頂点 c と c`は c と c`が同じ罫線(ハリ)を共有している時にリンクされる（下の図 1 参照），ただし本論処理アルゴリズムの便利のためではもっとも外側の矩形のサイズは 1 x 1 と仮定する(3D と 4D の場合も同様).



矩形分割 *D*　　*G*(*D*)の頂点周りの局所構造: 2 頂点はハリ(罫線)を共有する時リンクされ　　*D* に対する 8 次格子 G(*D*)

図 1 矩形分割と、対応する 8 次格子の構造

平面図形は以下の例のように表される．

例 1.平面図形(三角形)を表す 8 次格子の例．解像度低減無し(上段左)、有り(上段右)．回転前(下段左)、回転して解像度低減(下段右)　(視認性を高めるため白黒反転して描画)

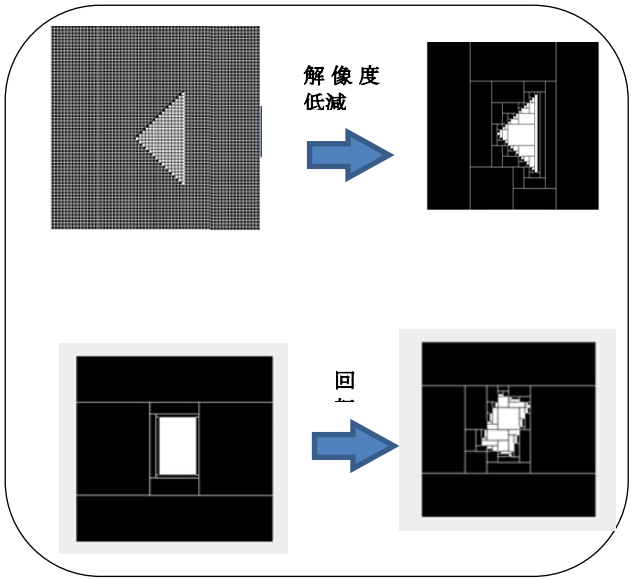
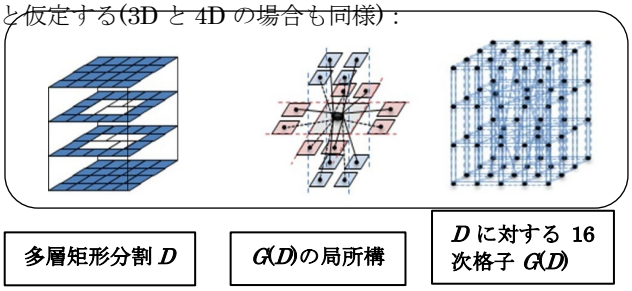


図 2　8 次格子による 2D 図形表現のイメージ

2.2 16 次格子: 複数の 2 次元図形のためのデータ構造[8]

16 次格子；以下の図 3 のように同じコーナー位置を共有する矩形に対応する頂点同士を辺でつないで多層化した格子グラフ, ただし最上層及び最下層の矩形のサイズは 1 x 1 と仮定する(3D と 4D の場合も同様)：

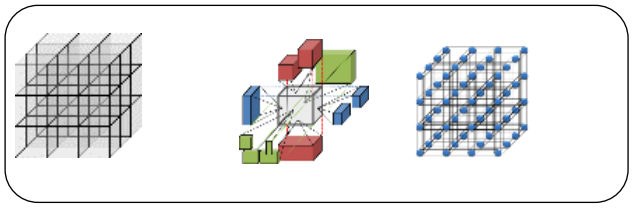


多層矩形分割 D | $G(D)$の局所構 | D に対する 16 次格子 $G(D)$

図 3　多層矩形分割と対応する 16 次格子の構造

2.3 24 次格子：3 次元図形のためのデータ構造[8]

8 次格子を拡張して、以下の図のように同じハリ位置を共有する直方体同士に対応する頂点同士を辺でつないで、得られる格子グラフを **24 次格子**という．24 次格子は $4k$ 分木の 8 分木に相当する．

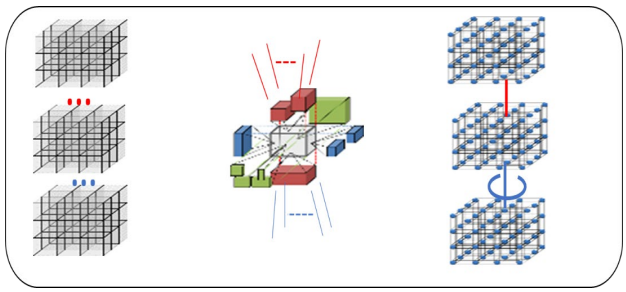


直方体分割 D(左) | G の頂点周りのリンク．D で同じハリを共有する頂点同士がつながる (中) | D に対応する 24 次格子 G (右)

図 4　直方体分割と対応する 24 次格子の構造

2.4 40 次格子：複数の 3 次元図形のためのデータ構造[10]

次に、24 次格子を §2.2 と同様に以下の図のように多層化する．以下で得られる格子グラフを 40 次格子という．



多層直方体分割 D (左) | $G(D)$ の局所構造 (中) | D に対する 40 次格子 $G(D)$ (右)

図 5　多層直方体分割と対応する 40 次格子の構造

2.5 時間連続な 4 次元超直方体分割のためのデータ構造[9]

24 次格子を 4 次元化して、以下の図のような **64 次格子**を定める(時間は離散的とする)．即ち

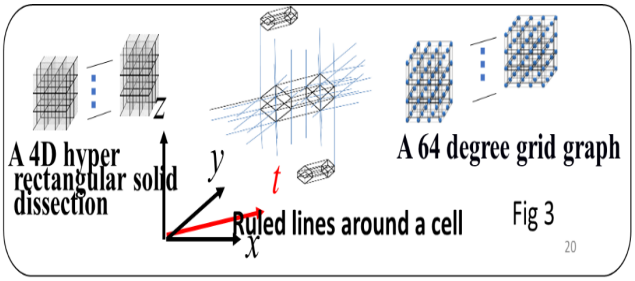


図 6　4D 超直方体分割と対応する 64 次格子の構造

64 次格子は $4k$ 木の 16 分木[3]に相当する．

2.6 複数の 4 次元超直方体分割のためのデータ構造[11]

§2.4 と同様にして、以下の **96 次格子**を構成する

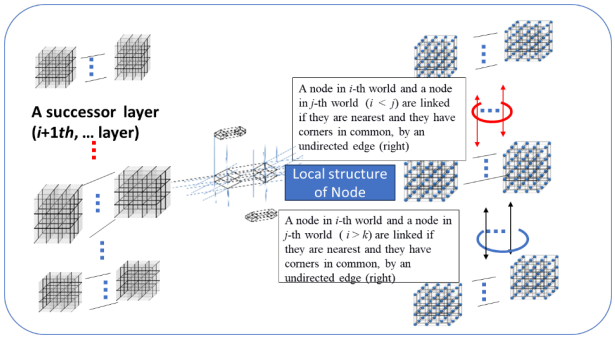


図 7 多層 4D 超直方体分割と対応する 96 次格子の構造

3. 重なり検知の定式化

以下では、各頂点が対応する物体に占有されているとき**黒色 (occupied) で色付け**されていて、占有されていないとき**白色 (vacant) で色付け**されているとする．物体はある層で黒色の頂点の集まりで表現される．2 次元の場合、即ち 2 つの物体は 2 つの層で表される．

3.1 完全重なり

定義 3.1　異なる層に属する二つの矩形 C、D (対応する頂点 c、d)は**完全に重なる<=def=>C, D (c, d)が占有されていて、C、D のコーナー全ての位置が共通(c, d が 4 つのコーナー辺すべてでリンクされている)．(次の図参照)**

図 8　矩形と対応する頂点の完全重なりのイメージ

3.2　部分的重なり

物体の衝突は部分的な重なりが引き起こす．そこで次に部分的重なりを定式化する．

定義 3.3　多層矩形分割 D を考える．異なる層に属する二つの図形 C、D (対応する頂点 c、d)は**部分的に重なる <=def=> C の中に矩形 x と D の中に矩形 y が存在して、x と y が完全に重なっている(以下の図参照).**

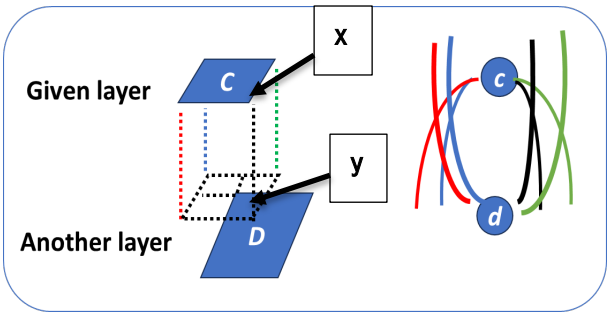


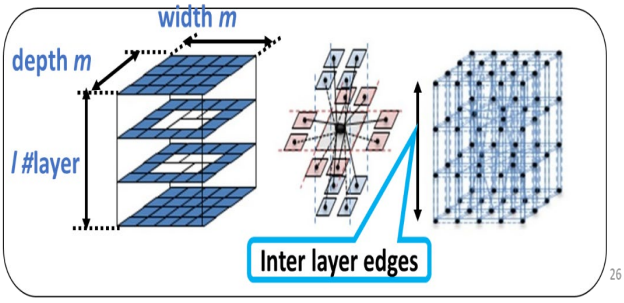
図９　矩形と対応する頂点の部分的重なりのイメージ

4. アルゴリズム

4.1 2 次元の場合

この節で重なり検出アルゴリズムを考える．

以下の図のような l 層の n x m 矩形分割(左)に対応する 16 次格子(右)を考える．この図で 16 次格子の上下方向の辺は層の間の辺で重なりに関係することに注意する．



l 層の m x n 矩形分割 D (左)

D に対する 16 次格子 $G(D)$のレイヤー間の辺のイメージ

図 10　多層矩形分割の例と対応する 16 次格子

(i)方法

概略　1 番上の層の n x m 個の各セル(矩形)について、最下層までリンクを辿って次を行う：

二つ以上の部分的に重なるセルが存在するかどうか、調べる；部分的に重なるセルが存在したら、“OVERLAPPED”を出力して停止．存在しなければ“NOT-OVERLAPPED”を出力して停止．

手順(概要)

Algorithm 2*DoverlapDetectionWith*16*grid*(*G*, *y*)
INPUT *G*(*V*, *E*, *loc*, *occupation*) : a multi-edge 16-ary grid graph with width *m*, depth *n*, layers *l*
loc : a map of *V* to N^4 that indicates the location (north, south, east and west walls) of a node
occupation : a map of V to {VACANT, OCCUPIED} that indicates object occupations.
OUTPUT *y*: NOT-OVERLAPPED / OVERLAPPED
WORK *M*: An *n* x *m* ARRAY that indicates occupations of *i-j* locations throughout whole layers while computation, that is, initially all cell values are VACANT, the value OCCUPIED indicates that the (*i*, *j*)-cell is occupied by exactly one object in algorithm execution, and the value OVERLAPPED indicates that the cell is occupied by two objects.

METHOD
Initially *set* y to “NOT-OVERLAPPED”;
Set all elements of M to “VACANT”;
for all cells c(*i, j,* 1) (1≦*i* ≦*n*, 1≦*j*≦*m*) in the perimeter layer 1 in *G* **do**
/* STEP A*/
repeat until visit reached to the bottom layer **do** *visit downward a cell c along with inter layer edges of the common northeast corner ;*
if *occupation*(*c*)=“OCCUPIED” and the area of the visited cell *c* is bounded by *i, i', j, j'* ruled lines **then**
for *x, y* bounded by *i, i', j, j'* **do**
if M(*x, y)*= “VACANT” **then** *set* M(*x, y*) to “OCCUPIED”
else /* M(*x, y)* is *already* “OCCUPIED”*/ **do** set *y* to “OVERLAPPED”; **STOP end**
end
end;
execute Steps *B, C, D* similarly to STEP *A* for the other 3 corner edges, respectively
end

(ii) 計算量

頂点に罫線座標が付随するため、一般に不均一な多層矩形分割に対して $O(n^2)$である．

物体に占有された全てのセルのサイズが 1 x 1 ならば、STEP　A は定数時間で実行されるため計算量は $O(n)$である．

4.2　3 次元の場合.

最上層の全ての直方体分割と全ての共通コーナー辺について、STEP A と同様の処理を行う.

時間計算量は 2 次元の場合と同様である.

Algorithm 3*DoverlapDetectionWith*40*grid*(*G*, *y*)
INPUT
G(*V*, *E*, *loc*, *occupation*) : a multi-edge 40-ary grid graph with width *m*, depth *n*, hight *h*, layers *l*
loc : a map of *V* toN^8 that indicates the location ({north, south, east, west) x {ceiling, floor} beams) of a node
occupation : a map of V to {VACANT, OCCUPIED} that indicates object occupations.
OUTPUT *y*: NOT-OVERLAPPED / OVERLAPPED
WORK *M*: An *n* x *m* x *h* ARRAY that indicates occupations of *i-j-k* locations throughout whole layers while computation, that is, initially all cell values are VACANT, the value OCCUPIED indicates that the (*i, j, k*)-cell is occupied by exactly one object in algorithm execution, and the value OVERLAPPED indicates that the cell is occupied by 2 objects
METHOD
Initially *set* y to "NOT-OVERLAPPED"**;**
Set all elements of M to "VACANT"**;**
for all cells c(*i, j, k,* 1) (1≦*i* ≦*n*, 1≦*j*≦*m,* 1≦*k* ≦*h*) in the perimeter layer 1 in *G* **do**
/* STEP A*/
2 次元の場合と同様に処理する
execute Steps *B, C, …, H* similarly to STEP *A* for the other 7 common corner edges, respectively
end

4.3　4 次元の場合

上と同様にアルゴリズムを構成する：

Algorithm 4*DoverlapDetectionWith*96*grid*(*G*, *y*)
INPUT
G(*V*, *E*, *loc*, *occupation*) : a multi-edge 96-ary grid graph with width *m*, depth *n*, hight *h*, period *p*, layers *l*
loc : a map of *V* toN^4 that indicates the location ({north, south, east, west) x {ceiling, floor} x {start, end} ruled lines) of a node
occupation : a map of V to {VACANT, OCCUPIED} that indicates object occupations.
OUTPUT *y*: NOT-OVERLAPPED / OVERLAPPED
WORK *M*: An *n* x *m* x *h* x *p* ARRAY that indicates occupations of *i-j-k-t* locations throughout whole layers while computation, that is, initially all cell values are VACANT, the value OCCUPIED indicates that the (*i, j, k, t*)-cell is occupied by exactly one object in algorithm execution, and the value OVERLAPPED indicates that the cell is occupied by 2 objects.
METHOD
Initially *set* y to "NOT-OVERLAPPED"**;**
Set all elements of M to "VACANT"**;**
for all cells c(*i, j, k, t,* 1) (1≦*i* ≦*n*, 1≦*j*≦*m,* 1≦*k* ≦*h*, 1≦*t* ≦*p*) in the perimeter layer 1 in *G* **do**
/* STEP A*/
2 次元の場合と同様に処理する
execute Steps *B, C, …, P* similarly to STEP *A* for the other 15 common corner edges, respectively
end

ここでは、最上層の全ての超直方体分割と全ての共通コーナー辺について、STEP A と同様の処理を行う.

時間計算量は 2 次元の場合と同様である.

5. おわりに

我々は以下を導入した：8*k* 次格子による重なりの定義と同じく 8*k* 次格子を用いた $O(n)$時間と $O(n^2)$時間の重なり検知アルゴリズムを示した. これらのアルゴリムは単一矩形で表現された物体を含む一般の不均一な矩形分割に関して、矩形双対グラフや 4*k* 分木の上の自明なアルゴズムより低い時間計算量で実行される.

§4.2 の結果は、3 次元静止物体の重なり検知と動きのある平面図形の重なり検知に応用可能であり、§4.3 の結果は動きのある 3 次元物体の重なり検知(衝突検知)に応用可能である.

土田賢省氏と野牧賢志氏に感謝します

参考文献

[1] R. A. Finkel and J. L. Bentley, Quad Trees: A Data Structure for Retrieval on Composite Keys, *Acta Informatica* 4, pp.1-9, 1974.

[2] C. L. Jackins and S. Tanimoto, Oct-trees and their use in representing three-dimensional objects *Computer Graphics and Image Processing* 14(3), pp. 249-270, 1980.

[3] N. Inamoto, Hexadecimal-Tree: A Time-Continuous 4D Interference Check Method. In: Falcidieno B., Kunii T.L. (eds) Modeling in Computer Graphics. *IFIP Series on Computer Graphics*. Springer, (1993)

[4] T. Yaku, Representation of Heterogeneous Tessellation Structures by Graphs, *Memoir of WAAP Meetings* 108, 6p, Dec., 2001.
URL http://www.waap.gr.jp/waap-rr/waap-rr-01-013.pdf

[5] R. Yokoyama, A. Kureha, T. Motohashi, H. Ogasawara, T. Yaku and D. Yoshino, Geographical concept recognition with the octgrid method for learning geography anc geology, *Proc.* 7 *th IEEEE ICALT*, pp. 470 – 471, 2007.

[6] S. Koka, K. Nomaki, K. Sugita, K. Tsuchida and T. Yaku, Ridge detection with a drop of water principle, *Proc. SITGRAPH ASIA* 2020 Posters, No. 34 2010

[7] G. Akagi, K. Anada, S. Koka, Y. Nakayama, K. Nomaki, T. Yaku: A resolution reduction method for multi-resolution terrain maps. *SIGGRAPH Posters* 2012: 86

[8] T. Yaku, K. Anada, K. Anzai, S. Koka, Y. Miyadera, K. Tsuchida, 8*k*-ary Grid Graph Models of Tabular Forms, *Springer LNCS* 8373, pp. 465-477, 2014.

[9] T. Yaku, K. Anzai K. Yokota and Y. Miyadera, A 64 degree grid graph model of the time-conbinuous 4D objects, *Proc. ACIT* 3, pp, 133 - 135, 2015

[10] T. Yaku, K. Anzai, Y. Miyadera and K. Yokota, A 40-degree grid model for multiple 3D objects, *Proc.*2016 *IEEE ICIT*, pp, 1678-1683, 2016.

[11] Takeo Yaku, Youzou Miyadera and Masanori Suzuki, 96-ary degree grid graphs for modeling of multiple time-continuous 4D objects, *Proc.*12th *KICSS*, pp.208-209, 2017.