2024年9月4日
FIT2024

8k次格子グラフモデルによる物体の重なり検知

夜久竹夫（日本大）　安斎公士（関東学園大）
横田健（日大学二中高）、横山隆介（応用オートマトン研）

FIT2024　　CA-005

2024年9月4日 広島工業大学 1a会場

スライド
PDF

用語

- *“8k-次格子グラフ”*・・罫線指向処理のための矩形分割のための疑似格子グラフ

位置づけ 問題とデータ構造の計算時間

問題→ データ構造↓	表現力	探索	結合	重なり検知 (複数(超)矩形分割)
矩形双対グラフ	全て	O(n)	O(n)	O(*n*^2)　(自明)
4*k*-分木	非螺旋のみ	O(logn)	O(1) (同一クラスター内) O(n) (一般)	O(*n*)　(自明)
8*k*-次格子	全て	O(n)	O(1)	O(*n*) ← the objects represented by 1 x 1 rectangles O(*n*^2)← otherwise

要約—

- 重なりの定式化
  - 2D 物体、16次格子による
  - 3D 物体、40次格子による
  - 4D 物体、96次格子による
- それらの重なり検知のためのO(*n*) and O(n^2) 時間手法

Keywords— *モデル化; 移動対象; 重なり; 高次元; 8-次格子*

引用文献

1. R. A. Finkel and J. L. Bentley, Quad Trees: A Data Structure for Retrieval on Composite Keys, *Acta Informatica* 4, 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), 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, T. Yaku et.al, 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. SIGRAPH ASIA* 2010 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.

12 S. Koka, T. Yaku, et. al., Tabular forms editing with a hexadecimel grid graph model, Proc. VL/HCC 253-254, 2011

13 K. Anada, T. Yaku, et. al., Algorithms for ridge and valley detection in terrain maps, *J. Comput. Methods in Sci. & Engin*. 17, 95 – 110, 2017.

1. はじめに

分野: CADや自律移動、効率的な計算

主題 :

複数物体のためのデータ構造

重なり検知

対象 : 複数の 2D, 3D, 4D物体

●我々のゴール:

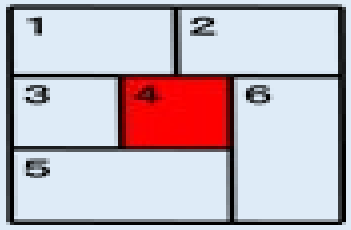
複数の 4D物体(移動する3次元物体)
の個別処理のための
データ構造とアルゴリズム

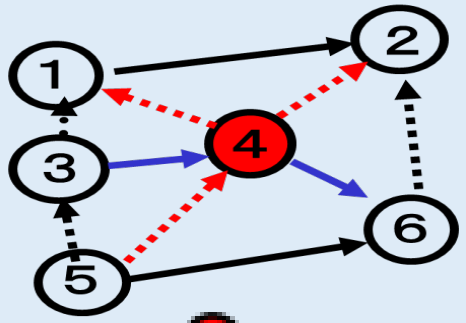
- 背景

矩形分割のグラフ表現

矩形双対グラフ

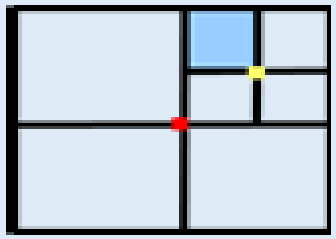
1800s

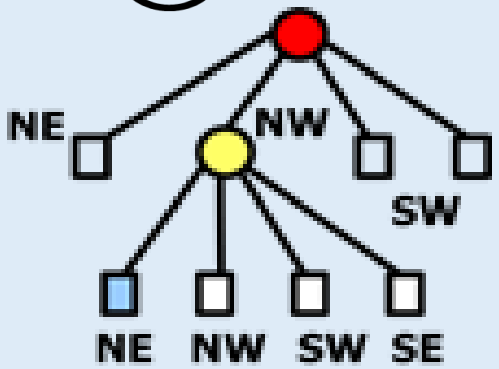




4*k*-分木[1]

Finkel et al, 1974



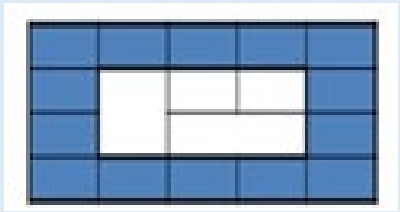


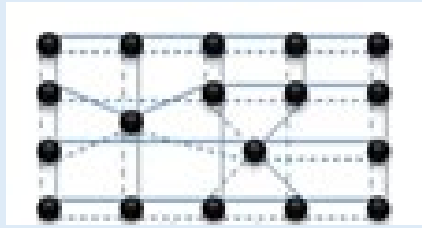
SW: SouthWest

・ ・ ・

8*k*-格子[8]

Yaku 2002～





背景 (続き)

- ８分木 for 3D, Tanimoto, 1980 [2]
- 16分木 for 4D, Inamoto, 1993 [3]
- 衝突検知、8分木、ゲームの世界
  自律移動、CAD
- 16次格子、多層8次格子, Yaku 2011 [12]

動機：

高速の重なり検知手法：期待

目的：

1. 8*k*-次格子を用いた重なりの定式化
2. 2D物体の重なり検知
3. 3D, 4D物体の重なり検知

結果：

1. 8*k*-次格子を用いた重なりの定義
2. 16次格子を用いた、2D対象のO(*n*) and O(*n*^2) 重なり検知アルゴリズム [8].
3. 40-次と96-次格子を用いた3D、4D対象のO(n) と O(n^2) 重なり検知アルゴリズム [8].

応用:

速い衝突検知システム：

CAD ・・・・部品配置の整合化

自律移動・・自律走行

飛翔物体

2. 準備

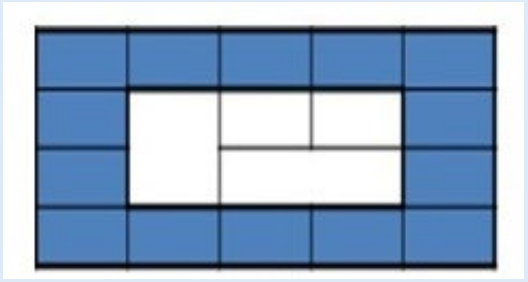
罫線処理を指向した”8*k*-次格子”の解説

(1) “8-次格子” : 単一の矩形分割とその罫線指向処理のためのデータ構造

定義2.1　(8次格子)

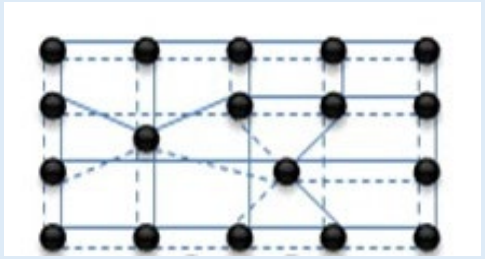
D : 矩形分割

(多重辺)グラフ G_D : D*に対する8次格子*　⇐def⇒　例で説明↓

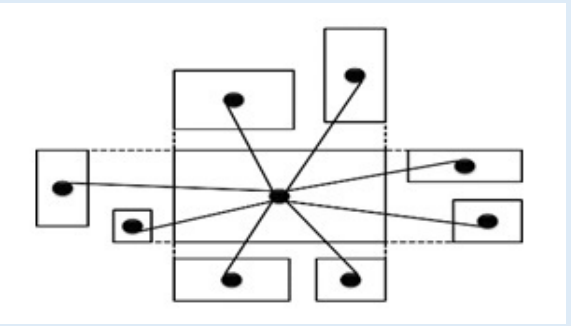


D : 矩形分割

Fig D



Dに対する8次格子 G_D



G_Dの局所構造
2頂点がリンク
⇔対応する矩形がもっとも近くで罫線(壁)を共有

■

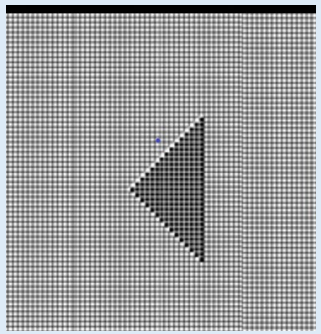
グラフ G : *8次格子*

<=def=> G : ある矩形分割に対する8次格子　■

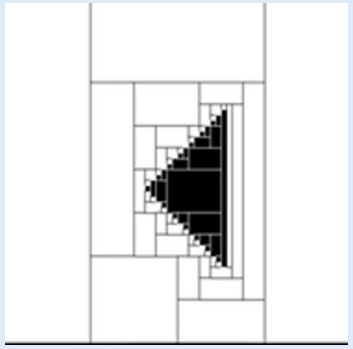
8次格子の応用 ：解像度低減と回転

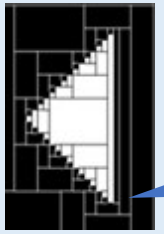
少ないセル -> 速い表示

8次格子は早いアルゴリズムを提供-> 速い表示処理

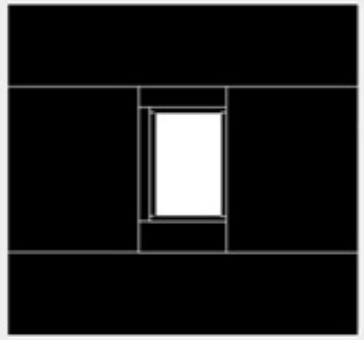


解像度低減

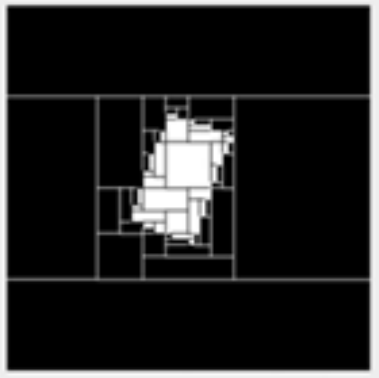




罫線のイメージ



回転

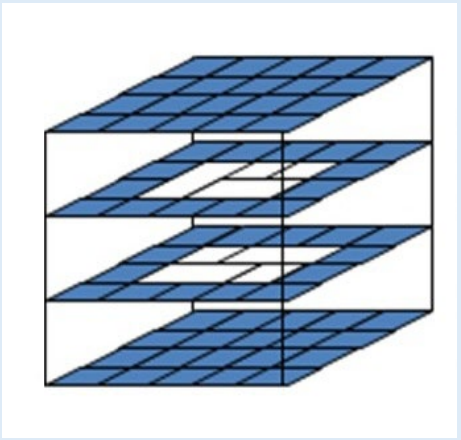


白黒逆

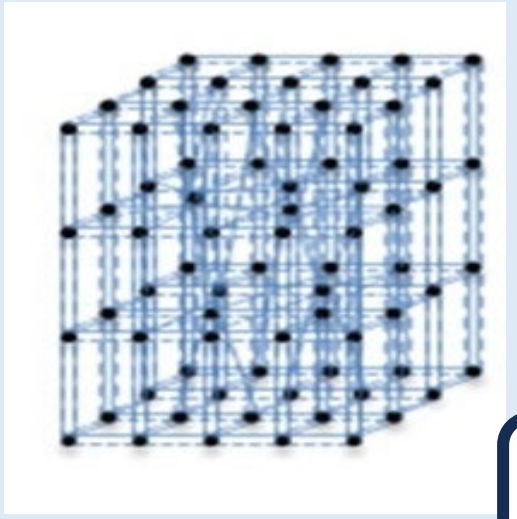
Fig E (Kubota)

(2) “16次格子”：複数矩形分割のデータ構造[12]

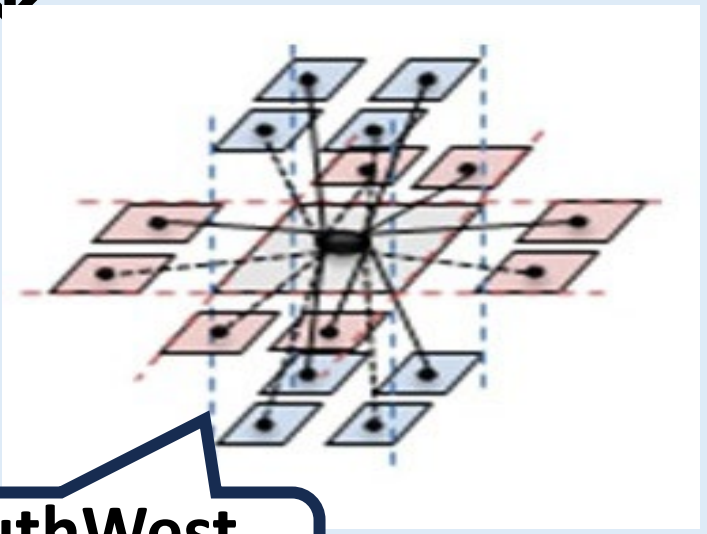
16次格子：共通のコーナーを持つ直近のセル同士をリンクして多層化、以下の図参照↓



多層矩形分割



対応する16次格子



局所構造

Fig G (Kureha)

16次格子の複数対象への応用

ターゲット

- **複数ページ表 (商用書類)**
- **多層地図 (地理学)**
- **複数平面部品 (CAD)**

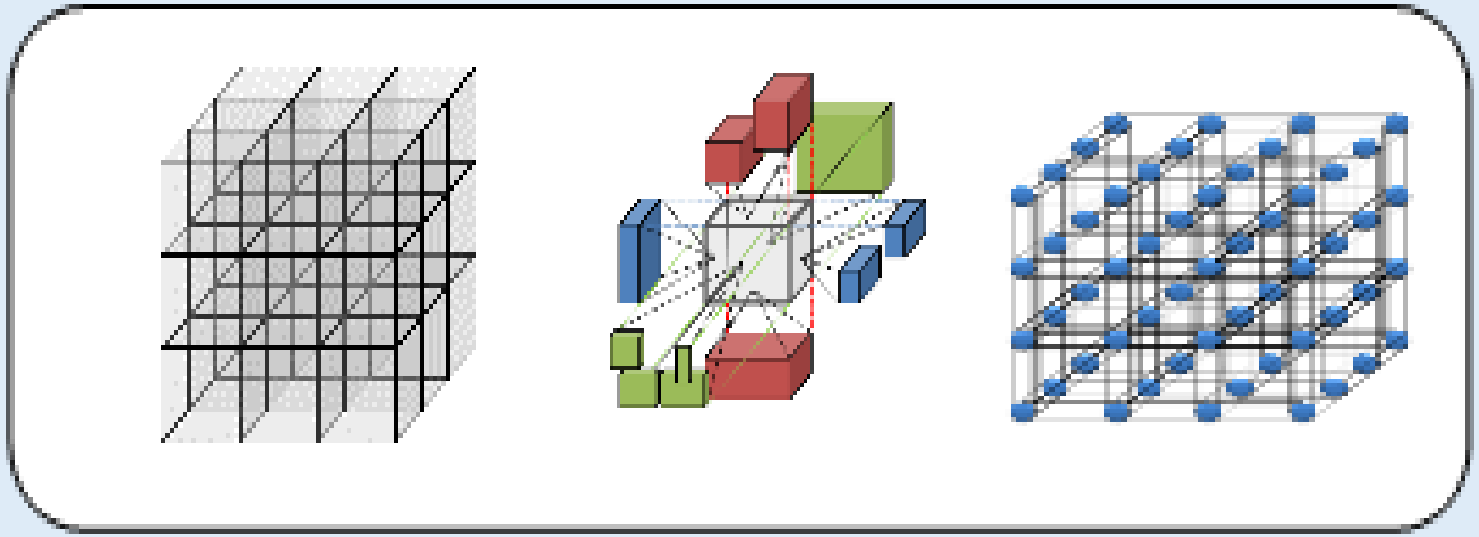
処理の例

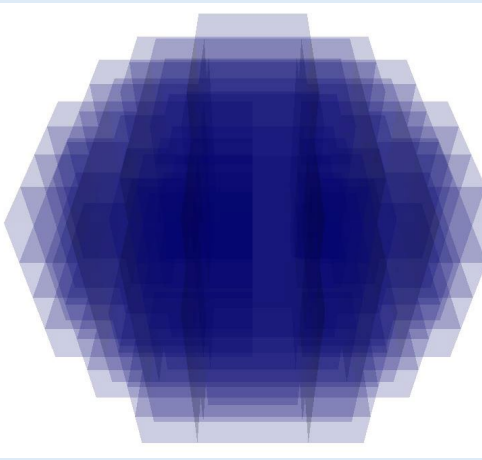
- **特定位置のセルの複数ページ表の値の合計**
- **個別の変換**
- **衝突検知・重なり検知**

3D

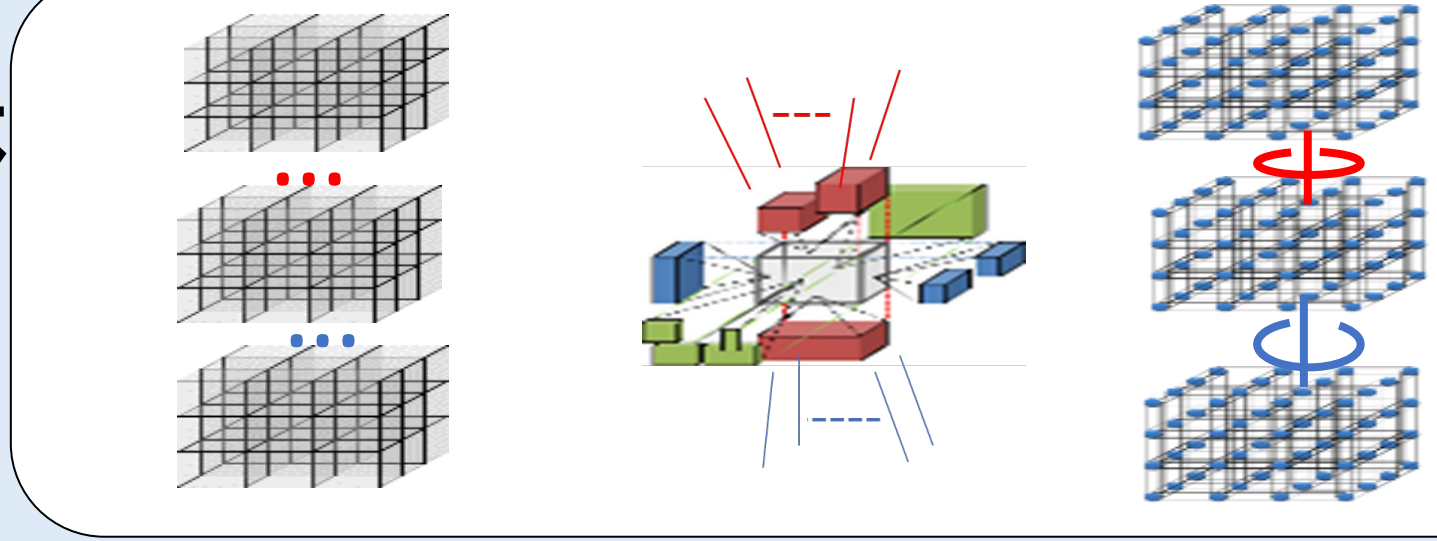
例

単一3D物体
24次格子

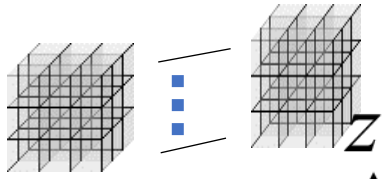




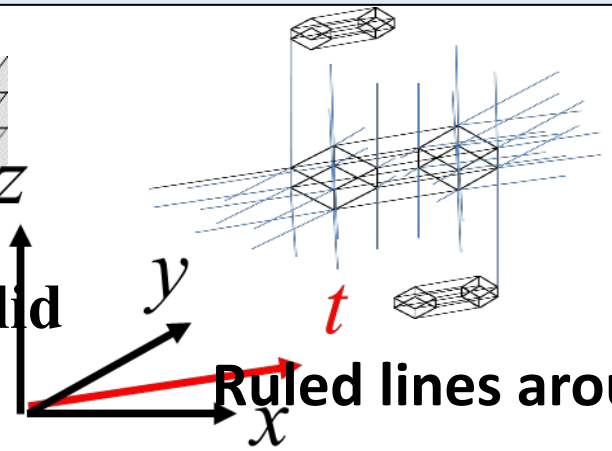
複数3D物体
40次格子



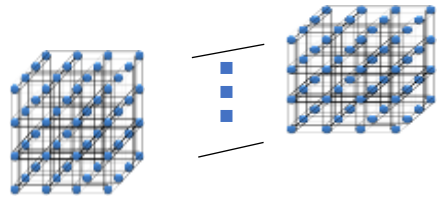
単一4D物体
64次格子



A 4D hyper rectangular solid dissection



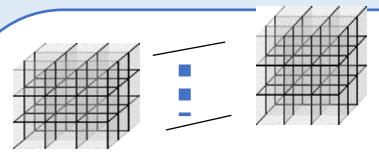
Ruled lines around a cell



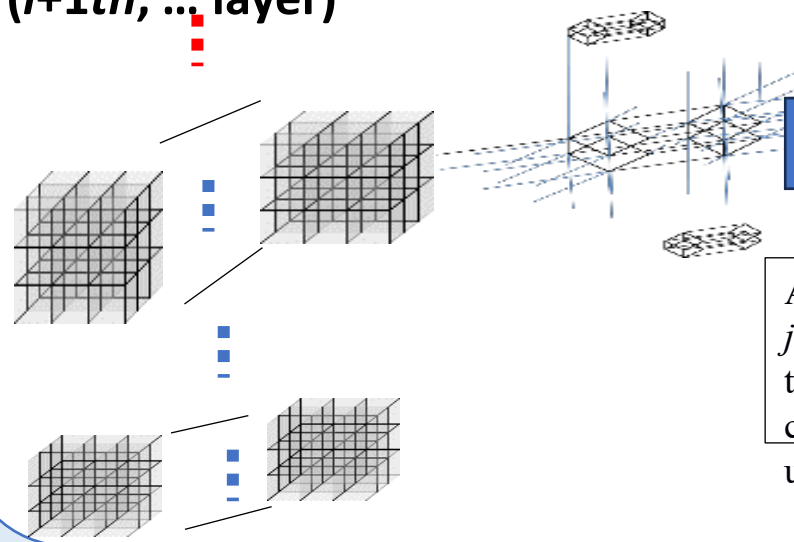
A 64 degree grid graph

Fig 3

複数4D物体
96次格子



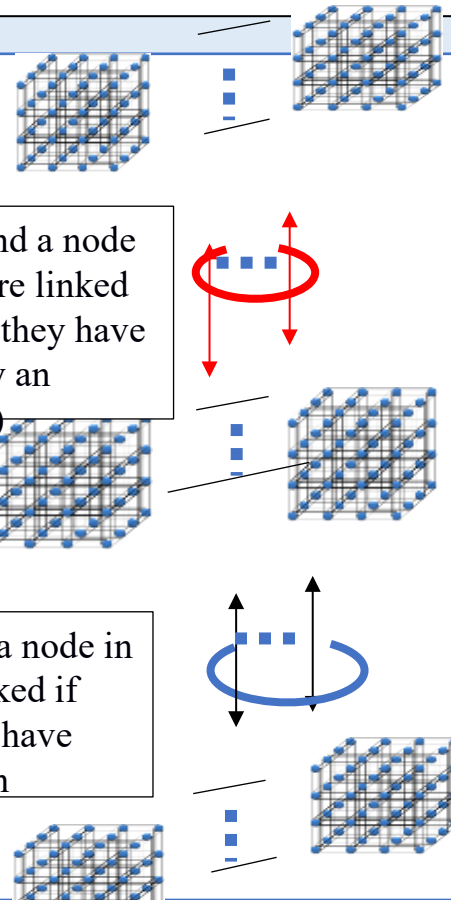
A successor layer
(*i*+1*th*, … layer)



A node in *i*-th world and a node in *j*-th world ($i < j$) are linked if they are nearest and they have corners in common, by an undirected edge (right)

Local structure of Node

A node in *i*-th world and a node in *j*-th world ($i > k$) are linked if they are nearest and they have corners in common, by an undirected edge (right)



3. 8^{K}次格子を用いた重なりのモデル化

（はじめに2D の場合）

次の16次格子を考える：

各頂点は黒●(占有)か白〇(空虚)で色付けと仮定、

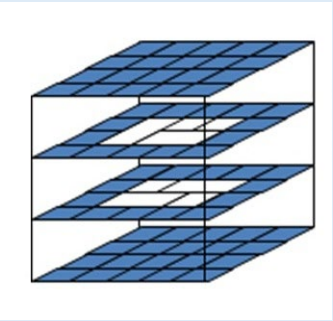
各物体はある層の黒の頂点の集まりで表現[7]、

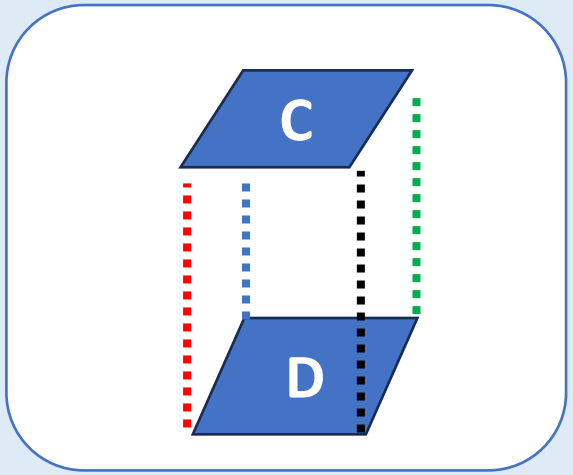
即ち二つの物体は二つの層で表現.

重なり(→衝突)の定義

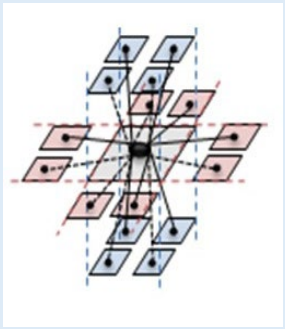
定義　(重なり)

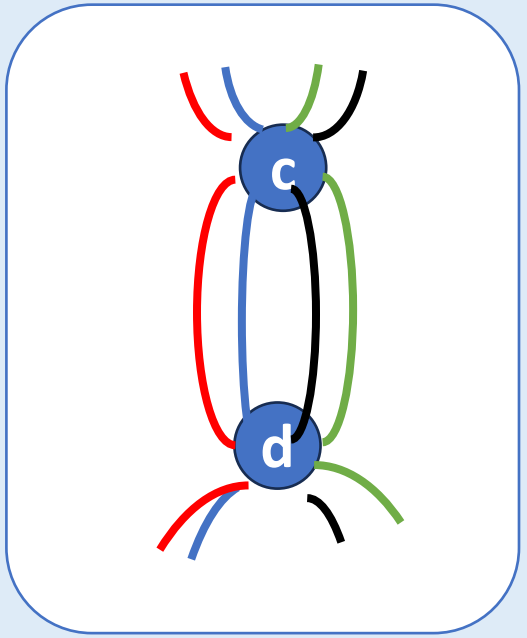
C、D **をそれぞれ多層矩形分割の層** ***l*** **、** ***k*** ***のセルとする.***
異なる層のセル ***C*** **と** ***D*** **は真に重なる**
<=def=> ***C, D*** **: 占有されている**
C **と** ***D*** **は４コーナーを共有している**　■





16次格子の"真に重なる" **も同様に定義される**





注意。 **この場合、二つのセルの重なり検知アルゴリズムは　O(1) 時間**

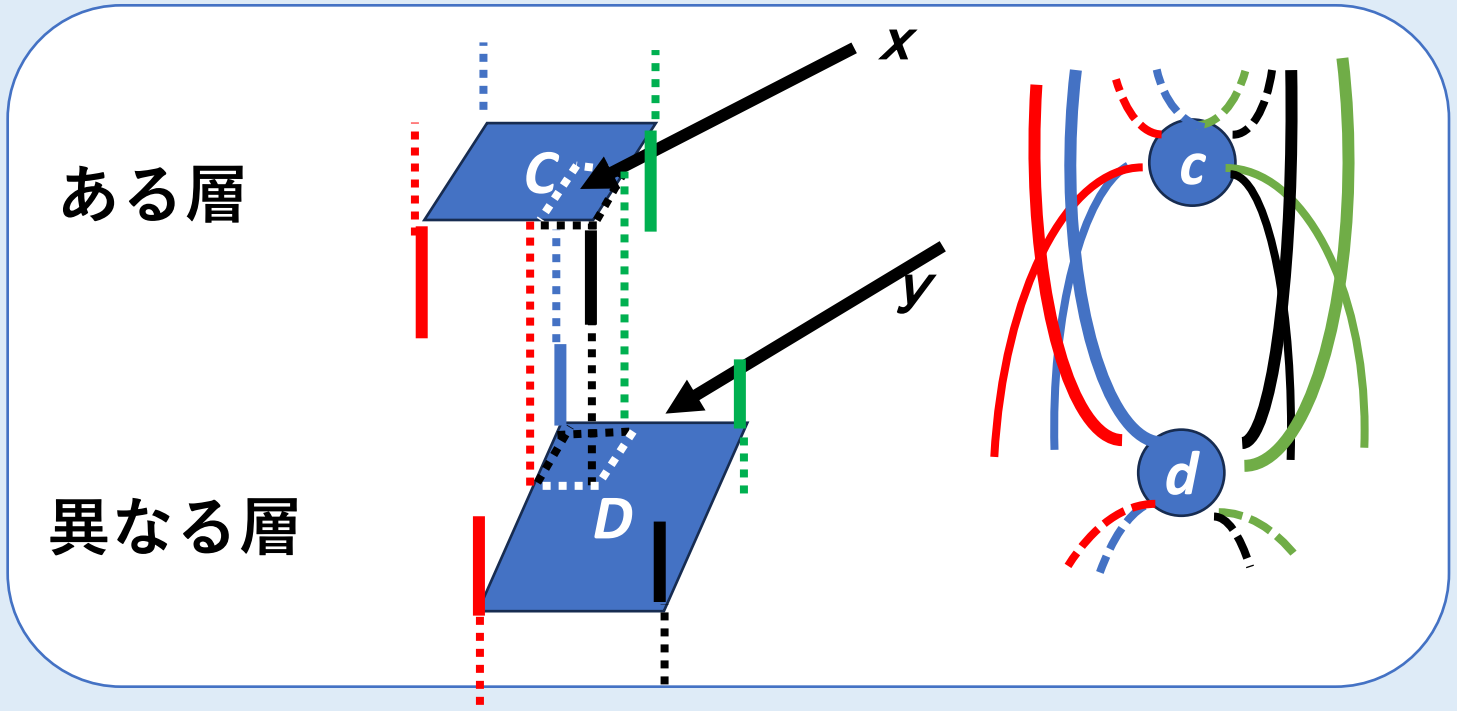
物体: 多層矩形分割の同一の層に属する連結された占有セル集合

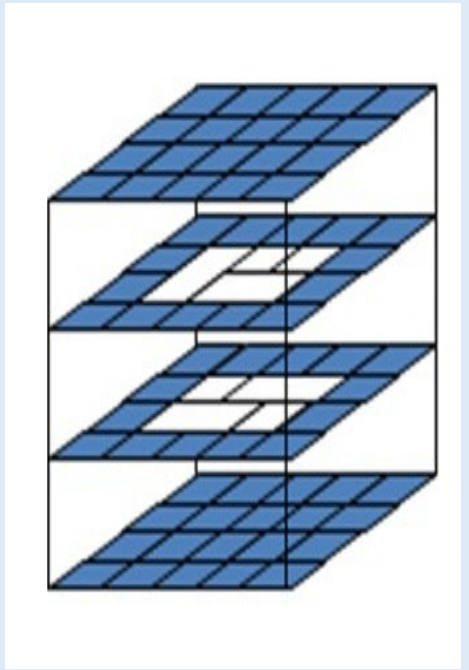
定義. 物体*P* と異なる層に属する物体*Q* : (部分的に)*重なる*

<=*def*=>ヨセル *C* in *P*, ヨセル *D* in *Q* : 下のように”部分的に重なる”

■

16次格子の*物体*と*物体の重なり*も同様に定義.



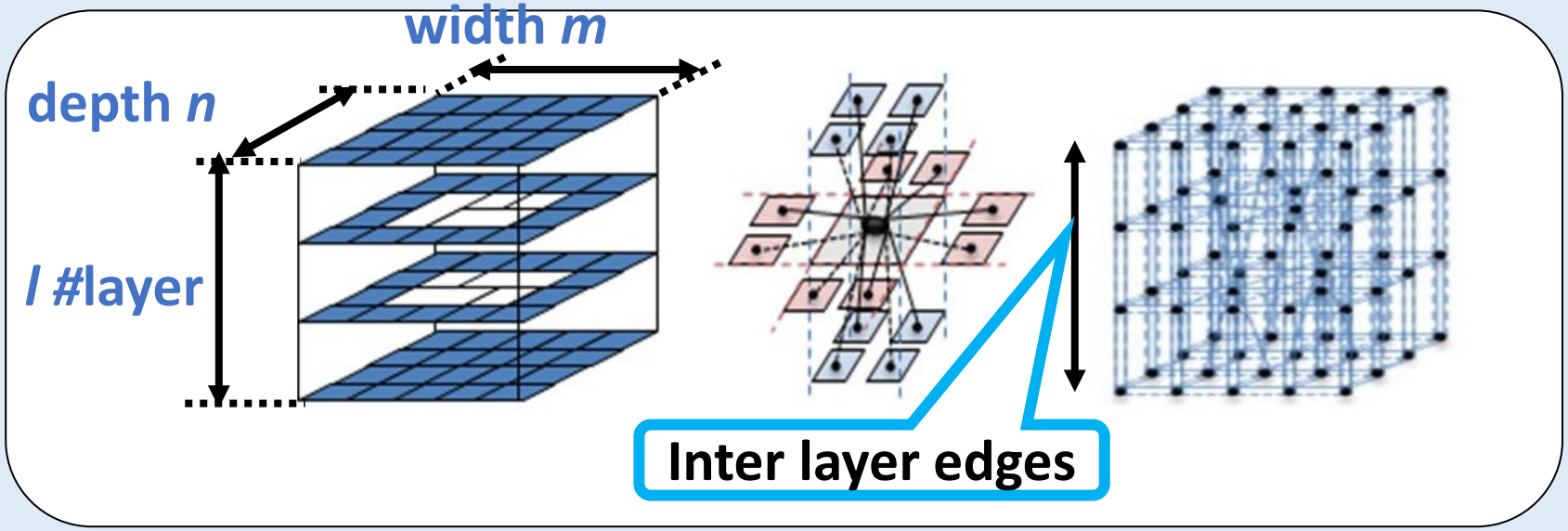


注意. この場合*C*と*D*の重なり判定は O(*n*) 時間■

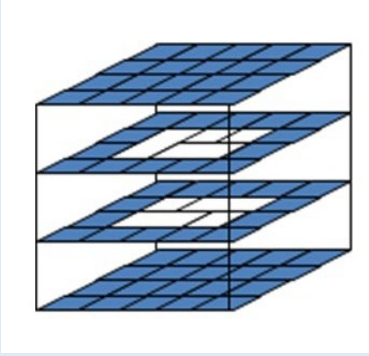
4. 重なり検知アルゴリズム

(1) 2D物体の重なり検知

l 層の n x m 矩形分割(例：左)と
対応する16次格子(例：右)を考える



Algorithm 2*DoverlapDetectionWith*16*grid*(*G*, *y*)



INPUT

G(*V*, *E*, *L*, *loc*, *occupation*, *v*0) : 幅*m*, 奥行 *n*, 層数 / の16次格子
L : **辺の北壁、南壁、東壁、西壁の種別**

loc : *V* → N^4 : セルの位置(**北壁、南壁、東壁、西壁の番号**)
occupation : *V* → {*VACANT*, *OCCUPIED*} : 占有／空虚の種別

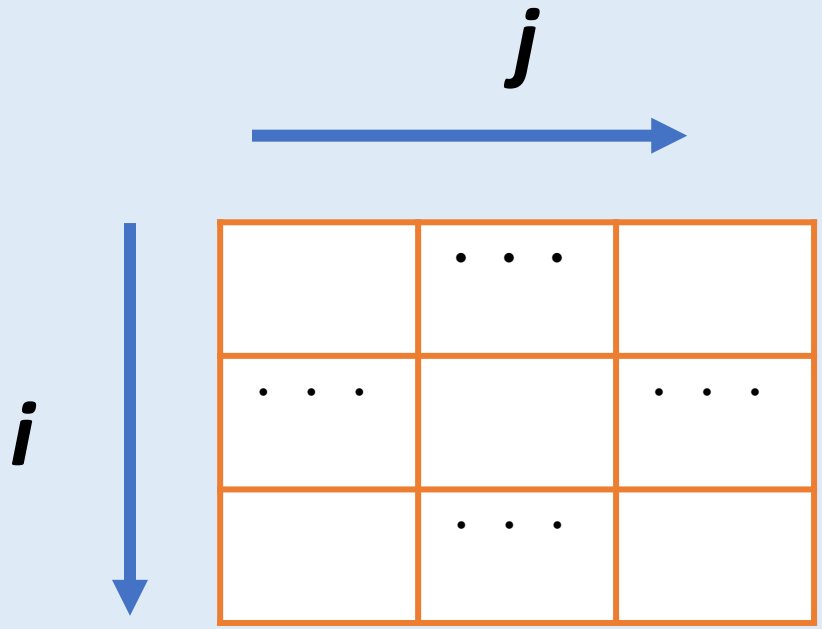
v0 : 基準頂点、*v*1,1と仮定

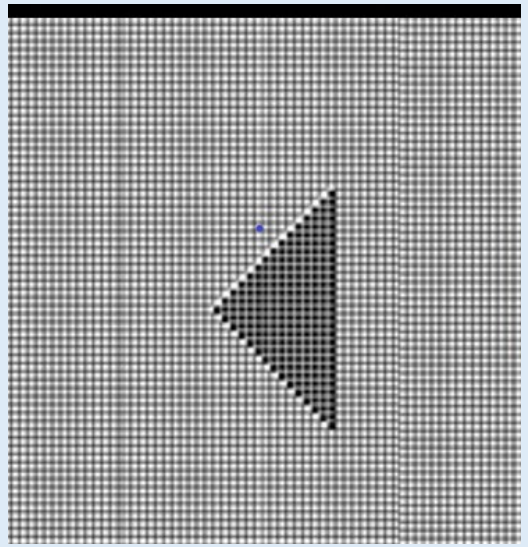
OUTPUT

y: NOT-OVERLAPPED / OVERLAPPED

補助変数K

M: An *n* x *m* ARRAY that indicates occupation of each *i*, *j* location while computation

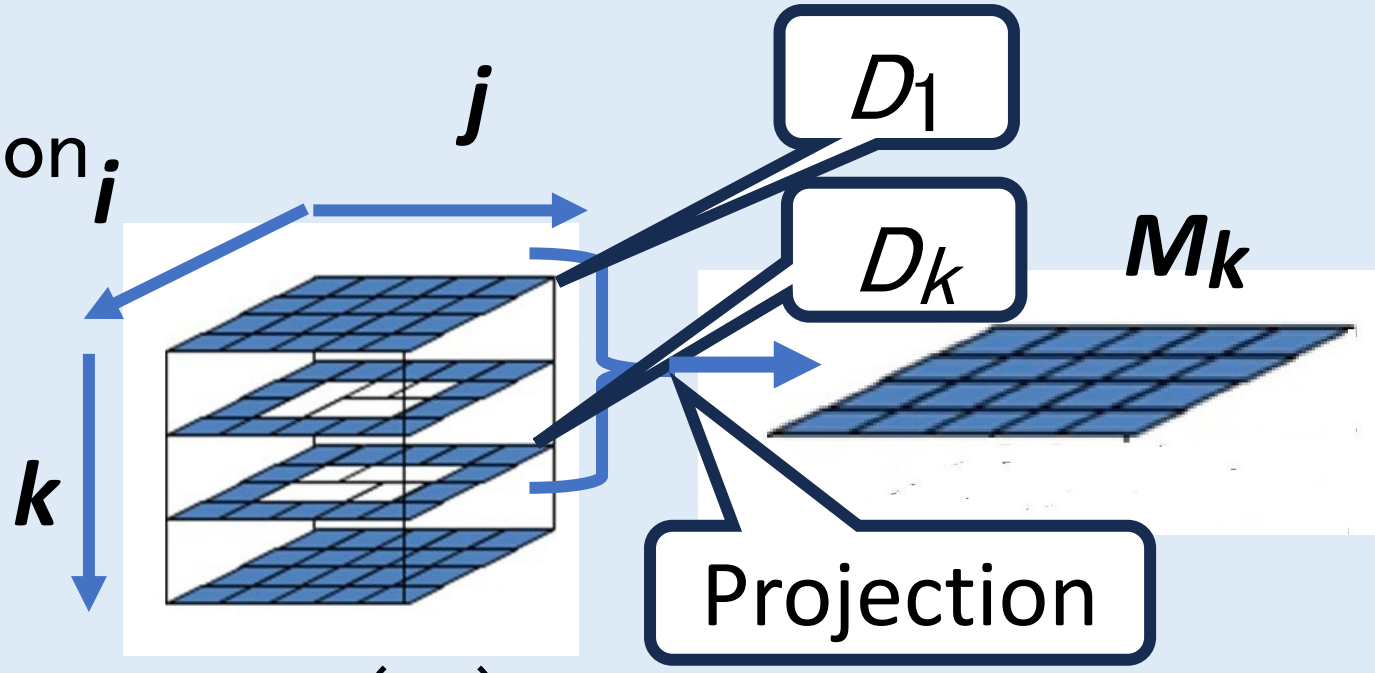




初期化で最初は　M の値は全て *VACANT*,

手順概要

Suppose that $\mathbf{M}_k$ is a projection of $D_1, \cdots, D_k$ with respect to occupation in STEP k.



In STEP k,

$\mathbf{M}_k(I, j)$ = VACANT if $D_1(I, j) = \cdots = D_k(I, j)$ = VACANT

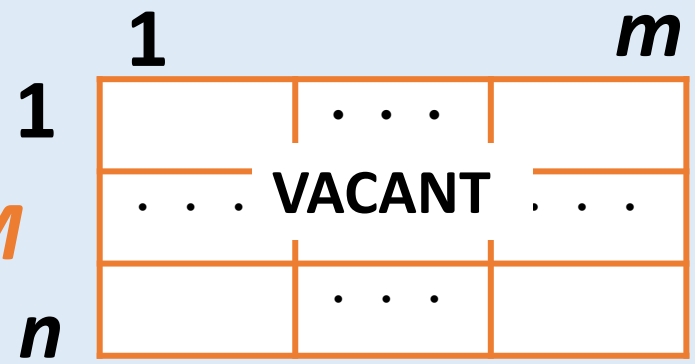
= OCCUPIED ⇐ exactly one $D_k'(i, j)$ of $D_1(i, j), \cdots,$

$D_k(i, j)$: OCCUPIED and the others : VACANT

if $\mathbf{M}_k(i, j)$ is already OCCUPIED and $D_{k+1}(i, j)$ = OCCUPIED then the output y is OVERLAPPED and the computation STOP.

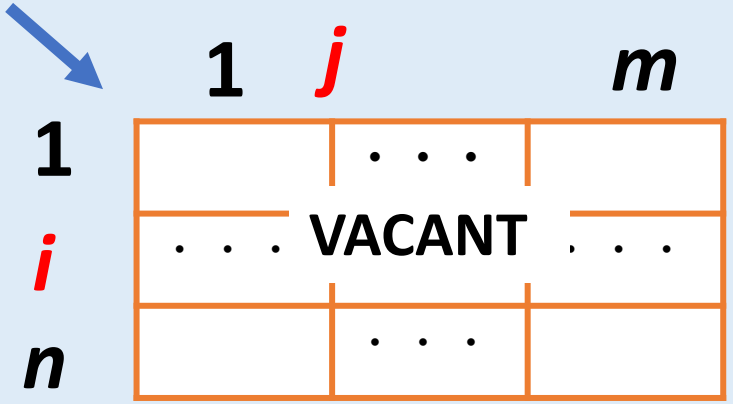
METHOD

The work matrix M



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 layer 1 in *G* **do**



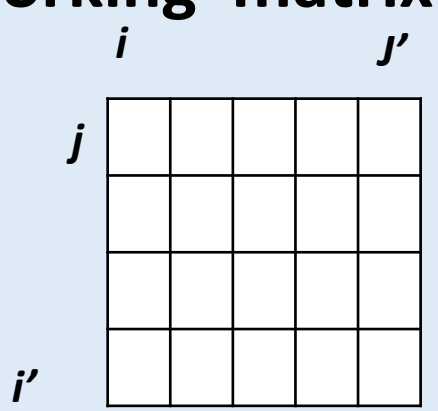
/* STEP A*/

repeat until the visit reaches to the bottom layer **do** visit
downward cells *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;

STEP Aの説明例 1
Mがvacant の場合

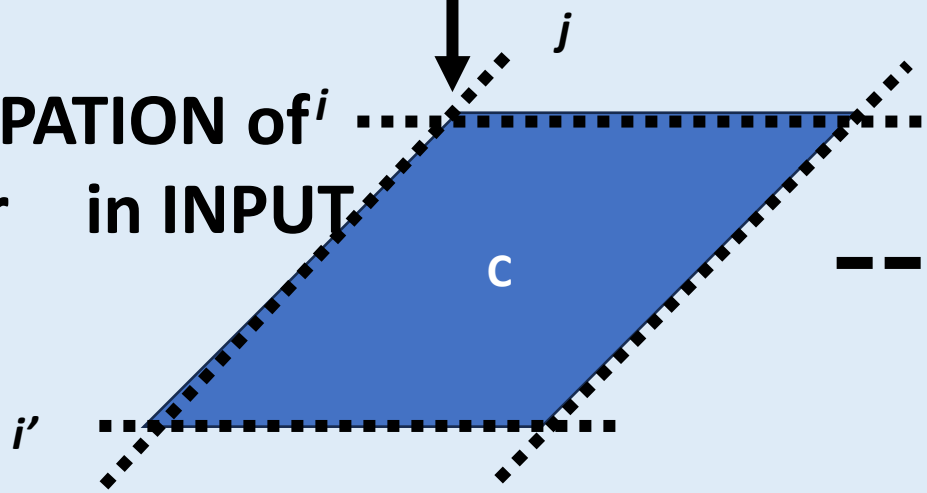
The working matrix M, a part



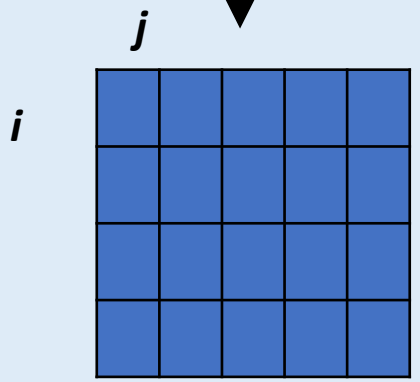
k-th repeat loop の後

Ex. OCCUPATION of *i*

k+1 layer in INPUT

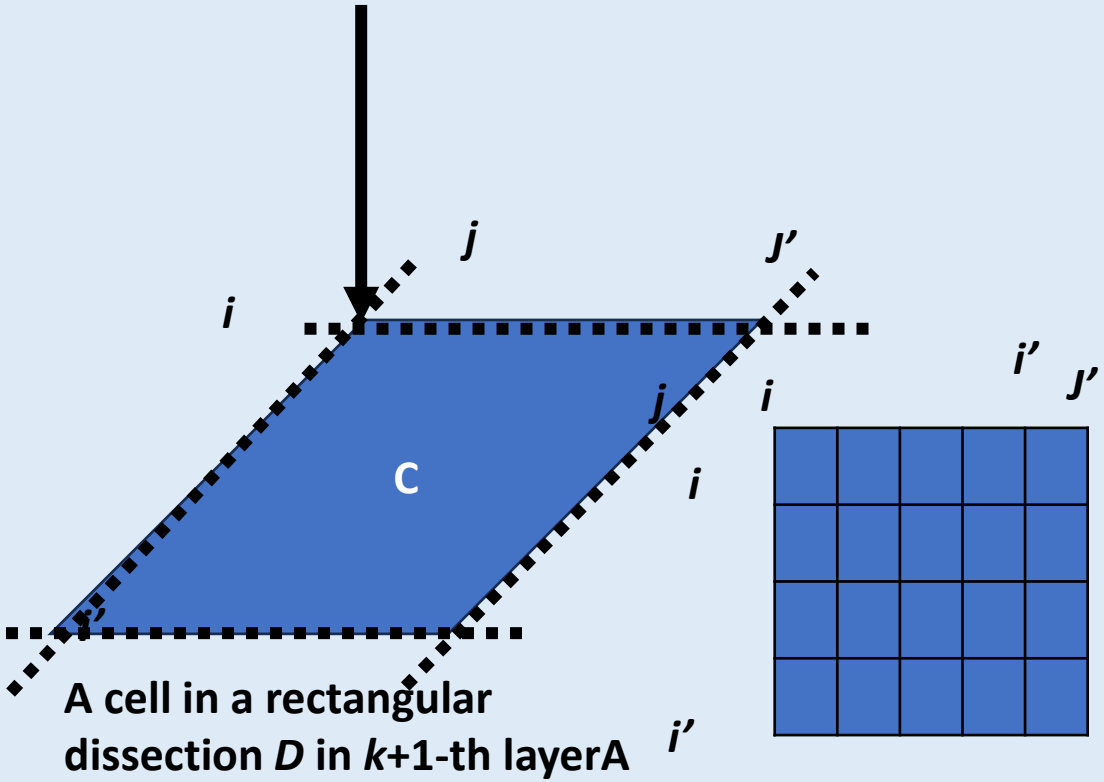


A cell in a rectangular dissection *D* in *k*+1-th layer



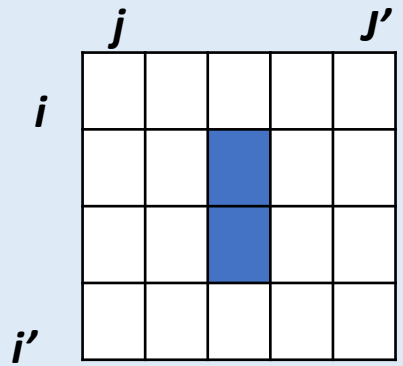
k+1-*th* repeat loop の後

STEP Aの説明例 2
Mがvacantでない場合



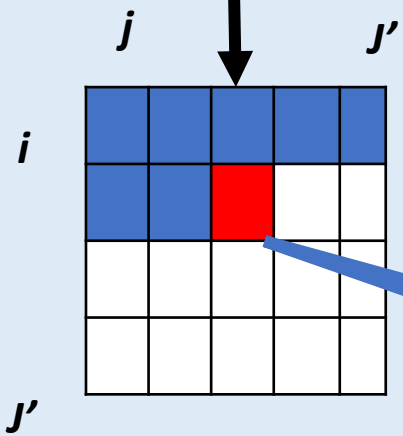
Ex. OCCUPATION of *k*+1 layer

The working matrix M, a part



***k-th* repeat loop**

Ex. OCCUPIED



***k*+1-*th* repeat loop**

STOP

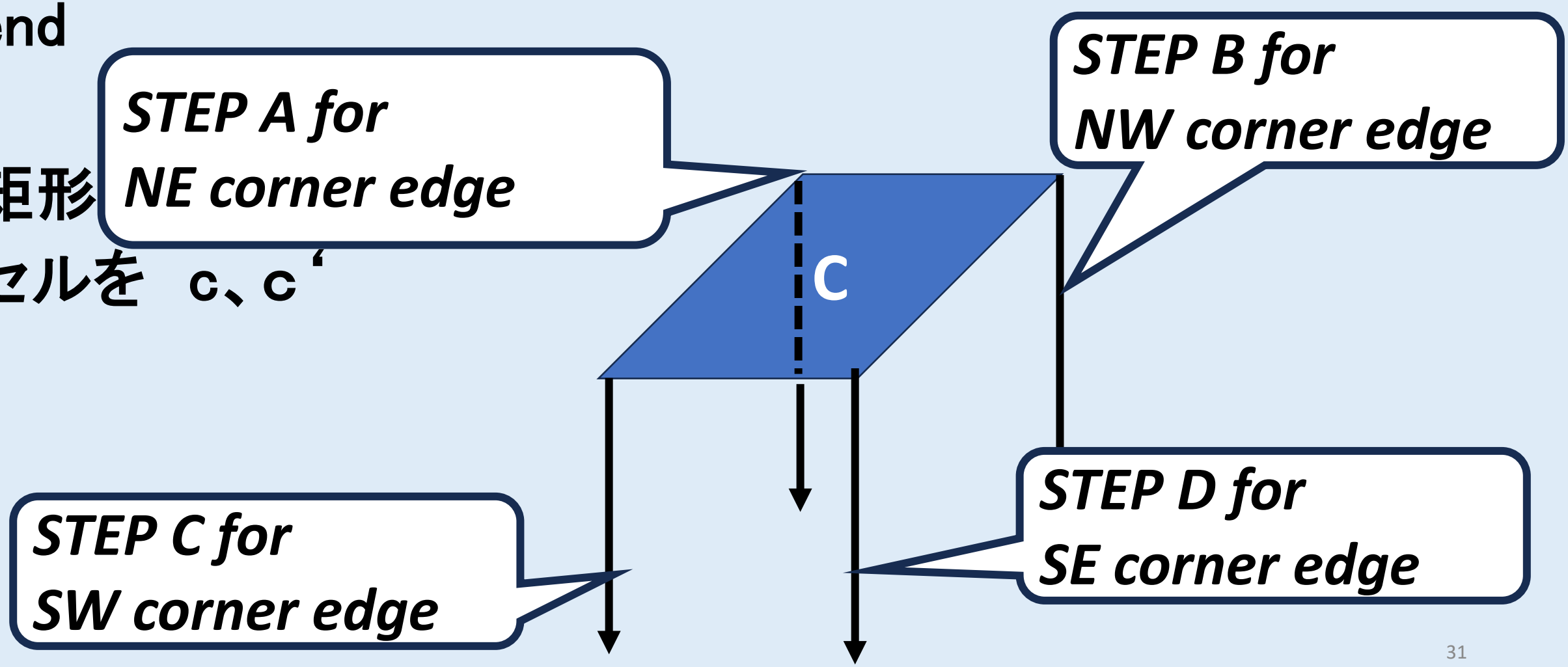
OVERLAPPED!

/*念のため他のコーナー処理→不要？*/

execute Steps *B, C, D* similarly to STEP *A* for the other 3 corner edges, respectively

end

矩形セルを　c、c‘



命題。多層型矩形分割Dで、部分的に重なる二つの図形*C*と*D*が存在

⇔ Algorithm 2DoverlapDetectionWith16grid(G_D, y)はOVERLAPPEDを出力する、ここでG_DはDを表す16-次格子グラフ

(ii) 計算時間

- 一般に O(*n*^2)

  ∵)各頂点の位置(壁の罫線番号)は各頂点に付随
  （もし無ければ、前処理でO(*n*)時間で付加

- セルのサイズが全て 1 x 1 ⇒ O(*n*)

  ∵)この場合 STEP A はO(1)時間

- 追加。 一般にもO(n)

  ∵) 各層に注目すると、頂点は2度はvisitされない、
  Mの要素は2度は書き換えられない

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8個について
- 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.

4.3　4次元 の場合

同様の結果を得る（予稿参照）

前処理アルゴリズムも同様

- 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

- 以下を導入：
  1. 8k次格子による重なりの定義
  2. 同じく8k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．(実際の計算時間はどちらが早いか分からない)

- O(n)のアルゴリムは一般の不均一な(超)矩形分割で表現された物体に関して、
  - 矩形双対グラフ上の自明なアルゴズムより低い時間計算量
  - 4k分木のグラフ上の自明なアルゴズムより適用範囲が広い

- 応用
- §4.2は3次元静止物体と動きのある平面図形の重なり検知に応用可能、
  §4.3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.

- 今後

  実装

  8次格子　データ済、他のアルゴリズム済

  16次格子　データ済　他のアルゴリズム済

  24次格子　データ済　他のアルゴリズム済

  40次格子　データ途中

  64次格子　データ途中

  96次格子　データ途中

- 土田賢省氏と野牧賢志氏に感謝します

THANK YOU

予備用

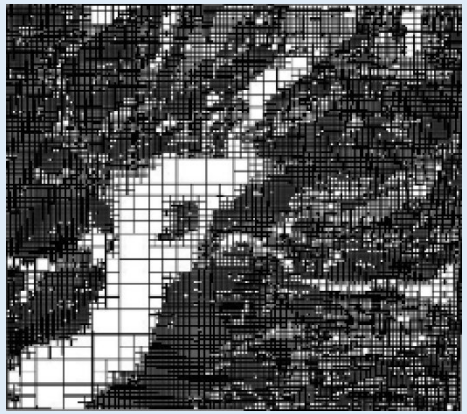
アルゴズムに16次である事はどう反映しているか？

　隣接しているセルを16個以上探す必要が無い

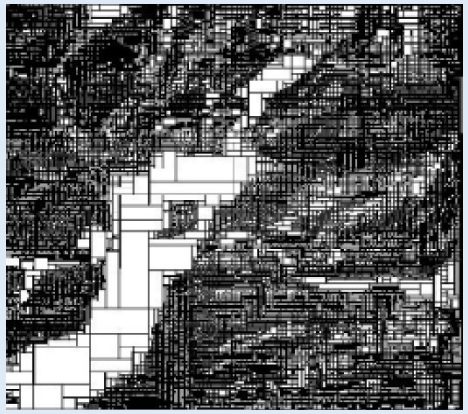
アルゴリズムに罫線はどう反映しているか？

　各セルの座標が入っている。

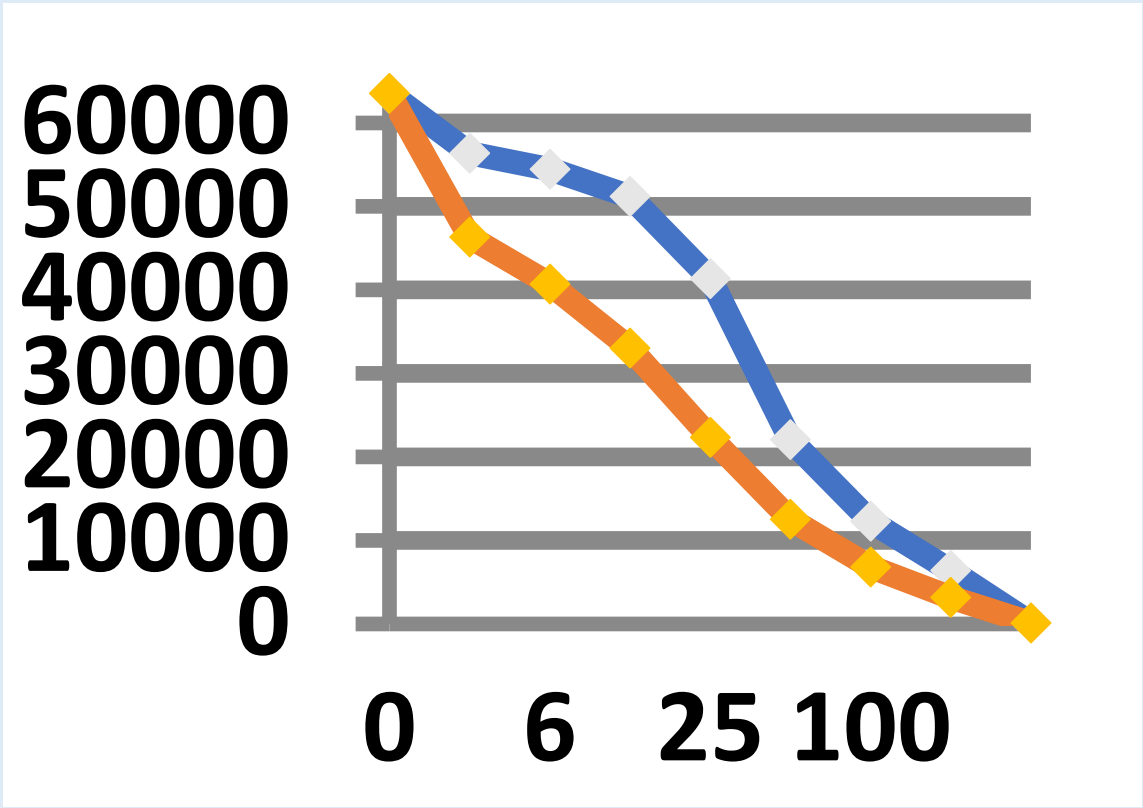
Comparison of quadtree method and the octgrid method with respect to resolution reduction　１５分



Quadtree methods(in blue)



Octgrid methods in orange



SIGGRAPH 2012 Posters