空間検索における索引付け技術の実装と評価

島根大学　総合理工学部　数理・情報システム学科
計算機工学講座　卒業論文
田中研究室
ｓ９９４２５－Ｕ　　喜代吉　容大

目　次

第１章　序章

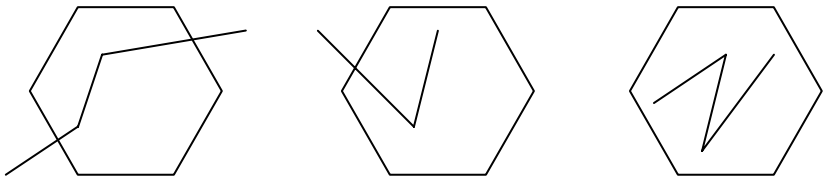
近年、画像、音などのマルチメディアデータを取り扱うことが多くなってきた。それとともに、多くのマルチメディアデータの中から、求めるデータをすばやく探し出す技術も発展していた。その中のひとつの手法として、索引付けがある。本研究では、特に木構造を用いた索引付けについて実装する。また、従来までの索引付けをしない幾何学的な検索方法と実際のデータを用いて比較することで、索引付け技術の有用性を示すことを目的とする。

第２章　幾何学的計算による空間検索

参考文献［３］

2.1　幾何学的な空間検索

本研究では、行政界と道路の Overlap を検索することに焦点を置いた。つまり、多角形と折れ線の Overlap である。Overlap するとは次の図に示すような場合である。

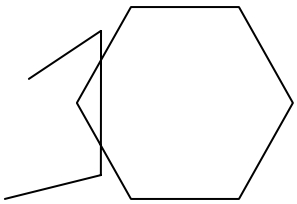


このような場合には Overlap すると言い、そうでなければ Overlap しないと言う。この定義は Cross（交差）や Within（包括）を含んでいるが、Overlap には、部分的な重なりなどの意味もあるため、この言葉を選んだ。また、この定義は、Open GIS Consortium によるものである。詳細は「http://www.opengis.org/」を参照されたい。

このような質問（以下クエリ）を幾何学的な手法によって解決する方法をいくつか考えてみた。すると、「点位置決定問題の応用」と「スラブ法」の大きく２つが挙げられる。他の方法もいくつか考えられるが、多角形の判定には凸多角形などのように制限を持ったものが多く、本研究の対象である行政界には不向きであった。また、スラブ法とは走査線を用いて、高速な検索を実現するものである。これは、あらかじめ前処理を要することから、索引付けに似たものがある。このようなことから、今回は「点位置決定問題の応用」を利用して検索の比較対照とすることにした。

2.2　検索アルゴリズム

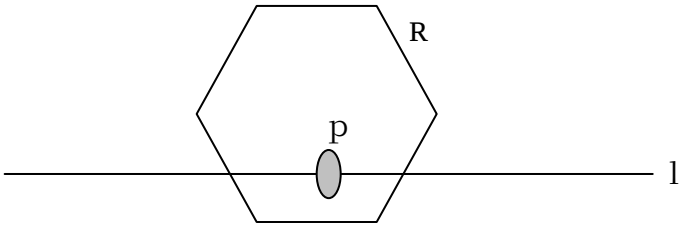
点位置決定問題とは内点判定問題のことである。つまり、「点 p は領域 R 内に存在するか」という問題である。この作業を、折れ線を構成するすべての点に対して調べることで、多角形と折れ線の Overlap が検出できる。しかし、折れ線の構成する点が領域内に存在しない場合でも、図のように Overlap する場合がある。



このような場合は、各辺同士の交差判定を行う必要がある。この方法は後に詳細に述べる。

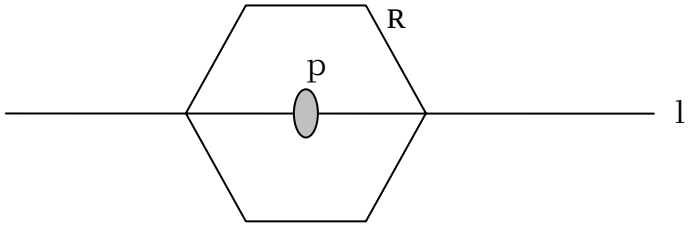
2.3　点位置決定問題

点ｐが領域 R の内部にあるか無いかは前処理なしで O(n)の手間で判定できる。



点ｐを通る水平線ｐを考える。l が R と交差しないならば､点ｐが領域 R の外部である。まず、直線 l が R の頂点と交差しない場合を考えてみよう。L を、直線 l の左から点ｐまでの線分と、領域 R の境界線との交点の数とすると、L が奇数のとき点ｐは領域 R の内部に存在し、L が偶数のとき点ｐは領域 R の外部ということになる。

次に、直線 l が R の頂点で交差する場合を考えてみる。



この様な場合、直線 l は左側で領域 R と２点で交差している。正確には、２辺の端点で交差している。しかし、この点位置決定問題においては、これを１点と判定する必要があ

る。そこで、次のような方法によって解決した。

辺の端点以外で交差した数：ａ　　辺の端点で交差した数：ｂ とすると交点数は「ａ＋（ｂ÷２）」と表せる。 ここで、ｂは必ず偶数となることに注意していただきたい。

以上の方法で、点位置決定問題は解決できる。ここで、点位置決定問題についてまとめてみる。

領域：R　　調べる点：ｐ　　無限遠点とｐとを結ぶ線分ｌとすると、ｌと R の境界線との交点数が奇数の場合、点ｐは領域 R 内に存在し、偶数の場合、点ｐは領域 R の外に位置する。

この方法で折れ線を構成する点すべてに対して行うことで、Overlap の検出を行うことができる。また、2.2 節で述べた交差判定については、この方法を行っていく際、折れ線の次の点との線分を考え、領域の各辺との交差判定を行うことで検出した。このことで、１度でも交差していたなら、そこで Overlap を検出でき、枝刈りの役目を果たした。この、点位置決定問題を用いた Overlap の検出における計算量は、最悪の場合、以下のとおりである。

領域データ数：N　折れ線データ数：M　とすると O(MN)

第３章　幾何学データにおける索引付け技術

参考文献［1］［2］

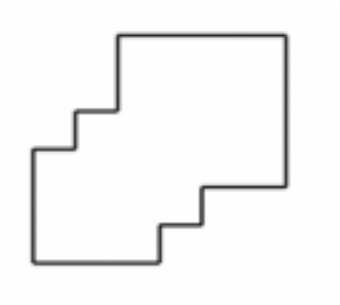
3.1　空間検索における索引付け技術

索引付け技術には、木構造を用いたものだけではなく、非常に多くの技術が存在する。そこで、本研究では２次元データを扱う木構造を用いた索引付け技術の中から、Quad-Tree と R-Tree の２つに絞って研究を進めることにした。

3.2　Quad-Tree と R-Tree の概要

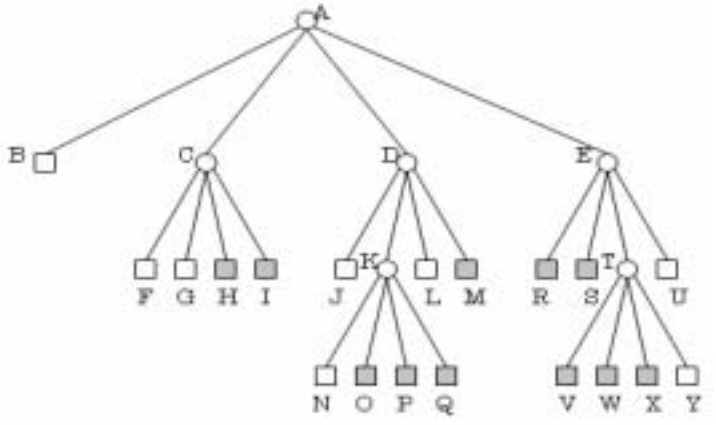
3.2.1　Quad-Tree

この索引付けは、４分木や４進木などとも呼ばれるものである。ある与えられた領域に対して、それを包括する正方形を考える。その正方形を根ノードとして、北東・北西・南東・南西に４等分する。このとき、それら４つのエリアにデータが存在すれば子ノードとし、存在しなければ子ノードとせず分割を終了する。この作業をある深さまで再帰的に作業することで木を生成するものである。



(a)

(b)

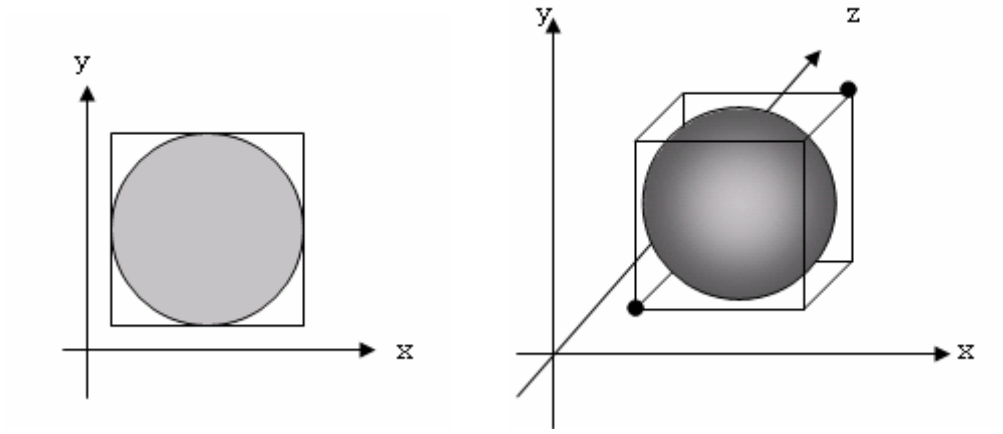


(c)

図は(a)のようなエリアが与えられたとき、(b)のように分割される。また、(c)は(b)をもとに Quad-Tree にしたものである。図からわかるように、データが存在しない場合、もしくはエリアの内側で、これ以上分割する必要がない場合はその時点で分割を終了して葉ノードとする。従って、葉ノードまでの深さは均一ではない。また、ある深さまで分割を繰り返すわけだが、この深さと探索の精度が大きく関係している。深さを深くすることで、葉ノードの大きさが小さくなり、限りなく実際のデータに近づくのである。つまり、深さを大きくするほど精度が上がると言える。しかし、深さを n とすると、ツリー作成の最悪の場合の計算量は O（$4^0+4^1+\cdots 4^n$）{O（4^n)} となる。このことから、深さを深くすると計算量は膨大に増えることになる。このように、索引付け技術は探索の前の処理に時間を必要とする。この前処理に必要とする時間も本研究における大きな焦点の１つである。

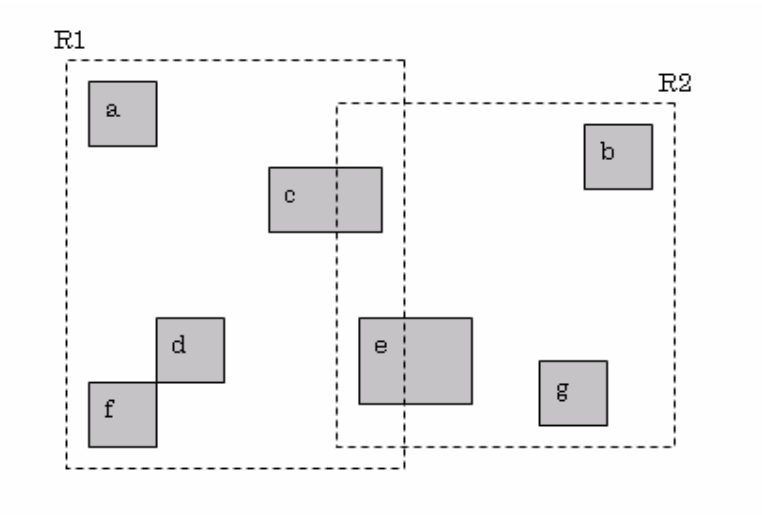
3.2.2 R-Tree

R-Tree の R とは Rectangle（長方形）から由来している。その名の通り、オブジェクトを MinimumBoundingRectangle（以下 MBR）と呼ばれる最小包括矩形によって管理するものである。これは、あるオブジェクトが入る最小の長方形のことである。また、３次元であるならば、長方形ではなく直方体となる。

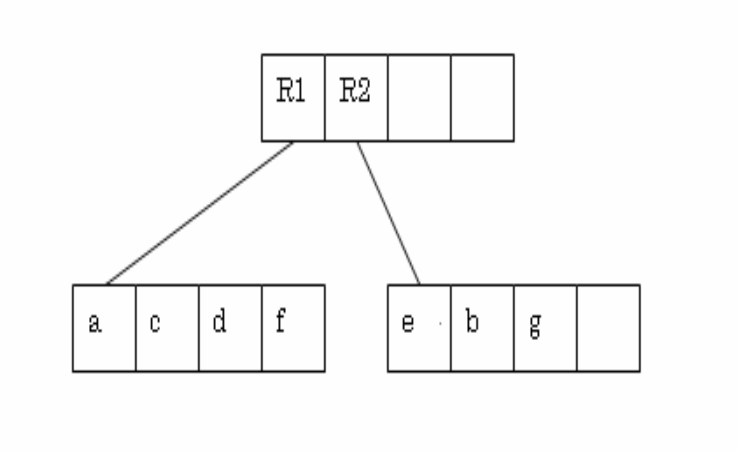


２次元の MBR　　３次元の MBR

R-Tree はツリー作成にあたって、すべてのオブジェクトを MBR に変換し根ノードから挿入していくものである。文字データなどの１次元データの索引付けで知られる B-Tree と同じように、すべてのオブジェクトは葉ノードに格納され、すべての葉ノードまでの深さは均一に保たれる。このことで、探索速度が均一になるのである。



(a)



(b)

図のように(a)のようなオブジェクトが与えられたとき、(b)のように R-Tree は生成される。１つのノード内のデータ数は「２×M」で M は自由に変えることができる。M をあまり大きくすると、ノード内の探索速度が遅くなるが、少なくするとツリーの深さが大きくなる。普通はディスクアクセス時間を減らすために、クラスタバイト数の倍数にするが、ここではあまり関係がないので M=２としてある。また、R-tree はツリー作成にあたって MBR を考えるが、MBR のオーバーラップを許している。精度については、Quad-Tree は深さを変えることで精度が上がった。一方、R-Tree は挿入という形をとっていることから、挿入するオブジェクトを細かくすることで精度が上がる。これは、挿入データが細かいほど MBR に変換したときの誤差が少なくなるからである。

3.3　索引付け方法の選択

ここまで、Quad-Tree と R-Tree の２つの木構造について簡単に説明してきた。そこで、本研究で扱う行政界と道路データについて考えてみる。

オブジェクト間の Overlap を検出するために、オブジェクトに対して１つの索引付けが必要となる。従って、Quad-Tree にはそれぞれのオブジェクトからツリーを生成すればよい。一方、R-tree の場合、１つの行政界にツリーを生成する場合、さらに細かいデータが必要となる。例えば島根県に索引付けする場合、松江市・平田市・・・というように島根県より小さい単位のデータが必要となる。本研究で取り扱うデータでは、市町村単位のデータを取り扱っており、これ以上の細かいデータを取り出すことができない。従って、R-tree による索引付けは本研究で取り扱うデータに対して不適切であった。以上のことをふまえ、Quad-tree のみに焦点を絞り、研究を進めることにした。

このように、索引付けには、取り扱うデータやクエリに対して、向き不向きがあることが示された。

第４章　Quad-tree

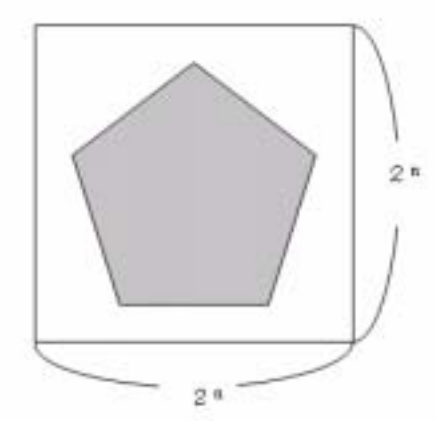
参考文献［１］

4.1　Quad-Tree 作成アルゴリズム

この節では、第３章で簡単に触れた Quad-Tree を作成するアルゴリズムについて詳しく述べる。

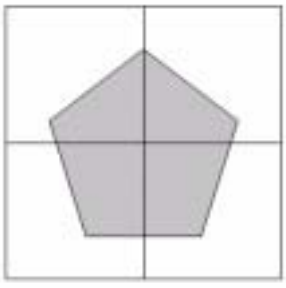
4.1.1　ルートノードの作成

Quad-Tree のルートノードは、与えられたエリアを包括する正方形である。Quad-Tree はルートノードをもとに４等分していくため、正方形の１辺の長さは２のべき乗で構成するとよい。これは、分割していく際、整数型ですべて処理をしていくためである。つまり、「与えられたエリアを包括し、かつ、１辺の長さが２のべき乗である正方形」を求めるのである。



4.1.2　エリアの分割

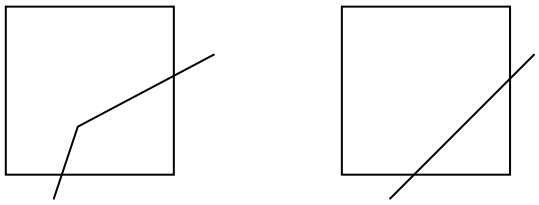
作成したルートノードを北西・北東・南西・南東に４等分する。



４等分割

分割した４つのエリアはそれぞれ、以下の場合に応じて処理を行う必要がある。

CASE１　データとのオーバーラップしている

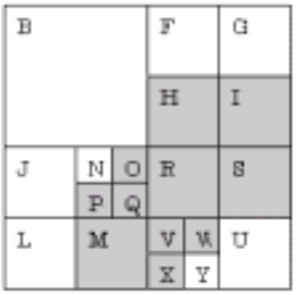


オーバーラップ

ここでいうデータとは、領域ならば境界線のことである。このオーバーラップの検出は第２章で述べた方法で調べる。つまり、領域を分割したエリア、データを折れ線として点位置決定問題を用いて判断する。このとき、オーバーラップしていた場合は子ノードにする。

CASE２　データとオーバーラップしていない

ここでの処理は、領域データとラインデータによって異なる。ラインデータであれば、子ノードにせず、そこで分割を終了すればよい。しかし、領域データの場合、その分解されたエリアが領域の内なのか外なのか識別する必要がある。内側であれば、子ノードにし、外側であれば子ノードにしない。両者ともに分割をここで終了する。この内外判定も点位置決定問題を利用する。



上図では、B・F・G・J・L はこの作業でエリア外となり、H・I・M・R・S はエリア内となり子ノードとなる。両者ともこれ以上分割されることはない。

4.1.3　再帰処理

以上の作業を再帰的に処理する。再帰を終了する条件とは、指定した深さに達することである。また、どのノードの大きさも「$2^n \times 2^n$」になっていることから、ルートノードの大きさからすべてノードの深さは求めることができる。

4.2 Quad-Tree の検索アルゴリズム

本研究では、オブジェクト間の Overlap を検出することである。この節では作成した Quad-Tree を用いてどのように Overlap を検出するのか説明する。

いま、比較する２つのオブジェクトａとｂに作られた Quad-Tree をそれぞれ A と B とする。

手順１　A と B のルートノードの Overlap を調べる。このとき、Overlap していなければオブジェクトａとｂは Overlap していないといえる。

手順２　A と B のルートノードが Overlap していた場合、それぞれ北西・北東・南西・南東どの位置で Overlap しているかを求める。求めたペアのノードに A と B それぞれ移動して、再び Overlap しているかを求める。この作業を両方が葉ノードになるまで繰り返す。

手順３　A と B 両方が葉ノードでかつ、ノードが Overlap していた場合、ａとｂは Overlap しているといえる。

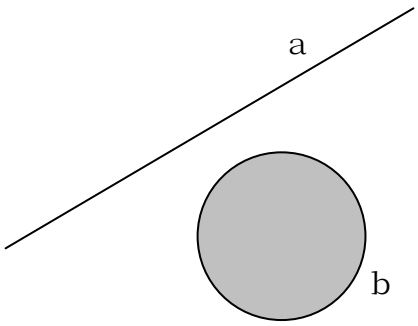
つまり、Quad-Tree の Overlap とは

「両方が葉ノードでかつ、ノードが Overlap している」

ペアが１つでも存在している場合のことである。

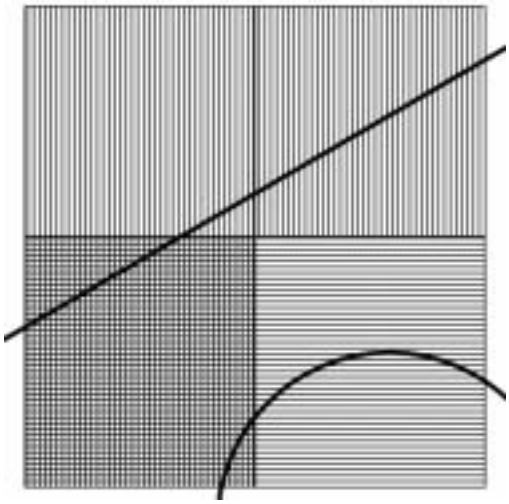
4.3 Quad-Tree の深さと精度

本節では、Quad-Tree の深さと精度の関係を、次の例を用いて説明する。深くするということは葉ノードの大きさを小さくするということである。また、本来はオブジェクトごとに Quad-Tree を生成するが、本節では、理解を容易にするため、同一のエリアでオブジェクトの境界付近に的を絞って考える。



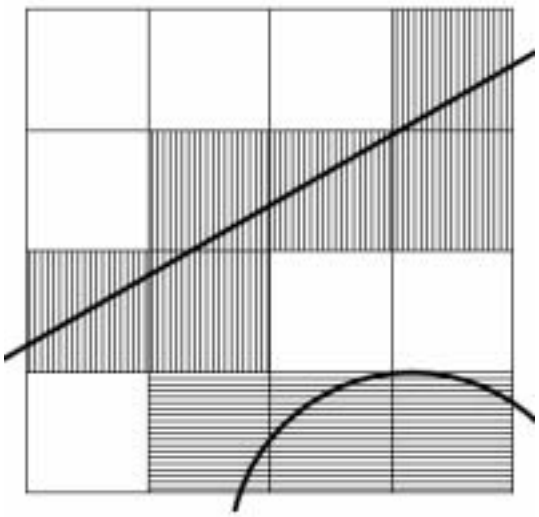
ａ：ライン型　　ｂ：エリア型

深さ：１

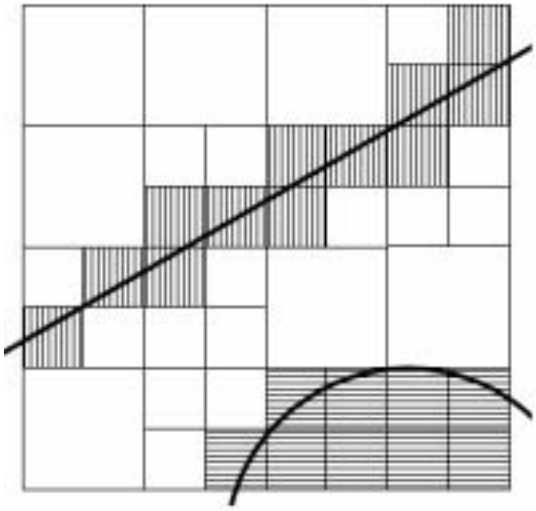


深さ：１では、左下のブロックで交わっている。従って、この状態で Overlap の検出を行うと、２つのオブジェクトは Overlap すると判断される。では、さらに分割を繰り返すとどのように変化するか考えてみる。

深さ：２



深さ：３



深さ：２では、交わっていないものの、接している。そして、深さ：３では、完全に離れている。従って、深さ３以上であれば、正しい Overlap の検出が行える。このように、ツリーの深さを深くする、すなわち葉ノードの大きさを小さくすると、探索の精度が上がることがわかる。

第５章　使用データの抽出（JMC マップ）

参考文献［4］［5］および JMC マップ付属 Readme.txt

5.1　JMC マップデータの概要

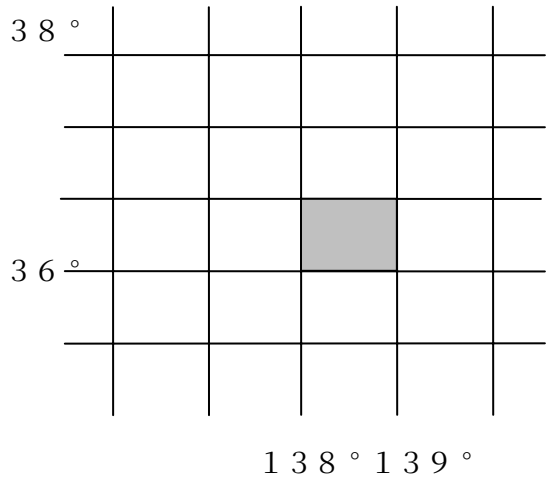
日本国土地理院の JMC マップは、20 万分１相当のベクトル形式の地図データである。データ項目は、行政界・海岸線、道路、鉄道、河川・湖沼、市区町村名等の記号・注記で、このうち、行政界・海岸線のデータは、構造化されたデータとなっており、各自治体（市区町村）をポリゴンとして認識することができる。また、データは、１次メッシュ（約 80㎞四方：20 万分１地勢図）単位に１つのファイルにまとめられている。

5.2　データフォーマット形式

5.2.1　１次メッシュと２次メッシュ

１次メッシュ単位のファイルになっており、さらに、ファイル内のデータは２次メッシュ単位の番号順に並んでいる。メッシュとは一定間隔の経緯線によって地域を分割するものである。

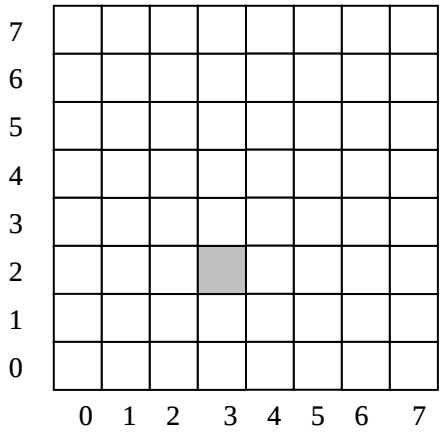
１次メッシュとは１度毎の経線と３分の２度ごとの緯線によって縦横に分割した区画のことである。１次メッシュコードは区画南端の緯度を１．５倍した２桁の数字と、西端軽度から１００を引いた数字を順に並べた４桁の数字からなる。



メッシュコードは
５　４　３　８　となる

36 × 1.5 = 54
138 － 100 = 38

２次メッシュは１次メッシュを縦横それぞれ８等分した区画で、経線方向については南から、緯線方向については西からそれぞれ０から７までの数値をつけ、経線方法、緯線方向の順に組み合わせた数字で定義される。



メッシュコードは

５　４　３　８－２　３

5.2.2　データレコード

２次メッシュ内のデータは、メッシュ・ヘッダー・レコードに続き、必要に応じてノードレコード、ラインレコード、エリアレコード、ポイントレコード、注記テキストレコードが続く。なお、ラインレコードには座標値レコード、エリアレコードにはエリア構成レコードが続く。ここでは、データ作成に必要なレコードのみ簡単に示す。

メッシュ・ヘッダー・レコード

項目	内容
レコードタイプ	M を記録
２次メッシュコード	２次メッシュコード
図名	当該 2 万 5 千分 1 地形図名
レイヤー総数	当２次メッシュに含まれるレイヤー総数
ノード総数	当２次メッシュに含まれるノード総数
ライン総数	当２次メッシュに含まれるライン総数
エリア総数	当２次メッシュに含まれるエリア総数
ポイント総数	当２次メッシュに含まれるポイント総数
レコード総数	当ヘッダーレコードを除いた当２次メッシュに含まれるノード総数

ラインレコード

項目	内容
レコードタイプ	L を記録
レイヤーコード	1:行政界・海岸線 2:道路 3:鉄道 5:河川･湖沼
データ項目コード	ライン項目コード表参照
ライン一連番号	レイヤー内で当ラインが何番目に位置するかを示す一連番号
ライン種別コード	ライン種別コード表参照
始点ノード番号	始点ノード番号
始点接続情報	0:図葉内ノード 1:隣接図葉に接続する図郭線上のノード 2:隣接図葉に接続しない図郭線上のノード
終点接続情報	始点接続情報と同じコード
左側 行政コード	ラインの向きに対して左側の行政コード
右側 行政コード	ラインの向きに対して右側の行政コード
座標点の数	当ラインを構成する XY 座標点の数 （始終点ノードを含む）

ライン項目コード

レイヤー	コード	データ項目
行政界・海岸線 1	1	都府県界
	2	北海道の支庁界
	3	郡市･特別区界
	4	町村･指定都市の区界
	5	海岸線
	9	図郭線
道路 2	1	高速道路及び自動車専用道
	2	一般国道
	3	主要地方道
	4	一般都道府県道

鉄道・河川･湖沼は除く

エリアレコード

項目	内容
レコードタイプ	A を記録
レイヤーコード	1:行政界・海岸線 2:道路 3:鉄道 5:河川・湖沼
データ項目コード	行政界レイヤーの場合は行政コードを記録
エリア一連番号	当レイヤー内で当エリアが何番目に位置するかを示す一連番号
代表点の座標値（X)	当エリア内代表点の X 座標値
代表点の座標値（Y)	当エリア内代表点の Y 座標値
ライン数	当エリアを構成するライン数(図郭線及びエリア内アイランドの外周を含む)にアイランドの数を加えた数

5.2.3　２次メッシュデータの原点と軸

２次メッシュ内の各座標値は、左下を(0,0)、右上を(10000,10000)とする正規化座標で記録され、ｘ座標は右（東）方向、ｙ座標は上（北）方向となる。

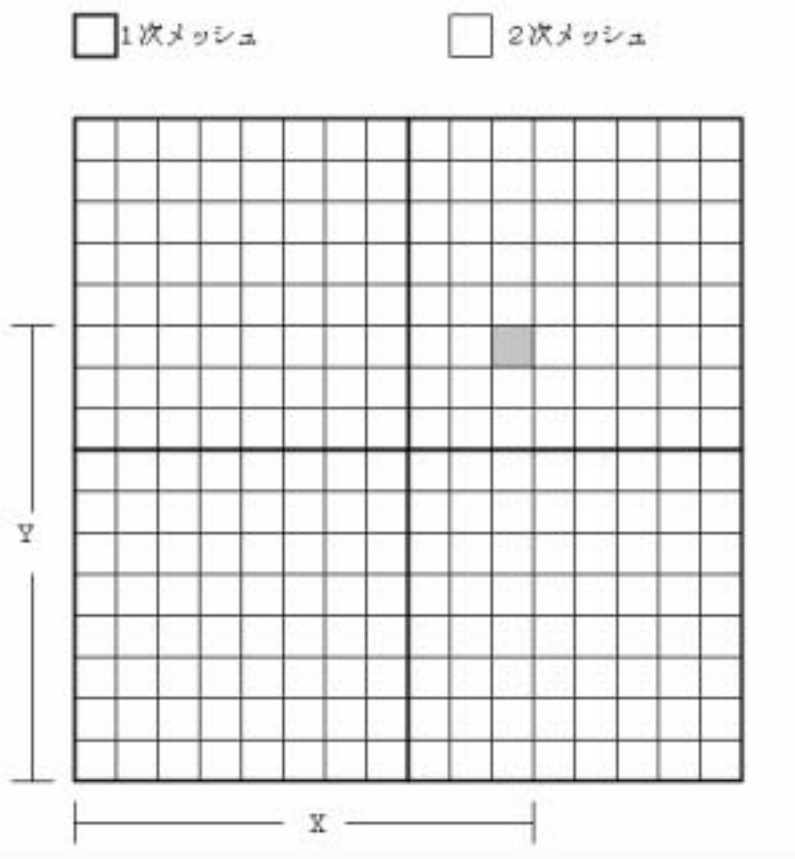
5.3　使用データ抽出アルゴリズム

5.3.1　使用データの概要

本研究で使用するデータは行政界データと道路データである。ただし、行政界データにおいては、細かな島などは除くものとする。また、それぞれのデータは２次元データであり、ユークリッド平面に座標として表現できるものである。

5.3.2　データの座標変換

5.2.3 節で述べたが、２次メッシュ内の各座標値は、左下を(0,0)、右上を(10000,10000)とする正規化座標で記録され、ｘ座標は右（東）方向、ｙ座標は上（北）方向となる。従って、基準となる原点を１つに統一する必要がある。このため、２次メッシュコードから次のような変換を行った。



新ｘ座標　＝　旧ｘ座標　＋　（Ｘ　×　１００００）
新ｙ座標　＝　旧ｙ座標　＋　（Ｙ　×　１００００）

Ｘ・Ｙの計算は、使用ファイル名もしくは、メッシュ・ヘッダー・レコードから知ることができる。

5.3.3　行政界データの抽出

行政界データはエリアデータである。従って、まずエリアレコードを参照する。エリアレコードでは、行政コードとその行政を構成するライン番号が得られる。また、ライン構成番号は行政界を常に右に見ながらの構成になっている。そのライン番号をもとに、行政界のデータを取り出す。

手順１　エリアレコードを参照し、行政コードから求める行政エリアを認識する。
手順２　エリアレコードから、エリア構成ライン番号を摘出する。この番号は行政界を常に右に見ながらの構成になっている。また、マイナスがついている場合は逆順という意味である。
手順３　ラインレコードを参照し、摘出したライン番号のデータをとりだす。このとき、マイナスの有無に注意する。
手順４　すべてのデータをとりだしたら、データを連結させる。エリアデータであるので、かならず始点と終点が同じになる。

これら４手順でデータを取りだせるが、この時点では、小島など複数のエリアデータが混在している。データ数を比較し、最もデータ量の多いもののみを取り出すことで、小島などのデータを除去する。

5.3.4　道路データの抽出

道路データは複数のラインデータである。JMC マップで認識可能な道路は、高速道路及び自動車専用道・一般国道・主要地方道・一般都道府県道の４種類である。これ以上のデータは認識できない。

手順１　ラインレコードを参照し、レイヤーコードから道路データを認識する。
手順２　ライン項目コードから求めたい４種類のデータを認識する。
手順３　ラインデータを取り出す。

これら３手順で道路データを取り出せる。道路は複数のラインデータのため、連結する必要はあまりない。本研究では国道９号線と国道４３１号線のデータを使用している。これは、JMC マップでは認識できないため、手作業によって取り出した。

第６章　結果および考察

6.1　索引付けの検証方法

ここまで、幾何学的計算と Quad-Tree の２つの方法で Overlap の検出方法を説明してきた。本章では、第２章・第４章で述べた方法をもとに、隠岐と松江市周辺の２種類のデータを使用し、索引付け技術の検証を行ってみる。空間検索の検証に伴い、「前処理にかかる時間」・「検索にかかる時間」・「記憶領域」の３項目を比較の指標にする。これら３項目の詳細は以下のとおりである

前処理にかかる時間
　ツリー作成時間のように、検索のためのデータ構造を構築する時間

検索にかかる時間
　１つのクエリに対して、応答するまでの時間

記憶領域
　データや、データ構造に必要とされる記憶領域

6.2　データの詳細および結果

本研究で用いたデータは隠岐と松江市周辺の２つであるが、詳細は以下のとおりである。

形状	地名	データ数	合計
多角形	西郷町	２５９９	１０９６１
	布施村	６１１	
	五箇村	９７１	
	都万村	１２０６	
	海士町	２０９０	
	西ノ島町	２５７８	
	知夫村	９０６	
折れ線	一般国道	１４７	６７１
	主要地方道	２５７	
	一般都道府県道	２６７	
多角形	松江市	１４５９	１０６５８
	安来市	１１４１	
	平田市	１３３６	
	鹿島町	１０３６	
	島根町	１３１７	
	美保関町	２１００	
	東出雲町	５７４	
	八雲村	７１９	
	玉湯町	４２０	
	宍道町	５５６	
折れ線	国道９号線	１４１	３１６
	国道４３１号線	１７５	

表からわかるように、松江市周辺の行政界の数が隠岐に比べ増えているが、隠岐と松江市周辺のデータ数はほぼ同じである。また、データの全体表示を付録に掲載する。

上記のデータをもとに、幾何学的計算と Quad-Tree を用いてそれぞれ Overlap を検出した。以下の表は探索回数１０回の平均結果である。また、表中の記号等の意味は以下のとおりである。

○：Overlap している　　×：Overlap していない

0 msec　：　測定不能

隠　岐

幾何学的探索による Overlap

地　名	一般国道		主要地方道		一般都道府県道	
		探索時間		探索時間		探索時間
西郷町	○	250 msec	○	331 msec	○	1078 msec
布施村	○	163 msec	○	254 msec	×	321 msec
五箇村	○	100 msec	○	134 msec	○	2 msec
都万村	×	362 msec	○	2 msec	○	463 msec
海士町	×	647 msec	×	1155 msec	○	196 msec
西ノ島町	○	4 msec	×	1424 msec	○	28 msec
知夫村	×	284 msec	×	499 msec	○	2 msec
合計時間	1810 msec		3799 msec		2090 msec	

Quad-Tree 作成時間

地方	葉ノードサイズ：１０２４	葉ノードサイズ：２５６
西郷町	138 msec	719 msec
布施村	18 msec	60 msec
五箇村	50 msec	158 msec
都万村	58 msec	206 msec
海士町	52 msec	239 msec
西ノ島町	135 msec	533 msec
知夫村	24 msec	86 msec
一般国道	6 msec	28 msec
主要地方道	14 msec	48 msec
一般都道府県道	14 msec	54 msec
合計時間	509 msec	2131 msec

Quad-Tree による一般国道の Overlap

地名	葉ノードサイズ：１０２４		葉ノードサイズ：２５６	
		探索時間		探索時間
西郷町	○	0 msec	○	0 msec
布施村	○	0 msec	○	0 msec
五箇村	○	0 msec	○	0 msec
都万村	○	0 msec	×	0 msec
海士町	×	0 msec	×	0 msec
西ノ島町	○	0 msec	○	0 msec
知夫村	○	0 msec	×	0 msec
合計時間	0 msec		0 msec	

Quad-Tree による主要地方道の Overlap

地名	葉ノードサイズ：１０２４		葉ノードサイズ：２５６	
		探索時間		探索時間
西郷町	○	0 msec	○	0 msec
布施村	○	0 msec	○	0 msec
五箇村	○	0 msec	○	0 msec
都万村	○	0 msec	○	0 msec
海士町	×	0 msec	×	0 msec
西ノ島町	×	0 msec	×	0 msec
知夫村	×	0 msec	×	0 msec
合計時間	0 msec		0 msec	

Quad-Tree による一般都道府県道の Overlap

地名	葉ノードサイズ：１０２４		葉ノードサイズ：２５６	
		探索時間		探索時間
西郷町	○	0 msec	○	0 msec
布施村	○	0 msec	×	0 msec
五箇村	○	0 msec	○	0 msec
都万村	○	0 msec	○	0 msec
海士町	○	0 msec	○	0 msec
西ノ島町	○	0 msec	○	0 msec
知夫村	○	0 msec	○	0 msec
合計時間	0 msec		0 msec	

松江市周辺

幾何学的探索による Overlap

地名	国道９号線		国道４３１号線	
		探索時間		探索時間
松江市	○	220 msec	○	155 msec
安来市	○	180 msec	×	364 msec
平田市	×	330 msec	○	52 msec
鹿島町	×	259 msec	×	337 msec
島根町	×	320 msec	○	382 msec
美保関町	×	513 msec	○	485 msec
東出雲町	○	104 msec	×	186 msec
八雲村	×	180 msec	×	237 msec
玉湯町	○	56 msec	×	138 msec
宍道町	○	58 msec	×	184 msec
合計時間	2220 msec		2520 msec	

Quad-Tree 作成時間

地名	葉ノードサイズ：１０２４	葉ノードサイズ：２５６
松江市	158 msec	634 msec
安来市	68 msec	240 msec
平田市	134 msec	463 msec
鹿島町	58 msec	198 msec
島根町	38 msec	150 msec
美保関町	207 msec	611 msec
東出雲町	12 msec	74 msec
八雲村	20 msec	110 msec
玉湯町	12 msec	48 msec
宍道町	30 msec	97 msec
国道９号線	16 msec	52 msec
国道４３１号線	20 msec	60 msec
合計時間	773 msec	2737 msec

Quad-Tree による国道９号線の Overlap

地名	葉ノードサイズ：１０２４		葉ノードサイズ２５６	
		探索時間		探索時間
松江市	○	0 msec	○	0 msec
安来市	○	0 msec	○	0 msec
平田市	○	0 msec	○	0 msec
鹿島町	○	0 msec	×	0 msec
島根町	○	0 msec	×	0 msec
美保関町	○	0 msec	×	0 msec
東出雲町	○	0 msec	○	0 msec
八雲村	○	0 msec	○	0 msec
玉湯町	○	0 msec	○	0 msec
宍道町	○	0 msec	○	0 msec
合計時間	0 msec		0 msec	

Quad-Tree による国道４３１号線の Overlap

地名	葉ノードサイズ：１０２４		葉ノードサイズ２５６	
		探索時間		探索時間
松江市	○	0 msec	○	0 msec
安来市	○	0 msec	×	0 msec
平田市	○	0 msec	○	0 msec
鹿島町	○	0 msec	○	0 msec
島根町	○	0 msec	○	0 msec
美保関町	○	0 msec	○	0 msec
東出雲町	○	0 msec	×	0 msec
八雲村	○	0 msec	×	0 msec
玉湯町	○	0 msec	○	0 msec
宍道町	○	0 msec	○	0 msec
合計時間	0 msec		0 msec	

表１

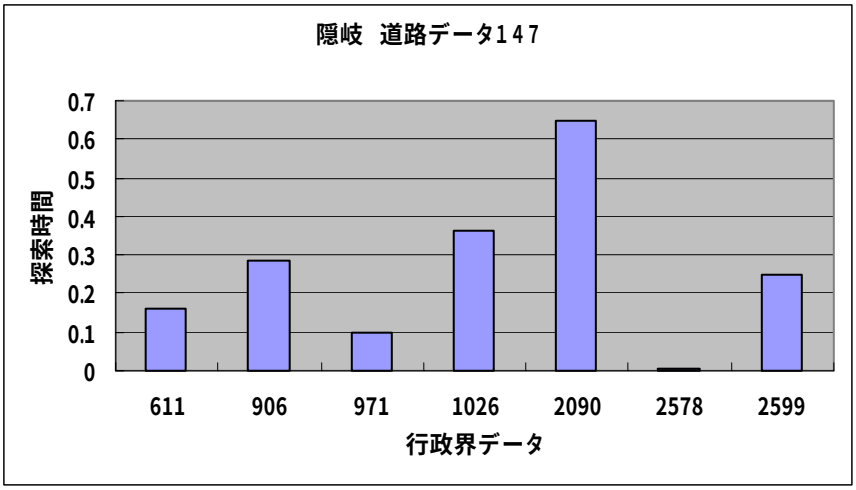

隠岐　道路データ１４７
探索時間
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
611
906
971
1026
2090
2578
2599
行政界データ

表２

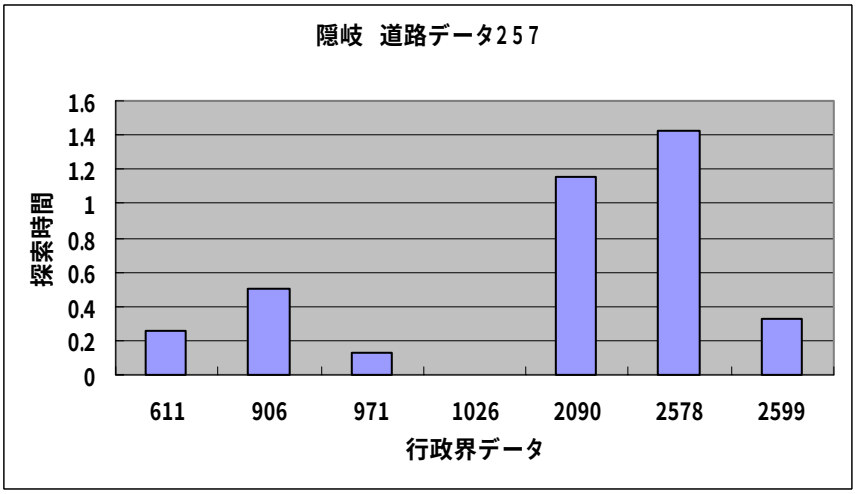

隠岐　道路データ２５７
探索時間
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
611
906
971
1026
2090
2578
2599
行政界データ

表３

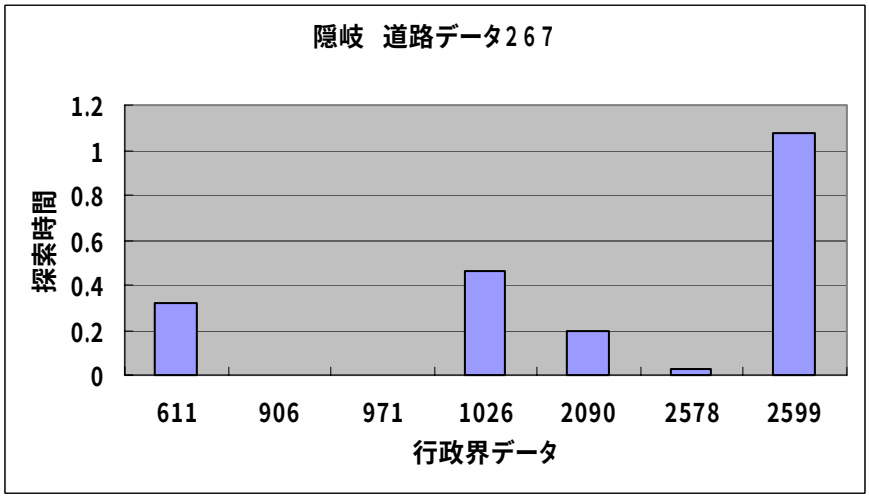

隠岐　道路データ２６７
探索時間
1.2
1
0.8
0.6
0.4
0.2
0
611
906
971
1026
2090
2578
2599
行政界データ

表４

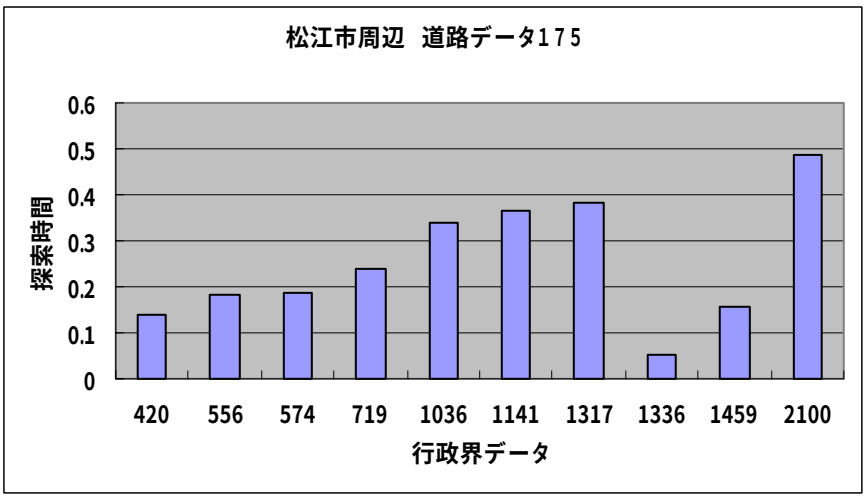

松江市周辺　道路データ175
探索時間
0.6
0.5
0.4
0.3
0.2
0.1
0
420
556
574
719
1036
1141
1317
1336
1459
2100
行政界データ

表５

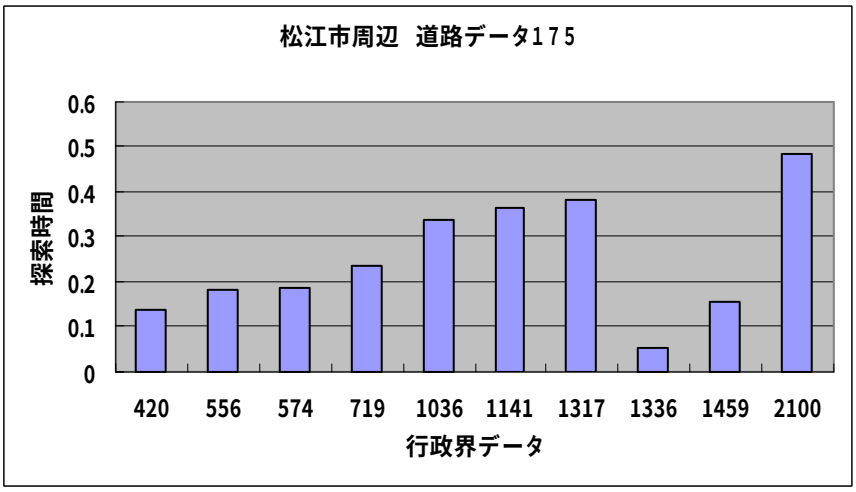
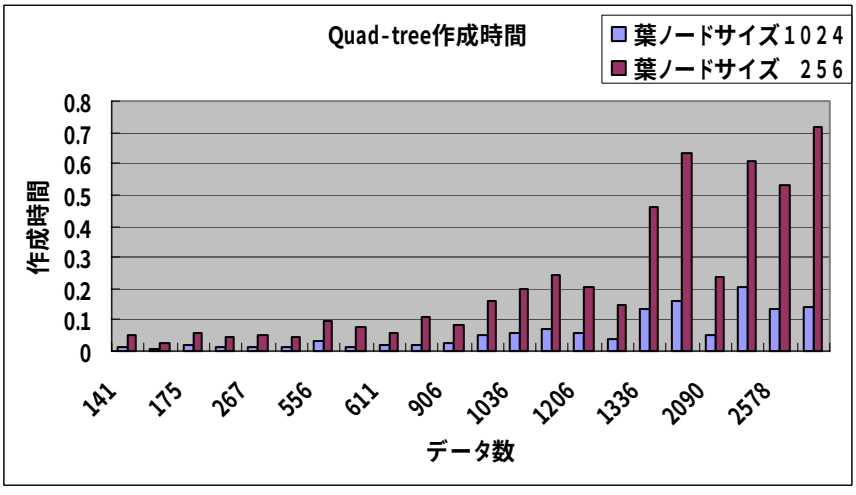

松江市周辺　道路データ175
探索時間
0.6
0.5
0.4
0.3
0.2
0.1
0
420
556
574
719
1036
1141
1317
1336
1459
2100
行政界データ

表６


Quad-tree作成時間
葉ノードサイズ1024
葉ノードサイズ　256
作成時間
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
141
175
267
556
611
906
1036
1206
1336
2090
2578
データ数

6.4　幾何学的計算の検証

まず幾何学的計算を用いた結果を見ていただきたい。隠岐と松江市周辺のデータ数はほとんど同じである。従って、隠岐の主要地方道で時間がかかっているものの、ほとんど同じ時間で結果を返している。点位置決定問題は、前処理なしで O(n)の計算量で求めることが出来る。従って、幾何学的計算による Overlap の検出は最悪の場合次のようになる。

行政データ：N　　道路データ：M　とすると　O(NM)

このことから、データの増加にともなって、Overlap の検出にかかる計算量はデータ数に比例して大きくなっていくことが推測される。実際の結果では、隠岐、松江市周辺それぞれ表１～３、表４～５のようになっている。また、記憶領域については、データのみ格納できればよい。従って、次のように考えることができる。

行政データ：N　　道路データ：M　とすると　記憶領域（N+M）

以上のことから、幾何学的計算による Overlap の検出は、大量のデータに対する問い合わせには向かないが、前処理は無く、必要メモリも少なくてすむと考えられる。

データ数の変化

前処理にかかる時間	：	データ数に関係なく前処理の必要はない
検索にかかる時間	：	データ数の増加に比例する
記憶領域	：	データを格納する領域のみ必要
探索結果	：	常に厳密解が得られる

6.5　Quad-Tree の検証

次に Quad-tree を用いた結果を見ていただきたい。どのクエリにも高速に結果を返していることが分かる。検索にかかった時間は幾何学的計算によるものよりも明らかに少ない。索引付けは、一度行ってしまえば、複数のクエリに答えることが出来る。そして、次のような結果が得られた。

幾何学的計算による検索時間の合計　≧ 索引付けにかかった時間　＋　索引付けを用いた検索時間の合計

「前処理かかる時間」とは Quad-Tree の作成にかかる時間のことである。ここで、深さとツリー作成計算量の関係を考えてみる。

深さ：ｎ　とすると　Ｏ（４^{0}＋４^{1}＋‥‥４^{n}）　｛Ｏ（４^{n})｝

これは、Quad-Tree は葉までの深さは均一ではないので、最悪の場合の計算量である。このことから、前処理にかかる時間は Quad-Tree の深さとともに、膨大に増えることが推測される。実際の結果では表６のようになっている。

そこで、深さの変化に伴う、検出結果を見てみる。葉ノードの大きさが１０２４の場合、幾何学計算による結果比較して分かるように、明らかに結果が異なる。一方、葉ノードの大きさを２５６にした場合、隠岐データについては幾何学的計算と同じ結果が得られたが、松江市周辺のデータでは結果が異なる。これは、4.3 節で説明した深さと制度の関係によるものである。オブジェクト同士が密接していた場合、深さを深くしなければ、正しい結果が得られないのである。このように、葉ノードの大きさを小さくすることで、正確な結果に近づくことはわかる。しかし、葉ノードをどの大きさにすれば正確な結果が得られるというわけではない。また、深さとツリー作成の計算量の関係から分かるように、葉ノードを小さくするほどツリーは深くなり、前処理にかかる時間が増加する。また、記憶領域についてもツリーの大きさに応じて大きくなる。つまり、ツリーを構成するノード数と等しいだけ記憶領域を必要とする。従って、最悪の場合の記憶領域はノード数から以下のように考えられる。

深さ：ｎ　とすると　記憶領域（４^{0}＋４^{1}＋‥‥４^{n}）

以上のことから、Quad-Tree による Overlap の検出は、ツリーを作成する時間とデータ構造に必要な記憶領域は増えるが、大量データに対する問い合わせにも高速に近似解を求めることが出来るといえる。

データ数の変化

前処理にかかる時間	：	データ数の増加とともに増加
検索にかかる時間	：	データ数にあまり関係なく高速
記憶領域	：	ツリーを構成するノード数必要
探索結果	：	近似解が得られる

深さの変化

前処理にかかる時間	：	深さの増加に伴い、膨大に増加
検索にかかる時間	：	深さにあまり関係なく高速
記憶領域	：	深さの増加に伴い、ツリーを構成するノード数も増加
探索結果	：	深さに応じて、厳密解に近づく

6.6 索引付けのあり方

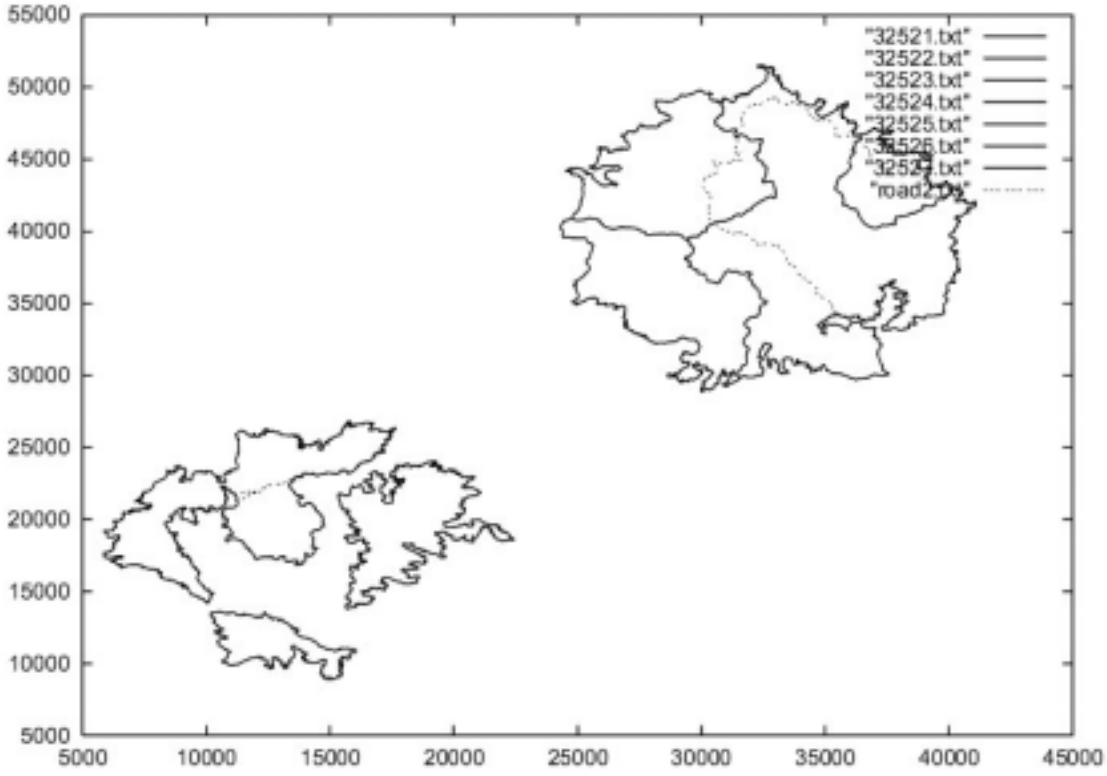
これまで、幾何学的計算と Quad-Tree について Overlap の検出をテーマに検証を行ってきた。幾何学的計算では正確な解を得ることができるが、データの増加に伴い、検索時間も増加していく。一方、Quad-Tree は大量のデータにも高速に近似解を求めることができた。このことから、Quad-Tree は検索範囲を高速に絞るという面ではかなりの効果が得られる。もちろん、近似的な解を求めるのであれば、問題はない。幾何学的計算についても、データが少なければ、短時間で正確な解を得ることができる。このようなことから、これら２つの技術を併用するなど、索引付け技術はかなりの利用価値があるといえる。近年、マルチメディアデータを取り扱うことが増える中、高速にデータを検索する技術として、索引付け技術は十分な有用性を発揮すると考えられる。

付録

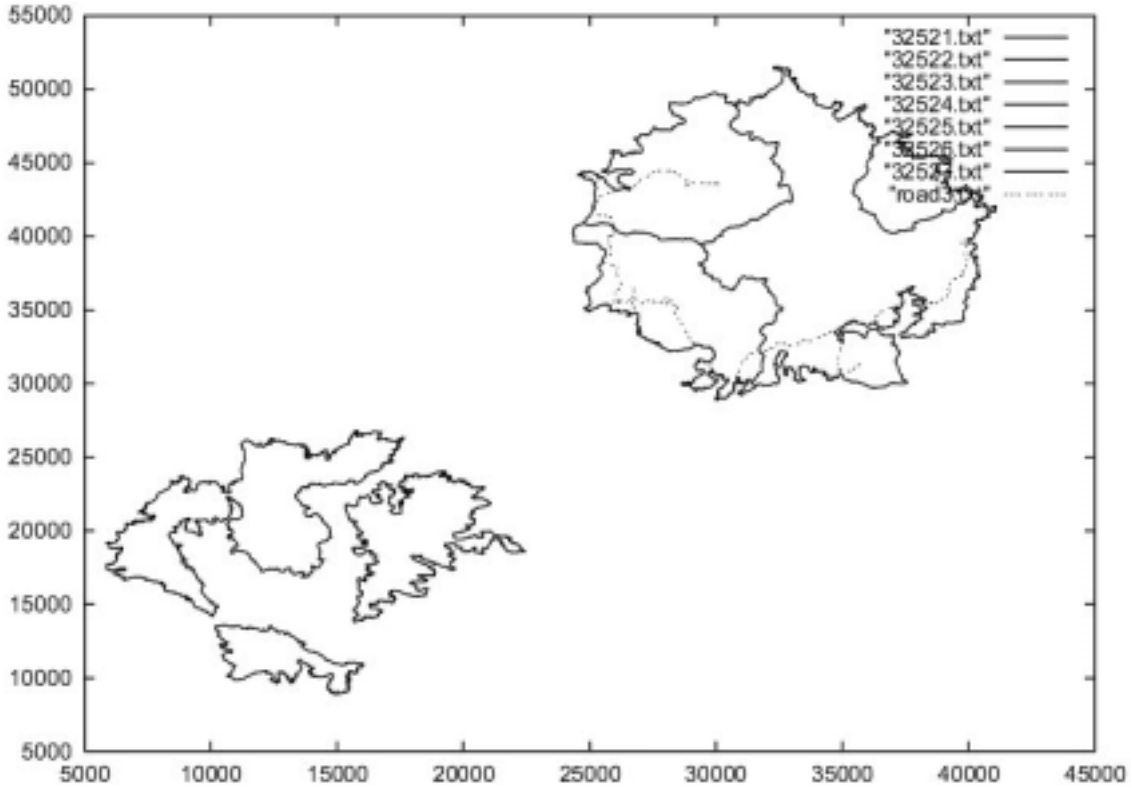
・データの画像表示

この表示は GNUPLOT によるものである。

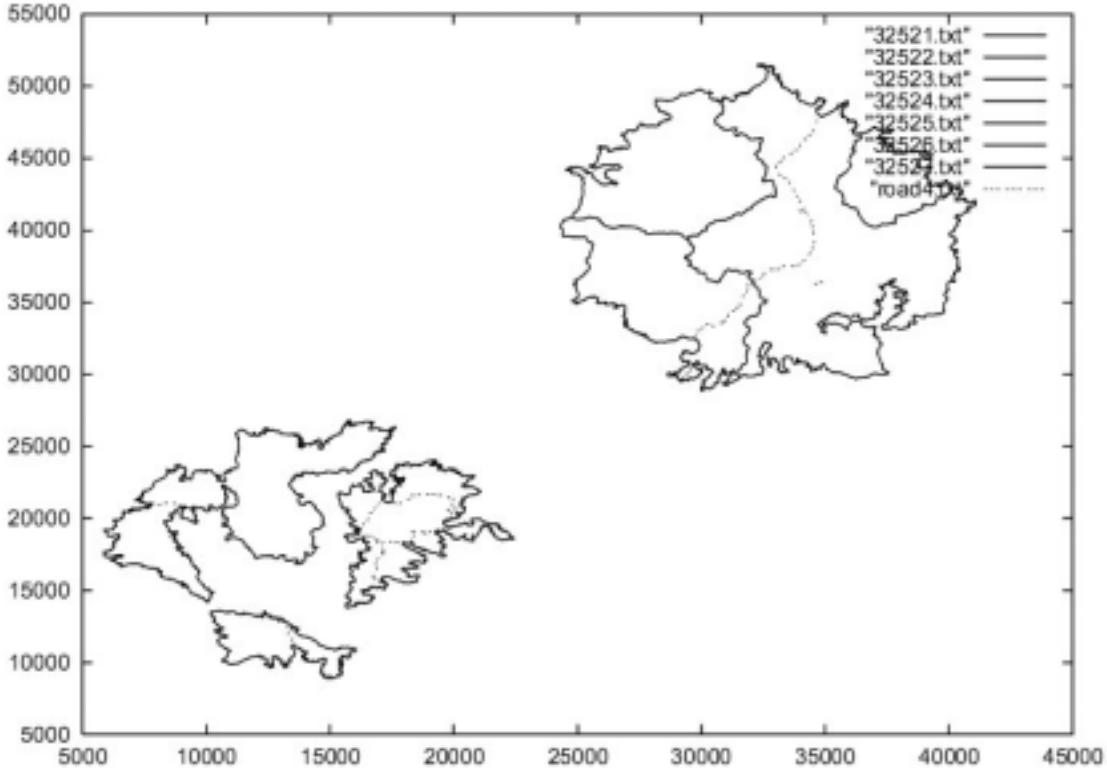
隠岐と一般国道



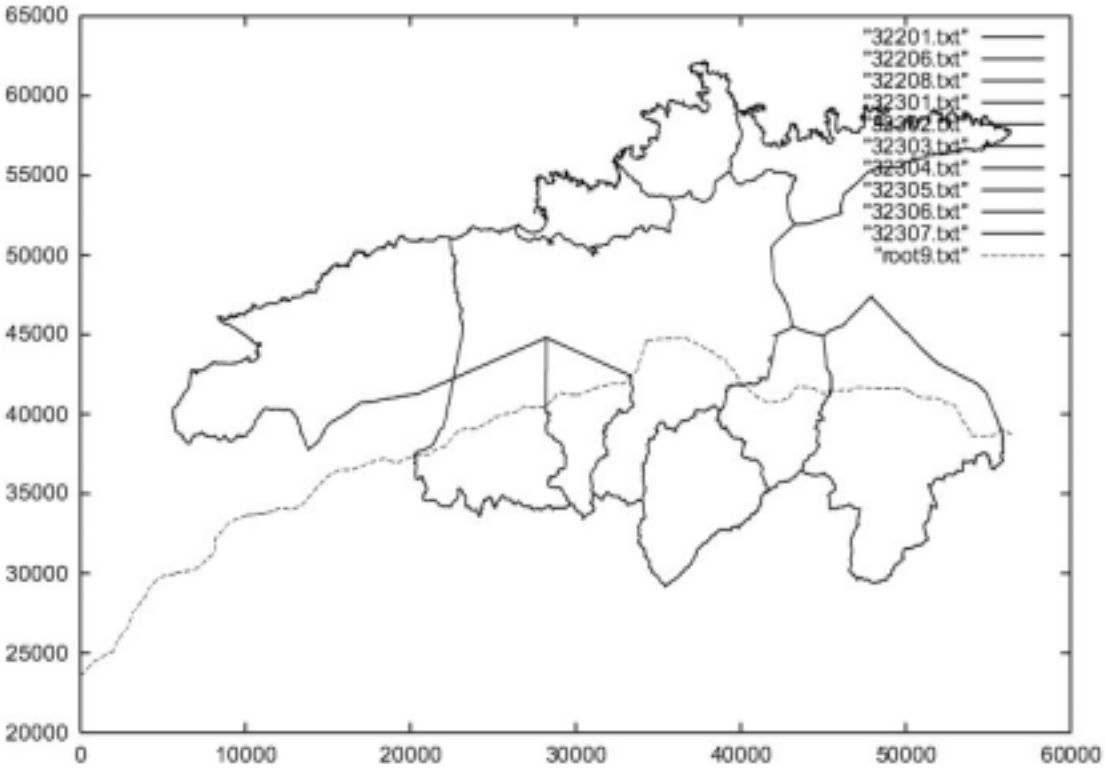
隠岐と主要地方道



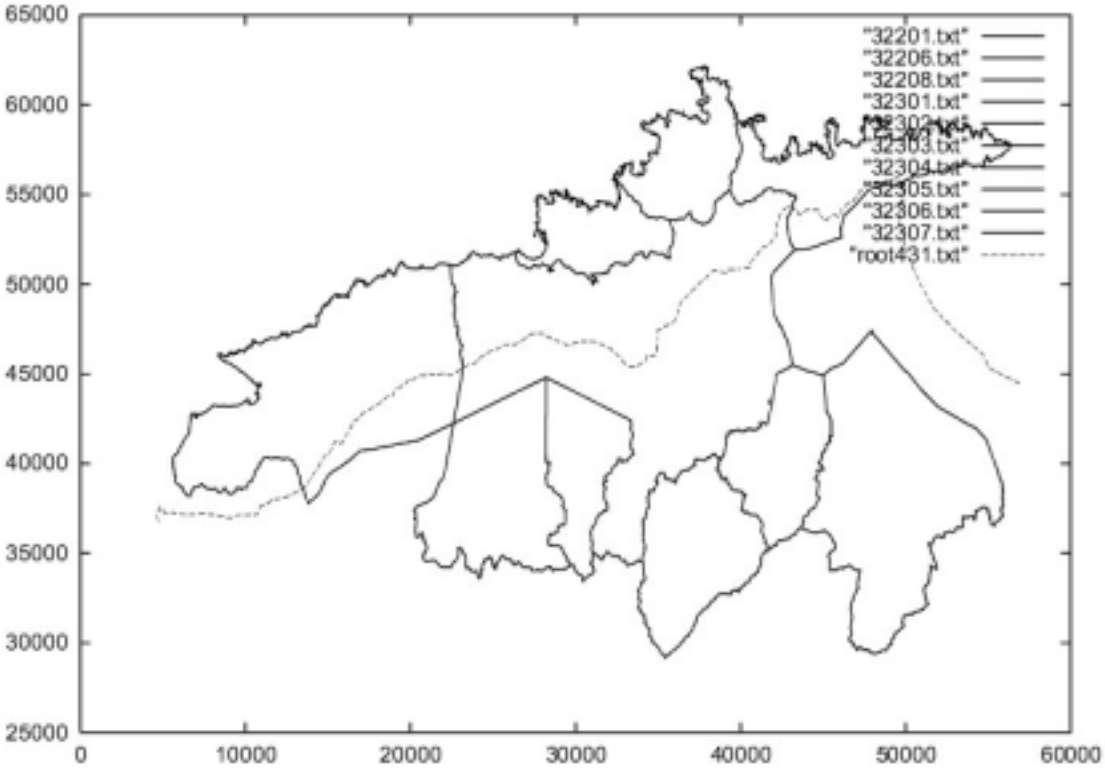
隠岐と一般都道府県道



松江市周辺と国道９号線



松江市周辺と国道４３１号線



参考文献

［1］HANAN SAMET　著
The Quadtree and Related Hierarchical Data Structures

［2］Elisa Bertino・Beng Chin OOI・Ron Sacks-Davis・Kian-Lee Tan
Justin Zobel・Boris Shidlovsky・Barbara catania　著
INDEXING TECHNIQUES FOR ADVANCED DATABASE SYSTEMS

［3］プレパラータ / シェーモス　著　　浅野孝夫 / 浅野哲夫　訳
計算幾何学入門　総研出版

［4］建設省国土地理院　監修　　（財）日本地図センター　編集・発行
数値地図ユーザーズガイド

［5］田中研究室　２０００年度卒業　金納陽晴　卒業論文

謝辞

本研究にあたり、最後まで熱心な御指導をいただきました田中教授には、心より御礼申し上げます。また、田中研究室の貫目さんには、本研究に関して数々の御協力と御助言をいただきました。厚く御礼申し上げます。なお、本論文、本研究で作成したプログラム及びデータ、並びに関連する発表資料等のすべての知的財産権を、本研究の指導教官である田中教授に譲渡致します。