部品のあいまいな位置に関する検索

赤石美奈[†] 田中 譲[††]

あいまいな位置に関する情報をもとに，必要なものを探し出す手法について述べる．著者らは，単純な機能を持つ部品（メディア）を合成し，それらの機能連携により複雑な機能を実現することを可能とするシステムをシンセティック・メディア・システムと呼ぶ．ここで扱うメディアを統合管理するためのデータベースでは，ユーザが必要とするものの自由な検索を支援する機能が必要である．本論では，部品の種類と位置に関する検索について述べる．種々雑多なものから，定型な書類や右下に挿絵が入っているページ等，デザインの一部が印象に残っているものを探す場合には，その位置を指定し検索を行うことが有効である．しかし，人間の位置に関する記憶はあいまいであるため，厳密な位置を指定することはできない．そこで，本論では，部品の位置に関する情報を抽出し，あいまい性を含んだコード化によりシグニチャを生成し，それらを集めたシグニチャ・ファイルを用いて検索を行う方法を提案している．また，検索時に，ユーザが，あいまい領域を拡張指定することも可能である．さらに，シグニチャ・ファイルの検索においては，本論で用いるシグニチャ・ファイルの特性に基づき，検索の高速化を図る方法について述べる．

Search for Every Composite Object that Satisfies Ambiguously Specified Relative Locations of Its Components

MINA AKAISHI [†] and YUZURU TANAKA [††]

This paper presents a search method for media objects from the ambiguous information about their components' positions. The synthetic media system provides its users with a toolkit for the construction of various interactive media objects including multimedia documents, desktop tools, and application systems. The database for this system needs to manage all kinds of media objects. It needs to support its user to retrieve what he wants out of the large varieties of composite media objects. If users remember the part of an application's layout design, such as 'a form having a picture at its upper right position', it is efficient to use such information about components positions as a query specification. However, it might be impossible to point out component positions exactly. Human's memory about spatial information is ambiguous. The method presented in this paper is based upon the use of signatures that ambiguously encode component positions and sizes. Users can retrieve the media objects that they want by ambiguously and partialy specifying their layouts. This paper also gives an efficient signature-file access method for the processing of such ambiguous query signatures.

1. はじめに

情報探索においては，ブラウジング，ナビゲーション，条件検索といった手法が提案されている．著者らは，何を検索条件として用いるかにより条件検索方法を二種類に分類する．一つは内容指定検索（コンテント・ベースト検索）であり，もう一つは文脈指定検索（コンテクスト・ベースト検索）である．現在，提供されているデータベースの条件検索においては，必要な情報の一部（いくつかの属性値等）を指定し，条件として利用する．例えば，文献データベースにおいては，著者名やタイトルなどを条件に検索を行う．著者らは，これを，内容指定検索と呼ぶ．これに対して，文脈指定検索は，必要な情報がどのような状況にあるか，どのような場面で利用されるかというコンテクスト情報を検索条件として情報を探す方法である．例えば，必要な文献が，「本棚の上から3段目の棚の右のほうにあった」などという情報を条件に検索を行う．種々雑多な情報を統合的に扱うシステムにおけるデータベースは，このような内容指定検索と文脈指定検索の機能を備え，それらを相補的に利用できる機構をユーザに提供する必要があると考える．

文献1)では，機能部品の合成により，さらに複雑な

† 静岡大学工学部知能情報工学科
Department of Computer Science, Faculty of Engineering, Shizuoka University
†† 北海道大学工学部電子情報工学専攻
Department of Electronics and Information Engineering, Faculty of Engineering, Hokkaido University

機能を持つ部品を実現するシステムにおいて，必要な機能の検索に，部品の種類とその結合関係に着目した文脈指定検索を取り入れている．

本論においては，各部品の位置に着目し，これを条件とする文脈指定検索とその高速化の手法について述べる．

2章では，機能部品の組合せにより，さまざまなアプリケーションを構築するシンセティック・メディア・システム IntelligentPad[2)~4)] について簡単に述べ，3章において，部品のあいまいな位置による検索について述べる．4章では，検索の効率化について述べ，5章においてまとめを述べる．

2. IntelligentPad システムの概要

IntelligentPad システムでは，文書，図表，静止画，動画，音声等のコンピュータ上の様々なメディアに加えて，アプリケーション・プログラムや，システムが提供する各種サービス・システムをも，すべて紙（パッド）として表現し，これらをダイナミック・メディア・オブジェクトとして統一的に扱う（**図1**参照）．それぞれに対応して各種のパッドが定義される．システムにより提供される基本的な部品をプリミティブ・パッドと呼ぶ．パッドは，『貼る』というメタファにより機能の合成ができる．合成されたパッドは，これを構成するプリミティブ・パッドの機能連携により，複雑な機能を持つ一枚のパッドとして定義される．様々なパッドを自由に組み合せることができ，すべてのパッドは，新たなパッドを構築するための部品として再利用できる．新たなアプリケーションの開発は，すべてを新たに開発するのではなく，それまでのシステムに欠けていたプリミティブな機能のみをパッドとして実現し，これと既存のパッドを合成することにより遂行できる．

なお，本論文においては，プリミティブ・パッドと合成パッドを区別する必要のない場合には，単に「パッド」と記述し，区別する場合には，それぞれ，「プリミティブ・パッド」，「合成パッド」と記述する．

3. パッドのあいまいな位置検索

あるアプリケーションを検索する際に，そのレイアウト・デザイン（の一部）が強く記憶に残っている場合には，部品の配置を条件に用いることが有効である．例えば，たくさんの書類の中から『右上に写真の貼ってある書類』や，辞書などの『左下に挿絵の入っているページ』等を探す場合には，写真や絵等の位置を検索条件として指定すると良い．図1のようなパッド群において，写真が右上にあることを条件にすると，これを満たす合成パッドを選択できる．また**図2**は，本パッドであるが，表紙や，各ページなど，すべてパッドの貼り合わせで実現されている．ページの挿絵の部分には絵を表示する機能をもつプリミティブ・パッドが貼られている．このような場合，絵の位置を検索条

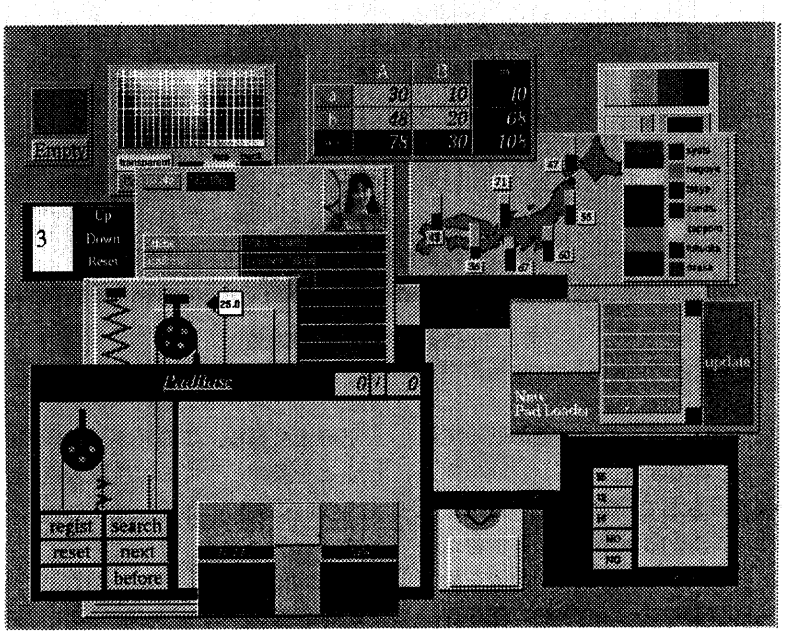


図1 IntelligentPad システムの画面ハードコピー
Fig. 1 A display hardcopy of the IntelligentPad system.

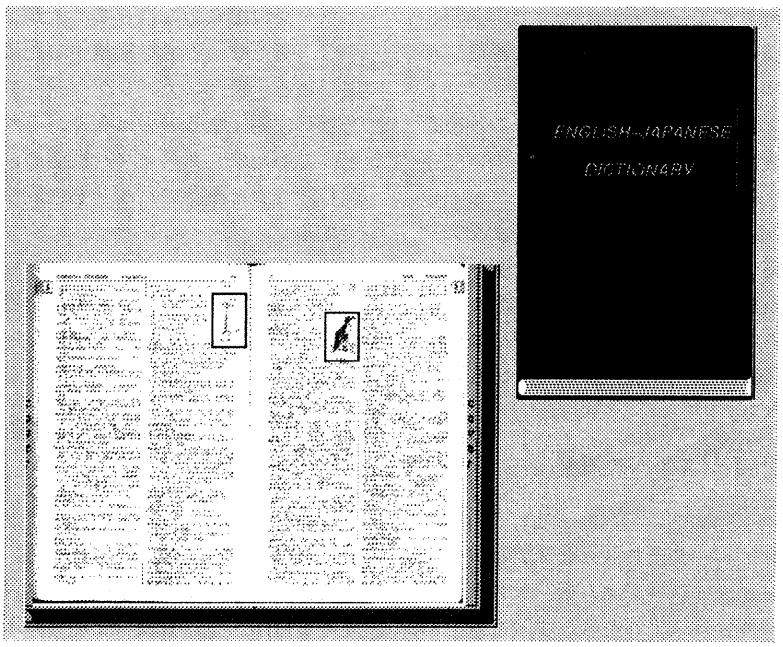


図2 本パッド
Fig. 2 A pad representing a book.

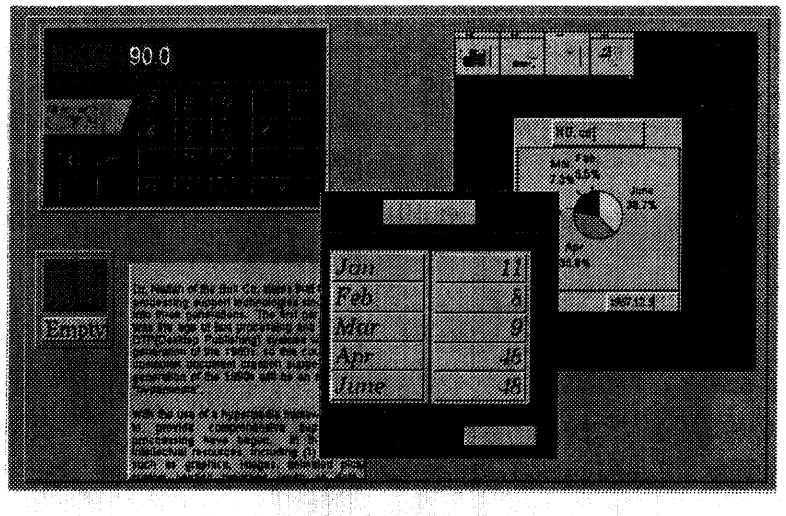


図3 IntelligentPad の作業環境例
Fig. 3 An example work space in the IntelligentPad system.

件とすることにより，指定された位置に絵の貼ってあるページを選択することができる．

また，『ある作業環境のもとで，作成していたレポート等を探したい』というようにコンテクスト情報から必要な情報を探すときに，その作業環境の配置を検索条件に用いてレポートを探すこともできる．**図3**は，電卓や表やグラフを用いてレポートを作成している所である．これらの道具のパッドが，一枚の台紙のパッドの上にある．このとき，電卓が左上にあったということを条件としてこの作業環境（台紙）を探すことにより，この時，作成していたレポートを得ることができる．

IntelligentPad システムは，システムで扱うすべてのものをパッドとして統一している．書類を構成する写真や文字記入欄などの部品はすべてパッドであり，さらに，書類自身もパッドである．さらに，その書類

を入れるフォルダ等もパッドとして提供されている．このため，パッドの位置による検索機能を提供することにより，アプリケーションのレイアウト検索だけではなく，作業環境の配置等も条件として検索できることになる．

位置情報を管理する手法はいろいろ提案されている．R-tree[5]，R⁺-tree[6]，R*-tree[7]，Cell-tree[8]などは，各領域を分割し，B-tree[9]のような木構造でデータを管理する．これらは，位置に関する座標データを格納している．しかし，人間の記憶はあいまいであり，それぞれの位置を厳密に限定できないため，厳密な座標データを保持する必要はない．本論では，あいまい性をもたせた位置の検索機構について述べる．部品の位置に関する情報をコード化してシグニチャを作成する．これらを集めてシグニチャ・ファイルとする．検索時にはシグニチャ・ファイルを調べて条件を満たすものを取り出す．本章では，シグニチャのコード化の方法，シグニチャ・ファイルの構成とマッチングについて述べる．

3.1 パッドの位置のコード化

アプリケーションを構築している各プリミティブ・パッドの台紙に対する位置と大きさをコード化し，シグニチャを作成する．コード化の手順を以下に示す．

コード化の手順

(step1) 台紙の領域を $m \times n$ に分割する．i 行 j 列目の領域を $c(i, j)$ とする（図4参照）．

(step2) 台紙上のすべてのプリミティブ・パッドについて，以下(step3)(step4)を行い，部品の名前と位置を表すビットマップの組を生成する．

(step3) プリミティブ・パッドの種類名を抽出する．

(step4) $i=1$ から m まで，以下を繰り返す．

$j=1$ から n まで，以下を繰り返す．

$c(i, j)$ と，部品の領域が交わる場合には'1'，交わらない場合には'0'として，位置を表すビットマップを作成する．

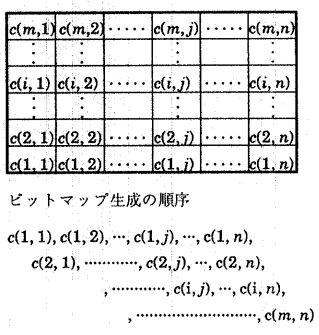


図4 台紙の分割領域とコード化の順序

Fig. 4 The partition of the base pad and the encoding scheme.

図5に，$m=n=4$ の場合のコード化の例を示す．図中の黒塗りと斜線の部分は，部品として用いられているプリミティブ・パッドの領域を示す．プリミティブ・パッドが，台紙の各分割領域と（少しでも）交わっている場合には，それに対応するビットが立てられる．このため，パッドの位置と大きさに関するビットマップには，分割領域の大きさ分のあいまい性が含まれることになる．プリミティブ・パッドの台紙に対する位置をコード化したビットマップをパッドの位置に関するシグニチャと呼び，それらを集めてシグニチャ・フ

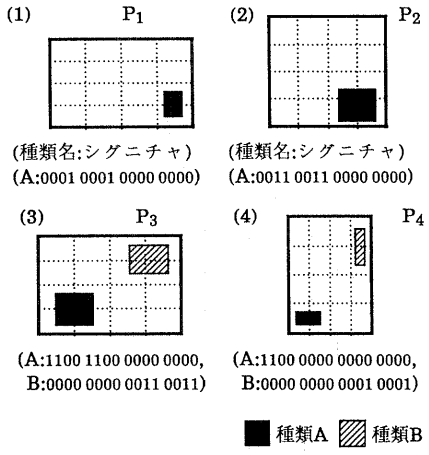


図5 部品の位置に関するあいまいなコード化

Fig. 5 Signatures that encode ambiguously specified components' positions.

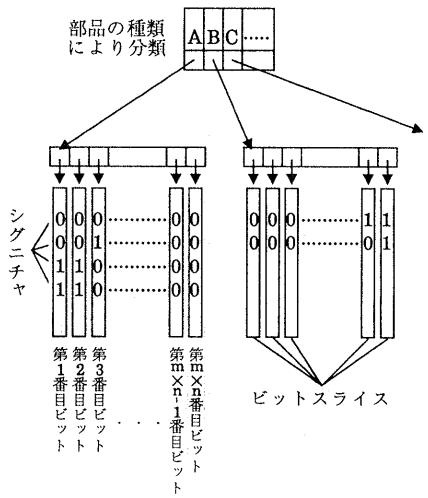


図6 シグニチャ・ファイルの構成

Fig. 6 The organization of a signature file.

ファイルを作成する．

シグニチャは，プリミティブ・パッドの種類ごとに分類し，Bit-Sliced Signature Files (BSSF)[10] 方式でシグニチャ・ファイルに格納する．BSSF 方式では，各ビットマップをビットごとに分割し，各ビットマップの同じ位置のビットを集めてそれぞれをファイルに格納する(**図6**参照)．まず，各シグニチャの第一番目のビットを格納し，次に第二番目のビットを格納する．以下同様に最後のビットまで，格納していき，ファイルを生成する．各ビットマップの同じ位置のビットごとに格納したファイルをそれぞれ列ファイルと呼ぶ．

3.2 シグニチャを用いたあいまい位置検索

検索時には，任意のパッドを配置して，位置に関する検索条件をパッドを用いて表現する．これを Query パッドと呼ぶ．Query パッドを構成するプリミティブ・パッドの種類と位置が検索条件として利用される．まず，Query パッドに対して前節と同様のコード化を行い各プリミティブ・パッドごとに Query シグニチャを作成する．Query シグニチャは，プリミティブ・パッドの種類名と対になっているので，そこで指定されている種類のパッドに関するシグニチャ・ファイルを検査し，Query シグニチャと同じシグニチャを有する合成パッドの台紙の ID 番号の集合を得る．その結果，Query パッドで指定した種類の部品が指定された位置にあるアプリケーションを得られる．**図7**の Q_1 を Query とすると，種類 A に対する 16 個の列ファイルを検査し，検索の結果，Q_1 と同じシグニチャを持つ台紙 P_1 (図5) が得られる．

また，パッドの位置に関するシグニチャは，既にあいまい性を含んだコード化がなされているが，ユーザは，これをさらに拡張して検索することができる．ユーザは，Query を指定する際に，拡張したい領域を指定する (図7における Q_2, Q_3 の網かけ部分)．このとき，あいまいな領域として指定された領域に関するビットはワイルド・カード'*'としてコード化される．検索時には，Query シグニチャの'*'以外の部分に関してのみ，それぞれに対応するシグニチャ・ファイルを検査し，'*'以外のビットが一致するものを探す．Q_2 を Query とすると，('*'に対応するビットを格納している列ファイルは検査の対象外となるため) 種類 A に対する 12 個の列ファイルを検査し，Query シグニチャの'*'以外の部分のビットが等しいシグニチャを持つ P_1 と P_2 (図5) が得られる．

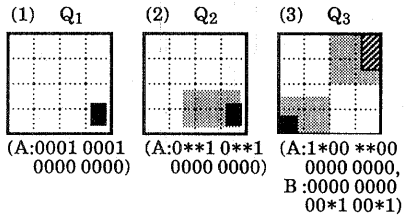


図7 query のコード化と検索
Fig. 7 The signature encoding of example queries.

Query パッドとして複数の部品の位置が指定されている場合は，個々の Query シグニチャを用いて先述の検索を行い，それぞれから得られた台紙の ID 集合から，共通なものを最終的な結果として返す．図7の Q_3 を Query として検索した場合，図5の P_3, P_4 を結果として得ることができる．

プリミティブ・パッドの種類を同定できない (覚えていない) 場合には，ワイルドカード・パッドというプリミティブ・パッドを用いて Query を表現する．ワイルドカード・パッドに対するシグニチャを Query シグニチャとした場合には，パッドの種類に関係なく，すべてのシグニチャ・ファイルを検査し，同じシグニチャを有する合成パッドを選び出す．また，パッドのカテゴリ分類に基づき，検索条件として用いられたプリミティブ・パッドと同じカテゴリに属するパッドの種類に対するシグニチャ・ファイルを検査するように拡張することも可能である．

検索の効率化に関しては，第4章において述べる．

3.3 領域分割とパッドの検索

データを絞り込むという観点からみれば，台紙の領域分割を細かくする方が良い．しかし，これを小さくしすぎるとユーザが，その位置を指定できなくなる．そこで，パッドの位置をコード化する際に分割する領域数と検索の関係について検討する．

まず，以下に示す手順で実験を行った．

(i)例題の番号をクリックすると，台紙の上に部品

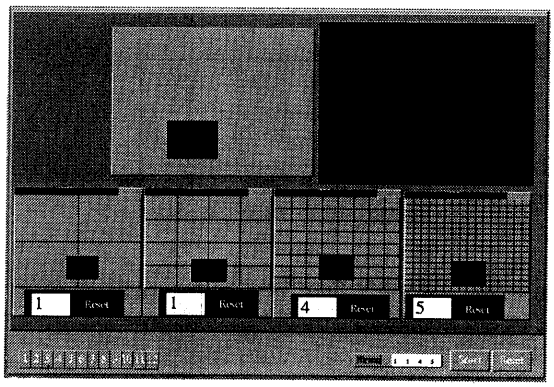


図8 部品のあいまいな位置による検索
Fig. 8 A search tool and queries that ambiguously specify components' positions.

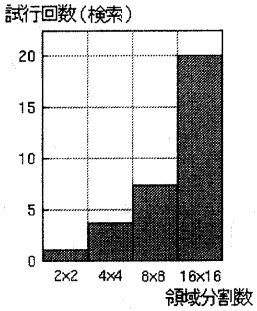


図9 領域分割数に対する試行回数の比較
Fig. 9 The required number of search trials as a function of the division granularity.

をひとつ配置した例題が画面に現れる（図8の上部参照）.

(ii)部品が5秒間点滅した後，例題は画面から消える.

(iii)その後すぐに，被験者は，与えられた例題を選択するためのQueryを作成する.（Queryを作成する際の，コード化の領域分割は図8の下部に示した，$m=n=2$（4分割），$m=n=4$（16分割），$m=n=8$（64分割），$m=n=16$（256分割）の4通りとし，分割数の少ないものから，Queryを作成した.）

図9に，これらの例題を選択するために発したQueryの試行回数の平均値（被験者8名）を示す．4分割においては，試行回数は，約1.1回であり，16分割では約3.7回，64分割では約7.3回，256分割では約20回という結果が出ている．さらに，30回以上の試行にもかかわらず，検索できなかった問題が，4分割，16分割では0%，64分割で5.2%，256分割では22.9%となっている．これらの結果より，256分割は，本研究の目的においては，明らかに不適である．また，ユーザは，検索時に，数回の試行で選出されない場合には，そのデータは無いと考え，検索をあきらめてしまう．これらのことを考慮すると，台紙の領域分割は，$m=n=4$〜ぐらいが妥当であると考えられる.

4. 検索の効率化

テキスト検索におけるシグニチャ・ファイルの格納形態及びファイル検査方法には以下のものが提案されている．Sequential Signature File (SSF)[11]は，シグニチャを順番にファイルに格納する．検索時にはファイル全体を走査しなければならないため，シグニチャ・ファイル自身が大きな場合には不適である．Bit-Sliced Signature Files (BSSF)[10]は各シグニチャのビットごとにファイルに格納するものである．検索時には，Queryシグニチャにおいてビットの立っている部分のファイルのみ走査すればよいため，SSFより速く検索できる．Frame-Sliced Signature File (FSSF)[12]は，SSFとBSSFの中間的なもので，シグニチャをいくつかのビットから成るフレームに分割し，フレームごとにファイルに格納する．検索時には，必要なフレームのみを走査すればよい.

これらのシグニチャ・ファイルにおいては，Queryシグニチャにおいてビットが立っている位置にビットが立っているかを調べればよい．Queryシグニチャでビットの立っていない部分は，'0'でも'1'でもかまわない．しかし，本論におけるパッドの位置検索においては，Queryで指定された位置にビットが立っていて，かつ，指定されていない位置にビットが立っていないものを探索しなければならない.

このため，SSFに比べて，アクセスするファイルの大きさが小さいという利点を持つBSSFの手法をそのまま，適用しても，アクセス領域はSSFとほとんど変わらず，BSSFの利点が生かされないことになる．そこで，ファイル構成はBSSFと同様にするが，その検査において，どのビットから検査するかによって比較するビット数が異なることに着目し，これを小さくすることにより検索の効率化を図った.

なお，この検索においては，False Drop（検索条件は満たさないが，Queryシグニチャの条件を満たしてしまうもの）はないため，これの検査に要する時間は考えなくて良い.

4.1 効率化のための基本方針

シグニチャ・ファイルは，台紙の分割領域に対応するビットごとに格納されている．この時，各ビットに対応するファイルを列ファイルと呼ぶ．検索時には，まず，最初のビット列ファイルを走査する．次の列ファイルをアクセスする際には，前の列ファイル走査において，Queryシグニチャで指定されたビットと同じであった部分に対応するビットのみを比較する（ただし，実際には，メモリアクセスはビット単位で行うわけではない．これに対する考察は4.4節において述べる）．**図10**の左の例では，左の列ファイルから順に走査する．1列目のファイルは，全領域を走査する．次に2列目のファイルは，1列目で条件を満たしたビットの位置（斜線部分）に応じた部分を比較すればよい．以下，同様に最後まで走査する．この時，アクセスする列ファイルの順序により，比較ビット数の，全ファイル（ビット数）に占める割合Pが異なる．Pを比較必要率と呼ぶ．図10の右の例は，同じファイルに対してアクセスする順序を変えたため，比較必要率Pの値

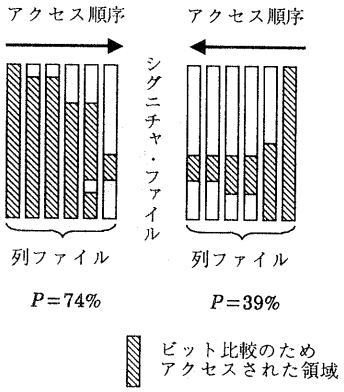


図10 アクセス順序による比較ビット数の違い
Fig. 10 The dependency of the required number of the compared bits on the different access orders.

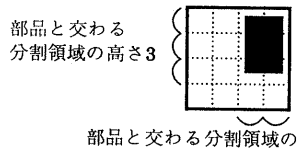


図11 台紙に占める部品の大きさと位置の例
Fig. 11 The size and the position of a component pad on its base pad.

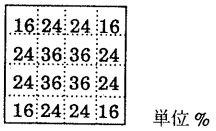


図12 各分割領域におけるビット1の立つ確率
Fig. 12 The probability of having the set bit in each segment.

が異なっていることを示す．ビットの比較に要する時間は，(1ビットの比較に要する時間)×(シグニチャ・ファイルの大きさ)×比較必要率で計算できる．そこで，同性能のマシンで，同一のシグニチャ・ファイルを検査した場合に，ビットの比較に要する時間の効率化の指標として，比較必要率を用いることとする．

シグニチャ・ファイルの検査時にアクセスする列ファイルの順序を変えることにより，比較を要するビット領域を小さくできることに着目し，以下の2つの方針により検索の効率化を図る．比較必要率 P を小さくするためには，(I)互いに相関関係の小さい分割領域に対するビットを格納した列ファイルを順にアクセスするとよい．また，このシグニチャ・ファイルは，重み（シグニチャにおける1の数）が少ないという特徴がある．このため，(II)各列ファイルの重みが，小さいものからアクセスすれば，候補数を早く絞り込め，比較ビット数を小さくできると考えられる．これについて，以下で述べる．

4.2 解析モデルとシグニチャ・ファイル（列ファイル）のアクセス順序

シグニチャ・ファイルへのアクセス順序により，検索が効率化されることを示すため，解析モデルと，アクセス順序について述べる．

台紙領域の分割を $m=n=4$ とする．このとき，部品の占める領域のパターンは100種類である（部品と交わる台紙の分割領域の幅が1の場合は4通り，2の場合は3通り，3の場合は2通り，4の場合は1通りで，合計10通りである．高さに関しても10通りある．よって，部品の占める領域のパターンは100通りとなる（図11参照）．そこで，各パターンが等しい頻度で存在するモデルを考える．つまり，部品の位置と大きさに偏りのないモデルを考える．なお，部品の種類は1種類とする．このモデルを用いてシグニチャ・ファイルを作成する．このとき，ファイル全体に対する重みの割合の平均は25%であり，台紙の各分割領域に対するビットでできる列ファイルにおける重みの割合はそれぞれ図12に示したとおりである．

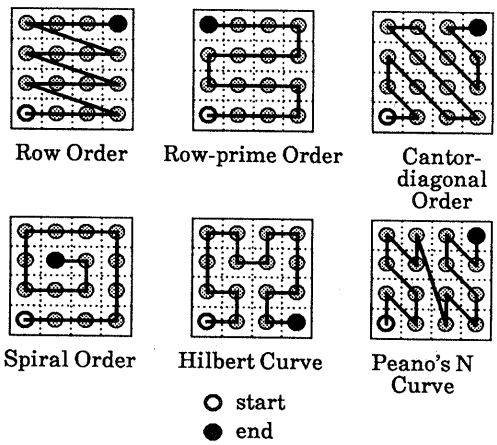


図13 ファイル・アクセス順序 その1
Fig. 13 The orders of file accesses.

図14 ファイル・アクセス順序 その2
Fig. 14 The orders of file accesses.

図13，14に，アクセスする順序を示す．それぞれ，図13，14に示した順序で，台紙の各分割領域に対応する列ファイルをアクセスするものとする．図13では，

実線に沿って交わる各領域に相当する列ファイルを順に比較し，●で終わる．これは，システマティックに定めた順序でアクセスした場合の例として代表的な順序を6種類示したものである．図14では，相関が小さい順にアクセスする順序を示したものである．点対称，線対称となる順序を省き（解析モデルの性質上，同じものと見なせるため），始点の異なる3種類を示した．なお，各領域間の相関は（先述のモデルに基づき）あらかじめ計算しておき，検索時にはファイルへのアクセス順番は確定しているものとする．

4.3 解析結果

前節で述べた解析モデルを用いて，列ファイルのアクセス順序により異なる比較必要率 P を比較する．Queryとして，解析モデルに用いたものと同じ100パターン（100個）のシグニチャを利用し，それぞれが要した比較必要率を調べた．

まず，図13に示したアクセス順序に従い，列ファイルを検査した場合に要した比較必要率を**図15**に示す．グラフの横軸は，シグニチャ・ファイル中の比較を要するビット数の全体に対する割合 P（比較必要率）であり，縦軸は，それを要したQueryの度数を示す．約5～40%の比較必要率 P に対して，ほぼ同数ずつのQueryが分布している．図15のグラフは，指定された列ファイルのアクセス順序がQueryに適している場合は，比較必要率 P の値を約10%ぐらいに押さえられることを示している．しかし，図13に示したアクセス順序は，すべてのQueryに適したアクセス順序ではないため，Queryにより，比較必要率 P の値にばらつきがあり，部品の大きさや位置により検索速度に差が出てくることになる．

次に，4.1節の検索の効率化の基本方針（I）「相関の小さい列ファイルからアクセスすることにより比較必要率 P を小さくできる」ことに基づき，図14に示したアクセス順序を適用した場合に要した比較必要率を**図16**に示す．Queryが要する比較必要率 P の値の範囲は約10～25%ぐらいとなり，Queryによるばらつきは，図15の場合よりせばまっている．また，図15の P の平均値は約25%であったが，図16においては，約18%となっている（**表1**参照）．

さらに，検索の効率化の基本方針（II）「列ファイルの重みの小さいものからアクセスすることにより比較必要率 P を小さくできる」ことに基づき，Queryシグニチャの'1'の立っているビットに対応する列ファイルに関しては，その重みが小さいものからアクセスする．そのあと，Queryシグニチャの'0'のビットに対応する列ファイルを，相関の小さい順に検査することとする．その場合に要した比較必要率を**図17**に示す．Queryの6割から7割が，約10～15%の比較必要率 P を要する結果となっており，Queryによるばらつきは，図15，16の結果に比べて最も小さくなっている．

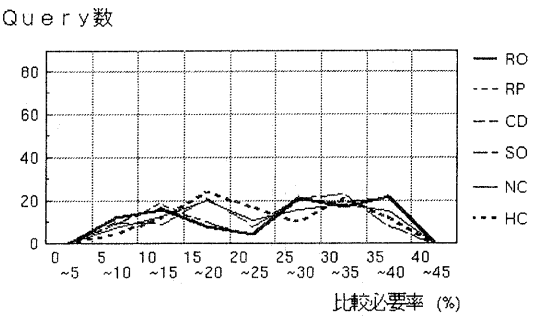


図15 アクセス順序の違いによる比較ビットの割合の比較
Fig. 15 The comparison of the number of compared bits for different file access orders.

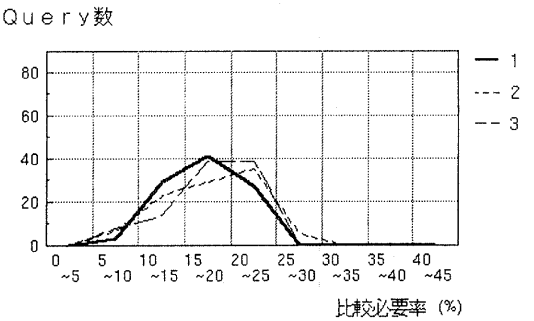


図16 アクセス順序の違いによる比較ビットの割合の比較（相関の小さい順）
Fig. 16 The comparison of the number of compared bits for different file access orders.

表1 アクセス順序の違いによる比較ビットの割合（平均値）
Table 1 The comparison of the number of compared bits for different file access orders.

比較順序	比較必要率 P (%)
Row Order (RO)	24.9
Row-prime Order (RP)	24.9
Cantor-diagonal Order (CD)	23.6
Spiral Order (SO)	25.0
Hilbert Curve (HC)	24.0
Peano's N Curve (NC)	23.9
相関の小さい順1	18.4
相関の小さい順2	17.8
相関の小さい順3	17.2
重みの小さい順1	10.8
重みの小さい順2	10.8
重みの小さい順3	10.7

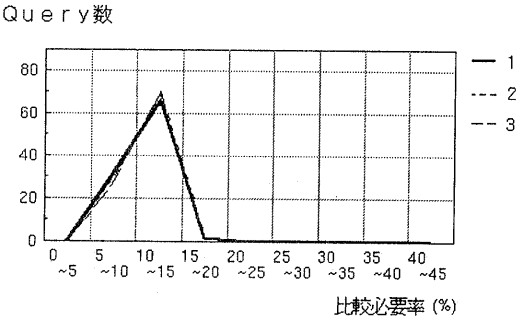


図17 アクセス順序の違いによる比較ビットの割合の比較（重みの小さい順）
Fig. 17 The comparison of the number of compared bits for different file access orders.

また，比較必要率 P の平均値も約11%と小さく押さえられている．ここで用いられている列ファイルは重みが小さいという特徴があるため，ビットが'1'であるという条件により，候補データの数を大幅に絞り込める．検索の初期の段階で，候補を大幅に絞り込んでおくことで，比較必要率 P を小さくできるのである．Queryに応じたアクセス順序により，列ファイルをアクセスするため，すべてのQueryに対して比較必要率 P を約1割程度に押さえられる．

以上の結果より，列ファイルのアクセス順序を変えることにより比較必要率 P を小さくできることを示した．あるQueryに適したアクセス順序は，ほかのQueryに適したアクセス順序であるとは限らない．そこで，本論では，シグニチャ・ファイルの特性に基づき，基本方針I, IIにより列ファイルへのアクセス順序を変えることを提案した．本論で用いたシグニチャ・ファイルは，重みが小さいという特徴がある．そこで，与えられたQueryシグニチャにおいてビットの立っている部分（'1'の部分）は，それに対応する列ファイルの重みが少ないものから順にビットの検査をする．次に，ビットの立っていない部分は，相関が小さい順にアクセスする．これにより，すべてのQueryに対して最適なアクセス順序となり，(パッドの数や条件を満たすパッドの数に依存せず) 比較必要率を平均約10%に押さえられることを示した．

なお，'1'のないQueryシグニチャ('0'と'*'から成るQueryシグニチャ）に対しては，基本方針(II)は，効果がないが，基本方針(I)の効果により，検索時に比較するビット数は平均約26%程度となる (Queryとしては，前節の解析モデルの100パターンのシグニチャの'1'を'*'に置き換えたものを利用した)．

4.4 メモリ・アクセスの単位

実際に，シグニチャ・ファイルを検査する際のメモリ・アクセス単位は，マシンに依存する．アクセス単位を8ビットと仮定する．列ファイルは，8ビットずつアクセスされ，この8ビット中に1ビットでも条件を満たすものがあれば，次にアクセスする列ファイルにおいても，その8ビットに対応する部分をアクセスしなければならない．このため，前節で述べた値をそのまま適用することはできない．しかし，それぞれのアクセス単位により，効率化の度合は異なるが，本論において述べた手法を適用することにより，検索の効率化を図ることができる*．

5. おわりに

本論において，部品の位置による検索について述べた．IntelligentPadシステムにおいて，部品の台紙に対するあいまいな位置を検索する機構を提供することにより，部品のだいたいの配置を条件として，必要なアプリケーションを検索することができる．また，同じ機構を用いることにより，位置による文脈指定検索が可能である．さらに，画像の色の分布をパッドの貼り合わせで表現するパッドなどの開発に伴い，本論で述べた検索法の適用範囲が広がっていくものと考えられる．

位置検索においては，部品の位置にあいまいさを持たせてコード化し，シグニチャ・ファイルを各ビットごとに格納する．ファイル検査時は，比較するビットの数を少なくするため，Queryシグニチャの'1'が立っている部分に対応する列ファイルの重みが少ない順にアクセスし，Queryシグニチャの'0'の部分に対応する列ファイルは，相関が小さい順にアクセスするとよいことを示した．

参考文献

1) 赤石美奈，田中 譲：IntelligentPadにおける部分構造検索，情報処理学会論文誌，Vol. 35, No. 2, pp. 232-242 (1994).
2) Tanaka, Y.: A Toolkit System for the Synthesis and the Management of Active Media Objects, *Proc. 1st Int. Conf. Deductive and Object-Oriented Databases,* Kyoto, pp. 269-277 (Dec. 1989).
3) Tanaka, Y.: A Synthetic Dynamic-Media System, *Proc. Int. Conf. Multimedia Information Systems,* Singapore, pp. 299-310 (1991).
4) Tanaka, Y., Nagasaki, A., Akaishi, M. and

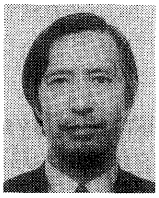
* テクニカル・レポート参照

Noguchi, T.: A Synthetic Media Architecture for an Object-Oriented Open Platform, *Proc. IFIP 12th World Computer Congress*, Madrid, Spain, pp. 104-110 (1992).
5) Guttman, A.: R-trees: A Dynamic Index Structure for Spatial Searching, *Proc. ACM-SIGMOD Int. Conf. Management of Data,* pp. 47-57 (1984).
6) Sellis, T., Roussopoulos, N. and Faloutsos, C.: The R^+-tree: A Dynamic Index for Multi-dimensional Objects, *Proc. 13th Int. Conf. Very Large Data Bases,* pp. 507-518 (1987).
7) Beckmann, N., Kriegel, H. P. and Schneider, R. and Seeger, B.: The R*-tree: An Efficient and Robust Access Method for Points and Rectangles, *Proc. ACMSIGMOD Int. Conf. Management of Data,* pp. 322-331 (1990).
8) Gunther, O. and Bilmes, J.: Tree-Based Access Methods for Spatial Databases; Implementation and Performance Evaluation, *IEEE Trans. Knowlledge and Data Engineering,* Vol. 3, No. 3, pp. 342-356 (1991).
9) Comer, D.: The Ubiquitous B-tree, *Computing Surveys,* Vol. 11, No. 2, pp. 121-138 (1979).
10) Faloutsos, C. and Chan, R.: Fast Test Access Methods for Optical Disks Designs and Performance Comparison, *Proc. 14th Int. Conf. VLDB,* Long Beach, CA, pp. 280-293 (1988).
11) Faloutsos, C. and Christodoulakis, S.: Signature Files: An Access Method for Documents and Its Analytical Performance Evaluation, *ACM Trans. Office Information Systems,* Vol. 2, No. 4, pp. 267-288 (1984).
12) Lin, Z. and Faloutsos, C.: Frame-Sliced Signature Filles, *IEEE Trans. Knowledge and Data Engineering,* Vol. 4, No. 3, pp. 281-289 (1992).

(平成6年9月8日受付)
(平成7年5月12日採録)

赤石 美奈（正会員）

昭和42年生．平成2年北海道大学工学部電気工学科卒業．平成4年同大学院修士課程修了．平成7年同大学院博士後期課程修了．工学博士．同年静岡大学工学部知能情報工学科助手．ヒューマンインタフェース，メディア・ベース等の研究に従事．ソフトウェア科学会，人工知能学会，電子情報通信学会各会員．



田中 譲（正会員）

昭和25年生．昭和47年京都大学電気工学科卒業．昭和49年京都大学電子工学専攻修士課程修了．工学博士．現在，北海道大学工学部電子情報工学専攻教授．データベースマシン，データベース理論，メディア・ベース，論理型プログラミング等の研究に従事．主たる著書「コンピュータ・アーキテクチャ」（オーム社，共著）．IEEE，ソフトウェア科学会，人工知能学会各会員．