フラクタル符号のベクトル集合間類似度に基づく検索の高速化手法

横山貴紀† 渡辺俊典† 古賀久志†

私たちは画像をフラクタル圧縮して得られるフラクタル符号の類似検索手法を提案した．この類似検索手法では，フラクタル符号をベクトル集合と見なし，ベクトル集合間に類似定義を与えることで，圧縮符号の直接検索を可能とした．画像の回転や拡大縮小，平行移動などの変動に対してロバストであり，既存のウェーブレット変換を用いた手法よりも良好な検索精度を示したが，実用に際し，類似度計算コストの削減が課題として残っていた．今回，1）ベクトル集合データに索引構造を導入することで類似度計算を高速化し，2）ベクトル集合データの要素数に対する類似度の上限値設定により類似度算出対象の画像数を削減する，という 2 つの戦略によって検索速度を大幅に改善した．

A Fast Fractal Code Retrieval Method Exploiting the Similarity of Vector Sets

Takanori Yokoyama,† Toshinori Watanabe† and Hisashi Koga†

We have proposed a fractal code retrieval method which decomposes a compressed code to a set of vectors, and exploits the similarity measured by the degree of one-to-one correspondence between two vector sets. This retrieval method is robust for various fluctuation of images. Although the retrieval performance of this method is better than conventional ones based on wavelet transform etc., it requires much retrieval time. In this paper, we propose an acceleration method for the fractal code retrieval to solve this problem. We introduce two following strategies in particular: 1) Retrieval system newly uses an index structure to store a set of vectors in order to perform the similarity computation efficiently. 2) We exploit the upper bound of the similarity easily derived from the cardinal numbers of vector sets to reduce the number of images for which the similarities have to be actually computed. These strategies contribute to the drastic improvement of the retrieval speed.

1. はじめに

一般にメディアデータのサイズは非常に大きいが，情報の冗長度は高いため，通常圧縮された状態でネットワーク上を流通し，データベースなどに蓄積される．このような圧縮されたメディア情報を対象とした検索手法の研究が近年さかんに行われるようになってきた[1)]．これらの検索手法により，メディアの圧縮符号だけをシームレスに扱う検索システムの実現が可能となる．

私たちは画像メディアについて研究を行い，フラクタル符号の検索手法[2)]を提案した．検索実験により，この手法が画像の変動にロバストな検索特性を持ち，公知の手法と比較して良好な検索精度を持つことを確認した．しかし，提案手法で定義した類似度の算出にかかる計算コストが高く，検索のたびに類似度を算出する必要があることから，検索速度の実用面での課題が残っていた．

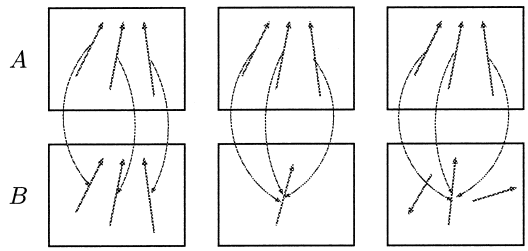
この問題について本稿では，類似度算出に必要な符号データに対し最近接点探索の索引構造を適用し，計算時間の短縮を試みる．また，類似度がとりうる上限値を利用して類似度算出対象の画像数を絞り込み，検索速度の高速化を図る．

本稿の構成は次のとおりである．2 章では既提案のフラクタル符号の検索手法について，その概要を改めて説明する．3 章では，検索手法を高速化する提案手法の詳細を説明し，4 章では，提案手法の有効性を実験により検証する．5 章では，本稿のまとめと今後の課題について述べる．

2. フラクタル符号の検索手法[2)]

フラクタル符号とは，画像 I 中の矩形領域 R_i に最も相似な領域 D_i を調べ，その相似関係を写像として

† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, University of Electro-Communications



(a) A, B が類似 (b) 要素数が異なる (c) 分布が異なる

図 1 写像 $f_B(A)$ による対応関係
Fig. 1 Correspondences induced by $f_B(A)$.

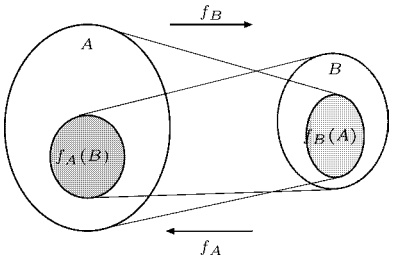


図 2 写像に基づく集合間の関係
Fig. 2 Relation between two vector sets based on f_B and f_A.

記録したものである．詳しくは文献 3) などを参照されたい．領域 R_i と D_i の相似関係は，それぞれの領域の左端点の座標 (x_{R_i}, y_{R_i})，(x_{D_i}, y_{D_i}) を用いて，4 次元ベクトル $(x_{R_i}, y_{R_i}, x_{D_i}, y_{D_i})$ として表現することができる．2 枚の画像を I_A，I_B とし，得られるフラクタル符号を A，B とする．ベクトルを a_i，b_i とし，$|\cdot|$ は集合の要素数を表すとすると，フラクタル符号は $A = \{a_1, \ldots, a_{|A|}\}$，$B = \{b_1, \ldots, b_{|B|}\}$ と，4 次元ベクトルを要素とする集合として表現することができる．

以上から，フラクタル符号間の類似問題は，ベクトル集合間の類似問題と見なすことができる．だが，集合 A，B の要素は同じ 4 次元の空間上に存在するものの，集合の要素が完全に一致することはほとんどなく，共通集合 $A \cap B$ や和集合 $A \cup B$ を基に類似性を求めても，有効な検索結果を得ることができない．

そこで本手法では，2 つの集合をそれぞれ他方へ写像し，写像による集合特性から類似度を求めることとした．写像は $f_B : A \to B$ を

$$f_B(a_i) = \arg\min_{b_j \in B} \|a_i - b_j\| \qquad (1)$$

とし，同様に $B \to A$ への写像を f_A とする．この写像により，2 つの集合間には**図 1** に示すような対応関係が構成される．集合 A，B が似ている場合には 1 対 1 の関係が構成され，両者の要素数が異なる場合や，要素数は同じでも要素の分布が異なる場合には，多対 1 の関係が構成される傾向を持つ．

この写像により，2 つの集合間には**図 2** のような関係が構成できる．検索手法では，以上の関係に着目して，次の指標を集合間の類似度として定義した．

$$s(A, B) = \frac{|f_B(A)| + |f_A(B)|}{|A| + |B|} \qquad (2)$$

これは，A と B が非常に似通った要素で構成されているときには $s(A, B) = 1$ となり，要素の構成が大きく異なる場合には 0 に近づく指標である．

実験により，この類似度による検索手法が画像の平行移動，拡大，回転などの変動に対してロバストな検索特性を持ち，ウェーブレット変換を用いた検索手法との比較でも良好な検索精度を示すことを明らかにした．

3. 検索手法の高速化

上記提案の類似度を用いる場合，検索のたびに，質問画像の符号と，すべての検索対象画像の符号との間で類似度を算出する必要がある．また，類似度算出自体にかかる計算コストも，文献 2) で用いた距離行列に基づく方法では非常に高い．そのため，検索対象の画像数が増えるにつれて検索時間が増大し，現実の検索システムとして実用的ではなかった．

そこで今回，1）ベクトル集合データに索引構造を導入することで類似度計算を高速化し，2）ベクトル集合データの要素数に対する類似度の上限値設定により，類似度算出対象の画像数を削減する，という 2 つの戦略を用いて検索速度を大幅に改善する．

3.1 索引構造を用いた類似度算出

類似度の単純な計算方法は，距離行列を生成し，最小距離にある要素をすべて探索するものである．集合の要素数を $M = |A|$ と $N = |B|$ とすると，距離行列の生成に $O(MN)$ と，最小の距離にある要素の決定に $O(MN + NM)$ の計算コストが必要となる．

だが，式 (1) の写像は，一方の集合から取り出した要素について，他方の集合から最近接点の要素を探索する問題であり，これはベクトル集合データに索引構造を適用することで，高速に探索することが可能となる．

今回は，集合の要素が低次元（4 次元）であること，最近接点の探索機能のみが必要であること，後述する部分集合に基づく検索に適していることから，索引構造として K-D-B 木[4] を用いることとした．

3.2 上限値による類似度算出対象の絞り込み

索引構造を用いることで類似度の算出時間は改善されるが，それでも検索対象の画像数に比例して検索時間は増加する．検索の高速化には，類似度算出に費やされる計算コストをできるだけ少なくする必要があり，その解決策の 1 つとして，類似度算出対象の画像数を削減する方法が考えられる．

類似度算出で最も計算コストを要するのは，式 (2) の $f_B(A)$，$f_A(B)$ の計算である．これらの写像を決定しない段階で検索対象を絞り込むことが望ましい．式 (2) の類似度 s の定義から，次式のような自明な上限値が存在する．

$$s(A,B) \leq \frac{2 \times \min(|A|,|B|)}{|A|+|B|} \tag{3}$$

この上限値を本稿では「**自明な上限値**」と呼び，式 (2) の類似度 s を上限値と区別するために以後「**厳密類似度**」と呼ぶこととする．この上限値を基に検索対象画像を類似度算出前に絞り込むことが可能となる．

また，以下のような上限値を考えることもできる．

$$s(A,B) \leq \frac{|f_B(A)|+\min(|A|,|B|)}{|A|+|B|} \tag{4}$$

これは $f_B(A)$ のみを厳密に計算し，$f_A(B)$ の計算は省略することで実現できる上限値である．前述の「自明な上限値」よりも，$f_B(A)$ 分だけ「厳密類似度」に近い上限値を与えることができ，検索対象画像数をさらに絞り込むことが期待できる．この上限値を「**片側上限値**」と呼ぶ．

以上の考えを拡張すると，片側（たとえば A）に含まれるすべての要素を写像するのではなく，一部の集合だけを写像し，その結果に基づいた上限値設定も可能となる．この部分集合に基づく上限値を「**部分集合上限値**」と呼び，次式のように定義する．

$$s(A,B) \leq \frac{2\min(|A|,|B|)-(|U|+|V|)+|f_B(U)|+|f_A(V)|}{|A|+|B|} \tag{5}$$

ここで U，V は，$U \subseteq A$，$V \subseteq B$ となる，上限値を求める際に写像する部分集合を表す．部分集合の要素数は $|U|,|V| \leq \min(|A|,|B|)$ が満たされない場合，該当する部分集合の要素数を $\min(|A|,|B|)$ に置き換える．

この「部分集合上限値」は，その定義から明らかなように「自明な上限値 ($U=\emptyset$，$V=\emptyset$)」および「片側上限値（$U=A$，$V=\emptyset$)」を含んでおり，$U=A$，$V=B$ の場合は「厳密類似度」と一致する．

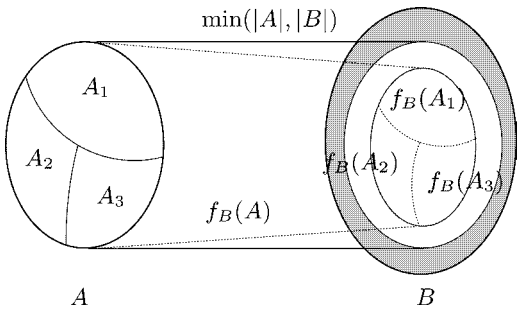


図 3 部分集合の写像
Fig. 3 Subset mappings.

3.3 索引構造に基づく部分集合と段階的検索

「部分集合上限値」は，その設定に用いる部分集合 U，V の大きさが A や B に近づくほど，「厳密類似度」の検索精度に近づくが，計算コストも大きくなる．単独の「部分集合上限値」を用いて検索精度の向上と，計算コストの削減をともに実現することは困難である．

そこで異なる部分集合を用いて複数の「部分集合上限値」を設定し，それらを組み合わせて段階的に検索対象を絞り込むことで，検索結果の上位の検索精度を保ちながら検索時間を短縮するアルゴリズムが効果的となる．これを本稿では「**段階的検索**」と呼ぶ．

以下では「段階的検索」の検索方法の詳細を述べるが，まず「段階的検索」に必要な部分集合の取り方と，索引構造との関係について述べ，その後に検索アルゴリズムを示す．

3.3.1 直和分割と上限値設定

複数の「部分集合上限値」を設定する際，ベクトル集合 A が直和分割されていると，以下のように効率良く計算することができる．

図 3 のように集合 A が部分集合 A_1，A_2，A_3 に分割されているとき，たとえば $U_1=\{A_1\}$，$U_2=\{A_1,A_2\}$，$U_3=\{A_1,A_2,A_3\}$ とすれば，3 種類の上限値が設定できる．このとき，U_1 の上限値を求めるときに使用した $f_B(A_1)$ は，次に U_2 の上限値を求める際，$f_B(U_2)=f_B(A_1)\cup f_B(A_2)$ であるので，再び計算する必要はない．この結果，U_2 の上限値の計算では，$f_B(A_2)$ のみを求めればよい．U_3 の場合も同様で，U_2 の計算結果を利用し，$f_B(A_3)$ だけ計算すればよい．

このように，複数の「部分集合上限値」の設定では，すでに求めた写像結果を用いることができる．部分集合が互いに素であれば，重複する領域の写像計算がないため，複数の「部分集合上限値」を効率良く求めることができる．

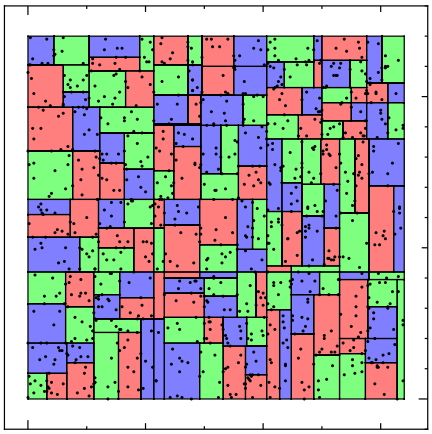


図 4 K-D-B 木に格納されたリーフノードとデータ点
Fig. 4 Leaf nodes with data in a K-D-B-tree.

3.3.2 K-D-B 木と部分集合

今回，ベクトル集合データの索引構造として K-D-B 木[4] を用いた．この索引構造は，k-d 木[5] と B 木[6] の性質が融合されたもので，データは該当するリーフノードの領域に格納する．

図 4 は 2 次元のランダムデータに対して作成した K-D-B 木の例である．リーフノードの領域が直和分割されているため，図の色分けのように，互いに素な部分集合を容易に生成することができ，複数の「部分集合上限値」を効率良く計算することができる．

また，同一のリーフノードに収められたデータは，互いにその距離が近くなる特性がある．このため，A, B が大きく異なっている場合，たとえば A のごく一部である $|U| \ll |A|$ のような部分集合 U の「部分集合上限値」によって，両者が異なることを判別できる可能性が高くなる．

その理由は，もし，A 中の互いに離散したデータ点を部分集合 U とし，これを写像した場合，集合 A，B の違いにかかわらず，B 内のデータ点と 1 対 1 の対応関係が構成される可能性が高く，「部分集合上限値」が 1 に近づく傾向を持つ．だが，データ間の距離が近い部分集合を写像するとき，集合 A，B が異なっている場合は，多対 1 の関係を構成する可能性が高くなるからである．

3.3.3 段階的検索のアルゴリズム

「部分集合上限値」では，使用する部分集合の大きさによって検索性能が異なってくる．異なる部分集合を用いて複数の「部分集合上限値」を設定し，適用範囲を「部分集合上限値」ごとに変え，段階的に対象を絞り込む「段階的検索」のアルゴリズムを以下に述べる．

ここでは質問画像のベクトル集合を A とし，検索対象データベースを $DB = \{B_1, \cdots, B_{|DB|}\}$ とする．$B_1, \ldots, B_{|DB|}$ は各画像のベクトル集合を表す．各上限値および厳密類似度の適用範囲をユーザの指定する数値 $K_1, \ldots, K_L$ によって制御する．L は設定した「部分集合上限値」数を表す．

step 1：「自明な上限値」の計算

DB 中の各ベクトル集合 $B_1, \ldots, B_{|DB|}$ について，A に対する「自明な上限値」をそれぞれ求め，その結果を降順にソートし，$i = 1$，$J = K_1$ とする．

step 2：「部分集合上限値」による絞り込み

DB 中の上位 J 以内のベクトル集合について，部分集合 U_i（注），$V = \emptyset$ の「部分集合上限値」を求め，降順にソートする．i を 1 増やし，$i \leq L$ であれば $J = K_i$ として step 2 を繰り返す．

（注） 質問側のベクトル集合 $A = \{A_1, \cdots, A_L\}$ とする，$U_i = \{A_1, \cdots, A_i\}$ とする．

step 3：検索結果の確定

DB 中の上位 K_{L+1} 以内のベクトル集合について「厳密類似度」を求め，その結果を基に順位をつけ，最終的な検索結果とする．

$K_{L+1} \leq K_L \leq \cdots \leq K_1$ を満たすように設定することで，step 2，3 における検索対象の絞り込みを実現する．

4. 実　　験

実験により提案手法の有効性を示す．提案手法を C 言語によって実装し，コンパイラは gcc 2.95.3 を用いた．実験は OS が Linux 2.4.20，CPU は Pentium 4 のクロック周波数 2.80 GHz，1 GB のメモリで構成される PC 上で行った．検索対象の画像を 1,264 枚用意し，このデータセットからフラクタル符号を生成し，圧縮符号データベースを作成した．画像の詳細や圧縮パラメータは文献 2) と同一である．

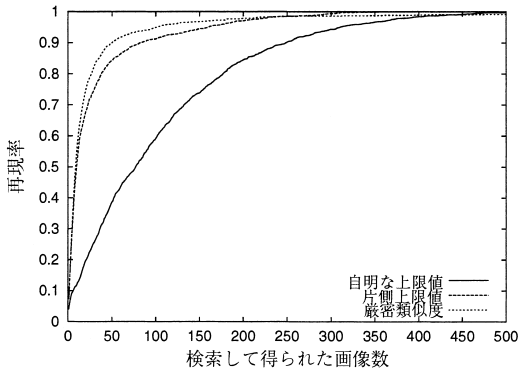
4.1 索引構造を用いた類似度算出の高速化

距離行列を用いた従来の提案手法では，質問画像と対象画像との間で距離を求め，この距離行列を基に写像を決定し，類似度を算出する．このとき，質問画像 1 枚あたり平均 221.1 秒の検索時間が必要であった．

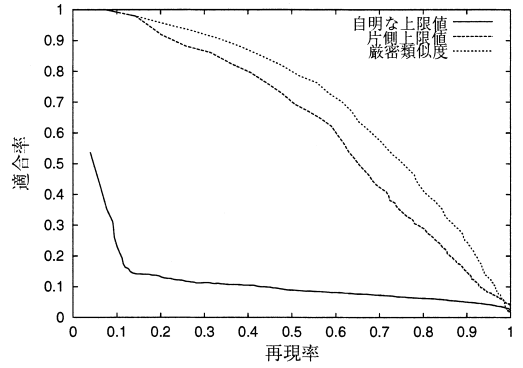
一方，符号から得られるベクトル集合データの索引構造に K-D-B 木を用いた場合，質問画像 1 枚あたり平均 87.4 秒であった．索引構造の導入による計算速度の改善が確認できる．

4.2 上限値を用いた類似度算出対象の削減

ここでは，各上限値の特性を検索実験によって明らかにし，その結果を基に上限値を組み合わせた「段階



(a) 検索して得られた画像数と再現率



(b) 再現率と適合率

図 5 「厳密類似度」,「自明な上限値」および「片側上限値」の指標に基づく検索性能

Fig. 5 Retrieval performances under the strict similarity, upper bound, and one-sided upper bound.

的検索」の検索性能を示す．なお,「部分集合上限値」はすべての上限値を含む定義だが，本実験では「自明な上限値」と「片側上限値」とは区別して扱う．

特性を検索精度で示すため，画像のデータセットからあらかじめ分類した 7 種類，合計 91 枚の類似画像群を用いた．検索精度の指標には以下を用いた．

$$\text{precision} = \frac{|\text{retrieved} \cap \text{relevant}|}{|\text{retrieved}|} \tag{6}$$

$$\text{recall} = \frac{|\text{retrieved} \cap \text{relevant}|}{|\text{relevant}|} \tag{7}$$

retrieved は検索された画像の集合を表し，relevant は検索結果として適切だと思われる画像の集合を表す．本実験においては質問画像が属する類似画像群を relevant とした．式 (6) は適合率，式 (7) は再現率である．

4.2.1 各上限値の検索精度

まず,「自明な上限値」と「片側上限値」,「厳密類似度」について，それぞれの指標を単独で用いて検索した結果を**図 5** に示す．図 5 (a) の横軸は検索して得られた画像数を，縦軸は再現率を表し，検索から得られる上位 500 の結果を示す．図 5 (b) の横軸は再現率を，縦軸は適合率を表す．以後，すべての検索結果は類似画像群 91 枚について得られた値の平均値である．

図 5 から,「自明な上限値」の結果は「厳密類似度」および「片側上限値」と比べて悪いことが分かる．しかし,「自明な上限値」の再現率は 500 位付近で 1.0 にほぼ漸近していると見なすことができる．これは「自明な上限値」による検索で得られる上位 500 以内に，質問画像に対する relevant な画像群がほぼすべて含まれることを意味する．このことから「自明な上限値」を用いて類似度算出対象の画像数を，全画像 1,264 枚から半分以下の 500 枚に削減することが可能となることが分かる．

検索時間は,「自明な上限値」の算出には 0.01 秒しかかからなかったが「片側上限値」には 42.1 秒と「厳密類似度」算出の半分の計算時間が必要であった．

次に「部分集合上限値」の結果を示す．この実験では，質問画像のベクトル集合 A の部分集合 U だけを用い，検索対象画像のベクトル集合 B の部分集合を $V = \emptyset$ として用いない．部分集合には質問画像のベクトル集合が格納されている K-D-B 木のリーフノードに基づいて得られる集合 A_1，A_2，A_3 を用いる．

「部分集合上限値」による検索性能を**図 6** に示し，参考として「自明な上限値」と「厳密類似度」による結果も合わせてプロットする．図中「1/3」は $U_1 = \{A_1\}$ を,「2/3」は $U_2 = \{A_1, A_2\}$ を表している．また「3/3」は $U_3 = \{A_1, A_2, A_3\}$ であり,「片側上限値」と同じ結果となる．

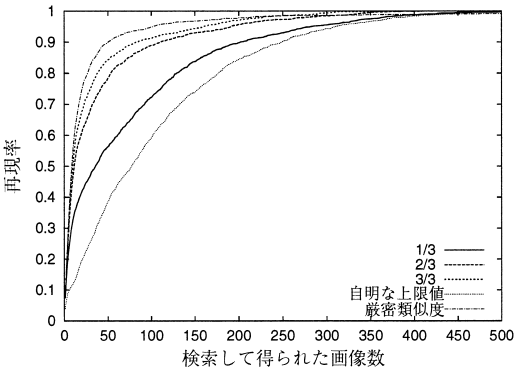
この図から「部分集合上限値」では,「自明な上限値」よりも検索精度が良くなることが分かる．また質問画像 1 枚あたりの計算時間は,「部分集合上限値 (1/3)」では 15.8 秒,「部分集合上限値 (2/3)」では 32.9 秒と,「片側上限値」よりも計算コストが削減できることが分かる．

4.2.2 段階的検索

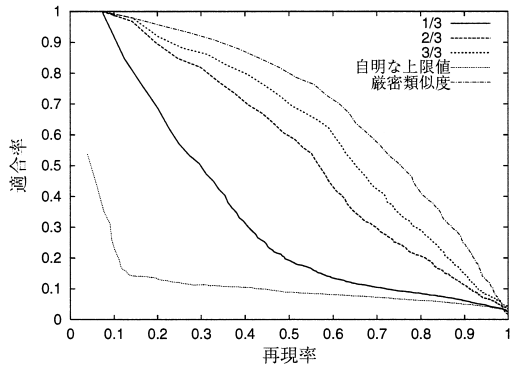
前項の結果に基づき「段階的検索」の適用範囲を設定する．上位 30 位までの再現率が 0.7 以上になるように，適用範囲を決定するパラメータ K_i を設定する．各パラメータを変えながら検索を複数回行ったうえで，**表 1** のように設定した．

得られた検索性能を**図 7** に示し，参考のため「自明な上限値」と「部分集合上限値 (1/3 と 2/3)」,「厳密類似度」による検索性能も合わせてプロットする．この図から，再現率の曲線が各上限値の適用範囲と検索特性に応じて変化することを読み取ることができる．

質問画像 1 枚あたりに必要な検索時間は平均 9.2 秒

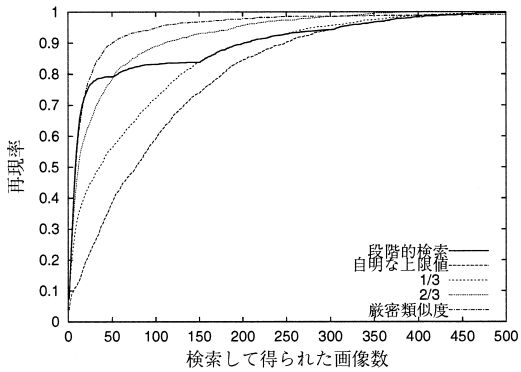


(a) 検索して得られた画像数と再現率

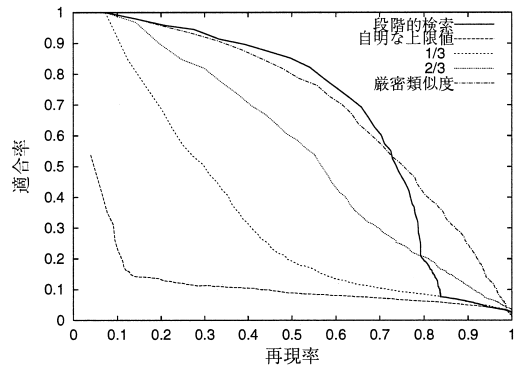


(b) 再現率と適合率

図 6 「部分集合上限値」に基づく検索性能
Fig. 6 Retrieval performances under the subset-mapping-based upper bounds.



(a) 検索して得られた画像数と再現率



(b) 再現率と適合率

図 7 段階的検索による検索性能
Fig. 7 Retrieval performance of the proposed stepwise retrieval method.

表 1 「段階的検索」での各パラメータ値
Table 1 Parameters in stepwise retrieval.

ステップ	各上限値および類似度	K_i
1	自明な上限値	–
2	部分集合上限値 (U_1)	$K_1 = 300$
	部分集合上限値 (U_2)	$K_2 = 150$
	片側上限値 (U_3)	$K_3 = 100$
3	厳密類似度	$K_4 = 50$

であり，厳密類似度だけを用いた場合の約 1/10 と，高速に検索できることが分かる．計算時間の内訳は，0.1%が「自明な上限値」，44.1%が「部分集合上限値 (1/3)」，22.1%が「部分集合上限値 (2/3)」，13.9%が「片側上限値」，19.8%が「厳密類似度」であった．

5. ま と め

本稿では，フラクタル符号のベクトル集合間類似度に基づいた検索手法について，検索の高速化手法を提案した．

まず，類似度算出に必要な符号データに索引構造の K-D-B 木を導入し，類似度の計算時間を改善した．次に，ベクトル集合データの要素数に基づいて類似度がとりうる上限値を設定し，類似度算出の対象画像数を削減した．また，符号データに用いた索引構造から直接得られる部分集合を用いて複数の「部分集合上限値」を設定し，これらを組み合わせて検索対象の画像数を段階的に削減する「段階的検索」を提案した．検索精度を高く保ったまま，従来よりも高速な検索が可能となることを示した．

提案した「段階的検索」では，各上限値や厳密類似度を求める範囲の指定が，最終的に得られる検索性能に影響を及ぼす．今回，これらを事前に行った実験結果に基づいて決定したが，今後はその自動設定法などを検討する必要がある．

謝辞 本研究について，東北学院大学の菅原研助教授から，貴重なご意見と多大なご助力を長年にわたり頂戴してきた．ここに記して感謝の意を表す．

参 考 文 献

1) Mandal, M.K., Idris, F. and Panchanathan, S.: A critical evaluation of image and video in-

dexing techniques in the compressed domain, *Image and Vision Computing*, Vol.17, No.7, pp.513–529 (1999).
2) 横山貴紀，菅原 研，渡辺俊典：フラクタル符号に基づく圧縮領域における類似画像検索手法，情報処理学会論文誌：データベース，Vol.45, No.SIG 4(TOD21), pp.11–22 (2004).
3) Fisher, Y. (Ed.): *Fractal Image Compression: Theory and Application*, Springer-Verlag New York, Inc. (1995).
4) Robinson, J.T.: The K-D-B-Tree: A search structure for large multidimensional dynamic indexes, *ACM SIGMOD Int. Conf. on the Management of Data*, pp.10–18 (1981).
5) Bentley, J.L.: Multidimensional Binary Search Trees Used for Associative Searching, *Comm. ACM*, Vol.18, No.9, pp.509–517 (1975).
6) Bayer, R. and McCreight, E.M.: Organization and Maintenance of Large Ordered Indexes, *Acta Informatica*, Vol.1, pp.173–189 (1972).

(平成 16 年 6 月 20 日受付)
(平成 16 年 10 月 7 日採録)

(担当編集委員 金子 邦彦)

横山 貴紀（学生会員）

2000 年電気通信大学電気通信学部電子情報学科卒業．2002 年同大学大学院情報システム学研究科博士前期課程修了．現在，同大学院博士後期課程に在籍．画像処理，圧縮領域における検索手法の研究に従事．

渡辺 俊典（正会員）

1971 年東京大学工学部航空工学科卒業．同年日立製作所入社，中央研究所，システム開発研究所．経営生産，LSI 設計，非線形最適化，学習機械，並列分散推論（ICOT プロジェクト）等諸システムの開発に従事．1992 年より電気通信大学大学院情報システム学研究科教授．工学博士．電子情報通信学会，日本写真測量学会，IEEE 各会員．専門はメディアデータ自動解析および情報システム表現法．

古賀 久志

1995 年東京大学大学院理学系研究科修士課程修了．同年（株）富士通研究所入社．2002 年東京大学大学院理学系研究科博士課程修了．理学博士．2003 年より電気通信大学大学院情報システム学研究科講師．ネットワーク，離散アルゴリズムの研究に従事．