2004－CVIM－145　(19)
2004／9／10

相互写像に基づくベクトル集合間類似度とその上限値

横山　貴紀†　　渡辺　俊典†　　古賀　久志†

† 電気通信大学大学院情報システム学研究科
E-mail: †{yokotaka,watanabe,koga}@sd.is.uec.ac.jp

概 要　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検討している．この検索手法では，フラクタル符号をベクトル集合と見なし，ベクトル集合間の類似度を用いて検索を行う．ベクトル集合間の類似度計算コストは高く，検索時間に実用上の問題があったが，類似度の上限値を用いて類似度算出の対象数を削減する手法により大幅な改善を実現した．本稿では，この既報告の上限値を拡張し，部分集合に基づく新たな上限値設定法を提案する．この新たな上限値は，既提案の上限値を一般化したものであり，部分集合の取り方によって多様な上限値設定が可能となる．上限値設定法の詳細と，部分集合の扱いを説明した後，画像検索への適用結果を示す．

A Vector Set Similarity Measure based on Bi-directional Mapping and Its Upper Bounds

Takanori YOKOYAMA†, Toshinori WATANABE†, and Hisashi KOGA†

† Graduate School of Infomation Systems, University of Electro-Communications
E-mail: †{yokotaka,watanabe,koga}@sd.is.uec.ac.jp

Abstract　We have proposed a fractal code retrieval method. There, we interpreted a fractal code as a vector set, and introduced a similarity measure between vector sets. This similarity measure required high computation cost. To reduce it, we proposed a method to use upper bounds of the similarity measure. In this report, we further improve the proposed upper bounds by using subsets of the vector sets, and introduce a new upper bound of the similarity measure. This new upper bound is a generalization of the already proposed upper bounds. After discussions on the details of the new upper bound, and we examine its properties through an image retrieval experiment.

1. はじめに

私たち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する手法を提案した [1]．このベクトル集合間の類似度算出に掛かる計算コストは高く，検索速度の実用面での課題があったが，類似度に上限値を設けることで類似度算出の対象数を削減する手法 [2] を提案し，検索速度が大幅に改善された．

今回，既提案の上限値を一般化した，新たな上限値設定法を提案する．既提案では自明と片側の 2 種類の上限値しか設定できなかったが，この新たな上限値では部分集合に基づいた多様な上限値設定が可能となり，類似度算出対象をユーザの要求に応じて柔軟に絞り込むことが可能となる．この上限値設定法の詳細と，実装時に重要となる索引データ構造，段階的な検索アルゴリズムについて考察した後，画像検索実験によりその性質を明らかにする．

以下，2. では，ベクトル集合間類似度の定義，3. では，既提案の類似度の上限値設定方法と，索引データ構造の導入，段階的検索について述べる．4.

では，新規提案の部分集合に基づく上限値設定法と，索引データ構造に着目した段階的検索について説明する．5. では，画像検索の実験を通して提案手法の性質を検証し，6. でまとめを行う．

2. ベクトル集合間の類似度に基づく検索手法

検索対象データがベクトルを要素とする集合である場合，このベクトル集合間に類似性の指標を与えるものが「**ベクトル集合間類似度**」である．

この類似度が有効な一例として，フラクタル符号の検索手法 [1] を以下で説明する．この類似度を用いたフラクタル符号の検索手法は，画像の平行移動，拡大，回転などの変動に対してロバストであり，ウェーブレットを用いた検索手法をしのぐ検索精度を示すことも実証した．

2.1 フラクタル符号のベクトル集合表現

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．フラクタル符号には，画像中の相似領域の関係が記録されており，領域 (x_{R_i}, y_{R_i}) に相似な領域を (x_{D_i}, y_{D_i}) とすると，相似関係を 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．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．このように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して扱うことができる．

2.2 ベクトル集合間の類似指標

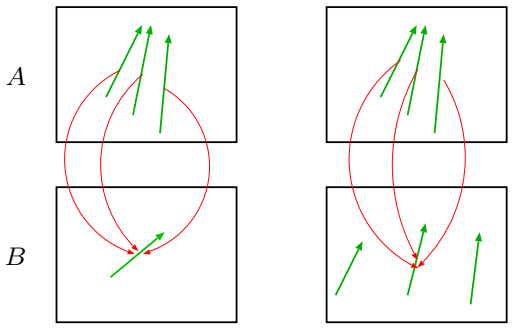
ベクトル集合 A，B それぞれが持つ要素は，同一空間上に存在するものの，互いの集合の要素が完全に一致することはほとんどなく，共通集合 $A \cap B$ や和集合 $A \cup B$ を基に類似性を求めても，有効な検索結果を得ることができない．このような特性を持つベクトル集合間に対し，集合の要素数と要素の分布の両者を反映した類似性が必要であると考えた．

そこで，2 つの集合をそれぞれ他方へ写像した時の，写像の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．図 1 に示すように，ベクトル集合間の要素数や，要素の分布が異なる場合，多対 1 の関係が構成される傾向を持つ．

図 2 に示すように，双方向の写像 f_A，f_B が構成できるので，これを用いてベクトル集合間 A，B の類似度を，以下のように定義する．



(a) 要素数が異なる　　(b) 要素の分布が異なる

図 1　写像 $f_B(A)$ による多対 1 の関係

Fig. 1　Many-to-one correspondence by $f_B(A)$ mapping.

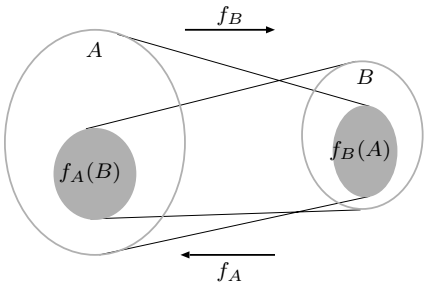


図 2　双方向の写像 f_A，f_B

Fig. 2　Bi-directional mappings f_A and f_B.

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

A と B が全く同じ要素で構成されている時，$s(A, B) = 1$ となる．

検索は質問側のベクトル集合とデータベース中の検索対象である個々のベクトル集合との間の類似度を求めることで行う．

3. 類似度の上限値と段階的検索

ベクトル集合間の類似度は計算コストが掛かる上に，検索の度に全ての検索対象との間で類似度を算出する必要があるため，検索対象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は実用的でない．

そこで文献 [2] では，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）類似度の上限値を設定し，3）類似度と上限値を段階的に適用することで類似度の計算回数を削減する，という 3 つの戦略を用いて検索速度を改善した．

3.1 ベクトル集合データへの索引データ構造の適用

類似度算出に必要な式 (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を探索するものである．そこで，ベクトル集合

データに索引データ構造を適用することで高速な探索が可能となる．

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から，索引データ構造として k-d 木 [3] を用いた．

3.2 上限値による類似度算出対象の絞り込み

索引データ構造を用いることで類似度算出時間は改善されるものの，検索の対象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，その解決策のひとつとして，類似度算出の対象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，写像 $f_B(A)$，$f_A(B)$ の計算である．よって，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．

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

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数を類似度算出前に絞り込むことが可能となる．

また，次のような上限値も考えることができる．

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

これは $f_B(A)$ のみを厳密に計算し，$f_A(B)$ の計算を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，$f_B(A)$ の写像分だけ「厳密類似度」に近い上限値となり，検索対象数をさらに絞り込むことができる．この上限値を「**片側上限値**」と呼ぶ．

3.3 段階的検索

検索精度（質問に対する適合性の高い検索結果を示す確率）は,「自明な上限値」が最も悪く,「片側上限値」そして「厳密類似度」の順に良くなるが，計算コストもこの順に高くなるというトレードオフ関係があり，単独の指標を用いて検索精度の向上と高速化を共に実現することは困難である．

そこで，これらの指標を組み合わせて用いることで高速に，しかも上位の検索精度を維持したまま検索を行う「**段階的検索**」が効果的である．以下にそのアルゴリズムを示す．

step 1. すべての検索対象について「自明な上限値」を求め，順位をつける．

step 2. step 1 で得られた結果の上位 K_1 以内について「片側上限値」を求め，順位をつける．

step 3. step 2 で得られた結果の上位 K_2 以内について「厳密類似度」を求め，検索結果とする．

各ステップにより，類似度算出対象を段階的に絞り込む機構となっている．K_1 および K_2 の設定によって検索時間と検索精度が変わる．文献 [2] の実験では，上位 30 位で再現率 0.7 を満たすように，$K_1 = 150$，$K_2 = 50$ を設定した．この結果，厳密類似度のみを用いた場合の約 1/12 の時間で，同精度の検索の実現が可能となった．

4. 部分集合に基づく上限値の設定法

既提案の上限値を拡張した，新たな上限値設定法について説明する．

4.1 部分集合上での上限値

「自明な上限値」および「片側上限値」の考え方を拡張し，片側の集合（例えば A）に含まれるすべての要素を写像するのではなく，集合の一部だけを写像し，その結果に基づく上限値を設定することが可能である．これを「**部分集合に基づく上限値**」と呼び，次式のように定義する．

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

ここで U，V は，上限値を求める際に写像する部分集合 $U \subseteqq A$，$V \subseteqq B$ を表す．$|U|, |V| \leq \min(|A|,|B|)$ が満たされない場合は $|U|$，$|V|$ を $\min(|A|,|B|)$ に置き換える．

この新規の上限値は，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（$U=A$，$V=\emptyset$)」を含む既提案の上限値の一般化であり，$U=A, V=B$ の時は「厳密類似度」と一致する．部分集合の取り方によって多様な上限値を生成することが可能となる．

上限値は部分集合 U，V が A，B に近づくほど，「厳密類似度」に近づくが，計算コストは高くなる．ここでも，単独で上限値を用いるのではなく，複数の上限値を段階的に用いて検索するアルゴリズムが効果的である．

4.2 索引データ構造と部分集合を用いた段階的検索

「部分集合に基づく上限値」の段階的検索では，ベクトル集合の領域が直和分割されている場合，効率的に検索を行うことができる．図 3 のように集合 A が A_1, A_2, A_3 に分割されている時，例えば $\{A_1\}$，$\{A_1, A_2\}$，$\{A_1, A_2, A_3\}$ の部分集合によって 3 種類の上限値が設定できる．この時，$V = \{A_1\}$ の上限値を求める時に使用した写像 $f_B(A_1)$ は，次に $V = \{A_1, A_2\}$ の上限値を求める

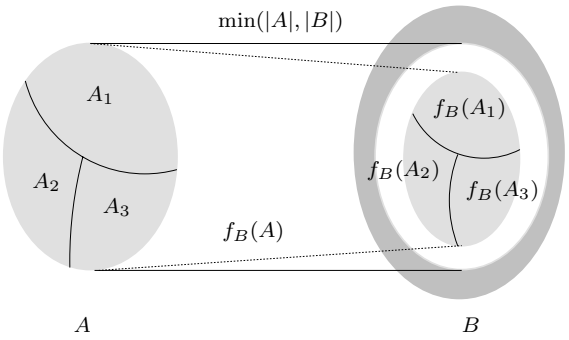


図 3　部分集合の写像
Fig. 3　Subset mappings.

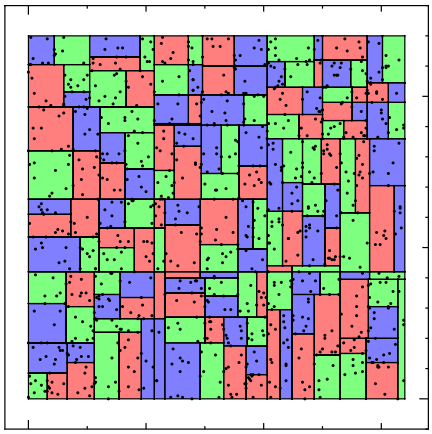


図 4　K-D-B 木に格納されたリーフノードとデータ点
Fig. 4　Leaf nodes with data in a K-D-B-tree.

際，$f_B(V) = f_B(A_1) \cup f_B(A_2)$ であるので，再び利用することができる．この結果，$V = \{A_1, A_2\}$ の上限値では，写像 $f_B(A_2)$ のみ計算すれば良いことになる．$V = A$ の場合も同様である．

このように，部分集合を用いた上限値に基づいて段階的に検索を行う場合，すでに写像した結果を用いることができる．この時，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無いため，上限値を効率良く求めることができる．

今回，ベクトル集合データに適用する索引データ構造に K-D-B 木 [4] を用いた．この索引データ構造は，k-d 木 [3] と B 木 [5] 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保持される．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．図のように領域が直和分割されているため，リーフノードの領域に基づいて，互いに素な部分集合を容易に取り出すことができる．

「部分集合に基づく上限値」の「段階的検索」のアルゴリズムを以下に述べるが，ここでは集合 $U \subseteqq A$ だけを用い，検索対象のベクトル集合の部分集合 $V \subseteqq B$ を用いない場合（つまり $V = \emptyset$）のアルゴリズムを説明する．

step 1.　すべての検索対象について「自明な上限値」を求め，その結果に順位をつけ，$J = K_1$ とする．

step 2.　上位 J 以内の検索対象について「部分集合 U_i（注）に基づく上限値」を求め，その結果に順位をつけ i を 1 増やし，$i \leqq N$ であれば $J = K_i$ として step 2 を繰り返す．

（注）　質問のベクトル集合 $A = \{A_1, \cdots, A_N\}$ とする時，$U_i = \{A_1, \cdots, A_i\}$ とした．$A_1, \ldots, A_N$ は互いに素な集合であるため，例えば U_1 で写像した部分集合 A_1 は，U_2 で再び写像する必要はない．

step 3.　step 2 で得られた結果の上位 K_{N+1} 以内について「厳密類似度」を求め，最終的な検索結果とする．

既提案の段階的検索と同じく，$K_1, \ldots, K_{N+1}$ の設定により，得られる検索精度と検索時間が異なる．

5. 実　　験

フラクタル符号のデータベースを対象とした検索実験から，提案手法の性質を分析する．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 2.80GHz，1GB のメモリで構成される PC 上で行った．

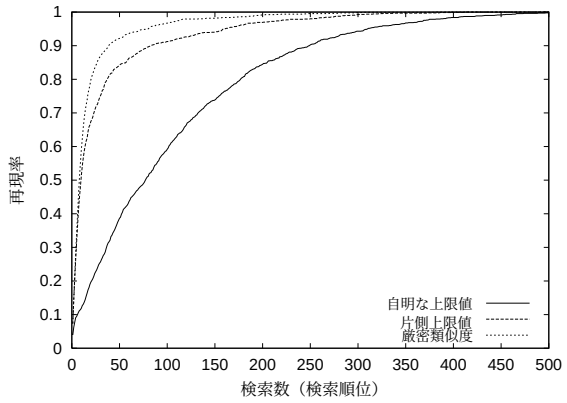
画像数は 1,264 枚を用意し，このデータセットからフラクタル符号を生成し，検索対象の圧縮符号データベースを作成した．画像の詳細や圧縮パラメータ，再現率および適合率の算出に用いた 6 種類計 91 枚の類似画像群などは文献 [1] と同一である．

5.1　上限値の検索精度

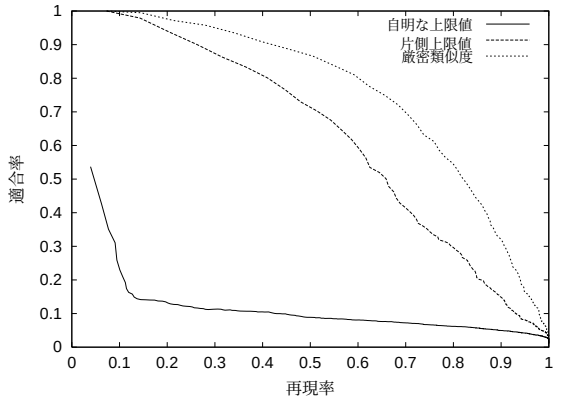
まず「自明な上限値」と「片側上限値」，そして「厳密類似度」のそれぞれの指標を用いて検索した結果を図 5 に示す．

図 5(a) の横軸は検索画像数を，縦軸は再現率を表す．この図では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．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わかる．検索時間は，「厳密類似度」の検索に 87.4 秒，「自明な上限値」には 0.01 秒しか掛からなかったが「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．



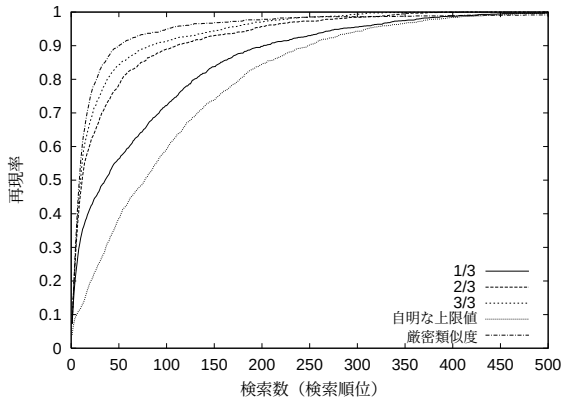
(a) 検索数と再現率



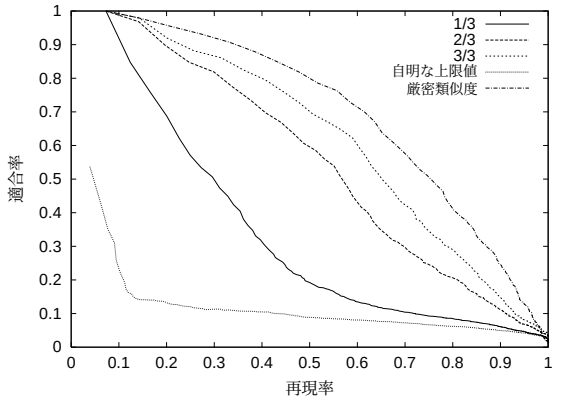
(b) 再現率と適合率

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結果

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



(a) 検索数と再現率



(b) 再現率と適合率

図 6　「部分集合に基づく上限値」に基づく検索結果

Fig. 6　Retrieval performances under the subset mapping based upper bounds.

5.2　部分集合に基づく上限値の検索精度

次に「部分集合に基づく上限値」について，異なる部分集合から得られた 3 種類の上限値の性能を示す．部分集合は，質問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領域を 3 分割して得られる集合 A_1, A_2, A_3 を用いた．

実験結果を図 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している．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，大きく検索精度に開きのあった「片側上限値」と「自明な上限値」の間に,「部分集合に基づく上限値」の結果が位置していることがわかる．これは「部分集合に基づく上限値」を導入することで，多様な「段階的検索」の組み合わせが可能になることを意味する．

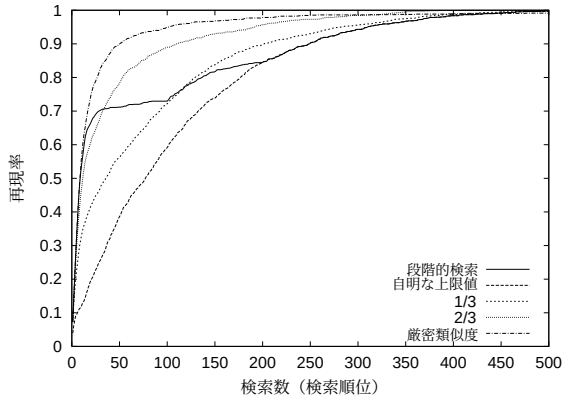
5.3　部分集合に基づく上限値の段階的検索

前節の結果に基づき，各上限値の適用範囲を設定する．検索では上位 30 位までの再現率が 0.7 以上になるように設定した,「段階的検索」の各処理ステップで用いるパラメータを表 1 に示す．

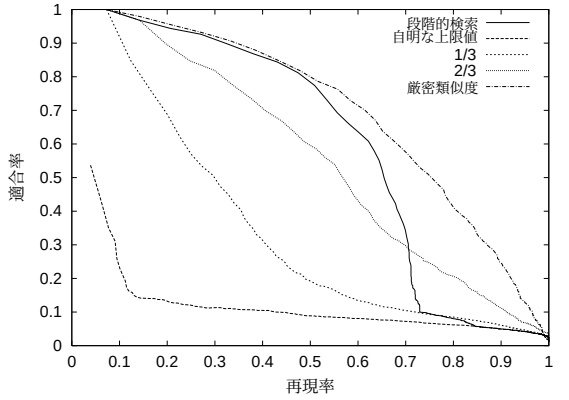
得られた検索結果を図 7 に示す．参考のために「自明な上限値」および「部分集合に基づく上限値 (1/3 と 2/3)」と「厳密類似度」による検索結果も合わせてプロットしている．この図から，再現率の曲線が各上限値の特性に応じて変化することを読み取ることができる．質問画像 1 枚あたりに必要な検索時間は平均 14.3 秒であり，厳密類似度だけを用いた場合の約 1/6 と，高速に検索できることがわ

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)	$K_1 = 200$
	部分集合上限値 (U_2)	$K_2 = 100$
	片側上限値 (U_3)	$K_3 = 70$
3	厳密類似度	$K_4 = 50$



(a) 検索画像数と再現率



(b) 再現率と適合率

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

かる．

5.4 考　　察

一般化した上限値により，部分集合を用いて多様な上限値設定が可能となるので，これらを用いて段階的検索を組み立てた．実験結果から，既提案の「自明な上限値」および「片側上限値」だけを用いた場合よりも，検索精度を柔軟に調整することが可能となることがわかる．部分集合に基づく上限値設定と，これを用いた段階的検索の有効性の一端が示されたと考える．

しかし，既提案の上限値を用いた質問 1 つあたりの検索時間が平均 6.4 秒だったのに対し，今回提案した手法は平均 14.3 秒であった．これは，今回実装した K-D-B 木の性能が，既報告で用いた k-d 木より劣っていたため，結果として既報告を超える検索時間の改善を実現することができなかったものと考えられる．この問題について今後継続して研究し，改善を目指す．

6. ま と め

本稿では，ベクトル集合間の類似度を用いた検索手法の検索時間を改善する上限値設定法について，部分集合に基づく新たな上限値設定法を提案した．従来の上限値では実現できなかった多様な上限値の設定と段階的検索が可能となった．今後は段階的検索における各ステップのパラメータの自動設定法などを検討する予定である．

ベクトル集合間の類似度は，フラクタル符号だけでなく，ベクトル集合と見なすことができる各種メディアへの応用が可能であると考えられる．今後はさまざまなメディアデータへの適用を試みていきたい．

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渡り頂戴してきた．ここに記して感謝の意を表します．

文　　献

[1] 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).

[2] 横山貴紀, 渡辺俊典, 古賀久志: フラクタル符号のベクトル集合間類似度に基づく画像検索手法, 情報科学技術レターズ (FIT2004) (2004). 掲載予定.

[3] Bentley, J. L.: Multidimensional Binary Search Trees Used for Associative Searching, *Communications ACM*, Vol. 18, No. 9, pp. 509–517 (1975).

[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] Bayer, R. and McCreight, E. M.: Organization and Maintenance of Large Ordered Indexes, *Acta Informatica*, Vol. 1, pp. 173–189 (1972).