距離空間インデックスを用いた類似画像検索の実装と評価

4V－6

岩崎 雅二郎
（株）リコー ソフトウェア研究所

1 はじめに

近年、WWWやデジタルカメラの普及により画像データを容易に取込めるようになり、身の回りに画像データが氾濫しつつある。そこで大量の画像データからユーザの所望する画像を検索する技術が脚光を浴び始めた。画像検索の一手法として画像から抽出した画像特徴を逐次比較（画像特徴間距離を算出）して類似する画像を検索する方法がある。この方法では画像数が少量の場合には問題はないが、大量になると必然的に遅くなる。そこで、画像特徴のインデックスを生成し検索の高速化が必要となる。

多次元データである画像特徴をR-tree[1]のような多次元インデックスを利用して検索する方法が考えられる。しかし、画像特徴として代表的な色のヒストグラム特徴ではビン間の相関を考慮するヒストグラム距離(式1)が一般に利用される。多次元空間インデックスはユークリッド距離を前提にしているので、このヒストグラム距離により近傍検索することが困難である。一方距離空間インデックスではある条件さえ満たせば如何なる距離定義でも利用ができる。そこで、既存の距離空間インデックスを改良し画像検索に適用した上で比較評価を行った。

2 距離空間インデックス

距離空間インデックスではオブジェクト（画像特徴）間の距離のみに基づいて距離空間を順次分割しツリー型インデックスを生成する。オブジェクトをO、オブジェクト間の距離（類似度）を$D(O_x, O_y)$とすると、距離空間インデックスでは次のような条件を満さなければならない。

1. $D(O_x, O_y) = D(O_y, O_x)$
2. $D(O_x, O_y) > 0, O_x \neq O_y$
3. $D(O_x, O_x) = 0$
4. $D(O_x, O_y) \leq D(O_x, O_z) + D(O_z, O_y)$

Implementation and evaluation of similarity search for images using distance space based index
Masajiro IWASAKI
Software Research Center, RICOH Co., Ltd.

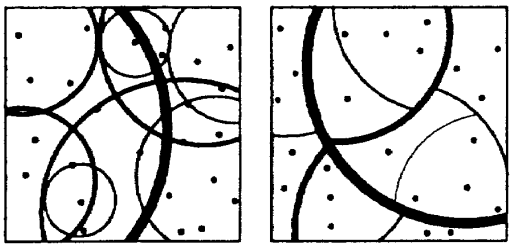

図 1: 距離空間インデックス

距離空間インデックスの一つであるM-tree[2]では各空間は複数の中心オブジェクト及び半径によって指定される閉領域により複数に分割される。分割された領域はさらに同様にして順次分割され、多分岐のツリー（図2）としてインデックスが生成される。図1にM-treeの空間分割の例を示す。また、M-treeは動的にデータを登録することを考慮したアルゴリズムである。

もう一つの距離空間インデックスであるvp-tree[3]では、各空間は一つの中心オブジェクト(vantage point)と半径によって内側、外側に二分され、同様に順次分割し二分岐のツリーが生成される。図1にvp-treeの空間分割の例を示す。なお、M-treeと異なりvp-treeは動的にデータを登録することを考慮していない。

3 M-tree、vp-treeの改良

上記M-treeとvp-treeは、1)検索速度が不十分である、2)動的にデータを登録できない、といった問題点がある。そこでこれらの問題点を解決すべく以下に示すM*-tree及びdvp-treeを構築した。

M*-tree

M-treeではツリーが飽和状態になると登録処理により生成されたノードはB-treeのようにルートノードに伝播し最終的にルートノードでツリー全体が繰り下ることでツリーが成長する。その時に下位のノードを包含する領域（二次元の場合には図1のような円）をルートノードとして生成する。

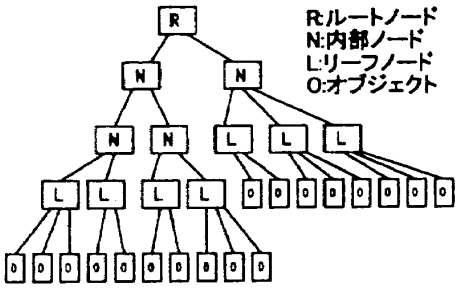

図 2: ツリーインデックス

したがって、成長するごとにノードの領域が急激に大きくなる傾向があり、その結果、領域間のオーバラップが増加し、検索速度の低下を招く。そこで、R-tree と同様にリーフノードの親ノードが飽和した時にリーフノードとの間に新たに生成した内部ノードを挿入することで成長するように改良を加えた。

dvp-tree

vp-tree は予め全登録データが用意され、そこから適宜データをサンプリングして空間を分割するので追加登録ができない。そこで、登録オブジェクトを包含するリーフノードにオブジェクトを追加し、そのノードがオーバーフローした時に vp-tree と同様に領域を分割するように変更することで、動的にデータを登録できるように改良した。さらに、M-tree と同様に中心オブジェクト及びその中心オブジェクトとオブジェクト間の距離をリーフノードに加え、これらの値を基に検索時に検索範囲内か否かをオブジェクトにアクセスする前に判断することでオブジェクトへのアクセス回数を減らし検索の高速化の改良を行った。

4 性能測定

Sun Ultra2(UltraSPARC-II 296MHz) 上でフォトやクリップアートなどの 25,458 画像を登録し M*-tree、dvp-tree、そして比較のため M-tree の 3 形式のインデックスを生成し N 個の近傍検索を行い検索性能を測定した。M-tree 及び M*-tree の内部ノードの分岐数は 5、リーフノードの分岐数は 10 とした。dvp-tree のリーフノードの分岐数は 10 とした。また、画像からは色のヒストグラム（54 ビン）を抽出し、画像特徴間距離として以下の式で表されるヒストグラム距離を採用した。

$$D(O_x, O_y)^2 = \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}(O_{x,i} - O_{y,i})(O_{x,j} - O_{y,j}) \quad (1)$$

なお、a_{ij} は i 番目のビンと j 番目のビンの類似度であり、$O_{x,i}$ は i 番目のヒストグラムのビンを示す。

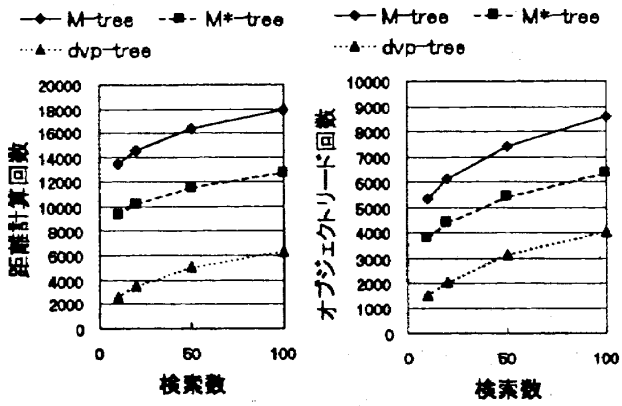

図 3: 距離計算回数及びオブジェクトリード回数

距離空間インデックスは比較的小さく（本評価において M-tree：約 930KB、M*-tree：約 780KB、dvp-tree：約 990KB）、かつ、近年のメモリの低価格化によりメモリ上にインデックスを保持した方が高速に検索することが可能である。したがって、ノードはメモリ上に、リーフノードにリンクされるオブジェクトは二次記憶に保持している。このようなデータ構成から画像特徴間距離計算とオブジェクトのリードが検索処理の大部分を占める。そこで、検索画像数を 10、20、50、100 と変化させた場合の距離計算回数及びオブジェクトリード回数を図 3 に示す。図より距離計算回数、オブジェクトリード回数ともに M-tree、M*-tree、dvp-tree の順で性能が高く（値が低く）なっていることがわかる。当然のことながら検索数が増えると各値は増加する。

5 結論

M*-tree 及び dvp-tree が M-tree より高速に検索が可能であることが示された。特に vp-tree を改良した dvp-tree では、検索数が 10 なら逐次距離を計算する場合と比較し距離計算回数は約 1/10、オブジェクトリード回数は約 1/20 となっており、dvp-tree の有効性が確認された。

今後は現状の dvp-tree の各ノード空間の分割数を増やすことにより距離計算回数を減らし、さらなる検索の高速化を目指す予定である。

参考文献

[1] Guttman, A., R-trees: a dynamic index structure for spatial searching, Proc. ACM SIGMOD Int. Conf., pp.47-57, 1984

[2] Ciaccia, P., Patella, M., Zezula, P., M-tree: An Efficient Access Method for Similarity Search in Metric Spaces, Proc. of the 23rd VLDB Conf. Greece, 1997

[3] Yianilos, P. N., Data Stuctures and Algorithms for Nearest Neighbor Search in General Metric Spaces, ACM-SIAM Symp., pp.311-321, 1993