九州大学学術情報リポジトリ
Kyushu University Institutional Repository

クラスタリングに基づく情報の検索と視覚化

堀田，政二

https://doi.org/10.15017/1398256

出版情報：九州芸術工科大学，2001，博士（工学），課程博士
バージョン：
権利関係：

第7章

次元削減とクラスタリングに基づくフィルタリングによる画像検索の高速化

7.1 まえがき

データ検索では，大量のデータを高次元ベクトル空間で検索する必要があり，高速化が望まれる．そこで本章では，クエリに近いk個の画像を選び出すkNN検索を次元削減とクラスタリングに基づくフィルタリングによって高速化する方法を提案する[40].

まずはじめに，Hafnerら[14]によって提案された2次形式距離と距離の不等式に基づくフィルタリングによってkNN検索の計算時間を短縮する方法を概説する．このフィルタリングをk平均法に適用することにより，クラスタリングの計算時間が短縮されることを実験で示す．続いてクラスタリングと三角不等式を利用するFukunagaら[19]のフィルタリング法を概説し，特徴ベクトルの次元削減とクラスタリングとを組合せたフィルタリングを用いてkNN検索の計算量を削減する方法を提案する．実験によりkNN検索がフィルタリングによって高速化されることを示す.

7.2 2次形式距離と特徴ベクトルの次元削減

本章の検索では画像間の距離をカラーヒストグラム間の2次形式距離で測る．そこで，まず最初にHafnerら[14]によって提案された2次形式距離と特徴ベクトルの次元削減法について概説し，距離の不等式に基づくフィルタリング法を示す.

7.2.1 カラーヒストグラムの2次形式距離

画像を n 色のカラーヒストグラム $\boldsymbol{h} = [h_1, \ldots, h_n]^T$ で表す．ここで h_i は第 i 色の割合とする．$\sum_{i=1}^{n} h_i = 1$ となるように規格化しておく．色 i の Lab 値を L_i, a_i, b_i とする．色 i と j の類似度を

$$s_{ij} = e^{-\alpha[(L_i-L_j)^2+(a_i-a_j)^2+(b_i-b_j)^2]} \tag{7.1}$$

とする．本論文では $\alpha = 0.001$ としている．$S = [s_{ij}]$ を色類似度行列と呼ぶ．画像 p のヒストグラム $\boldsymbol{h}_p$ と画像 q のヒストグラム $\boldsymbol{h}_q$ の2次形式距離の2乗は

$$\begin{aligned} QFD^2(\boldsymbol{h}_p, \boldsymbol{h}_q) &= \sum_{i=1}^{n}\sum_{j=1}^{n}(h_{pi} - h_{qi})s_{ij}(h_{pj} - h_{qj}) \\ &= (\boldsymbol{h}_p - \boldsymbol{h}_q)^T S(\boldsymbol{h}_p - \boldsymbol{h}_q) \end{aligned} \tag{7.2}$$

で定義される [14]．これは L_1 距離 (インタセクションと等価) や L_2 距離 (ユークリッド距離) よりも人の知覚に近いことが知られている．この2次形式距離は計算量が多いので標準化しておく．色類似度行列 $S = [s_{ij}]$ の固有値分解を $S = U\Lambda U^T$ とする．$U = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n]$ は固有ベクトル $\boldsymbol{u}_i$ を並べた直交行列，Λ は固有値を並べた対角行列である．すると式 (7.2) は $QFD^2(\boldsymbol{h}_p, \boldsymbol{h}_q) = (\boldsymbol{h}_p - \boldsymbol{h}_q)U\Lambda U^T(\boldsymbol{h}_p - \boldsymbol{h}_q)$ となり，ヒストグラム $\boldsymbol{h}$ を

$$\boldsymbol{x} = \Lambda^{1/2}U^T\boldsymbol{h} \tag{7.3}$$

により $\boldsymbol{x}$ に変換すると，式 (7.2) は $\boldsymbol{x}$ のユークリッド距離の2乗

$$\begin{aligned} D^2(\boldsymbol{x}_p, \boldsymbol{x}_q) &= (\boldsymbol{x}_p - \boldsymbol{x}_q)^T(\boldsymbol{x}_p - \boldsymbol{x}_q) \\ &= \sum_{i=1}^{n}(x_{pi} - x_{qi})^2 \end{aligned} \tag{7.4}$$

となる．S はすべての画像で共通なので，すべての画像のヒストグラムを式 (7.3) で予め標準化しておく．この標準形の上位 $l \leq n$ だけの $D_l^2(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sum_{i=1}^{l}(x_{pi} - x_{qi})^2$ で $D^2(\boldsymbol{x}_p, \boldsymbol{x}_q)$ を近似したのが低次元距離であり $D_l^2(\boldsymbol{x}_p, \boldsymbol{x}_q) \leq D^2(\boldsymbol{x}_p, \boldsymbol{x}_q)$ が成り立つ．次元削減によるフィルタリングはこの不等式に基づいている．

7.2.2 次元削減に基づくフィルタリングによる kNN 検索の高速化

m 枚のデータベース画像のなかからクエリ画像に近いものを k 個選び出す kNN 検索を考える．クエリ画像のヒストグラムを式 (7.3) で標準化したものを $\boldsymbol{q}$，データベース画像 i のヒストグラムを式 (7.3) で標準化したものを $\boldsymbol{x}_i (i = 1, \ldots, m)$ とし，それらの低次元距離を $D_l^2(\boldsymbol{q}, \boldsymbol{x}_i)$，元の次元での距離を $D^2(\boldsymbol{q}, \boldsymbol{x}_i)$ とする．

Hafner ら [14, 41] の領域検索法を単純に kNN 検索に拡張すると次のようなアルゴリズム (kNN-FDR: kNN search with filtering by dimensionality reduction) になる [15, 16]：

アルゴリズム kNN-FDR:

Step 1: $D_l^2(\boldsymbol{q}, \boldsymbol{x}_i)$ をすべての i について計算し，これが小さい順に k 個の i を選ぶ.

Step 2: 選ばれた i すべてについて $D^2(\boldsymbol{q}, \boldsymbol{x}_i)$ を計算し，そのうちの最大値を求める.

Step 3: その最大値を ϵ として，$D_l^2(\boldsymbol{q}, \boldsymbol{x}_i) \leq \epsilon$ を満たす i を選ぶ.

Step 4: 選ばれた i すべてについて $D^2(\boldsymbol{q}, \boldsymbol{x}_i)$ を計算し，これが小さい順に k 個出力する.

この手法では探索範囲を半径 $\sqrt{\epsilon}$ の円内に限定している．しかし，この ϵ は必要十分な値よりも大きい．Seidl ら [17] は ϵ が最適な値に収束する次の改良アルゴリズム (kNN-OFDR: kNN search with optimal filtering by dimensionality reduction) を提案した.

アルゴリズム kNN-OFDR:

Step 1: $D_l^2(\boldsymbol{q}, \boldsymbol{x}_i)$ をすべての i について計算し，$D_l^2(\boldsymbol{q}, \boldsymbol{x}_i)$ が小さい順に i をソートし直す．$\epsilon = \infty, i = 1$ とする.

Step 2: $D_l^2(\boldsymbol{q}, \boldsymbol{x}_i) > \epsilon$ なら選ばれたデータを出力して終了．$D_l^2(\boldsymbol{q}, \boldsymbol{x}_i) \leq \epsilon$ なら $D^2(\boldsymbol{q}, \boldsymbol{x}_i)$ を計算し，$D^2(\boldsymbol{q}, \boldsymbol{x}_i) \leq \epsilon$ ならその i を選び，$D^2(\boldsymbol{q}, \boldsymbol{x}_i) > \epsilon$ ならその i は捨てて Step 5 へ.

Step 3: 選ばれたデータの個数が k 未満なら Step 5 へ．k 以上なら k 番目に小さい $D^2(\boldsymbol{q}, \boldsymbol{x}_i)$ を ϵ として Step 4 へ.

Step 4: 選ばれたデータのうちで $D^2(\boldsymbol{q}, \boldsymbol{x}_i) > \epsilon$ となるものは捨てる.

Step 5: $i < m$ ならば $i = i + 1$ と増やして Step 2 へ．$i = m$ ならば選ばれたデータを出力して終了.

kNN-FDR よりも kNN-OFDR の方が高速 [17] なので，本章では次元削減によるフィルタリングには kNN-OFDR を用いることにする．クエリに近い画像を 1 個だけ選ぶ場合 (NN 検索) には，このアルゴリズムで k を 1 とおけばよいが，少し簡略化でき，次 (NN-OFDR: NN search with optimal filtering by dimensionality reduction) のようになる.

アルゴリズム NN-OFDR:

Step 1: $D_l^2(\boldsymbol{q}, \boldsymbol{x}_i)$ をすべての i について計算し，$D_l^2(\boldsymbol{q}, \boldsymbol{x}_i)$ が小さい順に i をソートし直す．$\epsilon = \infty, i = 1$ とする．

Step 2: $D_l^2(\boldsymbol{q}, \boldsymbol{x}_i) > \epsilon$ なら選ばれたデータを出力して終了．$D_l^2(\boldsymbol{q}, \boldsymbol{x}_i) \leq \epsilon$ なら $D^2(\boldsymbol{q}, \boldsymbol{x}_i)$ を計算し，$D^2(\boldsymbol{q}, \boldsymbol{x}_i) > \epsilon$ なら Step 3 へ．$D(\boldsymbol{q}, \boldsymbol{x}_i)^2 \leq \epsilon$ ならばその i を選び，$\epsilon = D^2(\boldsymbol{q}, \boldsymbol{x}_i)$ として Step 3 へ．

Step 3: $i < m$ ならば $i = i + 1$ と増やして Step 2 へ．$i = m$ ならば選ばれたデータを出力して終了．

以上のアルゴリズムのようにすると，単純に元の次元ですべての距離を計算する方法よりも計算量を削減できる．

7.3 k 平均法によるクラスタリング

本章では以上の次元削減によるフィルタリングと組み合わせて，クラスタリングによるフィルタリングも使う．クラスタリングには k 平均法を使うことにする．参考のため k 平均法のアルゴリズムも書いておく．第 i データベース画像のヒストグラムを $\boldsymbol{h}_i$ とし，それを式 (7.3) で直交変換したベクトルを $\boldsymbol{x}_i$ とする．クラスタの数を k 個とし，第 j クラスタの代表点を $\boldsymbol{r}_j$ とする．

アルゴリズム k-means:

Step 1: $\xi = 0$ とする．代表点 $\boldsymbol{r}_j^{(\xi)} (j = 1, ..., k)$ を適当に設定する．

Step 2: すべてのデータ $\boldsymbol{x}_i$ について自分に最も近い代表点 $\boldsymbol{r}_j^{(\xi)}$ を求める．

Step 3: すべての j について $\boldsymbol{r}_j^{(\xi+1)} = \sum_{i \in S_j} \boldsymbol{x}_i / \sum_{i \in S_j} 1$ を計算する．ここで S_j は $\boldsymbol{r}_j$ に近い $\boldsymbol{x}_i$ の集合を表す．

Step 4: すべての j で $\boldsymbol{r}_j^{(\xi+1)} = \boldsymbol{r}_j^{(\xi)}$ なら終了．そうでなければ $\boldsymbol{r}_j^{(\xi)} = \boldsymbol{r}_j^{(\xi+1)} (j = 1, ..., k)$ として Step 2 へ．

7.3.1 フィルタリングによる k 平均法の高速化

上のアルゴリズム k-means のなかの Step 2 は NN 検索であるから上記のアルゴリズム NN-OFDR を使えば高速化できる．これをアルゴリズム kM-OFDR(k-means with NN-OFDR) と呼ぶことにする．更にここでは，Step 1 の代表点の初期設定にも低次元

を利用する方法も試みた．まず低次元で上記の k-means アルゴリズムによってデータをクラスタリングする．このときの初期代表点は乱数とする．この低次元でのクラスタリングによって得られる各データのクラスタへの所属に基づいて，各クラスタに所属するデータの重心を元の次元で求め，それを初期代表点として kM-OFDR アルゴリズムを実行する．この 2 段階のクラスタリング法を 2SkM-OFDR(2 Step kM-OFDR) アルゴリズムと呼ぶことにする．

7.3.2 クラスタリングの実験

例として，サイズ 150×150 の風景写真 1300 枚を 10 個のクラスタに分割する実験を，kM-OFDR と 2SkM-OFDR とで比較した．色数 n は 512 とした．図 7.1 にクラスタリングの平均計算時間を示す．横軸は低次元数 l である．縦軸は各次元で 100 回実験した平均の計算時間である．kM-OFDR と 2SkM-OFDR の優劣は低次元数 l によって変わるが，この例では $l = 100$ 辺りの 2SkM-OFDR が最も高速である．ちなみに，フィルタリングを使わずに元の次元（512 次元）で k-means を行ったときの平均計算時間は 102.2 秒であった．次に，図 7.2 に平均量子化誤差を示す．量子化誤差は

$$\sum_{j=1}^{10} \sqrt{\sum_{i \in S_j} \| \boldsymbol{x}_i - \boldsymbol{r}_j \|^2} \tag{7.5}$$

である．$\boldsymbol{x}_i$ や S_j はアルゴリズム k-means の中で用いたものと同じである．これが小さいほど良いクラスタリングである．図 7.2 において，kM-OFDR では各次元で初期代表点が共通なので平均量子化誤差は一定となり，フィルタリングを使わずに元の次元で k-means を行ったときの平均量子化誤差 87.9 と同じ値になった．2SkM-OFDR では，初期代表点を低次元でのクラスタリングで求めているので，kM-OFDR よりも平均量子化誤差が小さくなり，低次元数が大きくなると元の次元での k-means の平均量子化誤差に近づいている．量子化誤差によれば l が小さい 2SkM-OFDR が良いが，本章では，高速化を主目的としているので低次元数が $l = 100$ の 2SkM-OFDR を使うことにする．$l = 100$ の 2SkM-OFDR は量子化誤差は大きいが，本章の最終目的は kNN 検索であり，量子化誤差が大きくても後述するフィルタリングで回復される．

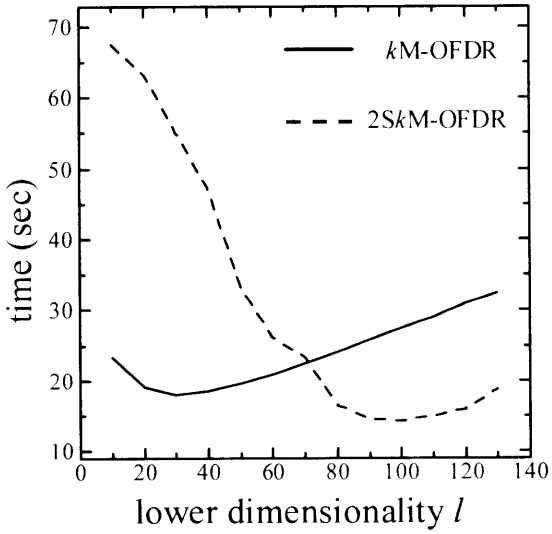

図 7.1: クラスタリングの平均計算時間

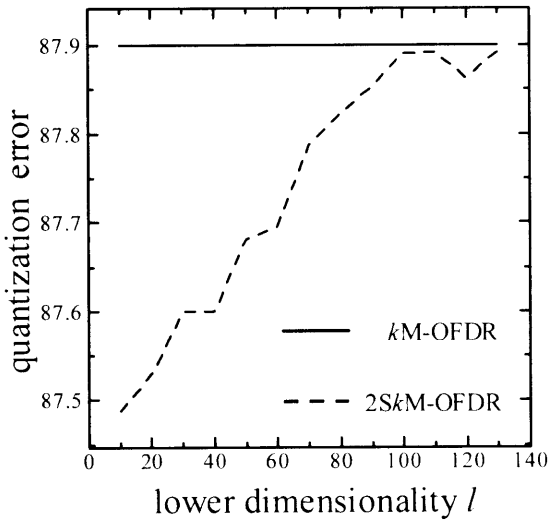

図 7.2: 平均量子化誤差

7.4 次元削減とクラスタリングとに基づくフィルタリングによるkNN検索の高速化

kNN検索の高速化に，次元削減と共にクラスタリングも用いる方法を考える．クラスタリングの利用においては，三角不等式によるフィルタリングを使う．Fukunagaら[19]はクラスタリングに基づく分枝限定法を提案しているが，そのアルゴリズムはクラスタリングだけを用いる方法としては高速であるが，次元削減とは組合せ難い．そこでここでは次元削減との適合性がよい方法を提案する．

その前に参考のためFukunagaら[19]のアルゴリズムを記しておく．クエリを$\boldsymbol{q}$，クラスタの代表点を$\boldsymbol{r}_j$，クラスタjに属すi番目のデータを$\boldsymbol{x}_{ij}$とする．クラスタの数をNとする．すべてのjについてクラスタの半径$D_j^{max} = \max_i D(\boldsymbol{r}_j, \boldsymbol{x}_{ij})$を予め計算しておく[1]．Fukunagaら[19]のアルゴリズムは階層的なクラスタリングに基づいて記述されているが，クラスタリングが1段だけの場合には次(kNN-OFC: kNN search with optimal filtering by clustering)のように書ける．

アルゴリズム kNN-OFC:

Step 1: $D(\boldsymbol{q}, \boldsymbol{r}_j)$をすべての$j$について計算し，$D(\boldsymbol{q}, \boldsymbol{r}_j)$が小さい順に$j$をソートし直す．$\epsilon = \infty, j = 1$とする．

Step 2: $D(\boldsymbol{q}, \boldsymbol{r}_j) > \epsilon + D_j^{max}$ならStep 5へ．$D(\boldsymbol{q}, \boldsymbol{r}_j) \leq \epsilon + D_j^{max}$ならStep 3へ．

Step 3: $D(\boldsymbol{q}, \boldsymbol{r}_j) > \epsilon + D(\boldsymbol{r}_j, \boldsymbol{x}_{ij})$ならそのデータは捨てる．$D(\boldsymbol{q}, \boldsymbol{r}_j) \leq \epsilon + D(\boldsymbol{r}_j, \boldsymbol{x}_{ij})$なら$D(\boldsymbol{q}, \boldsymbol{x}_{ij})$を計算し，$D(\boldsymbol{q}, \boldsymbol{x}_{ij}) > \epsilon$ならそのデータは捨て，$D(\boldsymbol{q}, \boldsymbol{x}_{ij}) \leq \epsilon$ならそのデータを選ぶ．

Step 4: 選ばれたデータの数がk以上になったらk番目に小さい$D(\boldsymbol{q}, \boldsymbol{x}_{ij})$を新たな$\epsilon$として$D(\boldsymbol{q}, \boldsymbol{x}_{ij}) > \epsilon$となるデータを捨てる．

Step 5: $j = N$なら選ばれたデータを出力して終了．$j < N$なら$j = j + 1$と増やしてStep 2へ．

これがFukunagaらのアルゴリズムであるが，これは次元削減とは組合せ難いので，ここでは次のアルゴリズム(kNN-FC: kNN search with filtering by clustering)を使うことにする．すべてのjについて$D_j^{max} = \max_i D(\boldsymbol{r}_j, \boldsymbol{x}_{ij})$を予め計算しておく．

[1]三角不等式を利用するため，データ間の距離はユークリッド距離$D(\cdot, \cdot)$であることに注意．

アルゴリズム kNN-FC:

Step 1: $D(\boldsymbol{q}, \boldsymbol{r}_j)$ をすべての j について計算し，これが最も小さい j について $D(\boldsymbol{q}, \boldsymbol{x}_{ij})$ が小さい順に k 個の i を選ぶ.

Step 2: 選ばれた i すべてについて $D(\boldsymbol{q}, \boldsymbol{x}_{ij})$ を計算し，そのうちの最大値を求める.

Step 3: その最大値を ϵ として，$D(\boldsymbol{q}, \boldsymbol{r}_j) \leq \epsilon + D_j^{max}$ を満たすクラスタ j すべてのなかから $D(\boldsymbol{q}, \boldsymbol{r}_j) \leq \epsilon + D(\boldsymbol{r}_j, \boldsymbol{x}_{ij})$ を満たすデータ i を選び出す.

Step 4: 選ばれたデータのなかから $D(\boldsymbol{q}, \boldsymbol{x}_{ij})$ が小さい順に k 個のデータを選んで出力する.

アルゴリズム kNN-FC と kNN-OFC との関係は前節の kNN-FDR と kNN-OFDR との関係と良く似ている．すなわち，kNN-FDR と kNN-FC では ϵ が固定であるのに対し，kNN-OFDR や kNN-OFC では逐次に逓減される．kNN-FDR や kNN-OFDR での次元削減による不等式を三角不等式に変えたものがそれぞれ kNN-FC と kNN-OFC であるとも言える．次元削減だけを使うときには kNN-OFDR が kNN-FDR よりも高速であるように，クラスタリングだけを使う場合には kNN-FC よりも kNN-OFC の方が速いと思われる．しかし，kNN-OFC は次元削減とは組合せ難く，kNN-OFDR と組合せた kNN-FC は kNN-OFC よりも速くなる可能性があり，実際それは下記の実験で確認される.

アルゴリズム kNN-FC では Step 2 以外の 3 つの step すべてにおいて次元削減によるフィルタリングを利用することができ，結局以下のアルゴリズム kNN-FDRC (kNN search with filtering by dimensionality reduction and clustering) が得られる．すべての j について $D_j^{max} = \max_i D(\boldsymbol{r}_j, \boldsymbol{x}_{ij})$ を予め計算しておく.

アルゴリズム kNN-FDRC:

Step 1: NN-OFDR により $D^2(\boldsymbol{q}, \boldsymbol{r}_j)$ が最も小さい j を選ぶ.

Step 2: 選ばれた j について kNN-OFDR により $D^2(\boldsymbol{q}, \boldsymbol{x}_{ij})$ が小さい順に k 個の i を選び，そのうちの $D^2(\boldsymbol{q}, \boldsymbol{x}_{ij})$ の最大値を ϵ とする.

Step 3: $D(\boldsymbol{q}, \boldsymbol{r}_j) \leq \sqrt{\epsilon} + D_j^{max}$ を満たす j をすべて選ぶ.

Step 4: 選ばれた j について $D(\boldsymbol{q}, \boldsymbol{r}_j) \leq \sqrt{\epsilon} + D(\boldsymbol{r}_j, \boldsymbol{x}_{ij})$ を満たす i をすべて選ぶ.

Step 5: 選ばれた i について kNN-OFDR により $D^2(\boldsymbol{q}, \boldsymbol{x}_{ij})$ が小さい順に k 個の i を出力する.

7.5 kNN 検索の実験

例として 7.3.2 節の風景写真 1300 枚を用いて kNN 検索を行った．色数 n は 512 とした．$k = 10$，すなわちクエリに近い画像 10 枚を検索した．まず最初に，フィルタリングを使わずに全探索で kNN 検索をした場合の検索時間は 0.32 秒であり，Fukunaga らの方法 [19] では 0.16 秒であった．次に，次元削減だけによる 7.2 節の検索法 kNN-OFDR と，次元削減とクラスタリングを組合せた 7.4 節の kNN-FDRC の検索時間を調べた．kNN-FDRC では，クラスタリングは予めオフラインで済んでいるものとし，検索にかかる時間だけを測った．図 7.3 に kNN-OFDR と kNN-FDRC の検索時間を示す．kNN-FDRC でのクラスタ数は 10 個とした．図 7.3 の横軸は低次元の次元数 l であり，縦軸はクエリ 1 枚あたりの平均検索時間である．すべての l において，クラスタリングを併用することによって検索時間が短縮されている．l が非常に小さいところでは，短縮率はそれほどでもないが，l を小さくし過ぎても悪いので，通常は小さ過ぎない程度の l を用いるのが安全である．

図 7.3 では，kNN-FDRC のクラスタリングに要する時間は除いているが，データベースが更新されたらクラスタリングをやり直す必要がある．したがって，データベースが頻繁に更新されるような状況では，クラスタリングの時間も考慮する必要がある．上記の kNN-FDRC の実験での 2SkM-OFDR では，低次元での k 平均法は平均 28 回の反復で収束し，元の次元の kM-OFDR は高々2 回で収束した．よって，ほとんどの時間は低次元でのクラスタリングに費やされている．データベースが更新される場合を考えると，1 部のデータが更新される場合がほとんどであろうから，更新前の代表点を初期値とすれば，k 平均法は 2.3 回程度で収束するものと思われる．したがって，図 7.1 での $l = 100$ の 2SkM-OFDR は 15 秒ほどかかっているが，データベース更新時では約 10 分の 1 の 1.5 秒ほどで済むものと思われる．$l = 100$ での kNN-OFDR と kNN-FDRC の図 7.3 の検索時間の差は約 0.03 秒であるから，データベースの更新頻度が 50 回の検索に 1 回以下であれば kNN-FDRC の方が kNN-OFDR よりも高速になると見積もられる．

また，上記の kNN-FDRC では，クラスタ数を 10 個とした．検索時間は，クラスタの個数によって図 7.4 のように変わる．クラスタが少な過ぎると各クラスタ内のデータが多くなり，クラスタ内の探索に時間がかかり，逆にクラスタが多過ぎるとクラスタを選択するのに時間がかかるようになる．よって適度のクラスタ数のときに検索時間が最も短くなり，この最適なクラスタ数は l が小さいほど小さいが，事前にこのような最適数を知ることは難しいと思われる．上記の実験でクラスタを 10 個にした

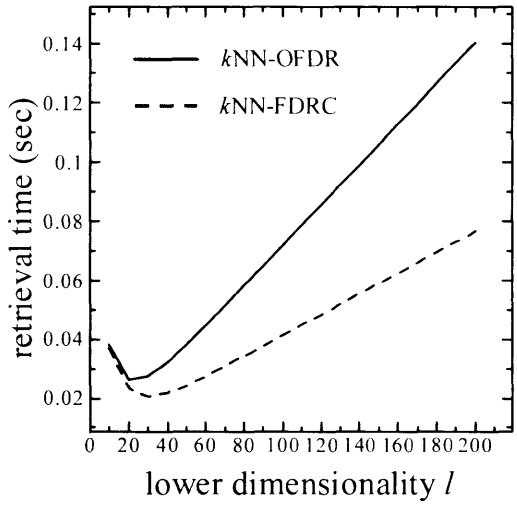

図 7.3: 平均検索時間

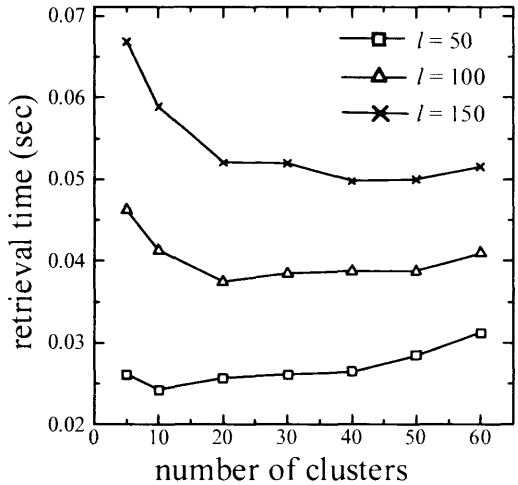

図 7.4: 次元数と検索時間の関係

のは，図 7.4 からわかるように l が小さい方が検索が速く，そのときに最適なクラスタ数は 10 個程度であるからである．

7.6 むすび

画像検索を高速化するためのフィルタリング法として，特徴ベクトルの次元削減に基づく距離の不等式とクラスタリングに基づく三角不等式とを利用する手法を提案し，実験により画像検索の時間が短縮できることを示した．提案手法における次元削減とクラスタリングを階層的に拡張することが今後の課題である．