Vol.2016-CVIM-203 No.24
2016/9/5

クエリの偏りを利用した学習型最近傍探索問題とその一解法

香川椋平† 和田俊和†

概要：最近傍探索は，類似画像検索や，BoF 特徴ベクトルの生成，画像間の特徴の対応付けなど，多岐にわたって用いられている．手法としては Hash 型，木探索型，森探索，グラフ探索，など様々な手法があり，問題としては，1−最近傍探索，k−最近傍探索，Radius 探索，などがある．近年では格納データの分布の次元が上昇することによって探索速度が低下するという問題を回避するために，近似最近傍探索がよく使われている．これに対し，本稿では，クエリ側の分布を利用した検索の高速化問題を提案する．これを「学習型最近傍探索問題」と呼ぶことにする．この分布は逐次与えられるクエリ自身から学習することもできるため，近似最近傍探索と近似のない最近傍探索を並列動作させればオンライン学習もできる．本稿では，その一歩として，クエリに基づいて k-d tree を再構築し，木探索のみで最近傍探索を行う高速化手法を提案する．学習した k-d tree を用いた最近傍探索時間と精度の比較を行い，その活用法を論じる．

キーワード：最近傍探索，k-d tree，プライオリティサーチ，学習型最近傍探索

Trainable nearest neighbor search exploiting query bias

Ryohei KAGAWA† Toshikazu WADA†

Abstract: Nearest neighbor search is widely used in many applications, such as, similar image search, BoF vector construction, keypoint matching between two images, and so on. Nearest neighbor search problem includes three major sub-problems, 1-nearest neighbor search, k- nearest neighbor search, and radius search. Also, the problem can be classified into two problems depending on their accuracy, approximate and accurate search problems. The algorithms can be classified depending on the data structures used for indexing; table, tree, forest, and graph. These well organized classification and criteria imply that this research field is well matured. In this report, however, we further propose a new problem in this field for accelerating the search speed exploiting the query vector distribution. We call this "training problem of nearest neighbor search". This problem is to accelerate the nearest neighbor search by providing query vector distribution. One solution for solving problem is to run both accurate nearest neighbor search and trainable approximate search. That is, if their answers don't coincide, the approximate search algorithm learns the correct answer by accurate search algorithm. By iterating this process, the approximate search gradually approaches to the accurate search while keeping the speed. This report examines one example of this framework based on k-d tree algorithm and confirmed the effectiveness.

Keywords: nearest neighbor search , k-d Tree , prioritysSearch , nearest neighbor search

1. はじめに

最近傍探索とは，あらかじめデータベースに格納されたベクトル集合X_0の中から，クエリベクトル$\boldsymbol{q}$との距離が近いベクトルを見つけ出す探索計算の総称であり，以下の問題を含む．

- 最近傍ベクトル$\boldsymbol{NN}_1(\boldsymbol{q};X_0)$を一つだけ見つける 1-NN サーチ．

$$\boldsymbol{NN}_1(\boldsymbol{q};X_0)=\underset{\boldsymbol{x}\in X_0}{\operatorname{argmin}}\, d(\boldsymbol{q},\boldsymbol{x}),$$

  但し，$d(\boldsymbol{q},\boldsymbol{x})$は$\boldsymbol{q}$と$\boldsymbol{x}$の間の距離である．
- 近傍ベクトルの集合$KNN(k,\boldsymbol{q};X_0)$を 1 番目から k 番目まで求める k-NN サーチ

$$KNN(k,\boldsymbol{q};X_0)=\{\boldsymbol{NN}_1(\boldsymbol{q};X_0),\cdots,\boldsymbol{NN}_k(\boldsymbol{q};X_{k-1})\},$$

  但し，$\boldsymbol{NN}_m(\boldsymbol{q};X_{m-1})=\underset{\boldsymbol{x}\in X_{m-1}}{\operatorname{argmin}}\, d(\boldsymbol{q},\boldsymbol{x})$
  および，$X_m=X_{m-1}\setminus\{\boldsymbol{NN}_m(\boldsymbol{q};X_{m-1})\}$とする．
- クエリを中心として指定した半径内に存在するベクトル集合$RNN(d,\boldsymbol{q};X_0)$を求める radius サーチ

$$RNN(d,\boldsymbol{q};X_0)=\{\boldsymbol{x}\in X_0 | d(\boldsymbol{q},\boldsymbol{x})\le d\}$$

また，正確な最近傍を求める場合，X_0の分布の次元が上昇すると，全探索と同程度に効率の悪い計算になってしまうという「次元の呪縛」と呼ばれる現象が発生する．これを回避するために，近年では厳密ではない近似的な最近傍探索を求める問題として定式化されることが多くなっている．

これらの問題を解くためのアルゴリズムは，使用するデータ構造によって分類することができ，線形構造（表），木構造，グラフ構造，森，などを用いたものに分けられる．例えば，近似最近傍探索では表探索（hash 法）を用いたものや森（random forest）を用いたもの，木構造を用いたものには，k-d tree や V-P tree, R-tree, S -tree，k-means tree (vocabulary tree)など様々なものがある．

このように，最近傍探索は問題と手法がよく整理された研究分野であるが，未だに検討されてこなかった問題があ

† 和歌山大学
Wakayama University

る．それは，データの分布ではなく，クエリの分布を利用するという観点である．

例えば，二分探索木の場合，構造上の平衡性を最適性の基準とすると，与えられるクエリが一様に分布することを強制することになる．これに対して，クエリが持つ統計的偏りを考慮する場合は，探索に要するコストの期待値を最小化するようにするため，構造上の平衡性は崩れるが，実際に探索を行ったときのコストの最適性は保証される．

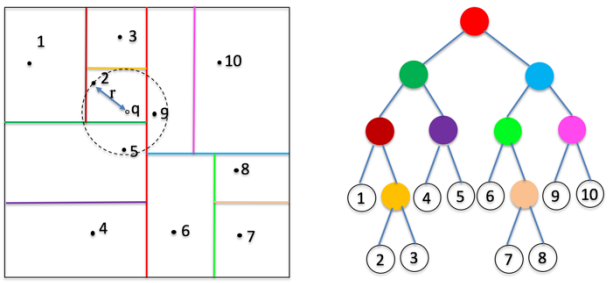
これと同様に，最近傍探索のデータ構造やアルゴリズムに関する過去の研究はクエリの統計的偏りを考慮しないものが殆どであり，実質的な意味での最適性を保証してはいない．このため，本報告では，クエリの偏りを考慮した最近傍探索アルゴリズムについて検討する．

クエリの偏りを利用する場合，探索コストの期待値を最小化するという観点，事前に与えられたクエリと似たクエリに対して，効率よく探索を行うという観点，の二つがある．本報告では，主に後者の観点に立ち，1-NN の近似最近傍探索の効率化を図る方法について検討する．

2. 関連研究

本研究では，木構造を用いた最近傍探索アルゴリズム，特に k-d tree を用いた最近傍探索アルゴリズムを高速化するため，この観点から見た関連研究について述べる．

木構造を用いた最近傍探索の代表例は，Bentley の考案した k-d tree[1]である．これは，与えられたデータ点で空間を Box 状に区切る空間分割に対応する木構造であり，データは木の全ノードに格納される．しかし，現在 k-d tree と呼ばれるデータ構造[2][3]は，当初の k-d tree とは異なっており，全てのデータは別々の葉に格納されるようになっている．これは，図 1 に示すようにデータの集合を Box に分割していき，最終的に 1 つの Box が 1 つのデータを含むようになるまで分割することに対応したデータ構造である．



(a) (b)

図 1. 空間分割と k-d tree の関係

この k-d tree には，空間分割を行う軸（次元）とその閾値が節に格納されており，クエリが与えられたときに以下のようにして最近傍探索を行う．

1. 節に格納された軸の値が閾値に対して小さいか大きいかで，木枝を辿り，葉に到達する「木探索」．
2. 到達した葉に格納されたデータを暫定的な最近傍とし，それよりも近距離に存在する可能性のある Box について優先度付きキューを用いて探索する「priority search」．（この際に調査する Box は，図 1 (a)のように，クエリを中心とし，葉に格納されたデータとの距離rを半径とする超球と交わる Box の全てである．）

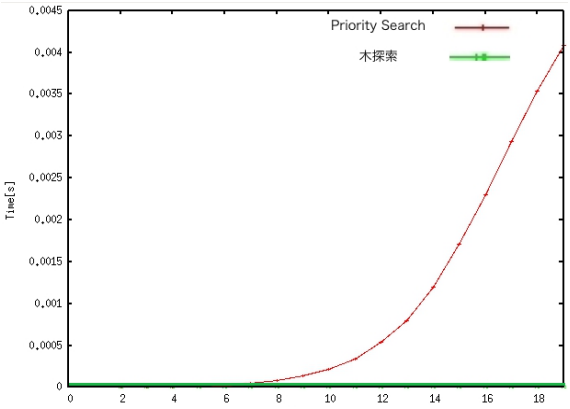
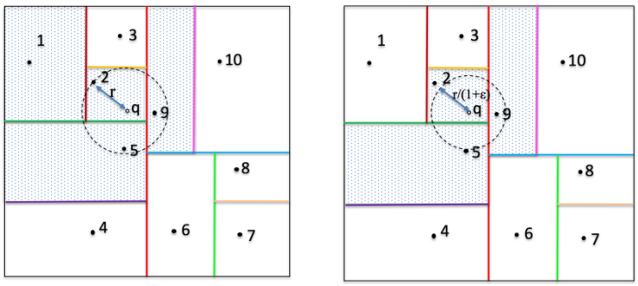


図 2. 木探索と Priority Search の探索時間の比較[7]より

このアルゴリズムは，許容解を求めておきそのコストを緩和解によって求められる下界値が上回った場合，探索を打ち切るという，分枝限定法あるいは A*アルゴリズムの一種であることが知られている．これは，木構造だけでなく，どのようなデータ構造を用いた場合でも，当てはまることが知られている．

実際に，木探索と priority search の消費時間について調査した報告があるが，図 2 に示すように，priority search は木探索に比べて大幅に計算時間がかかり，この傾向は次元の増加に伴い顕著になっていく．[4]

このことは，仮に priority search が省略できれば大幅な高速化が達成できることを意味している．しかし，木探索だけではおおまかな近似最近傍探索しか行えず，探索精度の調整は行えない．このため，k-d tree の場合は，クエリと葉に格納されたデータとの距離rに一定の比率$1/(1+\varepsilon)$をかけて図 3 に示すように超球の半径を短くし，調査すべき Box の個数を減らすという手法が用いられてきた．[3]



(a) $\varepsilon = 0$ の場合 (b) $\varepsilon \geqq 0$ の場合

図 3. k-d tree における priority search の範囲

これ以外にも，ハッシュを用いる方法[5][6][7]やランダムフォレストを用いる方法[8]などでも近似最近傍探索は盛んに研究されているが，単一の木探索のみで，ある程度正確な答えが出せれば，リアルタイム性が要求されるアプリケーションの高速化などに有用である．

3. 学習型最近傍探索問題

本稿では，登録データ分布の偏りのみではなく，クエリ側の偏りも用いた問題を提案する．これを行う観点としては，前述の通り，

1）探索コストあるいは探索時の誤差の期待値を最小化するという観点
2）事前に与えられたのと似たクエリに対して，手間のかからない方法で精度のよい探索を行うという観点

の二つがある．

前者は，頻度の高いデータに対しては，高速・高精度な処理が行え，頻度の低いデータに対しては比較的低速・低精度な処理でも構わないという立場に立ち，クエリ集合全体に対する速度または精度の期待値を向上させるという観点である．例えばカスケード型 ADA Boosting による顔検出器に与えられるデータの大半が顔データでは無く，明らかに顔ではないデータを early reject して，顔の可能性があるものについては，手間をかけて識別するといったデータの種類によって処理の手間を変えることと似ている．

後者はクエリをキャッシングしておき，似たクエリに対しては過去の経験に基づき即座に答えを出すという手法を指す．本報告では，この観点から priority search を行わない簡便な探索アルゴリズムの開発を行う．

本報告では，後者の観点から，一旦探索したデータを木構造に組み込むことで，priority search を用いずに探索の精度を向上させる方法について提案する．

これは，図 4 に示すように，クエリ$\boldsymbol{q}$に対して k-d tree の 2 段階のサーチ（木探索と，priority search）を行い，それぞれの段階で暫定最近傍$\boldsymbol{NN'}$と確定最近傍$\boldsymbol{NN}$を求め，

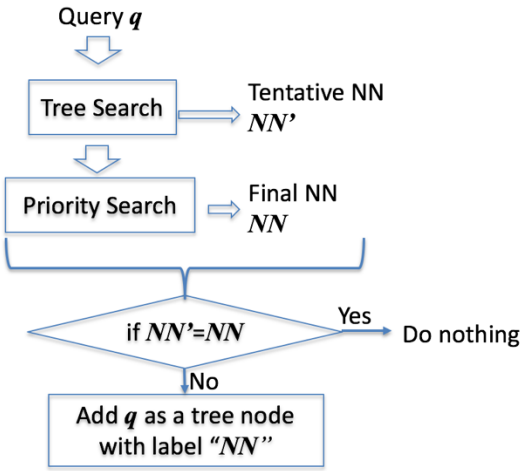
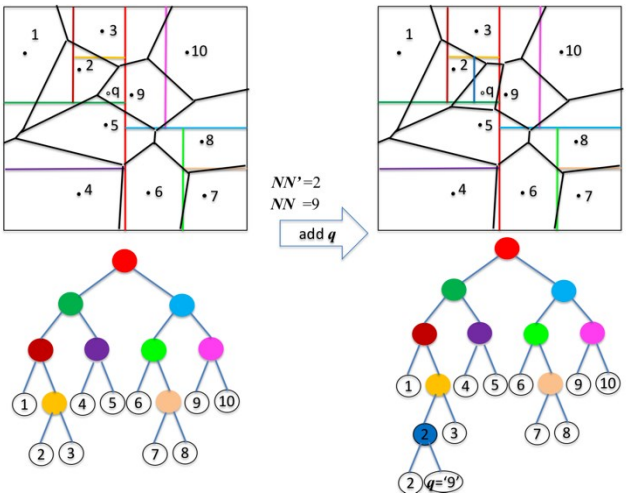


図 4. クエリによる k-d tree の改良



(a) $\boldsymbol{q}$ を付加する前　　(b) $\boldsymbol{q}$ を付加した後

図 5. クエリの付加による k-d tree の改良の効果

これらが一致しなかった場合には，クエリ$\boldsymbol{q}$をデータとして k-d tree に追加する．但し，この$\boldsymbol{q}$にはラベル"$\boldsymbol{NN}$"をつけておき，木探索でこのデータが見つかった場合には$\boldsymbol{NN}$を最近傍であると判断するようにする．

この様子を図 5 に示す．同図(a)では，木探索の結果は 2 となり，最近傍は 9 となるケースである．この場合，クエリ$\boldsymbol{q}$を k-d tree に付加する．このクエリに類似したデータが再びクエリとして与えられた場合，木探索の結果は 9 となる．このデータの付加により，2,5,3 の Voronoi 領域は狭くなり，結果的に誤りが生じるケースも生じるが，その誤りを修正するために新たなクエリが付加されるため，結果的には誤りは修正されていく．

このようにして大量のクエリを与えて k-d tree にクエリを付加し，修正しておけば，木探索の結果が priority search の結果と一致するようになる．しかし，多次元データになった場合，境界の面積が広がるため，このような配置は行えない．このような問題点がある場合でも，実際に与えられるクエリと，学習時に与えるクエリがほぼ同じ分布に従う場合には，木探索のみで，ほぼ正しいデータの探索が行えるものと考えられる．この手法は図 4 のように木探索と priority search を共に実行して結果を比較するようにしておき，実際にクエリを与えながらオンライン学習をすれば，最近傍探索を行いながら木探索の精度を上げることもできる．

4. 実験

提案した手法で学習した k-d tree の有効性を確認するために，

- 乱数で発生させた 2 次元データを用いて，付加されるクエリが Voronoi 領域のどこに分布するかの確認

- SIFT 特徴量データベースから抽出したデータを用いた既存手法との探索時間・精度の比較

に関する実験を行った結果について述べる．

以下の実験では，CPU：i7-2600(3.4GHz)，メモリ：16GB を搭載した計算機を使用して行った．

4.1 付加データの分布

提案手法は，前述の通り，「木探索と$\varepsilon = 0$とした priority search を実行し，双方の結果が異なる場合，正解のラベルを付けたクエリを元のデータに付加して，k-d tree を再構築する」というものである．この処理を繰り返すことによって付加されるデータの分布が Voronoi 領域とどのような関係にあるのかを確認する．

【実験の条件】
登録データ数は 10，トレーニング用クエリ数は 100000 とし，それぞれ初期値の異なる一様乱数で発生させた．
【実験の手順】
I. 登録データから k-d tree を構築する
II. トレーニング用クエリを逐次与え，必要に応じて登録データにクエリを付加し k-d tree を再構築する．
III. 全トレーニング用クエリに対して II の処理を行う．

図 6 にこの結果を示す．尚，この図では，k-d tree に登録されているデータをラベルごとに色を変えて表示している．また図 7 に参考として，同じ登録データに対する Voronoi 領域を示す．

図 6，図 7 より，学習したデータは Voronoi 領域の境界付近に多く分布しており，境界の表裏で異なるラベルを持っていることが確認できる．
これは，Voronoi 領域の境界上に母点となる登録データと同一ラベルの付加データを密に与えれば，木探索の結果と priority search の結果が一致するためである．この結果から，提案手法によって上記の分布に近い結果が得られていることが確認できる．しかし，境界近く以外にも Voronoi 領域内側に分布している付加データが存在する．これらのデータは，トレーニングの初期段階で与えられたクエリが付加されたものであると考えられるが，基本的には不要なものである．これらはトレーニング用クエリを与える順序を最適化すれば除去することが可能なものである．

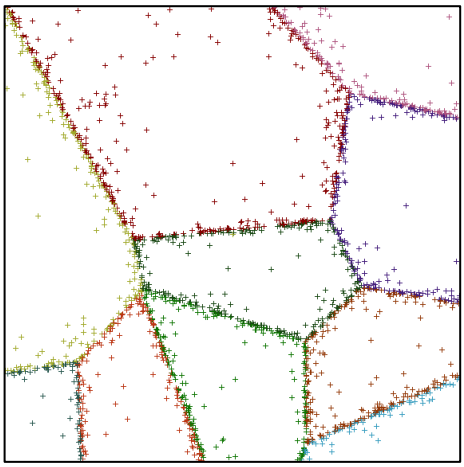


図 6　トレーニング済み k-d tree のデータ分布

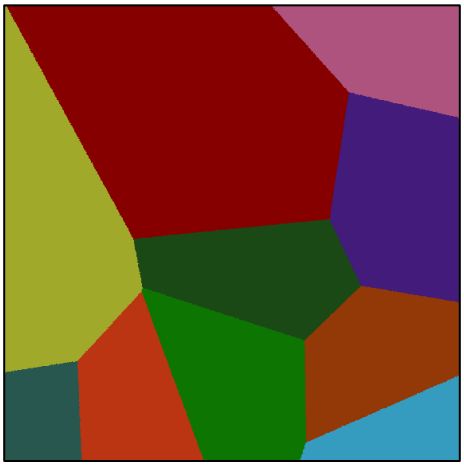


図 7　図 6 の登録データに対する Voronoi 領域

以下，この Voronoi 境界から離れた位置に分布するデータの除去を行う方法について検討する．トレーニングを行った結果，学習データは Voronoi 領域の境界付近に高い密度で分布し，境界から離れた部分では疎らに分布することが分かっている．この性質を利用し，トレーニングデータの並べ替えを行う方法を以下に述べる．

学習 k-d tree に登録されたデータそれぞれについて k-最近傍探索を行い，探索されたデータの分散を求める．この分散が小さいほど周囲のデータの密度が高いと言える．つまり，そのデータは Voronoi 領域の境界付近に分布していると見なすことができる．実際に k=10 でそれぞれの学習データについて分散を計算した．学習したデータを分散で昇順に並び替え，その順に初期状態の k-d tree に再トレーニングを行った．その結果を図 8 に示す．

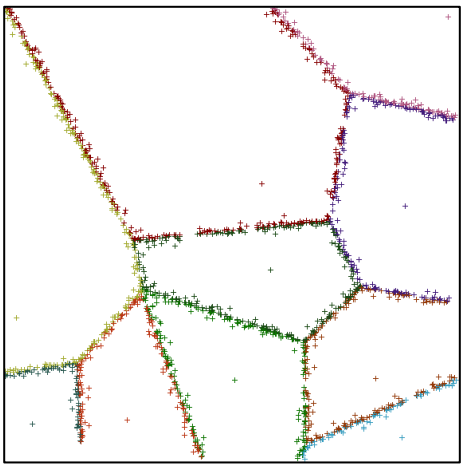


図 8　再トレーニング後の k-d tree 付加データの分布

図 8 から Voronoi 境界から離れた位置に分布しているデータの除去ができることが分かる.

4.2 探索時間・精度比較

k-d tree を用いた既存の最近傍探索である Priority Search を行う最近傍探索[2][3]と学習した k-d tree を用いた最近傍探索の探索時間・精度の比較実験を行った.

【実験の条件】

探索の精度の評価には，下記に示す誤差比を用いた.

$$\text{誤差比} = \frac{d(\textit{クエリ，探索結果データ})}{d(\textit{クエリ，最近傍データ})}$$

誤差比は 1 に近いほど精度が高く，誤差比が 1 であるときは完全な最近傍探索である.

登録データ数は 1000, トレーニング用クエリ数は 100000 検証用クエリ数は 100000 とし，すべて SIFT 特徴量データベース[9]の ANN_SIFT1B Learning set から抽出したデータより, 検証時に指定した次元数分を使用した. 登録データ，トレーニング用クエリ, 検証用クエリはすべて異なる SIFT 特徴から抽出したものを使用した.

Priority Search を行う最近傍探索の ε は 0 および 0.18 で検証を行った．0.18 は 5 次元時に学習した k-d tree を用いた最近傍探索と Priority Search の誤差比が同程度になるように調整した値である.

【実験の手順】

I. 登録データから k-d tree を構築する
II. トレーニング用クエリを逐次与え，必要に応じて登録データにクエリを付加し k-d tree を再構築する.
III. 全トレーニング用クエリに対して II の処理を行う.
IV. 検証用クエリを逐次与え，Priority Search を行う最近傍探索 (ε =0 および 0.18), 学習した k-d tree を用いた最近傍探索の探索時間と精度を計測する.

すべての検証用クエリを与え終わった後，検証用クエリ 1 つあたりの平均をとり，比較を行った.

図 9 にて各手法を用いた探索における探索時間について，横軸を登録データ，トレーニング用クエリ，検証用クエリの次元数．縦軸を探索精度として表示する.

図 10 では各手法を用いた探索における探索精度について横軸を登録データ，トレーニング用クエリ，検証用クエリの次元数．縦軸を実行時間として表示する.

図 9, 図 10 より, 誤差が同程度の場合でも Priority Search を行う最近傍探索よりも高速に探索を行うことができることがわかる.

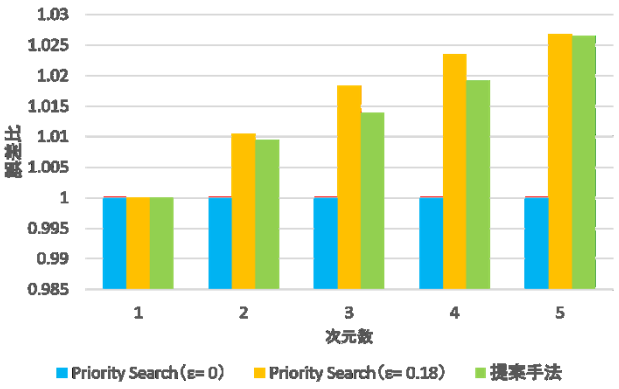


図 9　Priority Search と学習した k-d tree 探索の精度比較

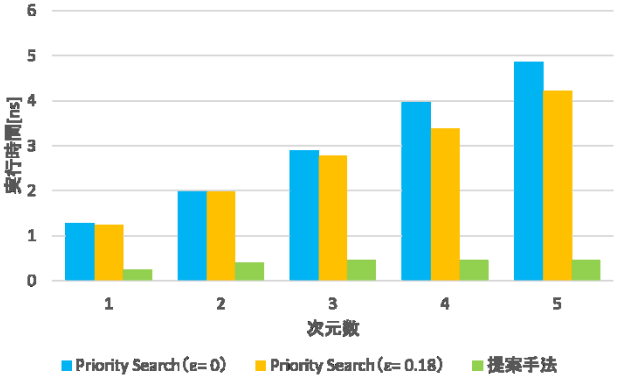


図 10 Priority Search と学習した k-d tree 探索の速度比較

5. おわりに

本稿では，クエリの偏りを考慮した最近傍探索アルゴリズムについて事前に与えられたクエリと似たクエリに対して効率よく探索を行うという観点から検討を行った． 1-NN の近似最近傍探索の効率化を図る方法について一旦探索したデータを木構造に組み込むことで，priority search を用いずに探索の精度を向上させる方法を提案した. そして, k-d tree を用いた既存の Priority Search を行う最近傍探索と学習した k-d tree を用いた最近傍探索の探索時間・精度の比較実験を行った．その結果，学習した k-d tree を用いた最近傍探索は Priority Search を行う最近傍探索と同程度の誤差比の場合でもより高速に探索が行うことができることを示した.

提案した手法は，トレーニングを行う手間はかかるが，平行実行することでオンライン上での学習もできる．これと比較実験から，この手法は以下の場合に適していると考えられる.

- 繰り返し最近傍探索が行われる
- リアルタイム性を求めるアプリケーション

今後の予定では，クエリの偏りを利用し探索コストの期待値を最小化するという観点からの手法について検討を行う．また，事前に与えられたクエリと似たクエリに対して効率よく探索を行うという観点も合わせた手法を検討する．

6. 参照文献

[1] J.L. Bentley, “Multidimensional binary search Trees used for associative searching,” Communication of the ACM, vol.18, No.9, pp.509-517, 1975.

[2] S. Arya, D.M. Mount, N. S. Netanyahu, R. Silverman and A. Wu, “An optimal algorithm for approximate nearest neighbor searching,” Journal of the ACM, Vol.45, pp.891-923, 1998.

[3] ANN: Library for Approximate Nearest Neighbor Searching
(http://www.cs.umd.edu/~mount/ANN)

[4] 大谷洋平，大池洋史，和田俊和．距離計算を行わない近似最近傍探索アルゴリズム: Grafting Trees. 電子情報通信学会技術研究報告 PRMU，Vol.112, No.441，pp.137-142，(2013).

[5] P. Indyk, R. Motwani, P. Raghavan, and S. Vempala. “Locality-preserving hashing in multidimensional spaces,” In Proceedings of the 29th ACM Symposium on Theory of Computing, pp. 618--625, 1997.

[6] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. “Locality-sensitive hashing scheme based on p-stable distributions,” In Proc. of the 20th ACM Symposium on Computational Geometry, pp. 253-262, 2004.

[7] Y. Weiss, A. Torralba, and R. Fergus, Spectral Hashing , in Proc. of NIPS2008 (Advances in Neural Information Processing Systems 21, , Curran Associates, Inc., eds. D. Koller and D. Schuurmans and Y. Bengio and L. Bottou, pp. 1753-1760, 2009

[8] Yi Lin and Yongho Jeon, “Random Forests and Adaptive Nearest Neighbors,” JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION, pp.101-474, 2002

[9] H. J. Laurent Amsaleg, “Datasets for approximate nearest neighbor search,” 7 2010. http://corpus-texmex.irisa.fr/