社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

隣接バケット探索を用いた近似最近傍探索手法の解析

武藤　大志† 　多田　匡志† 　岩村　雅一† 　黄瀬　浩一†

† 大阪府立大学大学院工学研究科　〒 599–8531 堺市中区学園町 1–1
E-mail: †{mutoh,tada}@m.cs.osakafu-u.ac.jp, †{masa,kise}@cs.osakafu-u.ac.jp

あらまし　近似最近傍探索は，クエリと最も距離が近い点を探索する最近傍探索の計算量，メモリ使用量を大幅に削減する手法である．近似最近傍探索において，メモリ使用量ををさらに減少させることが重要な課題である．本論文では，文献 [7], [8] で行われている隣接バケットを参照する近似最近傍手法のモデル化を行い，近似最近傍探索手法の代表的な手法である LSH よりもメモリ使用量を抑えて最近傍点を探索できることを実験と理論解析によって示す．

キーワード　近似最近傍探索, Locality Sensitive Hashing, 隣接バケット

Efficient Approximate Nearest Neighbor Search Based on Accessing Neighboring Buckets

Tomoyuki MUTO†, Masashi TADA†, Masakazu IWAMURA†, and Koichi KISE†

† Graduate School of Engineering, Osaka Prefecture University
1–1 Gakuencho, Naka, Sakai, 599–8531 Japan
E-mail: †{mutoh,tada}@m.cs.osakafu-u.ac.jp, †{masa,kise}@cs.osakafu-u.ac.jp

Abstract　Approximate nearest neighbor search is a technique which greatly reduces processing time and required amount of memory for nearest neighbor search. Further reduction of required amount of memory of the approximate nearest neighbor search is an important task. In this paper, we model a method to access neighboring buckets used in [7], [8], and reveal the method requires less amount of memory than LSH by an experiment and analysis.

Key words　Approximate nearest neighbor search, Locality Sensitive Hashing, Accessing Neighboring Buckets

1. ま え が き

近年，大規模なデータベースを用いて特定物体認識などを行う様々なアプリケーションが開発されている．このようなアプリケーションは蓄えられた大量のデータの中から類似のデータを探すことにより情報を処理するものであり，膨大なデータの中から処理時間 (時間計算量) を抑えて高速に類似例を検索することが求められる．それと同時に，メモリ使用量 (空間計算量) を削減することが，実用化する際に大きな役割を果たす．しかし一般に，検索の精度と，時間計算量，空間計算量の 3 者はトレードオフの関係にあり，精度を向上させようとすると時間計算量や空間計算量は大きくなってしまう．そこで，精度をなるべく下げることなく時間計算量や空間計算量を減少させることが重要な課題になる．

特定物体認識で用いられる SIFT [1] や PCA-SIFT [2] 等には最近傍探索が用いられる．最近傍探索は，ベクトルで表現されるデータの中から，クエリと最も距離が近いデータ (最近傍点) を探索するものである．高速な最近傍探索を実現するために，これまでに様々な改良手法が提案されているが，どの手法もデータ数や次元数に対して指数オーダーの時間計算量あるいは空間計算量を必要とする．

そのため，近似を用いて，最近傍探索と比べて時間計算量と空間計算量を削減することを目的とした近似最近傍探索という手法が考えられた．近似最近傍探索は，探索結果の誤りを許容することで，最近傍探索と比べて時間計算量と空間計算量を大幅に削減することができる．

近似最近傍探索の代表的な手法として，木構造を用いる Approximate Nearest Neighbor(ANN) [3] やハッシュを用いる Locality Sensitive Hashing(LSH) [4]〜[6] が知られている．このうち LSH は Indyk らによって，近似最近傍探索に要する空間計算量と時間計算量について解析的に考察されており注目を集めている．LSH はハッシュ関数を用いて距離を計算する点を限定しているので，最近傍探索と比べて時間計算量を大幅に削減することができる．一方，近似を行うが故に最近傍点が距離を計算する対象に含まれず，近似最近傍点と最近傍点が一致しないことがある．

LSH では，最近傍点と近似最近傍点が一致する確率を上げるために，ハッシュ関数を複数用いている．いずれかのハッシュ

関数で距離計算の対象になる点数はハッシュ関数が一つのときより増えるので，最近傍点を漏らす可能性が減る．しかし一方で，ハッシュ関数を増加させれば必要なハッシュテーブル数も増加するので空間計算量も大きくなってしまう．

では空間計算量を増やすことなく近似最近傍点と最近傍点が一致する確率を上げるにはどうしたらよいのだろうか．そもそも最近傍点はクエリの周辺にあるはずである．そのため，クエリの周辺の点を効率良く距離計算の対象とすることができれば近似最近傍点と最近傍点が一致する確率は上がるはずである．実際に野口らの手法 [7] や Principal Component Hashing(PCH) [8] では周辺の点を効率良く距離計算の対象とする工夫がされている．しかし，これまでこの方策の性能評価が十分に行われてこなかった．そこで本稿では，LSH 型の近似最近傍探索において前述した文献 [7], [8] ようなクエリの周辺の点を効率良く距離を計算する対象にする方策のモデル化を行い，性能を解析的に評価する．性能を評価するにあたり，これ以後，近似最近傍点と最近傍点が一致する確率を近似最近傍探索の精度とする．

2. Locality Sensitive Hashing

Locality Sensitive Hashing(LSH) [4]〜[6] はハッシュを用いた近似最近傍探索の代表的手法である．LSH(注1) は，複数のハッシュ関数を用いて距離を計算する対象になる点を求め，それらの点とクエリとの距離計算を行うことにより近似最近傍点を求めることができる．ここでは，LSH の中でもベクトル空間で用いることが出来る文献 [5] の LSH の概要について述べる．

LSH が近似最近傍点を求めることができるのは局所性に鋭敏な (Locality Sensitive) ハッシュ関数を用いているためである．局所性に鋭敏なハッシュ関数とは，距離が近い点同士は同じハッシュ値を取る確率が高く，距離が遠い点同士は同じハッシュ値を取る確率が小さいハッシュ関数である．

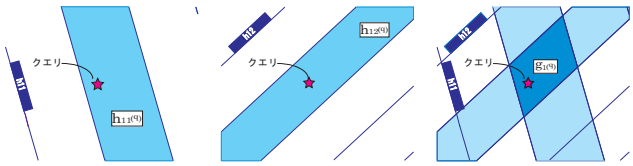
局所性に鋭敏なハッシュを実現するために，文献 [5] の LSH では次のようなハッシュ関数が用いられている．

$$h_{ji}(\boldsymbol{p}) = \left\lfloor \frac{\boldsymbol{a}_{ji} \cdot \boldsymbol{p} + b_{ji}}{w} \right\rfloor \quad (1)$$

ただし，$\boldsymbol{a}_{ji}$ は各次元の要素の値をガウス分布から独立にとってきた d 次元ベクトル，b_{ji} は区間 $[0, w]$ から一様に選ばれた実数である．添え字 i，j は後の議論で使用する．また，w はハッシュ幅である．LSH では $h_{ji}(q) = h_{ji}(p)$ となるような p が存在し得る空間を $\mathrm{S}^h_{ij}(q)$ とすると，この空間内に入った点を距離を計算する対象の点とする．図 5(a) の着色部分は $\mathrm{S}^h_{11}(q)$ を図示したものである．LSH はこのように，データ点が距離を計算する対象となる可能性のある空間を削減して近似をする．

ところが特徴空間が高次元の場合，$\mathrm{S}^h_{ij}(q)$ が大きくなってしまい，明らかに最近傍点になり得ない点も距離計算の対象とする効率の悪い探索になることがある．LSH はこの問題を緩和するためにハッシュ関数を複数個用いて近似近傍探索を行う．

(注1)：厳密に言えば，LSH は近似近傍探索の手法である．LSH で近似最近傍探索をするには，LSH で算出した近似近傍点に対して距離計算を行う．本稿では，この手法を LSH として用いる．



(a) $\mathrm{S}^h_{11}(q)$　(b) $\mathrm{S}^h_{12}(q)$　(c) $\mathrm{S}^h_{11}(q) \cap \mathrm{S}^h_{12}(q)$

図 1　g_j による距離計算対象範囲

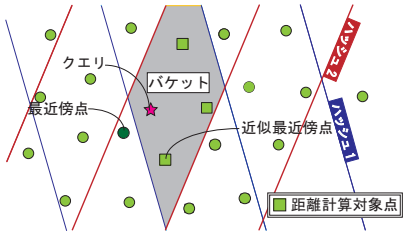


図 2　バケット $g_j(q)$

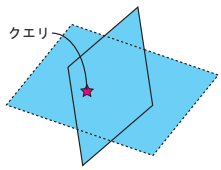


図 3　$S^g_i(q) \cup S^g_2(q)$

図 5(c) は $g_1 = \{h_{11}, h_{12}\}$ とした時の例で，g_1 を構成する h_{11}，h_{12} において $\mathrm{S}^h_{11}(q)$ と $\mathrm{S}^h_{12}(q)$ の両方に入った点のみを距離を計算する対象にする．これを一般化する．k 個のハッシュ関数 $h_{j1}, h_{j2}, \ldots, h_{jk}$ をランダムに選び，関数群 $g_j = \{h_{j1}, \ldots, h_{jk}\}$ を作る．このとき $g_j(q) = g_j(p)$，すなわち $^\forall i, h_{ji}(q) = h_{ji}(p)$ を満たすのみ距離を計算する対象とする．図 2 のように g_j において同じ値を取る空間，つまり $S^g_j(q) = \bigcap_{j=1}^{k} \mathrm{S}^h_{ji}(q)$ をバケットと呼ぶ．

さらに近似最近傍点の精度向上のために複数のハッシュ関数群を用いて，一度でもクエリと同じバケットに入った点を距離を計算する対象にする．このことを図 3 を例に説明する．ハッシュ関数群 g_1 と g_2 があったときこの中で近似近傍点になるのはバケット $S^g_1(q)$ 内の点とバケット $S^g_2(q)$ 内の点である．LSH は，このように関数群 g_j を L 個用いて $g_j (1 \leqq j \leqq L)$ において一度でもクエリと同じバケットに入った点と距離計算をして，距離が最小の点を近似最近傍点として返すことで近似最近傍探索を実現する．

3. 隣接バケット参照モデル

本節では，野口らの手法 [7] や Principal Component Hashing(PCH) [8] で用いられている LSH 型の近似最近傍探索を行う際に隣接するバケットを参照する方策をモデル化する．以降，隣接するバケットを参照する方策を Accessing Neighboring Buckets(ANB) と呼ぶ．

前述の通り，LSH を用いた近似最近傍探索の問題点は，g_j の数が少ない (L の値が小さい) と最近傍点が距離を計算する対象になる確率が低く，g_j の数が多い，(L の値が大きい) と時間計算量や空間計算量が大きくなってしまうということである．これは，LSH ではハッシュ関数をランダムに生成して用いるため，クエリ周辺の点を効率的に距離を計算する対象にできないからである．ANB はこの問題を解決する．

ANB でも式 (1) で示されるハッシュ関数を用いる．まず，

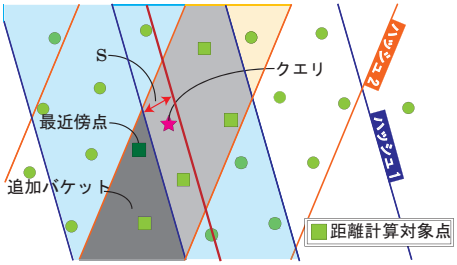


図 4　ANB の概要

ANB の目的を果たすためにどのような場合に最近傍点がクエリと異なるバケットに入り易いかを考える必要がある．クエリがハッシュ値の境界付近にある場合，ほぼ 1/2 の確率で最近傍点はクエリと異なるバケットに入る．このとき，その境界を挟んで隣のバケットに最近傍点が入ってしまう場合が考えられる．そこで図 4 のように閾値 s を設定し，クエリとハッシュ値境界の距離が s よりも近いときには，隣のバケットもクエリが入ったバケットと同様に扱うことにする．これを定式化する．h_{ji} に対して $h_{ji}(q) \pm 1$ となる隣接バケットをそれぞれ $\mathrm{S}_{ji}^{h^+}(q)$，$\mathrm{S}_{ji}^{h^-}(q)$ と置く．図 4 の場合を例にとると，h_{11} に対して s よりも近い所にクエリがあるので，$\mathrm{S}_1^{h^-}(q)$ 内の点も $\mathrm{S}_{11}^{h}(q)$ 内の点と同様に扱うことになる．それによってバケット $\mathrm{S}_{11}^{h}(q) \cap \mathrm{S}_{12}^{h}(q)$ 内の点だけではなくバケット $\mathrm{S}_{11}^{h^-}(q) \cap \mathrm{S}_{12}^{h}(q)$ 内の点も距離計算する対象にする．図 4 の場合では，バケット $\mathrm{S}_{11}^{h^-}(q) \cap \mathrm{S}_{12}^{h}(q)$ 内の点も距離計算する対象にすることによって，最近傍点が近似最近傍点と一致する．このように h_{jk} 毎に隣を探索するかどうかの判定を行えば，クエリが複数のハッシュ値境界の近くの値をとることになっても対応することができる．

ハッシュ関数に式 (1) を用いると $\boldsymbol{a}$ はハッシュ幅 w で区切られる．クエリとハッシュ値の境界との距離が s より小さいときに隣のバケットも探索するために

$$h^+(p) = \left\lfloor \frac{\boldsymbol{a}\cdot\boldsymbol{p}+b}{w} + \frac{s}{w} \right\rfloor$$

$$h^-(p) = \left\lfloor \frac{\boldsymbol{a}\cdot\boldsymbol{p}+b}{w} - \frac{s}{w} \right\rfloor$$

となるような $h^+(\cdot)$ と $h^-(\cdot)$ を考えて下記の処理を行う．

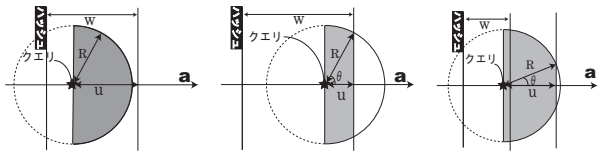
- もし $h(q) - h^-(q) \neq 0$ なら $h_{ji}(q) - 1$ 内の点も $h(q)$ 内の点と同様に扱う
- もし $h(q) - h^+(q) \neq 0$ なら $h_{ji}(q) + 1$ 内の点も $h(q)$ 内の点と同様に扱う

このような処理を行えば ANB を実現することができる．

ANB を近似最近傍探索に用いれば，LSH よりも用いるハッシュ関数の数を減らすことができると考えられる．それは，各ハッシュ関数群 g_j で最近傍点が距離を計算する対象となる確率が高くなれば，LSH よりハッシュ関数群 g_j の数を減らしても LSH と同等の確率で最近傍点と近似最近傍点を一致させることができるからである．

4. 理論解析

解析によって求めなければならないのは，精度と，時間計算量，空間計算量である．パラメータ s, k, L と精度，時間計算量，空間計算量の関係を導くことができれば，アプリケーションに用いる際の指標になる．



(a) 半超球がすべてバケットに含まれる　(b) 半超球がバケットに含まれない部分がある (隣を参照しない)　(c) 半超球がバケットに含まれない部分がある (隣を参照する)

図 5　解析モデル

4.1 精　　度

まず，最近傍点と近似最近傍点が一致する確率である精度から考える．精度はデータに依存するので，データ非依存の指標を得るため，確率モデルを用いた解析をする．各クエリ q から最近傍点 $p^*(q)$ までの距離のいずれよりも大きい R を考える．つまり，

$$^\forall q, \| q - p^*(q) \| \leqq R$$

であるとする．このとき，クエリを中心とした半径 R の超球を描き，その超球の内側の点をすべて距離を計算する対象にすることができたらその中に必ず最近傍点が含まれ，近似最近傍点と最近傍点が一致することになる．このモデルでは，精度の下限が，超球とクエリの入ったバケットの積集合空間が超球に占める割合の期待値によって求められる．なお，このモデルは左右対称なので，以下では超球の右半分 (半超球) だけを考える．

前述の割合の期待値を求めるために 図 5 のような解析モデルを考えて，$S_{ji}^h(q)$ と半超球の積集合が半超球に占める割合を考える．半超球の積集合の体積を Vo^d とおき，Vo^d が超球に占める割合を $P_{Vo}(u)$ とおく．図 5 のように u と θ をおく．u はクエリから大きいほうのハッシュ値境界までの距離である．

図 5 のそれぞれの着色部分の体積 Vo^d は $\varphi(u)$ を

$$\varphi(u) = \begin{cases} 0 & \text{図 5(a)} \\ cos^{-1}\frac{u}{R} & \text{図 5(b)} \\ cos^{-1}\frac{u+w}{R} & \text{図 5(c)} \end{cases}$$

とすると

$$Vo^d(\varphi(u)) = V^{d-1}(R)R^{d-2}\int_{\varphi(u)}^{\frac{\pi}{2}} \sin^d\theta \, d\theta \qquad (2)$$

で求めることができるので Vo^d が半超球の体積に占める割合は

$$P_{Vo} = Vo^d(\varphi(u))/\{V^d(R)/2\} \qquad (3)$$

で求められる．

次に $P_{Vo}(u)$ を用いて，h_{ij} で超球内の点が取れる確率 P_h を求める．$P_{Vo}(u)$ は u に依存するので，u に対して期待値を計算しなければならない．LSH ではハッシュ値の境界が一様乱数

表 1　場合分け毎の状態と精度を求めるための値

場合分け			事象	α	β	$\varphi(u)$
i		(a)	$E_{\text{i-a}}$	0	s	0
		(b)	$E_{\text{i-b}}$	s	R	$\cos^{-1}\frac{u}{R}$
		(c)	$E_{\text{i-c}}$	R	w	0
ii	1	(a)	$E_{\text{ii-1-a}}$	0	$R-w$	$\cos^{-1}\frac{u+w}{R}$
		(b)	$E_{\text{ii-1-b}}$	$R-w$	s	0
		(c)	$E_{\text{ii-1-c}}$	s	w	$\cos^{-1}\frac{u}{R}$
	2	(a)	$E_{\text{ii-2-a}}$	0	s	$\cos^{-1}\frac{u+w}{R}$
		(b)	$E_{\text{ii-2-b}}$	s	w	$\cos^{-1}\frac{u}{R}$
iii		(a)	$E_{\text{iii-a}}$	0	s	$\cos^{-1}\frac{u+w}{R}$
		(b)	$E_{\text{iii-b}}$	s	w	$\cos^{-1}\frac{u}{R}$

に従って決まるので，この u の値は $0 \leqq u \leqq w$ で一様であると考えることができる．u が $\alpha \leqq u \leqq \beta$ の値をとる事象を E として E での $P_{Vo}(u)$ の期待値を求める．u は $\alpha \leqq u \leqq \beta$ の値を一様にとるので，期待値は

$$\frac{1}{\beta-\alpha}\int_{\alpha}^{\beta}\frac{Vo^d(\varphi(u))}{V^d(R)/2}\,du \tag{4}$$

と求められる．P_h は $0 \leqq u \leqq w$ 間の u が取り得る事象ごとの期待値の和である．

以上のように u の値によってうまく場合分けを行えば P_h を求めることができる．ここで P_h の値は s，w，R の値に依存するのでこれ以降 $P_h(s,w,R)$ とする．本稿ではスペースの関係上，細かい場合分けの導出は，結果を表 1 にまとめて割愛する．

次に精度を導出する精度を $P_g(s,w,R)$ とおくと，$P_g(s,w,R)$ は，$1-[1-\{P_h(s,w,R)\}^k]^L$ で求めることができる [4]．まず，あるハッシュ関数群 g_j でクエリと同じバケットに入る確率を求める．ある点 p が各 h_{ji} で $h_{ji}(q)=h_{ji}(p)$ となる事象が起こる確率は $P_h(s,w,R)$ である．各 h_{ji} で $h_{ji}(q)=h_{ji}(p)$ となる事象は互いに独立なので，$^{\forall}i, h_{ji}(q)=h_{ji}(p)$ となる確率は $\{P_h(s,w,R)\}^k$ である．この確率 $\{P_h(s,w,R)\}^k$ から g_j で p がクエリと同じバケットに入らない確率を求めることができる．g_j でクエリと同じバケットに入らないという事象は g_j でクエリと同じバケットに入るという事象の余事象なので，その確率は $1-\{P_h(s,w,R)\}^k$ である．g_j は互いに独立なので用いた $g_j(1 \leqq j \leqq L)$ で一度もクエリと同じバケットに入らないという事象は確率 $[1-\{P_h(s,w,R)\}^k]^L$ で起こる．$^{\exists}j, g_j(q)=g_j(p)$ を満たす点 p は距離を計算する対象になるので，p が距離を計算する対象になる確率は $^{\forall}j, g_j(q) \neq g_j(p)$ の余事象の確率と等しい．以上より，$P_g(s,w,R)=1-[1-\{P_h(s,w,R)\}^k]^L$ が示せた．

この解析には 1 つ問題点がある．ハッシュ幅 w は，値を固定しても見掛け上のハッシュ幅は変動してしまう．$\| \boldsymbol{a} \|=A$ とおくと A の値は正規分布に従い，一定ではないからである．そのため本節で導出したすべての式の w を w/A で置き換えて考える必要がある．このようにして導出した $P_g(s,\frac{w}{A},R)$ の A に関する期待値を改めて $P'(s,w,R)$ とおくと，

$$P'(s,w,R)=2\int_0^{\infty}\frac{1}{\sqrt{2\pi}}\exp\left[\frac{-A^2}{2}\right]P'(s,\frac{w}{A},R)\,dA \tag{5}$$

で求められる(注2)．ここで $P'(s,\frac{w}{A},R)$ は $P_g(s,w,R)$ の w を w/A に置き換えたものである．w を w/A に置き換えたあとの (i)(ii)(iii) の条件式は

(i) $0<R<\frac{w}{A}$
(ii) $\frac{w}{A} \leqq R \leqq \frac{2w}{A}$
(iii) $\frac{2w}{A}<R$

となるので式 (5) は

$$\begin{aligned}
P'(s,w,R)= &\int_0^{\frac{w}{R}}\frac{2}{\sqrt{2\pi}}\exp\left[\frac{-A^2}{2}\right]P_{(1)}(s,\frac{w}{A},R)\,dA \quad \text{(i)}\\
+ &\int_{\frac{w}{R}}^{\frac{2w}{R}}\frac{2}{\sqrt{2\pi}}\exp\left[\frac{-A^2}{2}\right]P_{(2)}(s,\frac{w}{A},R)\,dA \quad \text{(ii)}\\
+ &\int_{\frac{2w}{R}}^{\infty}\frac{2}{\sqrt{2\pi}}\exp\left[\frac{-A^2}{2}\right]P_{(3)}(s,\frac{w}{A},R)\,dA \quad \text{(iii)}
\end{aligned}$$

と分解できる．以上のようにして精度 $P'(s,w,R)$ が求められる．

4.2　時間計算量

4.2.1　距離計算に要する時間計算量

次に距離計算に要する時間計算量について解析する．時間計算量の大部分を占めるのはクエリの近似最近傍点を求めるための距離計算である．ANB において，隣のバケット内の点も距離計算を行う為，必要以上に計算量が増加する可能性がある．

本稿では，ANB にかかる距離計算の時間計算量が LSH と比べてどうなるかという観点で解析を行った．解析の方法としては，単位面積あたりの点数が一定と仮定し，距離計算の計算量は面積に比例するとした．ハッシュの幅が w のとき，幅 w のハッシュで切り取られる空間の体積を $VOL(w)$ とおくと，A の変動によって見掛け上の幅は w/A になるので，実際には $VOL(w/A)$ となる．

$$2\int_0^{\infty}\frac{1}{\sqrt{2\pi}}\exp\left[\frac{-A^2}{2}\right]\left\{1-\left\{1-\left\{\frac{\mathrm{VOL}}{A}\cdot\frac{w+2s}{w}\right\}^k\right\}^L\right\}dA \tag{6}$$

を求めることにより LSH との距離計算量の比を求める．VOL/A は w が決まった時の空間 $\mathrm{S}_{ij}^h(q)$ の体積であり，ANB では，$2s/w$ の確率で隣のバケットも参照するので，探索するバケットの体積は LSH のときを VOL/A とすると $\frac{\mathrm{VOL}}{A}\cdot\frac{w+2s}{w}$ となる．

4.2.2　距離計算以外の時間計算量

距離計算以外にかかる時間計算量として主なものにハッシュの参照がある．ANB では LSH と比べてハッシュ参照回数が平均 $\frac{w+2s}{w}$ 倍になる．しかし一般には時間計算量全体の中で距離計算にかかる計算量が支配的になるため，本稿では距離計算の時間計算量を時間計算量として解析する．

(注2)：ガウス分布の確率密度関数は $\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}\exp\left[\frac{-A^2}{2}\right]dA$ であるが偶関数なので $2\int_0^{\infty}\frac{1}{\sqrt{2\pi}}\exp\left[\frac{-A^2}{2}\right]dA$ と等しくなる．

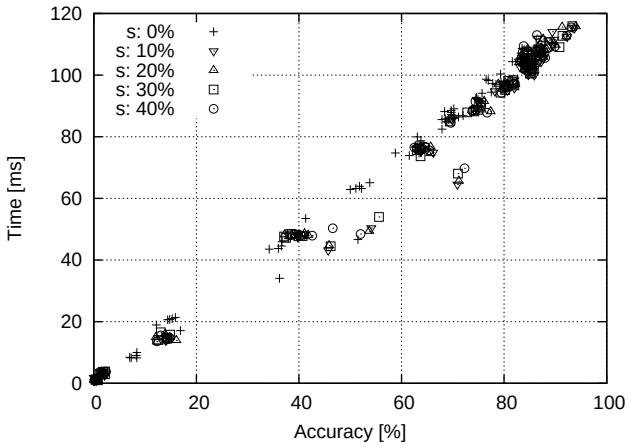


図 6　精度と計算時間の関係

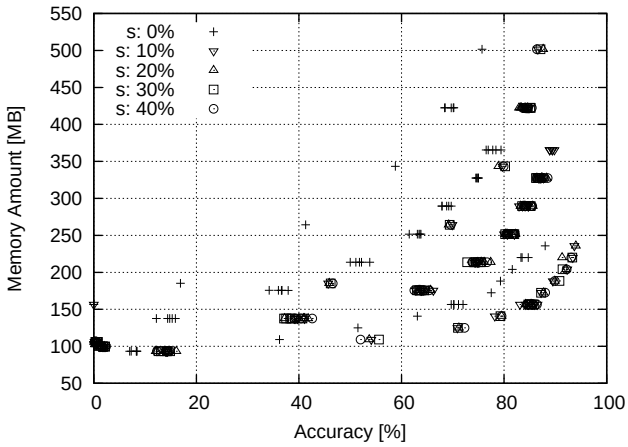


図 7　精度とメモリ使用量の関係

4.3　空間計算量

空間計算量の大部分を占めるのはハッシュテーブルの容量である．LSH と ANB では，h_{ji} 一つにつき一つのハッシュテーブルを準備する必要がある．h_{ji} の数は k と L の値に対して比例するため，本稿では kL を空間計算量の指標として用いる．

5.　シュミレーションと実装結果

5.1　実 験 結 果

100 次元空間に一様に分布する人工データを作成してシュミレーションを行う．各次元の要素の値が $[0, 10000]$ であるような 100 次元のデータを 100,000 点生成し，クエリを 100 点生成した．精度は，あらかじめ全探索によって最近傍点を求めておき，近似最近傍点が，全クエリ 100 点中何個一致したかにより算出した．ハッシュ幅 w を 100 とし，k と L の値を変動させた結果を図 6 と図 7 に示す．図 6，図 7 はそれぞれ精度と計算時間の関係，精度とメモリ使用量の関係を表している．図中の s が 0% が LSH に，s が 10%〜40% が ANB にそれぞれ対応する．図 6 より，LSH と ANB では計算時間にそれほど差がないことが分かる．また図 7 より，ANB は LSH と同等の精度を LSH よりも小さい空間計算量で実現できることが分かる．図 8 は精度が 80% から 85% の点について計算時間とメモリ使用量の関係を示したものである．これを見ると，パラメータをうまく設定すれば精度と時間計算量はそのままに，空間計算量を約 25% 削減できることが分かる．

以上の結果より ANB が空間計算量を削減するのに有効な手法であることが確認できた．

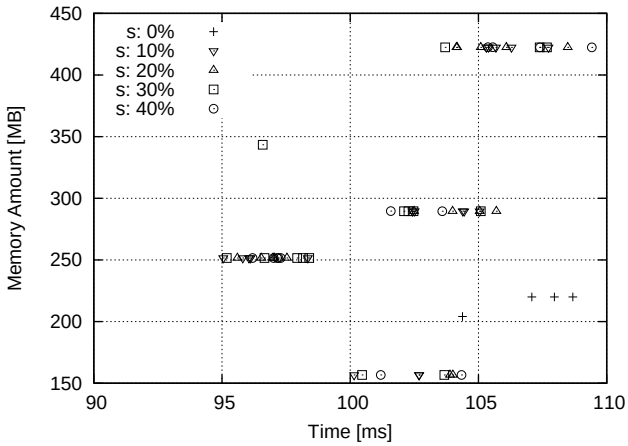


図 8　計算時間とメモリ使用量の関係 (精度 80%〜85%)

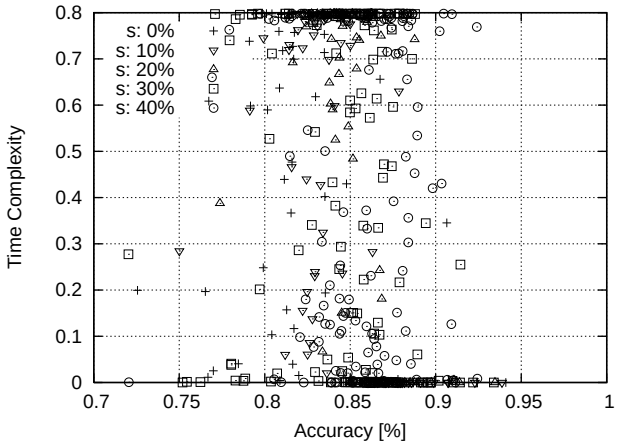


図 9　精度と時間計算量の関係

5.2　理　論　値

4 節で導いた解析結果を図示し，5.1 節の実験結果と類似の結果が得られるか検証する．4 節で得られた式には解析的に計算できない積分が含まれているため，Monte Carlo 法を用いた数値積分をする．精度は式 (5)，時間計算量は式 (7)，空間計算量は kL の値を用いる．データの次元数 d を 100，ハッシュ幅 w を 1.0，超球の半径 R を 0.99 として k と L の値を変動させて得られた結果を図 **??**と図 **??**に示す．図 9 と図 10 はそれぞれ精度と￥時間計算量の関係と精度と空間計算量の関係である．

今回のシュミレーションでは繰り返し回数が足りなかったため，結果が収束せずはっきりとした傾向を示すに至らなかった．また，図 9 では，精度と時間計算量の相関が全く現れていない．これは式 (6) の VOL の値を適切に設定できなかったためであると考えられる．しかし，図 10 より，ANB のほうが LSH よりも全体的に精度が高そうであるということが言える．

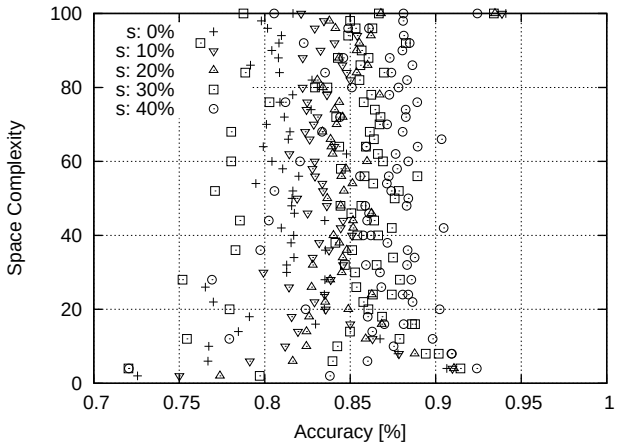


図 10　精度と時間計算量の関係

6. む す び

本論文では，LSH 型の近似最近傍探索において，文献 [7], [8] で行われている，隣接バケットを参照することにより精度を落とすことなく，メモリ使用量 (空間計算量) を減らすことができる手法をモデル化して解析した．実装結果より，ANB は空間計算量の削減に有効だと言える．今後の課題として，シュミレーション結果をしっかり出すことの他に，さらに隣のバケットも探索するような手法の解析などが挙げられる．

謝辞　本研究の一部は，平成 20 年度 SCAT 研究費助成ならびに科研費補助金基盤研究 (B)(19300062) の補助による．

文　献

[1] D. Lowe, “Distinctive image features from scale-invariant,” International Journal of Computer Vision, 2004.

[2] Y. Ke and R. Sukthankar, “Pca-sift: A more distinctive representation for local image descriptors,” Proc. of CVPR2004, pp.91–110, 2004.

[3] S. Arya, D. Mountand, N. Netanyahu, R. Silverman and A. Wu, “An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions,” Journal of the ACM (JACM), vol.45, pp.891–923, nov 1996.

[4] P. Indyk and R. Motowani, “Approximate nearest neighbors : Towards removing the curse of dimensionality,” In Proceedings of the 30th ACM Symposium on Theory of Computing(STOC’98), pp.604–613, may 1999.

[5] M. Datar, N. Immorlica, P. Indyk and V. S. Mirrokni, “Locality-Sensitive Hashing Scheme Based on p-Stable,” In-Proceedings of the 20th Annual Symposium on Computational Geometry(SCG2004), jun 2004.

[6] A. Andoni and P. Indyk, “Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions,” In Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS’06), pp.459–468, oct 2006.

[7] 野口和人，黄瀬浩一，岩村雅一，“近似最近傍探索の多段階化による物体の高速認識，” 画像の認識・理解シンポジウム (MIRU2007) 論文集，jul 2007.

[8] 松下裕輔，和田俊和，“Principal Component Hashing -等確率バケット分割による近似最近傍探索法-，” 画像の認識・理解シンポジウム (MIRU2007) 論文集，pp.127–134，jul 2007.