(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2012-247993
(P2012-247993A)
(43)公開日　平成24年12月13日(2012.12.13)

| (51)Int.Cl. | | F I | | | テーマコード（参考） |
|---|---|---|---|---|---|
| G06T 1/00 | (2006.01) | G06T | 1/00 | 200D | 5B050 |
| G06F 17/30 | (2006.01) | G06F | 17/30 | 320Z | |
| | | G06F | 17/30 | 170B | |
| | | G06F | 17/30 | 412 | |

審査請求 未請求　請求項の数 8　ＯＬ　（全 22 頁）

(21)出願番号　特願2011-119128 (P2011-119128)
(22)出願日　平成23年5月27日 (2011.5.27)

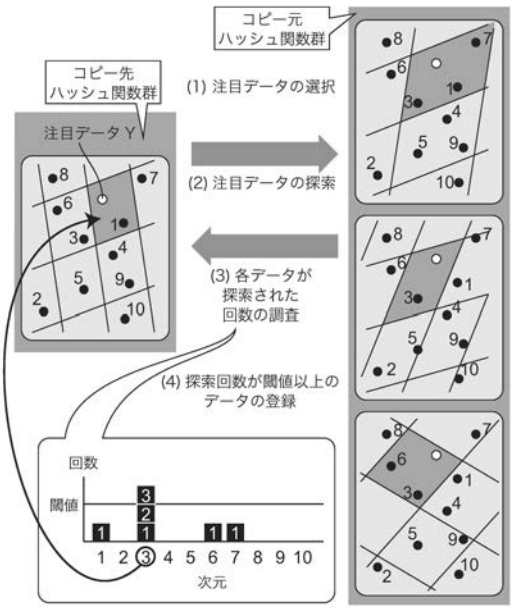
（出願人による申告）平成２２年度、独立行政法人科学技術振興機構、戦略的創造研究推進事業、産業技術力強化法第１９条の適用を受ける特許出願

(71)出願人　505127721
公立大学法人大阪府立大学
大阪府堺市中区学園町１番１号
(74)代理人　100065248
弁理士　野河　信太郎
(74)代理人　100145229
弁理士　秋山　雅則
(74)代理人　100159385
弁理士　甲斐　伸二
(74)代理人　100163407
弁理士　金子　裕輔
(74)代理人　100166936
弁理士　稲本　潔

最終頁に続く

(54)【発明の名称】近似最近傍探索に係るデータベースの登録方法および登録装置

(57)【要約】　（修正有）
【課題】近似最近傍探索の一手法であるLSHにおいて、必要な計算時間とメモリ使用量を削減する。
【解決手段】画像の特徴ベクトルを抽出してデータベースに登録するとき、各特徴ベクトルを複数のビンに分類するため、k個を一組の単位としてL組生成したハッシュテーブルに登録する。そして（ｉ）登録されたある特徴ベクトルを選んでその特徴ベクトルと同じ登録ビンの他の特徴ベクトルを特定し、（ii）各組ごとに、その組のk個の登録ビンのいずれにも登録されている他の特徴ベクトルの集合をその組のバケットとし、（iii）全L個のバケットのうち所定個以上のバケットに入っている特徴ベクトルを得、（iv）得られた特徴ベクトルを第１組のハッシュテーブルの各登録ビンにそれぞれ追加登録し、所定数の特徴ベクトルについて前記（ｉ）～（iv）による追加登録を実行した後、第１組を除く各組のハッシュテーブルを削除する画像データベースの登録方法。
【選択図】図４



【特許請求の範囲】
【請求項１】
コンピュータが、
画像に係るデータからそのデータの特徴を表す特徴ベクトルを抽出する工程と、
抽出された特徴ベクトルを前記データと共にデータベースに登録する登録工程とを備え、
前記データベースは、検索質問として画像に係るデータが与えられたとき、そのデータからクエリベクトルを抽出し、クエリベクトルから最も近いと推測される特徴ベクトルの探索を行うために用いられ、
前記登録工程は、各特徴ベクトルを複数のビンの何れか一つに分類して登録するためのハッシュテーブルを、k 個を一組の単位としてL 組（k ，L は２以上の整数）生成し、
各特徴ベクトルをそれらのハッシュテーブルにそれぞれ登録した後に、
（ｉ）登録されたある特徴ベクトルを選んでその特徴ベクトルと同じビンである登録ビンに分類された他の特徴ベクトルを特定し、
（ii）各組ごとに、その組のk 個の登録ビンのいずれにも登録されている他の特徴ベクトルの集合をその組のバケットとし、
（iii）全L 個のバケットのうち所定個以上のバケットに入っている特徴ベクトルを得、
（iv）得られた特徴ベクトルを第１組のハッシュテーブルの各登録ビンにそれぞれ追加登録し、
所定数の特徴ベクトルについて前記（ｉ）～（iv）による追加登録を実行した後、
第１組を除く各組のハッシュテーブルを削除することを特徴とするデータベースの登録方法。
【請求項２】
前記探索は、クエリベクトルにハッシュ関数を適用して、第１組の各ハッシュテーブルについて対応するk 個のビンを決定し、それらのビンのいずれにも登録されている特徴ベクトルの集合を求め、前記クエリベクトルをその集合に属する各特徴ベクトルと照合する請求項１に記載の方法。
【請求項３】
前記登録工程は、追加登録を行うために選択する特徴ベクトルを、一様乱数を用いて決定する請求項１または２に記載の方法。
【請求項４】
前記登録工程は、追加登録を行おうとする特徴ベクトルが、第１組のバケットに既に登録されているときは、その特徴ベクトルのさらなる追加登録を行わない請求項１～３のいずれか一つに記載の方法。
【請求項５】
前記登録工程は、予め定められたk およびL の値に基づいて処理を行い、
登録すべき特徴ベクトルのうち予め定められた割合の数だけ、追加登録を行う特徴ベクトルを選択し、
予め定められた個数以上のバケットに入った特徴ベクトルを追加登録する請求項１～４のいずれか一つに記載の方法。
【請求項６】
前記データベースは、各特徴ベクトルのベクトルデータと各特徴ベクトルの識別子とが対応付けられた対応表および各ハッシュテーブルを含み、
各ハッシュテーブルは、各ビンに登録される各特徴ベクトルを対応する識別子を用いて表す請求項１～５のいずれか一つに記載の方法。
【請求項７】
前記探索は、クエリベクトルと各特徴ベクトルとの距離を計算し、計算された距離に基づいてクエリベクトルから最も近いと推測される特徴ベクトルを決定する請求項１～６のいずれか一つに記載の方法。
【請求項８】
画像に係るデータからそのデータの特徴を表す特徴ベクトルを抽出する処理部と、

抽出された特徴ベクトルを前記データと共にデータベースに登録する登録部とを備え、
前記データベースは、検索質問として画像に係るデータが与えられたとき、そのデータからクエリベクトルを抽出し、クエリベクトルから最も近いと推測される特徴ベクトルの探索を行う探索装置に用いられ、
前記登録部は、各特徴ベクトルを複数のビンの何れか一つに分類して登録するためのハッシュテーブルを、k 個を一組の単位としてL 組（k ，L は２以上の整数）生成し、
各特徴ベクトルをそれらのハッシュテーブルにそれぞれ登録した後に、
（ｉ）登録されたある特徴ベクトルを選んでその特徴ベクトルと同じビンである登録ビンに分類された他の特徴ベクトルを特定し、
（ii）各組ごとに、その組のk 個の登録ビンのいずれにも登録されている他の特徴ベクトルの集合をその組のバケットとし、
（iii）全L 個のバケットのうち所定個以上のバケットに入っている特徴ベクトルを得、
（iv）得られた特徴ベクトルを第１組のハッシュテーブルの各登録ビンにそれぞれ追加登録し、
所定数の特徴ベクトルについて前記（ｉ）～（iv）による追加登録を実行した後、
第１組を除く各組のハッシュテーブルを削除することを特徴とするデータベースの登録装置。

【発明の詳細な説明】

【技術分野】

【０００１】

この発明は、画像に係るデータのデータベースへの登録方法および登録装置に関する。より詳細には、前記データベースの探索に適用される近似最近傍探索の手法に関する。

【０００２】

前記データベースは、例えば物体認識に用いられるものである。物体認識は、検索質問（クエリ）として物体の画像が与えられたとき、画像データベースに登録された各画像のうちクエリに最も近い画像、即ち物体をコンピュータを用いて探索する処理といえる。なお、ここでいう物体即ちオブジェクトは、人物や生物を含む広い意味の物体である。探索の処理手順としては、画像からその特徴を表すベクトルデータ（特徴ベクトル）を抽出し、抽出された特徴ベクトルを前記画像と共にその画像に対応する特徴ベクトルとを画像データベースに登録しておく。クエリが与えられたとき、そのクエリから特徴ベクトル（クエリベクトル）を抽出し、画像データベースに登録された各特徴ベクトルと照合する。その中で、クエリベクトルに最も近い特徴ベクトルを探索する。この探索を最近傍探索という。

なお、最近傍探索は、物体認識に限らず、他の様々な分野で用いられている。例えば、文字認識、画像検索をはじめとして、データの統計分類、データ圧縮、商品等の推薦システム、マーケティング、スペルチェッカー、DNAシークエンシングなどに適用される。この発明は、物体認識に限らずこれらの分野におけるベクトルデータの最近傍探索にも適用できる。

【背景技術】

【０００３】

最近傍探索は、データベースS 中からクエリベクトル（以下、単にクエリ）q と距離が最も近いベクトルデータ（以下、単にデータ）p ∈S を発見する命題である。最近傍探索ではクエリと全てのデータとの距離を計算すれば必ず正しい解が得られる。この単純な命題は、扱うデータの規模が大きくなると容易には解けなくなる。20 億ベクトルをデータベースに登録しておき、物体認識を行うタスクも存在する（例えば、非特許文献２参照）。そのため、最近傍探索の高速化は不可欠である。

【０００４】

最近傍探索の高速化には、木構造などでデータベースを構造化し、距離計算回数を削減することが有効である（例えば、非特許文献３参照）。しかし、構造化によってデータ以外の情報も記憶することになるため、より大きなメモリ使用量が必要になる。計算時間と

メモリ使用量にはトレードオフの関係があると考えられている。次元数が２より大きい場合に扱うデータ数n に対して計算時間が対数、メモリ使用量が線形に増加するアルゴリズムは知られていない（例えば、非特許文献４参照）。

【０００５】

この限界を超えるために近似最近傍探索が近年注目されている。これは最近傍探索において条件を緩和し、必ずしも最近傍データが得られなくてもよくしたものである。近似最近傍探索により、常に厳密な最近傍点を求める最近傍探索に比べて計算時間とメモリ使用量を大幅に削減できる。近似最近傍探索の代表的な手法としては、木構造を用いるApproximate Nearest Neighbor(ANN、例えば、非特許文献４参照）やハッシュを用いるLocality Sensitive Hashing(LSH、例えば、非特許文献１、５参照）、Spectral Hashing（例えば、非特許文献６参照）、Minwise Hashing（例えば、非特許文献７参照）などが知られている。近似最近傍探索によって得られるベクトルデータは、クエリベクトルq に最も近いと推測されたデータであるが、真に最近傍のデータとは限らない。

【先行技術文献】

【非特許文献】

【０００６】

【非特許文献１】M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni, “Locality-sensitive hashing scheme based on p-stable distributions,” Proc. 20th annual symposium on Computational geometry, pp.253-262, 2004

【非特許文献２】黄瀬浩一、野口和人、岩村雅一、“参照特徴ベクトルの増加による低品質画像の高速・高精度認識, ”信学論D, vol.J93-D, no.8, pp.1353-1363, Aug. 2010

【非特許文献３】片山紀生、佐藤真一、“Sr-tree：高次元点データに対する最近接検索のためのインデックス構造の提案, ”電子情報通信学会論文誌D, vol.J80-D1, no.8, pp.703-717, Aug. 1997

【非特許文献４】S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu, “An optimal algorithm for approximate nearest neighbor searching in fixed dimensions,” Journal of the ACM, vol.45, no.6, pp.891-923, Nov. 1998

【非特許文献５】P. Indyk and R. Motwani, “Approximate nearest neighbor: towards removing the curse of dimensionality,” Proc 30th Symposium on Theory of Computing, pp.604-613, 1998

【非特許文献６】Y. Weiss, A. Torralba, and R. Fergus, “Spectral hashing,” Advances in Neural Information Processing Systems, vol.21, pp.1753-1760, 2008

【非特許文献７】A.Z. Broder, M. Charikar, A.M. Frieze, and M. Mitzenmacher, “Min-wise independent permutations,” Journal of Computer and System Sciences, vol.60, pp.630-659, 2000

【発明の概要】

【発明が解決しようとする課題】

【０００７】

近似最近傍探索では、精度（最近傍データが正しく求まる確率）、計算時間、メモリ使用量にトレードオフの関係があると考えられる。そのため、ある一定の精度を実現するために必要な計算時間とメモリ使用量が問題となる。

【０００８】

この発明は、以上のような事情を考慮してなされたものであって、近似最近傍探索の一手法であるLSHに基づいて、同一精度を実現するために必要な計算時間とメモリ使用量を従来よりも削減できる手法を提供するものである。

【課題を解決するための手段】

【０００９】

この発明は、コンピュータが、画像に係るデータからそのデータの特徴を表す特徴ベクトルを抽出する工程と、抽出された特徴ベクトルを前記データと共にデータベースに登録する登録工程とを備え、前記データベースは、検索質問として画像に係るデータが与えら

れたとき、そのデータからクエリベクトルを抽出し、クエリベクトルから最も近いと推測される特徴ベクトルの探索を行うために用いられ、前記登録工程は、各特徴ベクトルを複数のビンの何れか一つに分類して登録するためのハッシュテーブルを、k 個を一組の単位としてL 組（k ，L は２以上の整数）生成し、各特徴ベクトルをそれらのハッシュテーブルにそれぞれ登録した後に、（ｉ）登録されたある特徴ベクトルを選んでその特徴ベクトルと同じビンである登録ビンに分類された他の特徴ベクトルを特定し、（ii）各組ごとに、その組のk 個の登録ビンのいずれにも登録されている他の特徴ベクトルの集合をその組のバケットとし、（iii）全L 個のバケットのうち所定個以上のバケットに入っている特徴ベクトルを得、（iv）得られた特徴ベクトルを第１組のハッシュテーブルの各登録ビンにそれぞれ追加登録し、所定数の特徴ベクトルについて前記（ｉ）～（iv）による追加登録を実行した後、第１組を除く各組のハッシュテーブルを削除することを特徴とするデータベースの登録方法を提供する。

【００１０】

また、異なる観点から、この発明は、画像に係るデータからそのデータの特徴を表す特徴ベクトルを抽出する処理部と、抽出された特徴ベクトルを前記データと共にデータベースに登録する登録部とを備え、前記データベースは、検索質問として画像に係るデータが与えられたとき、そのデータからクエリベクトルを抽出し、クエリベクトルから最も近いと推測される特徴ベクトルの探索を行う探索装置に用いられ、前記登録部は、各特徴ベクトルを複数のビンの何れか一つに分類して登録するためのハッシュテーブルを、k 個を一組の単位としてL 組（k ，L は２以上の整数）生成し、各特徴ベクトルをそれらのハッシュテーブルにそれぞれ登録した後に、（ｉ）登録されたある特徴ベクトルを選んでその特徴ベクトルと同じビンである登録ビンに分類された他の特徴ベクトルを特定し、（ii）各組ごとに、その組のk 個の登録ビンのいずれにも登録されている他の特徴ベクトルの集合をその組のバケットとし、（iii）全L 個のバケットのうち所定個以上のバケットに入っている特徴ベクトルを得、（iv）得られた特徴ベクトルを第１組のハッシュテーブルの各登録ビンにそれぞれ追加登録し、所定数の特徴ベクトルについて前記（ｉ）～（iv）による追加登録を実行した後、第１組を除く各組のハッシュテーブルを削除することを特徴とするデータベースの登録装置を提供する。

【発明の効果】

【００１１】

この発明による登録方法は、所定個以上のバケットに入っている特徴ベクトルを第１組のハッシュテーブルの各登録ビンにそれぞれ追加登録した後、第１組を除く各組のハッシュテーブルを削除するので、近似最近傍探索の一手法であるLSHに基づいて、同一精度を実現するために必要な計算時間とメモリ使用量を従来よりも削減できる。即ち、第１組を除く（Ｌ －１）組のハッシュテーブルを格納するメモリを削減でき、かつ、第１組を除く（Ｌ －１）組のハッシュテーブルの探索に要する時間を削減できる。なお、この発明の近似最近傍探索に係る手法は、従来のLSHを用いた近似最近傍探索に代えて適用できるだけでなく、他の手法による近似最近傍探索に代えて適用することができる。

この発明の登録装置についても、前記登録方法と同様の作用効果を奏する。

【００１２】

このように計算リソースとしての計算時間やメモリ使用量を削減する場合、最近傍探索を用いる識別器で行われているように、データベースに登録されるデータ数の削減を考えるのが一般的と思われる（例えば、和田俊和、“空間分割を用いた識別と非線形写像の学習:（1）空間分割による最近傍識別の高速化，”情報処理，vol.46，no.8，pp.912-918，Aug．2005参照）。この発明によれば、逆にデータベースに登録されているデータを重複登録し、データベースに登録されるデータ数を増加させることによってこれを実現する。この発明の手法は一見逆説的であるが、従来手法であるLSH に対して18%の計算時間と90%のメモリ使用量で同等の精度を実現できることを実験で確認した。さらに、この要因は、前記非特許文献１に示されているLSHの探索効率の基準ρを用いて説明することができる。

【００１３】
この発明において、登録すべきデータからは１つの特徴ベクトルが抽出されてもよいし複数の特徴ベクトルが抽出されてもよい。データから特徴ベクトルを抽出する手法としては、周知のものが適用できる。例えば、後述する実験例では、LBP 特徴の抽出手法を用いている。ただし、これに限定されるものでなく、例えば、局所特徴量として周知のSIFTや他の手法を用いることができる。
また、この発明において、クエリベクトルは各特徴ベクトルの抽出と同様の手法を用いて抽出する。
【００１４】
この発明において、一つのバケットはk 個のハッシュテーブルを用いて特定される。そして、クエリベクトルに最も近いと推測される特徴ベクトルを、クエリベクトルに対応する第１組のバケットに登録された各特徴ベクトルの中から決定する。
この発明の登録方法によれば、登録に際してk ×L 個のハッシュテーブルを一時的に生成するが、最終的には第１組に該当するk 個のハッシュテーブルに特徴ベクトルの追加登録を行った後、第１組を除く各組のハッシュテーブルを削除する。
よって、探索は、第１組のk 個のハッシュテーブルを用いる。
【図面の簡単な説明】
【００１５】
【図１】従来のLSH において、距離計算対象の絞り込みの様子を示す説明図である。
【図２】従来のLSH において、局所性に鋭敏なハッシュ関数の記述に係る説明図である。
【図３】この発明による近似最近傍探索を示す第１の説明図である。
【図４】この発明による近似最近傍探索を示す第２の説明図である。
【図５】この発明による近似最近傍探索が処理時間の面で有効であることを示す実験結果のグラフである。
【図６】この発明による近似最近傍探索がメモリ使用量の面で有効であることを示す実験結果のグラフである。
【図７】この発明による近似最近傍探索で、データがコピー元ハッシュ関数群でy 回探索される確率を示すグラフである。
【図８】この発明による近似最近傍探索で、閾値t とデータが追加登録される確率との関係を示すグラフである。
【図９】従来のLSHを適用した物体認識に係るデータベースの構造を示す説明図である。
【発明を実施するための形態】
【００１６】
以下、この発明の好ましい態様について説明する。
前記探索は、クエリベクトルにハッシュ関数を適用して、第１組の各ハッシュテーブルについて対応するk 個のビンを決定し、それらのビンのいずれにも登録されている特徴ベクトルの集合を求め、前記クエリベクトルをその集合に属する各特徴ベクトルと照合してもよい。このようにすれば、特徴ベクトルの追加登録がなされた第１組のハッシュテーブルのみを用い、削除された第２～第L 組のハッシュテーブルを用いないで探索を行うことができる。
【００１７】
また、前記登録工程は、追加登録を行うために選択する特徴ベクトルを、一様乱数を用いて決定してもよい。登録時に各バケットに入る特徴ベクトルの分布と複数回の探索時に各バケットに対応するクエリベクトルの分布が同一だと仮定すると、追加登録を行うために選択する特徴ベクトルを一様乱数に基づいて選択すれば特徴ベクトルの分布が密なバケットは多数対応し、粗なバケットはあまり対応しない。よって、多くのクエリベクトルが入力されることが期待されるバケットで追加登録が多く実行されるようにできる。
【００１８】
さらにまた、前記登録工程は、追加登録を行おうとする特徴ベクトルが、第１組のバケットに既に登録されているときは、その特徴ベクトルのさらなる追加登録を行わないよう

にしてもよい。このようにすれば、同一の特徴ベクトルが重複して登録されることがないので、探索の際に重複した距離計算を行って計算時間を無駄に費やすことを回避できる。

【００１９】

前記登録工程は、予め定められたk およびL の値に基づいて処理を行い、登録すべき特徴ベクトルのうち予め定められた割合の数だけ、追加登録を行う特徴ベクトルを選択し、予め定められた個数以上のバケットに入った特徴ベクトルを追加登録するようにしてもよい。これらの値は、実験例においてｋ，Ｌ，β，ｔで表されており、適当な値を用いることによって真の最近傍点が探索される確率を高められる。よって、経験的にあるいは解析によってそれらの好適な値を予め定めておくことができる。

【００２０】

また、前記データベースは、各特徴ベクトルのベクトルデータと各特徴ベクトルの識別子とが対応付けられた対応表および各ハッシュテーブルを含み、各ハッシュテーブルは、各ビンに登録される各特徴ベクトルを対応する識別子を用いて表してもよい。このようにすれば、各ハッシュテーブルにベクトルデータを登録する場合に比べ識別子を登録するだけでよいので、各ハッシュ表のメモリ使用量を削減できる。

【００２１】

さらにまた、前記探索は、クエリベクトルと各特徴ベクトルとの距離を計算し、計算された距離に基づいてクエリベクトルから最も近いと推測される特徴ベクトルを決定してもよい。このようにすれば、ベクトルの距離計算によってクエリベクトルから最も近いと推測される特徴ベクトルを決定することができる。

ここで示した種々の好ましい態様は、それら複数を組み合わせることもできる。

以下、図面を用いてこの発明をさらに詳述する。なお、以下の説明は、すべての点で例示であって、この発明を限定するものと解されるべきではない。

【００２２】

≪基礎となる従来のLSH（Locality Sensitive Hashing）の説明≫

この発明の詳細な説明を述べる前に、その基礎となる従来のLSHについてまず説明する。ここでは、ベクトルデータを対象としたp-stable LSH（前記非特許文献１参照）について述べる。

LSH による近似最近傍探索は、次の2 ステップで実現される。

（１）クエリとの距離を計算すべきデータ、即ち、距離計算対象を選択する。

（２）前記（１）の距離計算対象に対してクエリとの距離を計算し、これに基づいて最近傍データを決定する。

【００２３】

ここで（２）に近似処理は含まれていないことに注意しておきたい。つまり、（１）で求める距離計算対象に真の最近傍データが含まれていれば必ず探索は成功する。以下では、LSH において精度を決定する（１）の処理、すなわち距離計算対象の絞り込みがどのように実現されているかを説明する。

LSH による距離計算対象の絞り込み

最初に次式で与えられるLSH で使用されるハッシュ関数h(v)について述べる。

【００２４】

【数１】

$$h(\boldsymbol{v}) = \left\lfloor \frac{\boldsymbol{a} \cdot \boldsymbol{v}}{w} \right\rfloor \qquad (1)$$

ここで引数v にはデータp あるいはクエリq を与える。a はデータを射影するd 次元ベクトルであり、d 次元正規分布に従って定められる。w はハッシュ幅であり、ビンの幅を決めるためのパラメータである。

【００２５】

なお、式（１）は本稿において重要でない項b を省いている。本来のハッシュ関数は次

式で与えられる。
【数２】

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

ここで$b_{ji}$は区間［0，w］から一様乱数により定められた実数である。
【００２６】
LSHが式（１）のハッシュ関数を用いて距離計算対象を絞り込む様子を図示したのが図１である。星はクエリqで、丸はデータpを表す。
まず、図１（ａ）は、１つのハッシュ関数のみを用いて距離計算対象を決定する様子を示している。結果を先に述べると、図中の灰色地の部分に存在するデータが距離計算対象である。このようにLSHでは式（１）のハッシュ関数を用いて、クエリと同じハッシュ値（インデックス）を持つデータのみを距離計算対象とする。その計算には前述のベクトルaが使用される。幾何的な説明をすれば、ベクトルaにデータやクエリを射影して、等間隔に区切られたビンのどれに属すかでハッシュ値が決定される。クエリと同じビンに入ったデータは同じハッシュ値を持つので距離計算対象となる。このことを改めて定義すると、「h(q) = h(p) を満たす全てのデータpを距離計算対象とする」となる。
なお、厳密には、ハッシュ関数を用いてある参照用データ（キー）に対応する値をすばやく参照するため複数のビンにデータを分類し登録するデータ構造をハッシュテーブルと呼び、狭義のハッシュ関数はあるキーに対応するビンを示す値（ハッシュ値）を与える関数を指す。しかし、この明細書ではデータ構造と関数は一体不可分のものと考えて区別していない。狭義のハッシュ関数を指す場合の他、前記データ構造を示す場合にハッシュ関数あるいはハッシュ関数群の語を用いている。なお、異なるハッシュテーブルは異なる（狭義の）ハッシュ関数を用いてデータを登録する。
【００２７】
さて、次は図１（ｂ）である。注意して図１（ａ）を見ると、距離計算対象にはクエリから遠い点も含まれていることがわかる。この状態は距離計算対象を削減するという観点から言えば効率が悪い。そこで、図１（ｂ）のようにハッシュ関数を複数個（k個）用いて距離計算対象を更に絞り込む。ここで、これらk個のハッシュ関数から成るハッシュ関数群を次式のように定義する。
【００２８】
【数３】

$$g_i(\boldsymbol{v}) = \{h_{i1}(\boldsymbol{v}), h_{i2}(\boldsymbol{v}), \ldots, h_{ik}(\boldsymbol{v})\} \qquad (2)$$

【００２９】
複数のハッシュ関数は添字で区別する。図１中のベクトルaに添字が付いて$a_{ji}$となっているのはそのためである。１番目の添え字jはバケットの番号を表す。２番目の添字iはハッシュ関数を表す。このとき、各ハッシュ関数の距離計算対象の積集合をバケットと呼び、クエリのあるバケット（図１（ｂ）の灰色地領域）内のデータを距離計算対象とする。
【００３０】
ここで改めて図１（ａ）と図１（ｂ）を見てみると、実はどちらの図においても真の最近傍データは距離計算対象に含まれていないことに気付く。そこで真の最近傍データが距離計算対象に含まれる確率を上げるために図１（ｃ）に示すようにバケットを複数（L個）用いることにする。そして、各バケットの距離計算対象の和集合を最終的な距離計算対象とする。図１（ｃ）では、２つの灰色地領域のいずれかに含まれるデータが距離計算対象になる。
【００３１】
ここで、データベースの構造について簡単に触れておく。データベースSは、具体的に

は複数のハッシュテーブル（ハッシュ関数を用いたデータ構造の群、即ちハッシュ関数群）で構成されている。データベースS にデータを登録する場合、そのデータのID、データから抽出した特徴ベクトルのベクトルデータおよびベクトルIDを対応付けた対応表１１をメモリに格納する。ハッシュテーブルのビンには、ベクトルIDを格納する。登録すべきハッシュのビンは、式（１）を用いて計算する。なお、衝突が起こる場合はリスト構造で連結する。

【００３２】

図９は、従来のLSHを適用した物体認識に係るデータベースの構造を示す説明図である。図９の例では、k 個のハッシュ関数からなるハッシュ関数群$g_i(v)$をL 個用いて近似最近傍探索を行う。この場合、各基底のベクトルに対応する（L ×k ）個のハッシュテーブルを一時的にメモリ上に確保する。ただし、最終的には、コピー先ハッシュ関数群に該当するk 個のハッシュテーブルを残し、他のハッシュテーブルをメモリから削除する。

【００３３】

図９を図１と対応させて説明する、図１（ａ）のようにハッシュ関数が１つの場合、データベースは１つのハッシュテーブル$h_{11}(v)$を有する。図１（ｂ）のようにハッシュ関数が２つの場合、データベースは２つのハッシュテーブル$h_{11}(v)$、$h_{12}(v)$からなるハッシュ関数群$g_1(v)$を有する。図１（ｃ）のようにバケットが２つの場合、データベースは２つのハッシュ関数群$g_1(v)$および$g_2(v)$を有する。

【００３４】

バケット数L とLSH の性能の関係

LSH の性能は、ハッシュ関数で使用するw の他に、２つのパラメータ、即ち、ハッシュ関数の数k およびバケット数L によって決まる。このうちバケット数L について、精度、計算時間、メモリ使用量との関係を考える。

ｉ）精度：バケット数L の増加に伴って距離計算対象が単調に増加する。そのため、精度は単調に増加する。

ii）計算量：バケット数L の増加に伴い、参照するハッシュテーブル数が単調に増加し、また距離計算対象も単調に増加する。そのため、計算量は単調に増加する。

iii）メモリ使用量：LSH ではハッシュテーブルを構築する際にメモリを使用する。バケット数L の増加に伴って必要なハッシュテーブル数が単調に増加するため、メモリ使用量も単調に増加する。

【００３５】

局所性に鋭敏なハッシュ関数と探索効率

局所性に鋭敏なハッシュ関数の探索効率は、非特許文献１に示されている。後述する分析の基盤とするため、ここで紹介する。

式（１）のハッシュ関数は局所性に鋭敏(Locality-Sensitive)なハッシュ関数と呼ばれる。局所性に鋭敏なハッシュ関数とは、近いベクトル同士は同じハッシュ値を取る確率が高く、遠いベクトル同士は同じハッシュ値を取る確率が低いという性質を持つハッシュ関数である。数式を用いて具体的に書けば次のようになる。また、図２は以下の記述を図示したものである。

【００３６】

【数４】

$$\begin{cases} \mathrm{P}[h(\boldsymbol{q}) = h(\boldsymbol{p})] \geqq p_1 & \text{for } \boldsymbol{p} \in B(\boldsymbol{q}, r_1) \\ \mathrm{P}[h(\boldsymbol{q}) = h(\boldsymbol{p})] \leqq p_2 & \text{for } \boldsymbol{p} \notin B(\boldsymbol{q}, r_2) \end{cases} \tag{3}$$

ここでB（q, r）はクエリq から半径r 以内にある点の集合を表す。したがって式（３）は、クエリq から距離$r_1$ 以内の点は確率$p_1$ 以上でクエリと同じハッシュ値を持ち、クエリq から距離$r_2$ 以上の点は確率（１－$p_2$）以上でクエリと別のハッシュ値を持つ。ここで$r_1 < r_2$ かつ$p_1 > p_2$ を満たすとする。

【００３７】
局所性に鋭敏なハッシュ関数を用いるLSH の探索効率は、次式で与えられる基準ρを用いて記述される。
【数５】

$$\rho = \frac{\log 1/p_1}{\log 1/p_2} \qquad (4)$$

ρは、最近傍の点がクエリq から距離$r_1$ 以内にある確率$p_1$ が大きければ小さくなり、かつ、最近傍の点がクエリq から距離$r_2$ 以上にある確率$p_2$ が小さければ小さくなる。よって、ρの値は小さいほどよい。
【００３８】
非特許文献１によると、必要なメモリ使用量は$O(dn+n^{1+\rho})$で、計算時間のほとんどは$O(n\rho)$回の距離計算で占められる。ここで、O（M）あるいはO（Mρ）は、問題を解くために必要なおおよその計算量の表記方法であって、例えば、O（M）はMが定まったときの計算量が$\alpha_1 M+\alpha_2$以下で収まることを表す。ただし、$\alpha_1$，$\alpha_2$は定数である。また、例えば、O（$M^3$）は$\alpha_1 M^3+\alpha_2 M^2+\alpha_3 M+\alpha_4$以下で収まることを表す。ただし$\alpha_1$，$\alpha_2$，$\alpha_3$，$\alpha_4$は定数である。また、dはベクトルデータの次元数であり、n は扱うベクトルデータの数である。このとき、$O(n^{\rho}\log_{1/p_2} n)$回のハッシュ関数の評価が必要である。
【００３９】
≪この発明に係る最近傍探索の手法≫
この発明は、近似最近傍探索の手法において、データベースにデータを重複登録することで計算時間とメモリ使用量を削減する手法を提案する。図３にこの発明の手法の概要を示す。まず、図３（ａ）では大き目のバケット数L を持つLSH を構築する。そして図３（ｂ）のように、１つのバケット（「コピー先」バケット）に残りのバケット（「コピー元」バケット）の情報をコピーし、図３（ｃ）のようにコピー元バケットを削除する。これにより、バケット数が大きいときの性能を少数のバケットのみで実現できる。
【００４０】
以下、この発明の手法の処理の詳細を、図４を参照しながら述べる。最初に前節の「コピー元バケット」と「コピー先バケット」の代わりに、「コピー元ハッシュ関数群」と「コピー先ハッシュ関数群」を作成する。どちらのLSH にも同じデータが登録されているが、ハッシュ関数やバケットが異なる。そして、
【００４１】
手順（１）：コピー先ハッシュ関数群に登録されているデータを１つ選ぶ。説明の都合上、このデータをY と呼ぶ。次に
手順（２）：Y をクエリに見立ててコピー元ハッシュ関数群で探索し、
手順（３）：コピー元ハッシュ関数群でY と同じバケットに入った各データについて、同じバケットに入った回数を数える。その後、
手順（４）：同じバケットに入った回数が閾値t以上のデータのみを選択して、コピー先ハッシュ関数群のY が属していたバケットに追加登録する。ただし、既に登録済みの場合は追加登録の対象とはしない。
【００４２】
この処理を、Y とするデータを変えながら一定数のデータに対して行う。全データのうち、この処理に用いるデータの割合をβで表す（０≦β≦１）。最後にコピー元ハッシュ関数群を破棄し、コピー先ハッシュ関数群を通常のLSH の代わりに用いる。即ち、コピー先ハッシュ関数群のハッシュテーブルのみを残し、コピー元ハッシュ関数群のハッシュテーブルのあったメモリ領域を開放してデータベースを構築する。なお、LSH には「バケットに登録する」という概念はなく、データベース内の各ハッシュテーブルのビンに登録された距離計算対象の積集合を求めることによりバケットが決まる。よって、データのバケットへの追加登録は、具体的には当該バケットを構成する全てのハッシュ関数にデータを

登録する。

【００４３】

データの追加登録は、クエリとしての物体の画像が与えられたときその画像に対応するクエリベクトルが多く発生するバケットで実行するのが効果的である。データの分布とクエリの分布が同一だと仮定すると、Y とするデータを一様乱数に基づいて選択すればデータの分布が密なバケットは多数選ばれ、粗なバケットはあまり選ばれないため、前述のようにクエリが多く発生するバケットでデータの追加登録が多く実行される。そのため、本稿ではデータの分布とクエリの分布が同一だと仮定し、Y とするデータを一様乱数に基づいて選択する。

【００４４】

なお、この実施形態では、データの重複登録によってメモリ使用量が大幅に増加しないように実装した。具体的には、距離計算に用いるベクトルデータを保持しておくテーブルをハッシュテーブルとは別に用意して、各ハッシュ値を持つデータの番号のみをハッシュテーブルに保持した。これにより、重複登録によって増加するメモリ使用量はデータの番号を表す分のみとなる。

【００４５】

≪実験例≫

従来のLSH とこの発明による近似最近傍探索手法の結果を比較し、この発明の有効性を確かめる実験を行った。実験にはMulti-PIE Face database （R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker,“Multi-pie,”Proc. 8th IEEE Int'l Conf. on Automatic Face and Gesture Recognition, 2008参照）に含まれる754,200 枚の画像に顔検出（T. Mita, T. Kaneko, B. Stenger, and O. Hori,“Discriminative feature co-occurrence selection for object detection,”IEEE Trans. PAMI, pp.1257-1269, July 2008参照）を適用し、得られた316,089枚の画像に正規化（T. Kozakaya and O. Yamaguchi,“Face recognition by projection-based 3d normalization and shading subspace orthogonalization,”Proc 7th Int' Conf. on Automatic Face and Gesture Recognition, pp.163-168, 2006参照）を施して、928 次元のLBP 特徴（T. Ahonen, A. Hadid, and M. Pietikainen,“Face description with local binary patterns: Application to face recognition,”IEEE Trans. PAMI, vol.28, no.12, pp.2037-2041, Dec. 2006参照）を抽出し、さらに主成分分析によって100 次元に圧縮した。これらの特徴ベクトルからランダムに10,000 をデータベース登録用に選び、別の10,000 をクエリ用に選んだ。

【００４６】

事前に全探索で最近傍データを求めておき、それぞれの近似最近傍探索手法で求めた近似最近傍データが一致した割合を精度とした。計算機はOpteron 6174 (2.2GHz) を用いた。追加登録に用いるデータ数に関するパラメータβの値として0.001, 0.01, 0.1を用いた。これらはそれぞれ追加登録に10 個、100 個、1000個のデータを用いることを意味する。

【００４７】

図５に精度と処理時間の関係を、図６に精度とメモリ使用量の関係を表したグラフを示す。図５および図６に係る実験で、距離計算対象を絞り込むために用いるハッシュ関数の数に係るパラメータは$k_1=k_2=1$、ハッシュ幅に係るパラメータは$w_1=w_2=1000$で、各グラフともに共通である。また、表１に精度、計算時間、メモリ使用量の比較結果を抜粋する。表１に係る実験は、パラメータとして、$k_1=k_2=1$, $w_1=w_2=1000$を用いた。コピー先ハッシュ関数群のパラメータは$w_1$, $k_1$, $L_1$ のように添字1 をつけて表し、コピー元ハッシュ関数群のパラメータは$w_2$, $k_2$, $L_2$ のように添字2 をつけて表している。表１のパラメータt は、クエリに見立てたデータY と同じバケットに入った回数が何回以上のものを登録するかの閾値である。

【００４８】

【表１】

| 手法 | パラメータ | | | 精度 | 計算時間 | メモリ使用量 |
|---|---|---|---|---|---|---|
| | $\beta$ | $L_2$ | $t$ | (%) | (ms) | (MB) |
| LSH($L=1$) | | | | 46.5 | 0.32 | 16.0 |
| LSH($L=20$) | | | | 99.9 | 3.91 | 18.3 |
| 本発明 ($L_1=1$) | 0.001 | 1 | 1 | 68.0 | 0.54 | 16.0 |
| | 0.01 | 1 | 1 | 93.9 | 0.73 | 16.2 |
| | 0.1 | 1 | 1 | 99.3 | 0.69 | 16.4 |
| 本発明 ($L_1=1$) | 0.001 | 20 | 1 | 90.2 | 0.66 | 16.2 |
| | 0.01 | 20 | 1 | 99.2 | 0.72 | 16.4 |
| | 0.1 | 20 | 1 | 99.9 | 0.69 | 16.4 |
| 本発明 ($L_1=1$) | 0.001 | 20 | 10 | 49.1 | 0.34 | 16.0 |
| | 0.01 | 20 | 10 | 65.4 | 0.47 | 16.0 |
| | 0.1 | 20 | 10 | 90.1 | 0.63 | 16.2 |

【００４９】

図５および図６から、この発明の手法は従来手法であるLSH と比べて、同一の処理時間のときやメモリ使用量のときの精度が著しく向上していることがわかる。そのため、同等の精度で比べると、処理時間もメモリ使用量も減少している事が分かる。表１から、β=0.1, $L_2$ = 20, t = 1 のときに精度99.9%、計算時間0.69ms、メモリ使用量16.4MB を達成している。LSH においてL = 20 のときは精度99.9%、計算時間3.91ms、メモリ使用量18.3MB であるので、これらを比べると、計算時間が18%に、メモリ使用量は90%に削減されたことがわかる。これは、LSHと同等の精度を達成するのに必要なハッシュ関数群の数L が小さくてすむ事に起因していると考えられる。$L_2$ とt を変化させたときの性能を比べると、$L_2$ が大きいときのほうが性能がよく、t が大きいときは性能が悪かった。

【００５０】

≪分析≫

この発明の手法によるρの変化

前節でこの発明の手法の実験的な有効性を確認した。本節では、前述したLSH の探索効率の基準ρ（式（４）参照）がこの発明の手法ではLSH に比べて小さくなることを示し、解析的にこの発明の手法の有効性を示す。

LSH は局所性に鋭敏なハッシュ関数を用いるので、図４の手順（２）のように注目データの近傍点をコピー元ハッシュ関数群で探索すると、注目データに近い点ほど高い確率で探索され、遠い点ほど低い確率で探索される。そのため、コピー先ハッシュ関数群ではクエリと同じビンに属するデータが増加する。すなわち、クエリq から距離$r_1$ 以内の点がクエリと同じハッシュ値を持つ確率$p_1$ と、クエリq から距離$r_2$ 以上の点がクエリと同じハッシュ値を持つ確率$p_2$ は共に増大する。ただし、注目データに近い領域ではより多くのデータが探索されるため、$p_1$ は$p_2$ より大きくなる。この結果、ρが減少する。

【００５１】

コピー元ハッシュ関数群のバケット数$L_2$ と閾値t の関係

図４の手順（３）では探索回数が閾値t 以上のデータのみをデータベースに追加登録する。このときに用いる閾値を上手に調整できれば、近傍点として選ばれる確率が高い点を選択的に追加し、ρの減少をさらに促すことができる。本節ではどのような閾値が望ましいのかを考察する。

ここではコピー元ハッシュ関数群の$L_2$ 個のバケットを探索して、あるデータY がy 回

みつかったと仮定する。この事象が起こる確率P（y）は次式の二項分布で与えられる。

【００５２】

【数６】

$$\mathrm{P}(y) = \binom{L_2}{y} p^y (1-p)^{L_2-y} \qquad (5)$$

【００５３】

ここで

【数７】

$$\binom{L_2}{y} = \frac{L_2!}{y!(L_2-y)!} \qquad (6)$$

である。確率p は注目データからデータY までの距離の関数であり、両者の距離が近いほど大きくなる。

【００５４】

図７は式（５）をプロットしたもので、縦軸は、コピー元ハッシュ関数群でy 回探索される確率P（y）を表している。この式に値を入れて計算してみると、大きな確率p を持つY に近い点（例えばp = 0.5）と小さな確率p を持つY から遠い点（p = 0.3）では分布が異なり、近い点のほうが探索される回数ｙが多いことがわかる。また、データベースに追加登録をするか否かを決める探索回数の閾値t とデータが選択される確率の関係について考えてみると、閾値t をt = １としたときよりも、p = 0.5のときの期待値$0.5L_2$ としたとき（すなわち、図７（ａ）に示す$L_2$ = 10 の場合はt = 5，図７（ｂ）に示す$L_2$ = 20のときはt = 10）のほうがY から近い点と遠い点の確率差は大きいことから、適切に閾値を設定することが必要であること、そしてその値は最近傍点が探索される確率を基準に定められる可能性があることがわかる。

【００５５】

次に、コピー元ハッシュ関数群のバケット数$L_2$ がこの発明の手法の性能に及ぼす影響を考える。閾値t を先程と同様にp = 0.5 のときの期待値$0.5L_2$ としたとき、図８（ａ）よりも図８（ｂ）のほうがY から近い点と遠い点の確率差は大きい。したがって、コピー元ハッシュ関数群のバケット数$L_2$ を大きくすれば、性能が向上する場合があると考えられる。これは実験結果とよく一致する。

【００５６】

前述した実施の形態の他にも、この発明について種々の変形例があり得る。それらの変形例は、この発明の範囲に属さないと解されるべきものではない。この発明には、請求の範囲と均等の意味および前記範囲内でのすべての変形とが含まれるべきである。

【産業上の利用可能性】

【００５７】

この発明は、ハッシュに基づく近似最近傍探索手法であるLSHにおいて、データベースに登録されているデータの重複登録によって、同一精度を実現するために要する計算時間とメモリ使用量を削減する手法を提供する。実験によりこの発明の手法の有効性を確認し、さらに非特許文献１で用いられているLSH の探索効率の基準ρを用いて解析的に性能が向上することを示した。

この発明の手法を用いることによって、近似最近傍探索によるデータの探索に必要な計算時間とメモリ使用量を従来よりも削減することができる。

【００５８】

データを余分に登録すると直感的に性能に悪影響が出ると予測されるが、それに反して性能が向上したのは、クエリ（の予測値）の近傍点のみを選択的にバケットに登録することにより、真の最近傍点がクエリと同じバケットに入る確率が上昇したからである。そし

て、このことによる計算時間とメモリ使用量の上昇がごくわずかであったことが考えられる。解析的には、LSH の探索効率の基準であるρを減少することができたためであると考えられる。

【符号の説明】

【００５９】

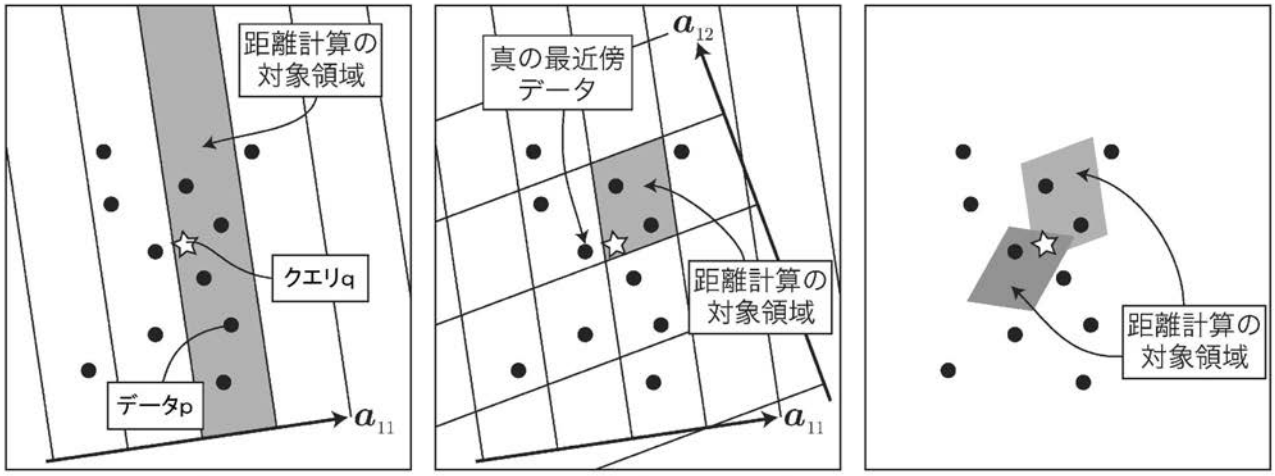
１１：対応表

a：ベクトル

p：データ

q：クエリ

【図１】

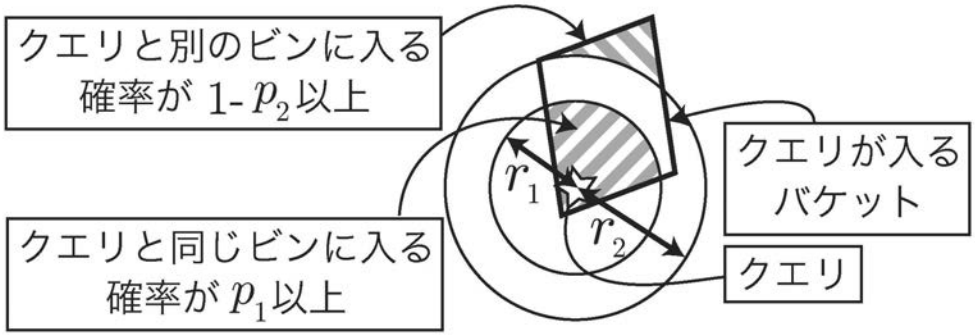



(a) ハッシュ関数が 1 つの場合.

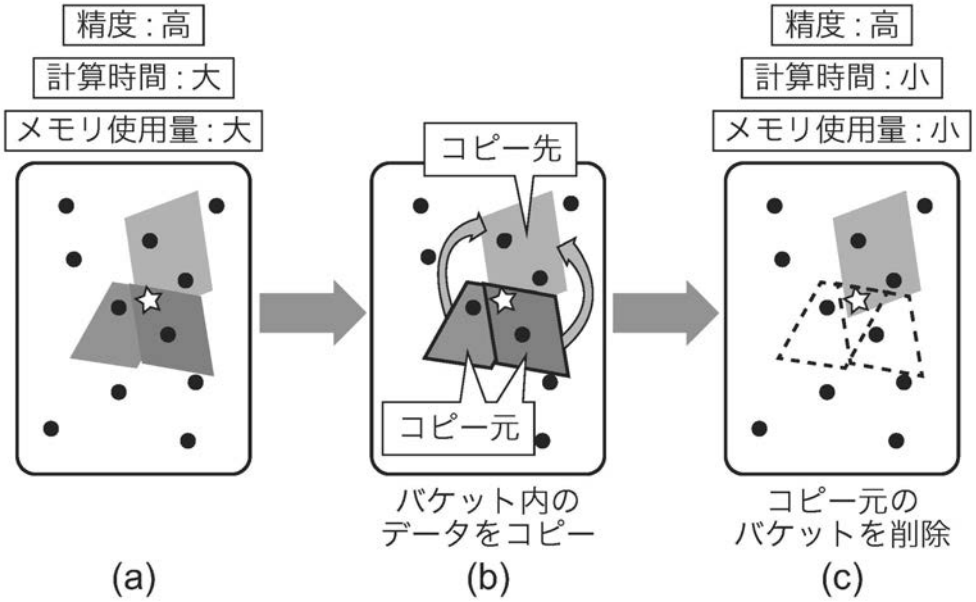
(b) 2 つのハッシュ関数で作られるバケットが 1 つの場合.

(c) バケットが 2 つの場合.

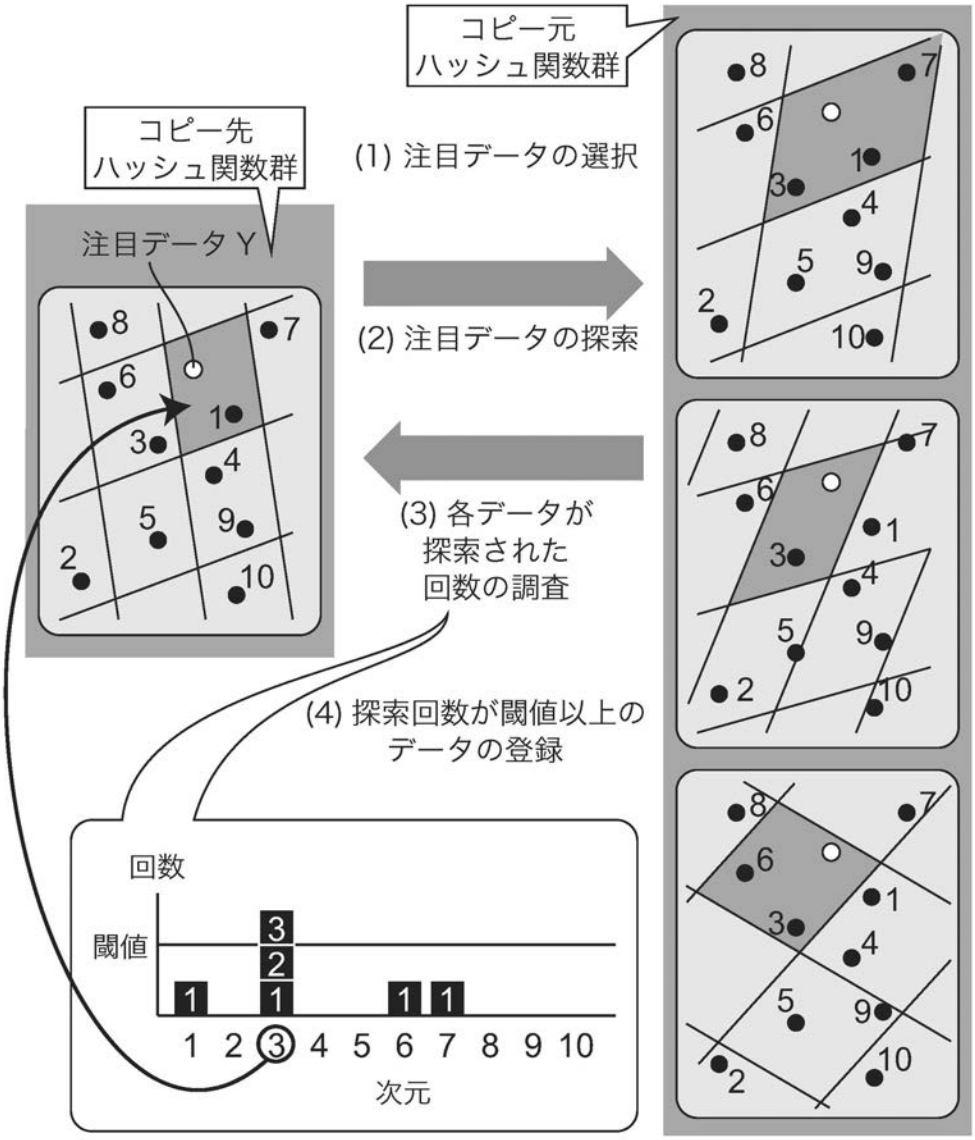
【図２】


クエリと別のビンに入る
確率が 1-$p_2$以上
クエリと同じビンに入る
確率が $p_1$以上
$r_1$
$r_2$
クエリが入る
バケット
クエリ

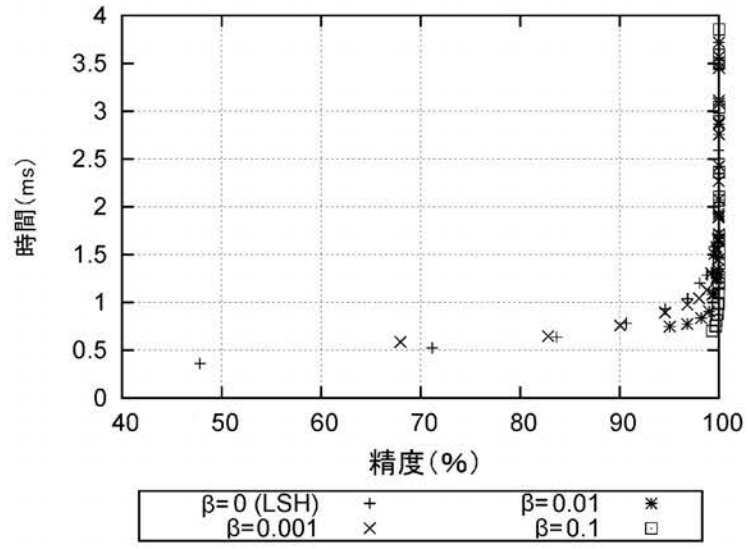
【図３】


精度：高
計算時間：大
メモリ使用量：大
コピー先
コピー元
バケット内の
データをコピー
精度：高
計算時間：小
メモリ使用量：小
コピー元の
バケットを削除
(a)
(b)
(c)

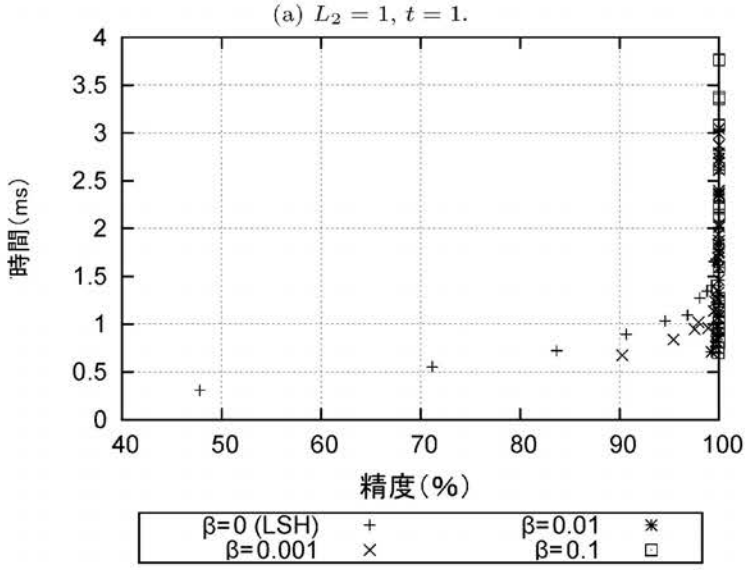
【図４】


コピー元
ハッシュ関数群
コピー先
ハッシュ関数群
注目データＹ
(1) 注目データの選択
(2) 注目データの探索
(3) 各データが
探索された
回数の調査
(4) 探索回数が閾値以上の
データの登録
回数
閾値
1 2 3 4 5 6 7 8 9 10
次元

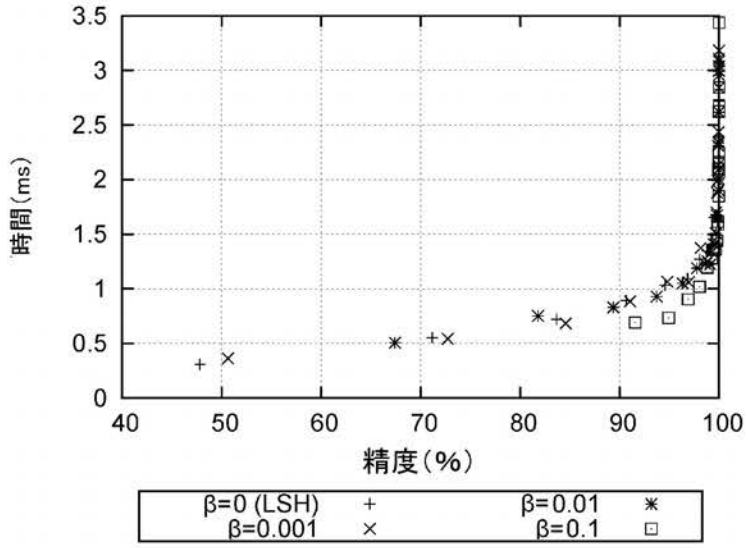
【図５】




(a) $L_2 = 1$, $t = 1$.

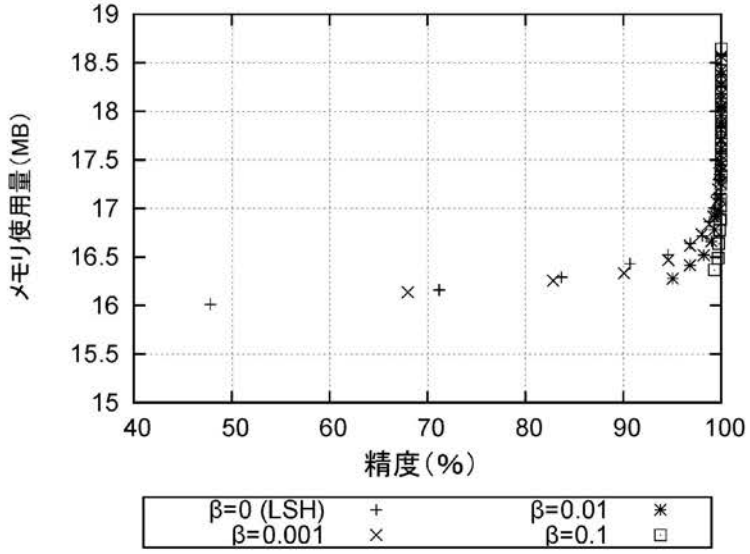



(b) $L_2 = 20$, $t = 1$.

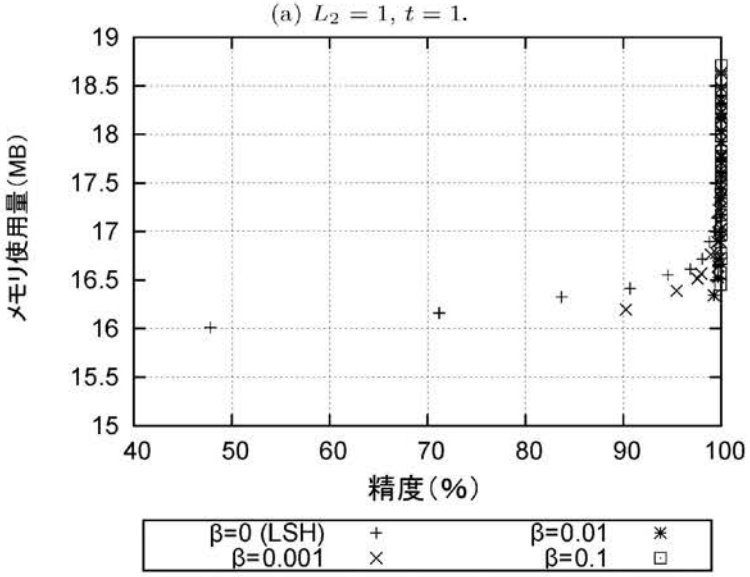



(c) $L_2 = 20$, $t = 10$.

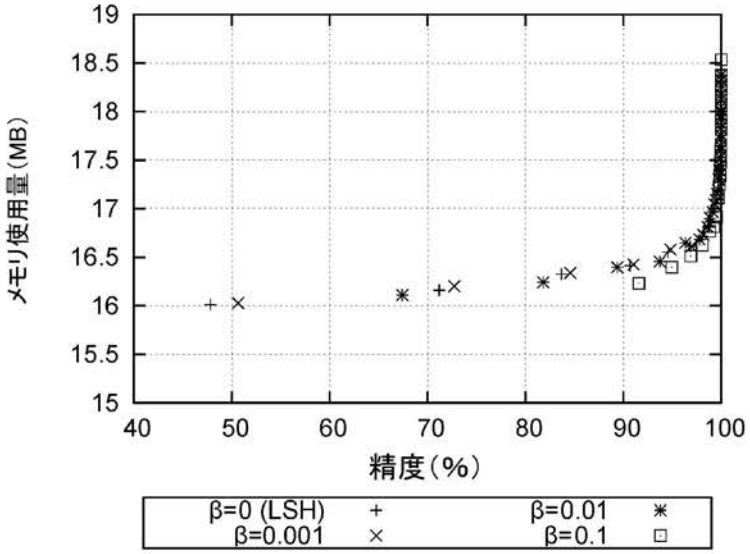
【図６】




(a) $L_2 = 1$, $t = 1$.

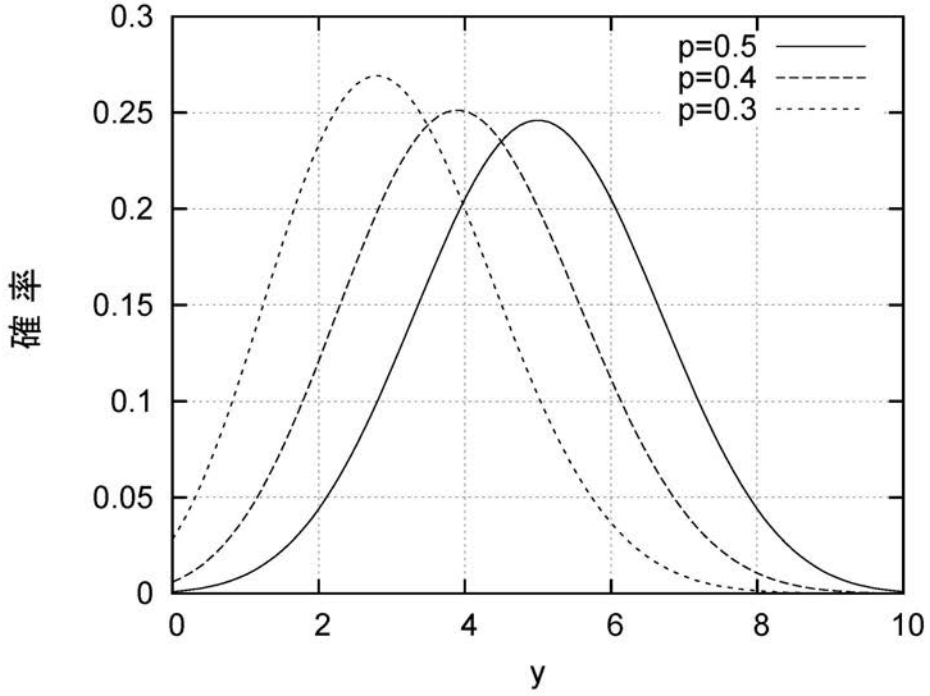



(b) $L_2 = 20$, $t = 1$.

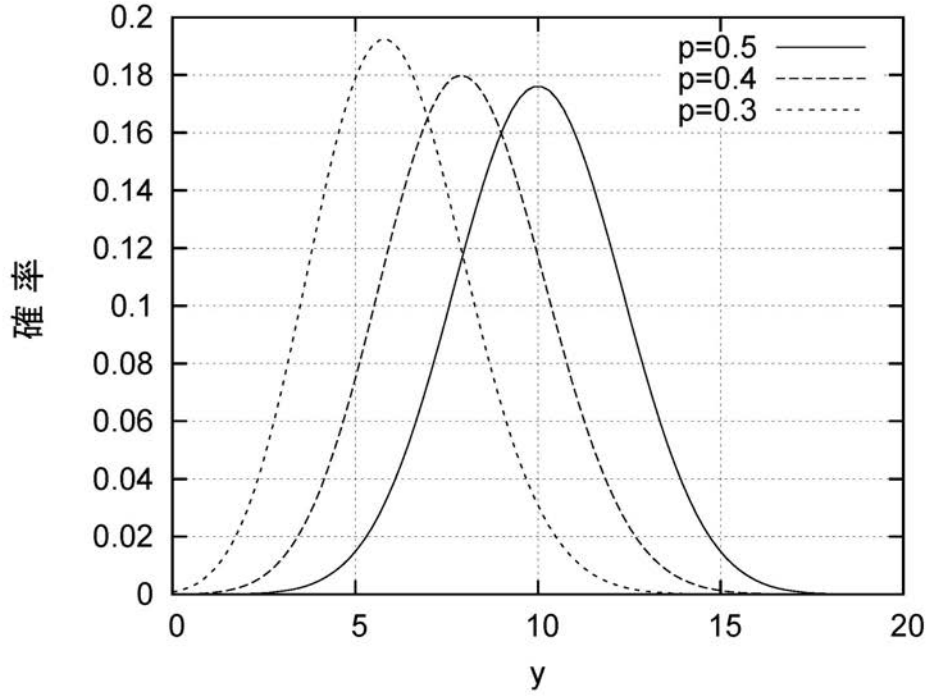



(c) $L_2 = 20$, $t = 10$.

【図７】

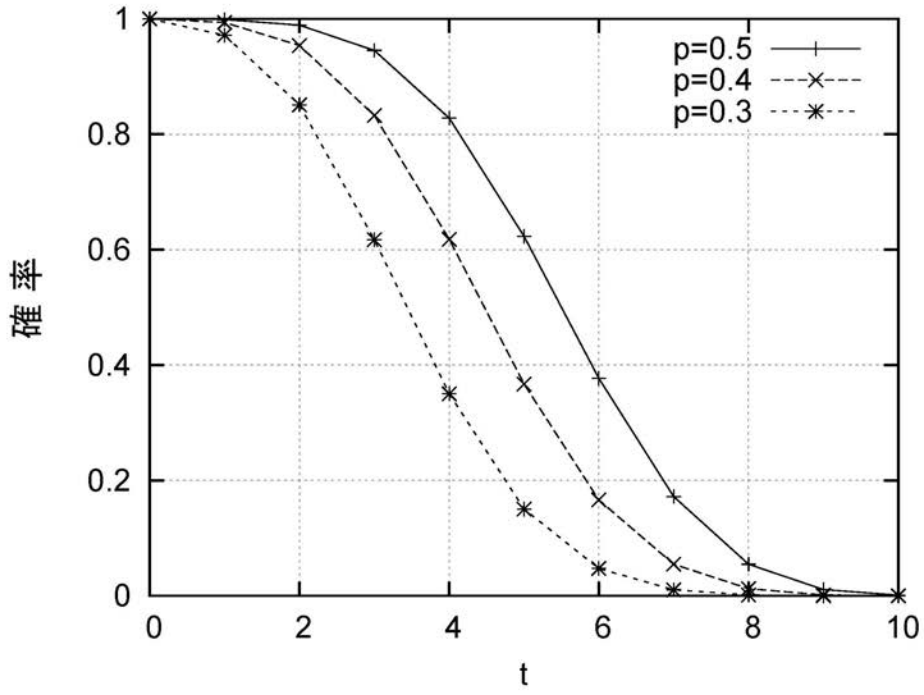



(a) $L_2 = 10$ のとき.

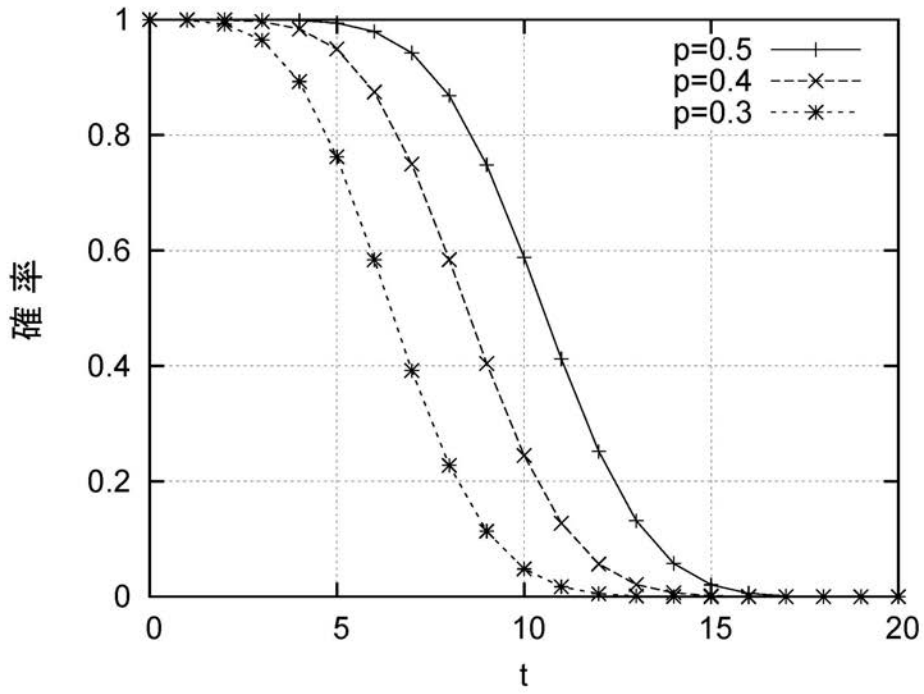



(b) $L_2 = 20$ のとき.

【図８】

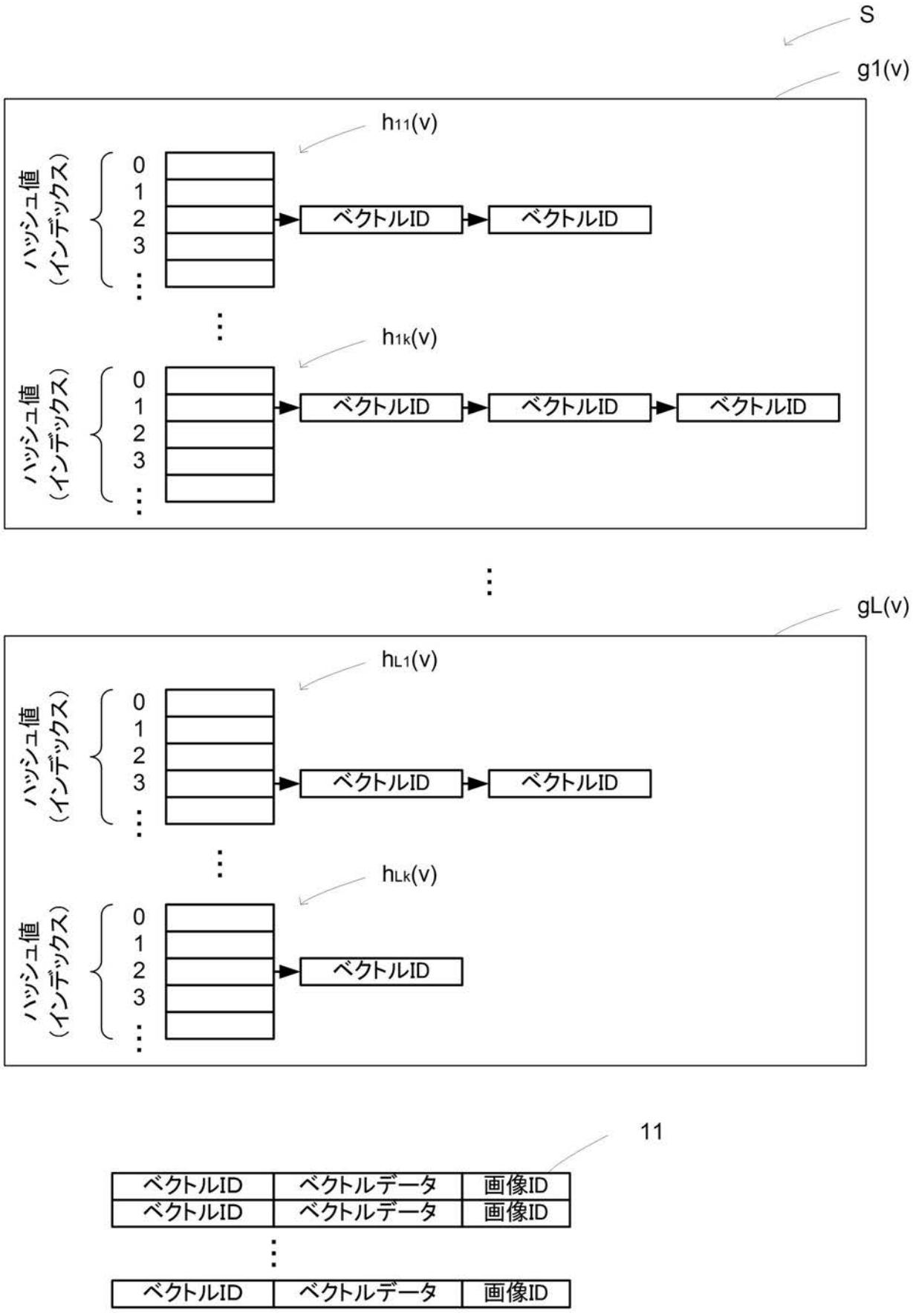



(a) $L_2 = 10$ のとき.




(b) $L_2 = 20$ のとき.

【図９】


S
g1(v)
h11(v)
ハッシュ値
（インデックス）
0
1
2
3
ベクトルID
ベクトルID
h1k(v)
ベクトルID
ベクトルID
ベクトルID
gL(v)
hL1(v)
ベクトルID
ベクトルID
hLk(v)
ベクトルID
11
ベクトルID
ベクトルデータ
画像ID
ベクトルID
ベクトルデータ
画像ID
ベクトルID
ベクトルデータ
画像ID

フロントページの続き

(72)発明者　岩村　雅一
大阪府堺市中区学園町１番１号　公立大学法人大阪府立大学内
(72)発明者　黄瀬　浩一
大阪府堺市中区学園町１番１号　公立大学法人大阪府立大学内

Ｆターム(参考) 5B050 AA09　BA10　BA15　EA04　FA02