B-012

GPU を用いた直積量子化の近似最近傍探索における並列化実装

GPGPU implementation of approximate nearest neighbor search using product quantization

村上 明男† 若谷 彰良‡
Akio Murakami Akiyoshi Wakatani

1. はじめに

大規模なマルチメディアコンテンツから物体やシーン認識を効率的で精度高く行うことは重要な研究テーマの一つである．高次元特徴ベクトルを用いて最近傍探索することによって実現する手法の一つとして，直積量子化を用いた近似最近傍探索が注目されている[1]．この手法は，直積量子化と粗量子化による転置インデックスを併用して探索範囲と計算量を削減し，高速化を図っている．しかし，大規模なリファレンスデータを対象とした近似最近傍探索はデータ量が多く，リアルタイムな探索には更なる高速化が望まれる．

一方，近年の GPU は高性能化が進み，汎用プロセッサの演算性能を大きく超えている．従来の GPU は専用ハードウェアで構成されていたので，グラフィック処理や画像処理に特化され，数値計算やデータベース処理などの汎用アプリケーションには用いられなかった．しかし，最近の GPU は CUDA (Compute Unified Device Architecture) [2] などにより，汎用プログラミング言語で GPU をプログラムできるようになっており，GPGPU (General-Purpose computing on Graphics Processing Units) と呼ばれる．GPGPU はデータ依存の少ない並列アプリケーションを容易に実装できる．最適なメモリ利用やスレッド構成の最適化など，高速実行するためのチューニング項目は多いが，GPU を用いた並列実行は身近な技術になってきている．

そこで，本稿では，直積量子化による近似最近傍探索における GPU を用いた並列化実装について述べる．2 章では直積量子化による近似最近傍探索のアルゴリズムについて述べる．3 章ではそれを GPU で並列化する手法について述べる．4 章では並列化手法の実装と性能評価を述べ，5 章では本稿のまとめを示す．

2. 直積量子化に基づく近似最近傍探索

文献[1]に掲載されている直積量子化に基づく近似最近傍探索について述べる．このアルゴリズムでは，リファレンスデータを予めベクトル量子化により空間的複雑さの削減をし，転置インデックスというデータ構造を利用してメモリ上に格納する．さらに，探索ベクトルとの距離計算を，予め計算したルックアップテーブルの参照に置換して高速化する．

2.1 転置インデックスの生成

N 個のリファレンスデータの中から適当な数のリファレンスベクトルを抽出し，ベクトル量子化の手法を用いて粗量子化コードブックを求める．次に，すべてのリファレンスデータを粗量子化コードブックで量子化し，同じコードブックに属するリファレンスデータを転置インデックスというデータ構造に格納する．転置インデックスには，リファレンスデータに対する情報として，D 次元のデータとそのインデックスの代表値であるセントロイドとの差（残差）を求める．D 次元の残差ベクトルは S 個の部分ベクトルに分けて，その部分ベクトルをベクトル量子化の手法を用いて直積量子化コードブックを求める．

2.2 近似最近傍探索

Q 個の探索ベクトル q_i に対して，以下の手順を行う．始めに，q_i を粗量子化コードブックを用いて量子化し，距離の近いインデックスを W 個選択する．選択したインデックスのセントロイドと q_i との残差を求め，その残差と直積量子化のコードブックとの距離を計算し，ルックアップテーブルに格納する．次に，選択したインデックスに属するリファレンスデータ r_j それぞれについて，q_i の部分ベクトルと r_j の部分ベクトルとの距離をルックアップテーブルから取り出し合計値を求め，距離の小さいものから K 個選択する．それを近似 K 最近傍とする．

3. 近似最近傍探索の GPU による並列化手法

2.2 節の近似最近傍探索において，GPU を用いた並列化について述べる．本稿では，一つの探索ベクトルに対する近似最近傍探索を並列化するため，Q 個の探索ベクトルそれぞれの処理は 1 個のスレッドブロックに割り当てる．スレッドブロック内では，始めに，転置インデックスに属するリファレンスデータを分割し，分割したリファレンスデータの中から K 個の最近傍候補をスレッドごとに求める．次に最近傍候補の中からスレッドブロック内の全体としての最近傍を求める．

ここで，K 個の最近傍を見つける場合，各スレッドでは何個の最近傍を見つけるようにすれば高精度な近似最近傍探索ができるかについて述べる．最近傍となる候補の出現動向を調べるため，近似 K 最近傍探索の予備実験を行った結果，転置インデックスに属する最近傍候補の個数にはばらつきがあり，またインデックスで粗量子化における距離の近い方（上位）に最近傍のリファレンスデータが多く分布することがわかった．P 並列のとき，各並列要素が K/P の最近近傍候補を生成し，全体でまとめると K 個の最近傍となるが，最近傍の分布が事前には確定できないため，各並列要素が最近傍となるリファレンスデータを選ばない場合も存在する．そこで，$m(K / P < m < K)$ を設定し，各並列要素が m 個の最近傍を生成し，その後，m×P 個から K 個の最近傍を選ぶ方法を考える．図 1 のように，インデックスに属するリファレンスデータを一次元に並べ (R 個)，各並列要素が R/P 個のリファレンスデータから m 個の最近傍を抽出して，その後 K 個の近似最近傍を求める場合，上位のインデックスの最近傍候補を担当する並列要素から全

†甲南大学大学院自然科学研究科
‡甲南大学知能情報学部

体としての最近傍が多く抽出される可能性があるため，m の値を大きめに設定する必要がある．そこで，図 2 のように，擬似行列転置を用いて各転置インデックスから均等に最近傍候補を選ぶ場合，各並列要素が担当するリファレンスデータのインデックスが分散するため，各並列要素から抽出される全体としての最近傍候補の数も小さくなると考える．つまり，図 2 の方法で m の値は図 1 の方法より小さくできると考えられる．

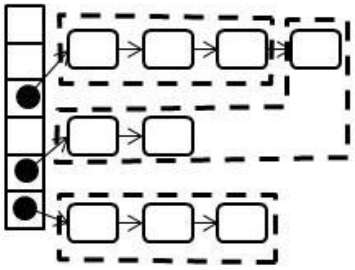

図 1: シンプルな方式

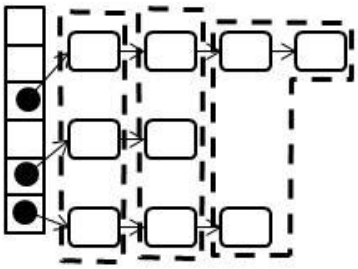

図 2: 擬似行列転置を用いる方式

元の近似最近傍アルゴリズムで検出される最近傍の中で，その m で検出された最近傍の割合を再現率と定義し，図 1 および図 2 の手法で P および m の値の違いによる再現率の変化を調べた．リファレンスデータは 100,000 個の 128 次元ベクトルで構成し，100 個の探索ベクトルに対し，256 個の近似最近傍 (= K) を見つける探索を行うと仮定する．P が小さい場合(P = 4 < K = 256) は，図 1 の手法では m = 92 で再現率が 100%になったのに対して，図 2 の手法では m = 74 で再現率が 100%になった．一方，P ≥ K の場合は，図 1 の手法でいずれの探索ベクトルに対しても m = 7 で再現率が 100%となり，図 2 の手法を用いる場合の効果があまり得られない．以上より，1 個の探索ベクトルに 1 個のスレッドブロックを割り当てる場合はスレッドブロック内のスレッド数が K よりも小さくなるので，擬似行列転置を用いると高速化の効果が大きいことがわかる．

4. 性能評価

3 章で述べた擬似行列転置を用いた並列化プログラムを CUDA で実装し，評価を行った．リファレンスデータは 100,000 個の 128 次元ベクトルで構成し，100 個の探索ベクトルに対し，256 個の近似最近傍 (= K) を見つける探索を行うと仮定する．また，1 個の探索ベクトルに 1 個のスレッドブロックを割り当て，スレッド数を P とする．並列化プログラムでは，各スレッドが P 分割された擬似行列転置のリファレンスデータから m 個の最近傍候補を求め，その後 P×m 個の最近傍候補から K 個の最近傍を求める．GPU は NVIDIA GeForce GTX590 (1024 コア) で，CPU は Intel Core i7 875K (2.93 GHz)，メモリは 4 GB を用いる．OS は Yellow Dog Enterprise Linux for CUDA を，CUDA のバージョンは 4.0 を用いる．

まず，CPU のみで近似最近傍探索を行った場合と GPU を用いて並列化を行った場合とを比較する．後者では 3 章で K = 256 のときに再現率が 100%となる P = 4, m = 74 をそれぞれ設定した．前者の実行時間が 5.032[sec]であったのに対して，後者の実行時間は 0.149[sec]であった．

次に，P の値を変化させてから P×m ≥ K×1.5 を満たす最小の m を決定し，並列化プログラムの実行時間を計測した．ただし，P = 1 のときは m = 256 として実行した．実行時間を表 1 に示す．今回の仮定では P = 16 のときに実行時間が最小となった．

表 1:並列化プログラムの 並列度ごとの実行時間

並列度(P)	m	実行時間[sec]
1	256	0.211
2	192	0.167
4	96	0.152
8	48	0.147
16	24	0.143
32	12	0.151
64	6	0.152
128	3	0.159
256	2	0.185

4. 終わりに

GPU を用いた直積量子化の近似最近傍探索における並列化実装の手法について述べた．P 並列で近似 K 最近傍探索を行うとき，P < K の場合に擬似行列転置を用いて十分な再現率を実現しながら高速化の効果を得ることができる．今後は，擬似行列転置を用いる場合にどのように P および m を設定すれば高い再現率を保ちつつ高速化できるかについて考えていきたい．

謝辞

本研究の一部は私立大学等経常費補助金特別補助「大学間連携等による共同研究」によるものである．

参考文献

[1] Herve Jegou, Matthijs Douze and Cordelia Schmid, “Product quantization for nearest neighbor search,” IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 33, no. 1, pp. 117-128, 2011.

[2] “Parallel Programing and Computing Platform | CUDA | NVIDIA,” http://www.nvidia.com/object/cuda_home.html.