ソーテッドコードブックベクトル量子化の並列処理

中野恵一[†] 笠原博徳[††]

ベクトル量子化は能率の良いデータ圧縮法のひとつとして音声や画像の量子化に利用されている．このベクトル量子化の単一プロセッサ上での高速処理のために，筆者らは，量子化歪みを増大させることなく，従来法に比べ歪み計算量を削減することができるソーテッドコードブックベクトル量子化(VQ-SC) をすでに提案している．本論文では，このVQ-SCにおける探索を並列化し，より大規模なベクトル量子化を高速に実現する手法を提案する．本手法では使用するマルチプロセッサ・アーキテクチャにより，1) コードブック分割並列探索，あるいは2) コードブック共有セルフスケジューリング並列探索，を使い分け，各種アーキテクチャのマルチプロセッサ上でVQ-SCの効率的な並列処理を実現する．本手法の有効性は，分散メモリ型マルチプロセッサAP1000およびcenju-3，共有メモリ型マルチプロセッサFX/4，分散共有キャッシュメモリ型マルチプロセッサKSR1上での，画像のベクトル量子化に要する処理時間の評価により確認された．

Parallel Processing for Fast Vector Quantization with Sorted Codebook

KEIICHI NAKANO[†] and HIRONORI KASAHARA[††]

This paper proposes two parallel processing schemes for fast vector quantization with sorted codebook. The vector quantization with sorted codebook (VQ-SC) algorithm has been proposed by the authors to minimize sequential processing time of vector quantization. The proposed two parallel search schemes for VQ-SC with 1) distributed codebook and 2) self-scheduling using shared codebook allow us to handle larger codebook sizes and vector dimensions with least encoding operations. The both schemes are used according to a multiprocessor architecture in hand. Effectiveness and practicality of the proposed schemes are demonstrated with real image encoding on two distributed memory multiprocessor AP1000 and cenju-3, a shared memory multiprocessor FX/4, and a distributed shared cache multiprocessor KSR1.

1. はじめに

近年，マルチメディア情報通信の要素技術として，画像や音声などのデータ圧縮技術が重要になっている．このデータ圧縮技術のうち，ベクトル量子化[1),2)]は，ベクトルの次元数を十分大きくすることでレート歪み限界と呼ばれる情報圧縮限界に漸近することが理論的に証明されている[3)]など，優れた符号化性能を示すことが知られている．このベクトル量子化は，入力ベクトル $X=(x_1,x_2,\cdots,x_M)$ に対して，コードブック $\{Y_1,Y_2,\cdots,Y_N\}$ の中から X との歪みを最小にする最適出力ベクトル $Y_i=(y_{i,1},y_{i,2},\cdots,y_{i,M})$ を探索し X を Y_i に写像することと定義される．ここで M はベクトルの次元数，N はコードブックのサイズである．

ベクトル量子化をナイーブに実現する全探索法[4)]では，入力されたベクトルに対する最適な出力ベクトルを発見するために，入力ベクトルとすべてのコードブックベクトルとの全次元にわたる歪み（ベクトル間距離）を計算するので，その計算量は $O(MN)$ であり，量子化時間が大きいという問題がある．そこで従来より様々な高速化手法[5)~7)]が提案されており，筆者らも，量子化歪みを増大させないことを保証しながら高速な探索を可能とするソーテッドコードブック・ベクトル量子化手法[8)] (Vector Quantization with Sorted Codebook: VQ-SC) をすでに提案している．

VQ-SCは，探索が終了した範囲で得られる暫定的な最小歪みである「歪みの上限値」と，入力ベクトルとそれに対して最終的に得られる最適出力ベクトルとの歪みがその値よりも小さくならないと保証できる

† オリンパス光学工業（株）映像システム部
Imaging System Department, OLYMPUS Optical Co., Ltd.
†† 早稲田大学電気電子情報工学科
Department of Electrical, Electronics and Computer Engineering, Waseda University

「歪みの下限値」との比較により，探索領域を限定し，歪み計算回数を大幅に減少させる手法である．

しかしベクトル量子化において高い符号化性能を実現するためには前述のとおり次元を大きくする必要があり，その場合，単一プロセッサでは処理時間の点で不十分である．そこで本論文では，VQ-SC における探索を，使用するマルチプロセッサ・アーキテクチャを考慮して並列化する2つの手法を提案する．そしてこれらの有効性を，分散メモリ型の AP1000 および cenju-3，主記憶共有型の FX/4，分散共有キャッシュメモリ型の KSR1 といった各種マルチプロセッサシステム上での，画像のベクトル量子化に要する時間の評価により検証する．

本論文では，まず2章で，VQ-SC のアルゴリズムを，従来手法と対比して説明する．次に3章で，VQ-SC の並列化手法について述べる．そして4章で実並列計算機上での画像データのベクトル量子化により，提案する並列 VQ-SC の並列処理効果を評価する．

2. VQ-SC による高速探索法

本章では，最適出力ベクトルの発見を保証しながら，歪み演算量を大幅に削減し高速化する VQ-SC のアルゴリズムと性能を，従来手法である全探索法や Partial-distance 法[7]と比較して説明する．

2.1 全探索法

全探索法は，すべてのコードブックベクトルとすべての次元にわたる，$O(MN)$ の歪み計算を必要とする．

2.2 Partial-distance 法

Partial-distance 法は，「歪みの部分和」d_K

$$d_K = \sum_{j=1}^{K} (x_j - y_{i,j})^2, \quad 1 \le K \le M \qquad (1)$$

と，探索が終了した範囲での最小歪み（暫定解）である「歪みの上限値」d_{upper}

$$d_{upper} = \min_i \left\{ \sum_{j=1}^{M} (x_j - y_{i,j})^2 \right\},$$

($Y_i \in$ 探索が終了したコードブックベクトルの集合)

$$i = 1, 2, \cdots, T \quad (T\text{番目までの探索終了}) \qquad (2)$$

とを比較し，さらに“良い”「歪みの上限値」を与える可能性のない歪み計算を途中で打ち切り，歪み計算回数を削減する手法である．これは，各コードブックベクトルの次元方向の計算を削減するが，全探索法と同様にすべてのコードブックベクトルを探索する必要がある．

2.3 VQ-SC

VQ-SC は Partial-distance 法と同様に，探索途中の暫定解を「歪みの上限値」とする．それに加え，ソートされたコードブックを用いて得られる「歪みの下限値」を導入する．最適解は「歪みの上限値」と「歪みの下限値」に挟まれて存在するので，探索途中でこれらが一致すれば，その暫定解が最適解として確定し探索を終了できる．すなわち VQ-SC では，Partial-distance 法と同様に次元方向の計算が削減されるのに加えて，探索するコードブックベクトル数も大幅に削減できる．

2.3.1 コードブックのソート

VQ-SC では，「歪みの下限値」を求めるためにソートされたコードブックを用いる．これは与えられたコードブックから次の手順で探索の前処理として求めておく．まずコードブック $\{Y_1, Y_2, \cdots, Y_N\}$ を構成する各ベクトルの同じ次元 $j\,(1 \le j \le M)$ の要素で，多重集合 $W_j = \{y_{1,j}, y_{2,j}, \cdots, y_{N,j}\}$ を生成し，さらにこれを非減少順にソートして，順序つき集合 $Z_j = \{z_{1,j}, z_{2,j}, \cdots, z_{N,j}\}$ を構成する．この Z_j のグループ $Z = \{Z_1, Z_2, \cdots, Z_j, \cdots, Z_M\}$ がソートされたコードブックである．なおこれには，要素 $z_{p,j}$ が本来コードブックベクトル Y_i に属しているといった情報も保持する．**図1** のコードブックに対応するソートされたコードブックを，数直線を用いて**図2** に示す．

2.3.2 歪みの下限値

「歪みの下限値」d_{lower} とは，

$$d_{lower} = \sum_{j=1}^{M} d_{lower,j} = \sum_{j=1}^{M} \min_p (x_j - z_{p,j})^2$$

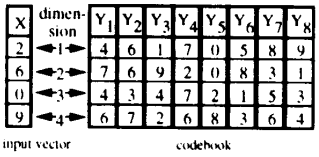

図1 ベクトル量子化の例
Fig. 1 An example problem of vector quantization.

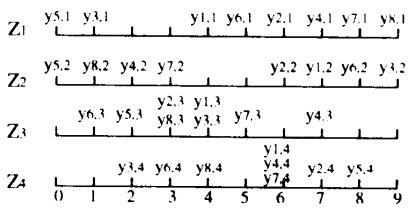

図2 図1 の例に対応するソートされたコードブック
Fig. 2 Sorted codebook corresponding to the example in Fig. 1.

$$\leq \sum_{j=1}^{M}(x_j - z_{optimal,j})^2 = d_{optimal} \quad (3)$$

($Z_p \in$ 探索の過程で，最適解を与えないことが判明したベクトルを除いたコードブックベクトルの集合) と定義される．ここで「歪みの次元ごとの最小値」$d_{lower,j}$ は，Z_j から，x_j との歪みが最小となる要素 $z_{p,j}$ を選択することで得られる．そして探索の過程で，$z_{p,j}$ が本来属するコードブックベクトル Y_i が最適出力ベクトルではないことが判明すれば，それを除外し次に小さい歪みを与える要素を選択し $d_{lower,j}$ を再計算する．すなわち $d_{lower,j}$ は単調増加する．またVQ-SCが対象とする歪みは，ベクトルの次元ごとの歪みの全次元にわたる線形和であるから，d_{lower} も探索の進行にともない単調増加する．結局式(3)に示すとおり，d_{lower} は最適出力ベクトルが与える歪み $d_{optimal}$ 以下であり，これに一致するまで単調に増加することが保証できる．

2.3.3 VQ-SCによる探索

以上述べた定義より，

$$d_{lower} \leq d_{optimal} \leq d_{upper} \quad (4)$$

がつねに成り立つ．そして探索の進行にともない，2.3.2項で示したとおり d_{lower} が単調増加する一方で，d_{upper} はより小さい暫定解に更新され，単調減少する．この結果 d_{upper} と d_{lower} が一致すれば，その値が $d_{optimal}$ となり，探索は終了する．なお探索順序の決定にはソートされたコードブックを用いて，次元ごとに，最も小さい歪みを与える要素が本来属しているコードブックベクトルを候補とし，これらを次元の順にラウンドロビンに選択する．すなわちVQ-SCでは次元レベルで最小歪みを与えるベクトルが最適出力ベクトルとなる可能性が高いというヒューリスティックスを用いている．

2.3.4 VQ-SCの性能

VQ-SCで256×256[画素]×8[ビット]画像データ(TV学会標準)を，Sun Microsystems社SPARCstation10（40 MHz, 1 MB external cache, 64 MB main memory）上で量子化し処理時間を計測した．その結果16次元4096個のベクトルを512個のベクトルからなるコードブックで量子化した場合，全探索法で30.31秒，Partial-distance法で13.66秒を要するのに対し，VQ-SCでは5.46秒と，従来手法に比べ60～80%以上，処理時間を短縮できることが確かめられた．

3. VQ-SCの並列探索手法

本章では，2章で述べたVQ-SCによる探索を並列化し，さらに高速なベクトル量子化を実現する手法を提案する．VQ-SCは2章で示したとおり従来の高速ベクトル量子化法に比較して非常に効率の良い探索が可能であるが，ベクトルの次元数あるいはコードブックが大きくなると，他の手法と同様に探索すべきベクトルの数は増える傾向にある．ところがベクトル量子化の符号化性能向上には次元の増大は不可欠[9]であり，より歪みの少ない符号化のためにはコードブックサイズも大きくする必要がある．そこでVQ-SCをマルチプロセッサシステム上で並列処理し高速化することを考える．VQ-SCの並列処理では，a）画像の領域分割並列化，b）コードブック分割並列化，c）コードブック共有・探索ベクトルレベル並列化，の3つの形態に基づく手法が考えられる．このうちa）画像の領域分割による並列処理は並列性も大きく，その実現も容易である．しかしJPEG[10]のように隣接領域の符号化結果を後続の領域の符号化に利用する場合や，リアルタイムアプリケーション等で画像データが通信路を介して順次送られ処理すべきデータが一括して得られない場合等には，適用が困難である．このため，本論文では上述のような場合にも適用できるb）およびc）のアプローチによる並列化手法を提案する．以下では図1の例を使って，b）に対応する1）コードブック分割並列探索法，およびc）に対応する2）コードブック共有セルフスケジューリング並列探索法について説明する．これらの手法は，使用するマルチプロセッサ・アーキテクチャの違い，すなわちプロセッサ間通信性能・ローカルメモリの有無等を考慮して使い分けられ，各種アーキテクチャ上での高速なベクトル量子化を可能とする．

3.1 コードブック分割並列探索法

本探索法では，各プロセッサが分割されたコードブックをそれぞれ独立にVQ-SCにより探索する．分割されたコードブックの例を，図1のコードブックを2分割した場合について，その分割コードブックに対応するソートされたコードブックと合わせて**図3**に示す．

本探索法は，この分割コードブックに関する探索処理の大半を基本的に各プロセッサで独立に実行できるので，プロセッサ間通信に比較的大きな時間を要する分散メモリ型マルチプロセッサにおいても実現可能である．ただしより良い上限値を用いて歪み計算を打ち切るために，プロセッサ間通信時間を考慮しつつ可能な限り上限値を共有/交換することが効果的である．以下では，図3の分割コードブックを用いてプロセッサ2台で並列処理する様子を示した**図4**を例として参照しながら説明する．

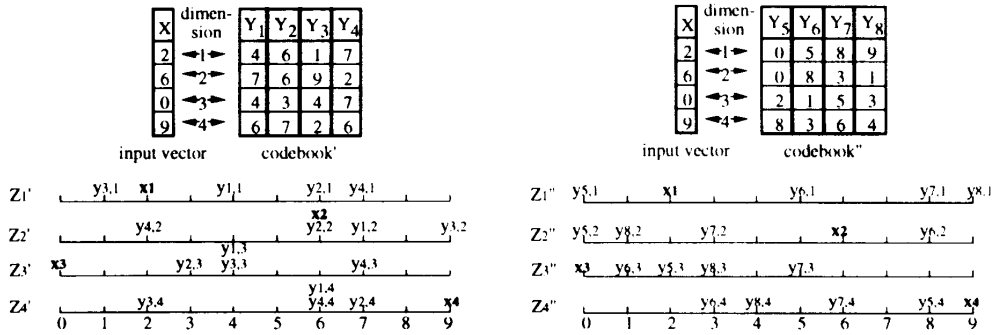

図3 図1の例に対応する分割されたコードブック
Fig. 3 Divided codebook corresponding to example of Fig. 1.

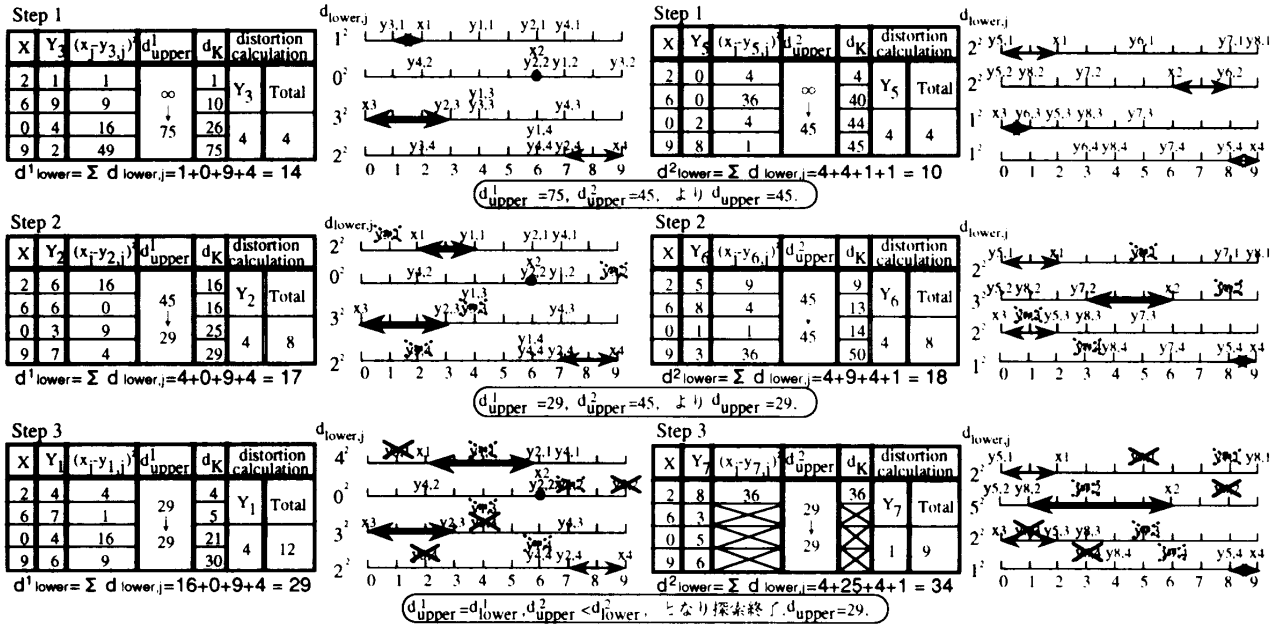

図4 コードブック分割による並列探索
Fig. 4 Parallel search with divided codebook.

Step 1 最初に各プロセッサでは，ローカルな上限値をそれぞれ ∞（実際には十分大きな値）に初期設定する．すなわち $d^1_{upper} = d^2_{upper} = \infty$ とする．そしてすべてのプロセッサで同一の入力ベクトル X を対象として歪み計算する．例では，$X = (2,6,0,9)$ である．

[上限値計算] 各プロセッサでは，第1次元 $(j = 1)$ から順に，その次元に関して X の要素との間で「最小の歪みを与えるコードブックベクトル要素（最小歪み要素）」が属するコードブックベクトルを歪み計算の対象として選択し，これと入力ベクトルとの歪みがローカルな上限値より小さい場合には更新する．

- PE1 第1次元 $(j = 1)$ の X の要素 x_1 の最小歪み要素は $y_{3,1}$ なので $i = 3$ より $Y_3 = (1,9,4,2)$ を選択し，$j = 4$ まで計算（歪み計算4回累計4回）して $d_4 = 75$ となり，d^1_{upper} を ∞ から 75 に更新する．
- PE2 第1次元 $(j = 1)$ の x_1 の最小歪み要素は $y_{5,1}$ なので $i = 5$ より $Y_5 = (0,0,2,8)$ を選択し，$j = 4$ まで計算（歪み計算4回累計4回）して $d_4 = 45$ となり，d^2_{upper} を ∞ から 45 に更新する．

[上限値の交換] 各プロセッサで最新の上限値が求められたならば，これらを互いに交換し，グローバルな上限値 d_{upper} として最も小さい値を採用し，各プロセッサでのローカルな上限値も同じ値に更新する．例では，$d_{upper} = d^1_{upper} = d^2_{upper} = 45$ となる．

[下限値計算] 各プロセッサでは，X の要素と，それに対する最小歪み要素との歪みの和を求め，これを各プロセッサにおけるローカルな下限値とする．

- PE1 X の要素 x_1, x_2, x_3, x_4 の最小歪み要素は順に，$y_{3,1}, y_{2,2}, y_{2,3}, y_{2,4}$ である．ゆえにそれぞれの次元における歪みを加算して，$d^1_{lower} = 14$.

d^1_{upper} と d^1_{lower} は一致しないので Step 2 へ進む.

- PE2 x_1, x_2, x_3, x_4 の最小歪み要素は順に, $y_{5,1}, y_{6,2}, y_{6,3}, y_{5,4}$ である. ゆえに, $d^2_{lower} = 10$. d^2_{upper} と d^2_{lower} は一致しないので Step 2 へ進む.

Step 2

[上限値計算]

- PE1 x_2 の最小歪み要素から Y_2 を選択し, $d_4 = 29$ となるので, d^1_{upper} を 45 から 29 に更新する.
- PE2 x_2 の最小歪み要素から Y_6 を選択し, $d_4 = 50$ となるので, $d^2_{upper} = 45$ のままである.

[上限値の交換] $d_{upper} = d^1_{upper} = d^2_{upper} = 29$.

[下限値計算]

- PE1 Y_3 は最適出力ベクトルではないことが分かり, ソートされたコードブック Z' から除外する. その結果 x_1 の最小歪み要素は $y_{3,1}$ から $y_{1,1}$ となり歪みは 1^2 から 2^2 に増加する. ゆえに $d^1_{lower} = 17$. d^1_{upper} と d^1_{lower} は一致しないので Step 3 へ進む.
- PE2 Y_6 をソートされたコードブック Z'' から除外すると x_2 の最小歪み要素は $y_{6,2}$ から $y_{7,2}$ に, x_3 の最小歪み要素は $y_{6,3}$ から $y_{5,3}$ に増加し, $d^2_{lower} = 18$. d^2_{upper} と d^2_{lower} は一致しないので Step 3 へ進む.

Step 3

[上限値計算]

- PE1 第3次元 ($j = 3$), 第4次元 ($j = 4$) の最小歪み要素が属する Y_2 はすでに探索済みであるので, x_1 の最小歪み要素から Y_1 を選択し, $d_4 = 30$ となり, $d^1_{upper} = 29$ のままである.
- PE2 x_2 の最小歪み要素から Y_7 を選択し, $d_1 = 36$ となるので, $d^2_{upper} = 29$ のままである.

[上限値の交換] $d_{upper} = d^1_{upper} = d^2_{upper} = 29$.

[下限値計算]

- PE1 Y_1 を Z' から除外した結果 x_1 の最小歪み要素は $y_{1,1}$ から $y_{2,1}$ となり $d^1_{lower} = 29$. ここで d^1_{upper} と d^1_{lower} が一致し PE1 の探索は終了する.
- PE2 Y_7 を Z'' から除外した結果 x_2 の最小歪み要素は $y_{7,2}$ から $y_{8,2}$ となり $d^2_{lower} = 34$. d^2_{upper} より d^2_{lower} が大きくなり PE2 の探索は終了する.

以上で並列探索は終了し, グローバルな上限値 $d_{upper} = 29$ を与える Y_2 が最適出力ベクトルである.

すなわち逐次処理では5ステップ18回の歪み計算が必要であった[8]のに対し, コードブック分割による並列探索は3ステップ最大12回の歪み計算で処理を終了できる. ただし逐次処理では歪み計算対象が5個のコードブックベクトルであったのに対し, 6個が対象であり冗長な探索が必要となる可能性を示している.

3.2 コードブック共有 セルフスケジューリング並列探索法

本探索法では, 逐次処理と同じコードブックをすべてのプロセッサで共有し, 逐次処理の場合は次元順にラウンドロビンに決定する探索候補を, 次元によって各プロセッサで分担し探索する. そして歪み計算を効果的に打ち切るために, 上限値・下限値を共有することが有効である. また同一コードブックベクトルを複数回, 歪み計算の対象としないための探索ベクトル管理テーブルを共有する必要がある. ただしこれら共有データの代入には排他的アクセスが必要である. 以下では, 図3の例を2台のプロセッサで並列に処理する様子を示した**図5**を用いて説明する. ここでは探索候補には, PE1 が第1・第2次元の, PE2 が第3・第4次元の, 入力ベクトル要素に関する最小歪み要素が本来属しているコードブックベクトルを選択している.

Step 1 最初に, すべてのプロセッサで共有する上限値 d_{upper} を ∞ (実際には十分大きな値) に初期設定する. そしてすべてのプロセッサは同一の入力ベクトル X を対象に歪み計算する. 例では $X = (2, 6, 0, 9)$.

- PE1 x_1 の最小歪み要素は $y_{3,1}$ なので, 探索ベクトル管理テーブルの $i = 3$ に対応する Table[3] を lock し $Y_3 = (1, 9, 4, 2)$ を歪み計算の対象として選択し unlock する. $d_4 = 75$ となるので, d_{upper} を lock し ∞ から 75 に更新して unlock する. 一方ソートされたコードブック Z (図2) より, x_1, x_2, x_3, x_4 の最小歪み要素は順に $y_{3,1}, y_{2,2}, y_{6,3}, y_{5,4}$ である. そこで d_{lower} を lock し, 次元ごとの歪みを加算して $1^2 + 0^2 + 1^2 + 1^2 = 3$ として unlock する. d_{upper} と d_{lower} は一致しないので Step 2 へ進む.
- PE2 上述のように排他制御しながら, x_3 の最小歪み要素から Y_6 を選択し, $d_4 = 50$ となるので $d_{upper} = 50$ に更新する. Y_3 は最適出力ベクトルではないと分かり, Z から除外した結果 x_1 の最小歪み要素は $y_{3,1}$ から $y_{1,1}$ となり, $d_{lower} = 6$ に更新する. d_{upper} と d_{lower} は一致しないので Step 2 へ進む.

Step 2

- PE1 同様に x_2 の最小歪み要素から Y_2 を選択し, $d_4 = 29$ から $d_{upper} = 29$ に更新する. Y_6 を Z から除外すると x_3 の最小歪み要素は $y_{6,3}$ から $y_{5,3}$ となり $d_{lower} = 9$ に更新する. d_{upper} と d_{lower} は一致しないので Step 3 へ進む.
- PE2 同様に x_4 の最小歪み要素から Y_5 を選択

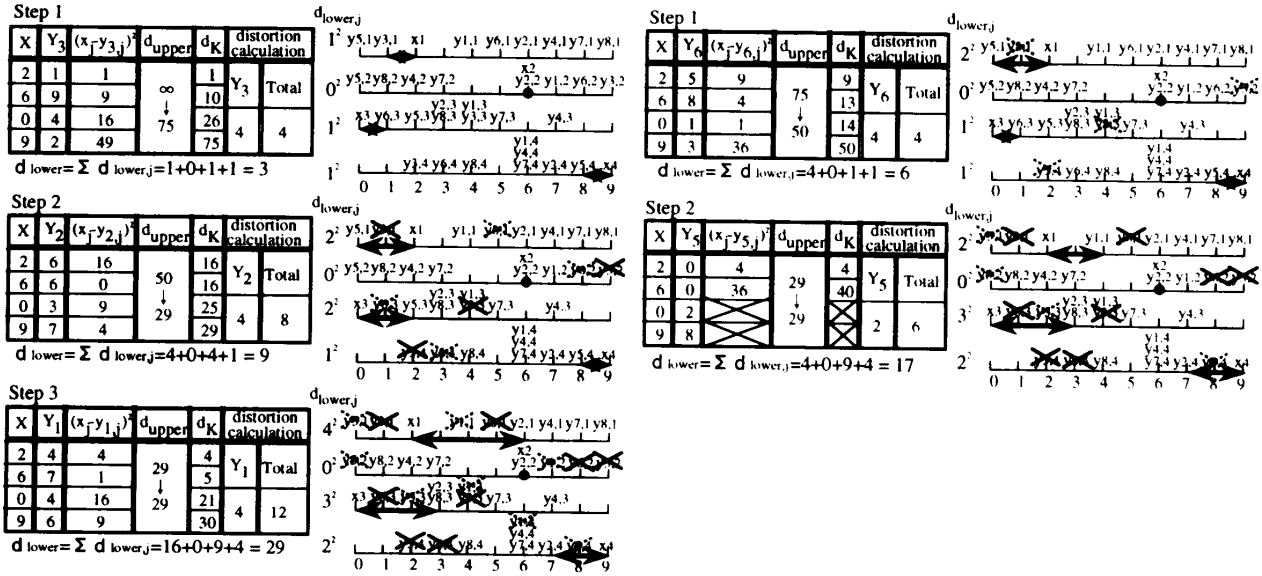

図5 セルフスケジューリング並列探索
Fig. 5 Self scheduled parallel search.

し，$d_2 = 40$ から $d_{upper} = 29$ のままにする．Y_5 を Z から除外すると x_3 の最小歪み要素は $y_{5,3}$ から $y_{2,3}$ ($y_{8,3}$ でも可)，x_4 の最小歪み要素は $y_{5,4}$ から $y_{2,4}$ となり $d_{lower} = 17$ に更新する．d_{upper} と d_{lower} は一致しないので Step 3 へ進む．

Step 3

- PE1　同様に x_1 の最小歪み要素から Y_1 を選択し，$d_4 = 30$ から $d_{upper} = 29$ のままにする．Y_1 を Z から除外すると x_1 の最小歪み要素は $y_{1,1}$ から $y_{2,1}$ となり $d_{lower} = 29$ に更新する．ここで d_{upper} と d_{lower} が一致し PE1 の探索は終了する．
- PE2　すでに $d_{upper} = d_{lower} = 29$ と一致しており PE2 の探索は終了する．

以上で並列探索は終了し，$d_{upper} = 29$ を与える Y_2 が最適出力ベクトルである．

すなわちコードブック共有による並列処理では，3 ステップ最大 12 回の歪み計算で処理を終了できる．特に本探索法は，逐次処理と同じ 5 個のコードブックベクトルだけを処理対象とし，並列化により冗長な探索が発生しない．ただし上述のとおり，上限値・下限値・探索ベクトル管理テーブルを更新するときに lock を掛ける必要があり，これが並列処理オーバヘッドとなる．

4. 性能評価

3 章で提案した並列探索手法の有効性を検証するために，AP1000[11]，cenju-3[12]，FX/4[13]および KSR1[14] というアーキテクチャの異なる実マルチプロセッサシステム上で，画像データのベクトル量子化に必要な処理時間を計測した．ここで処理の対象には，512 × 512 [画素] × 8 [ビット] の画像データを用いた．これをまず 8 × 8 のブロックに分割し 64 次元の 4096 個のベクトルとして，このすべてをトレーニング・シーケンスに用い PNN 法[15] でコードブックベクトル数 N=1024 のコードブックを作成した．このコードブックを用いて，同じ画像を提案する並列探索手法で処理し，それに要する時間を計測し速度向上率により比較する．

4.1 コードブック分割並列探索法

ここでは 3.1 節で述べたコードブック分割並列探索法を，AP1000，cenju-3，FX/4 および KSR1 を対象としてインプリメントした結果について述べる．

まず AP1000 では，通信コストが比較的小さな，全プロセッサ間でローカルな最小値を交換し同期的にグローバルな最小値を求める関数 (xy_imin) を，上限値の交換に利用する．**図 6** に示した AP (1) では図 4 の例で示したように各ステップで上限値を交換する．ただし探索終了までに処理すべきベクトルの数は通常プロセッサごとに異なるのに対し，同期的な関数 (xy_imin) を用いるために各プロセッサの探索進行状況を上限値と合わせて通信している．次に AP (2) では，すべてのプロセッサが探索を終了した後にのみ上限値を交換し，探索途中は交換しない．図 6 より，AP (2) は上限値を交換しないために歪み計算打ち切りの効果で劣るにもかかわらず AP (1) よりも高速化されており，上限値とプロセッサの状態を交換するオーバ

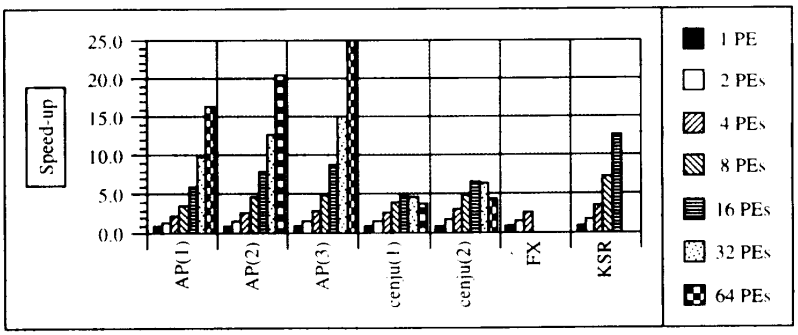

図6 コードブック分割並列探索法による速度向上率
Fig. 6 Speed-up ratio by parallel search with divided codebook.

ヘッドがかなり大きいことが分かる．そこで，これらの中間的な手法として，AP (3) では上限値の初期値だけを交換する．上限値の初期値は各プロセッサで必ず計算されるので，この直後であれば探索進行状況を調べることなく同期的な通信が実行できる．この結果，逐次処理と比べ8台で4.9倍，64台で25.0倍の速度向上が得られた．

次にcenju-3でAP (3) と同じ方式をインプリメントしcenju (1) に示す．しかしcenju-3ではプロセッサ数を増やしても速度向上につながらない．これは通信ライブラリとして使用しているPARALIB/CJが基本的な通信機能しかサポートしておらず，AP1000の(xy_imin) ほどに高速な通信が実現できないからである．そこでPARALIB/CJに含まれ，他プロセッサのローカルメモリへの書き込みを行う (CJrmwritem) を使用する．これを用いれば暫定解が更新されたときのみ通信すればよくなり，無駄な通信を削減することが可能となる．この結果cenju (2) のように改善され，8台で4.9倍，16台で6.5倍の速度向上が得られた．

次にFX/4では，上限値を共有して排他的に更新し，図6のFXに示されるように，4台で2.8倍という速度向上率が得られた．

最後にKSR1でもFX/4と同様にインプリメントし，8台で7.2倍，16台で12.6倍の速度向上が得られた．

結局コードブック分割並列探索法は，アーキテクチャの異なるシステムにおいても，ほぼ同等の性能が得られることが確かめられた．ただし並列化により冗長な探索を必要とし，処理すべきベクトル数も不均一になり，速度向上率はあまり大きくならない．しかし並列度は比較的大きく，処理プロセッサ台数の増加にともなって速度向上が得られ，高並列計算機により十分な処理速度を達成できる手法であることが確かめられた．

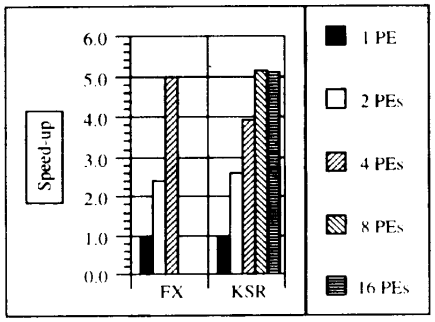

図7 セルフスケジューリング並列探索法による速度向上率
Fig. 7 Speed-up ratio by self scheduled parallel search.

4.2 コードブック共有セルフスケジューリング並列探索法

ここでは3.2節で述べたコードブック共有セルフスケジューリング並列探索法のFX/4およびKSR1でのインプリメント結果について述べる．**図7**から，FX/4では4台で5.0倍とスーパーリニアスピードアップを達成している．これはVQ-SCで用いるヒューリスティックスでは探索順序が必ずしも最適ではなく，並列探索すれば早期に良い上限値が発見できる場合があるためである．またKSR1では8台，16台でともに5.1倍と速度向上率が飽和している．これは現象的にはコードブック共有による共有データへのアクセス競合が原因として考えられるが，このアルゴリズムに関してはこのような競合が生じるのは多くの次元で探索候補が重なるようになることが本質的な原因と考えられる．すなわち，多くの場合，実質的には次元数だけの並列度はないと考えられる．ゆえに本探索法においては，次元数に比較して少ないプロセッサ数であれば，共有メモリ型のマルチプロセッサで非常に効率の良いVQ-SCの並列探索が実現できることが確かめられた．

5. おわりに

本論文では，VQ-SCを分散メモリ型および共有メモ

リ型マルチプロセッサシステム上で並列処理するための，1) コードブック分割並列探索法，および2) コードブック共有セルフスケジューリング並列探索法，という2つの手法を提案した．そして，これらの手法を，アーキテクチャの異なる実マルチプロセッサシステム上での画像データのベクトル量子化に適用し，その有効性と適用可能性を示した．なお本論文で提案した2つの並列探索手法を最適に組み合わせて，さらに高速化を図り，動画をも含む実用的なデータ圧縮システムへの適用を検討する予定である．また同期・通信のチューニング手法などの充実および評価が今後の課題である．

謝辞 本研究を進めるにあたり，計算環境をご提供いただいた富士通並列処理研究センター，キヤノンスーパーコンピューティングS.I.（株），NEC C&C研究所並列処理センターに感謝いたします．また日頃貴重なご助言とご指導をいただきます早稲田大学理工学部成田誠之助 教授に深謝いたします．

参考文献

1) Gersho, A. and Cuperman, V.: Vector Quantization: A Pattern-Matching Technique for Speech Coding, *IEEE Comm. Mag.*, Vol.21, No.9, pp.15–21 (1983).
2) 斎藤隆弘：画像符号化アルゴリズム (III)—ベクトル量子化—，テレビ誌，Vol.43, No.11, pp.1276–1284 (1989).
3) Gersho, A.: Asymptotically Optimal Block Quantization, *IEEE Trans. Inf. Theory*, Vol.25, No.4, pp.373–380 (1979).
4) Gray, R.M.: Vector Quantization, *IEEE ASSP Mag.*, Vol.1, No.2, pp.4–29 (1984).
5) Wu, H.-S.: Fast Search Algorithm for Vector Quantization, *Electron. Lett.*, Vol.28, No.5, pp.457–458 (1992).
6) Ra, S.-W. and Kim, J.-K.: Fast Weight-Ordered Search Algorithm for Image Vector Quantization, *Electron. Lett.*, Vol.27, No.22, pp.2081–2083 (1991).
7) Bei, C.-D. and Gray, R.M.: An Improvement of the Minimum Distortion Encoding Algorithm for Vector Quantization, *IEEE Trans. Commun.*, Vol.33, No.10, pp.1132–1133 (1985).
8) 中野恵一，笠原博徳：ソートされたコードブックを用いた高速ベクトル量子化，信学論（DII），Vol.J77-D-II, No.10, pp.1984–1992 (1994).
9) 加藤洋一，大久保栄，鈴木 豊：動画像ハイブリッド符号化方式の符号化効率—ブロックサイズの選定，DCT方式とVQ方式の比較，ループフィルタ—，信学論（B），Vol.J73-B-I, No.2, pp.92–99 (1990).
10) 大町隆夫，小野文孝：カラー静止画符号化国際標準方式 (JPEG) の概説（その1），画像電子学会誌，Vol.20, No.1, pp.50–58 (1991).
11) Ishihata, H., Horie, T., Inano, S., Shimizu, T. and Kato, S.: An Architecture of Highly Parallel Computer AP1000, *IEEE Pacific Rim Conf. on Comm., Comput. and Signal Process.*, pp.13–16 (1991).
12) 広瀬哲也，加納 健，丸山 勉，中田登志之，浅野由裕，稲村 雄：並列コンピュータCenju-3のアーキテクチャ，情報処理学会研究報告，Vol.94, No.66, pp.121–128 (1994).
13) 高橋義造（編）：並列処理機構，丸善 (1989).
14) Boman, E.: Experiences on the KSR1 Computer, Technical Report RNR-93-008, NAS Systems Division, Applied Research Branch, NASA Ames Research Center, Moffett Field, California (1993).
15) Equitz, W.H.: A New Vector Quantization Clustering Algorithm, *IEEE Trans. ASSP*, Vol.37, No.10, pp.1568–1575 (1989).

(平成7年9月1日受付)
(平成8年4月12日採録)

中野 恵一（正会員）

平成元年早稲田大学大学院修士課程修了．同年オリンパス光学工業（株）入社．平成7年早稲田大学大学院博士後期課程単位取得退学．工学博士．電子情報通信学会員，本学会誌編集委員．

笠原 博徳（正会員）

昭和55年早稲田大学理工学部電気工学科卒業．昭和60年同大学院博士課程修了．工学博士．昭和58～60年早稲田大学理工学部助手．昭和60年カリフォルニア大バークレー短期客員研究員，日本学術振興会第1回特別研究員．昭和61年早稲田大学電気工学科専任講師．昭和63年同助教授．情報学科助教授を経て，現在電気電子情報工学科助教授．平成元年～2年イリノイ大学Center for Supercomputing R&D客員研究員．昭和62年IFAC World Congress第1回Young Author Prize受賞．主な著書「並列処理技術」（コロナ社）．電子情報通信学会，電気学会，シミュレーション学会，ロボット学会，IEEE，ACM等の会員．