Vol.2012-AVM-76 No.2
2012/2/23

並列分散による映像イベント検出の高速化

後藤 充裕[†1]　佐藤 隆[†1]　東野 豪[†1]

映像イベント検出の高速化のための並列分散方式を提案する．映像を固定長の処理区間に分割する単純な方式では，処理区間の終了点がイベントの存在区間を横切る場合に，正確な検出結果が得られないという問題がある．そこで，本稿では，処理区間の終了点でイベントが検出され続けている場合に，必要な長さだけ適応的に処理区間を延長する方式を提案する．実験の結果，イベント検出精度を損なうことなく，高速な並列分散が実現できることを確認した．

A Parallel Distributed Processing Method for Fast Video Event Detection

Mitsuhiro Goto ,[†1] Takashi Satou [†1]
and Higashino Suguru [†1]

We propose a parallel distributed processing method for fast video event detection. Existing methods that divide a video into multiple fixed length partitions are not accurate because the boundary of the partitions terminates the detecting event before it ends. To solve the problem, we propose a new method that extends the partition to cover the event. The experimental results show that the proposed method can detect video events fast and accurately.

1. はじめに

動画共有サービスやライブ配信サービスが普及し，多くの映像がネットワーク上に流通するようになった．一般に，映像は画像・音データが時間経過と共に変化する時間メディアであり，視聴には映像長だけの時間を要する．そのため，映像内容の効率的な視聴を支援するイベント検出処理の重要性はますます高まってきている．なお，イベント検出処理とは，映像が持つ画像・音の特徴量をフレーム単位で抽出し，その時間的変化から特徴的な区間を定義して，映像の構造化を実現するものである[1)2)]．イベント検出の例として，フレーム間の画素輝度を抽出し，差分がしきい値以上となった点を映像の切れ目として検出するカット点検出[3)] や，2 枚のフレーム間の動きベクトルを計算し，それが一定方向に安定して持続している区間を検出するカメラワーク検出[4)] がある．

一般に，イベント検出はその処理に多くの時間を要するため，従来より，その高速化に向けて様々な試みがなされてきている．主なアプローチとしては，(1) 検出アルゴリズムの最適化[5)6)]，(2)GPU の利用[7)8)9)]，(3) マルチコア CPU・マルチサーバによる並列分散[10)11)] が存在する．本研究では，(3) のアプローチを採り，さらに，既存のイベント検出処理アルゴリズムへの変更を最小限に抑えながら，並列分散方式による高速化の実現を目指す．

映像を並列分散する最も単純な方法は，対象とする映像データを固定長の処理区間に分割し，それぞれの処理区間を CPU コアやサーバ上で動作するワーカに割当てる方法である．ワーカは，割当てられた処理区間に対してイベント検出処理を実行し，検出結果を出力する．しかしながら，このような単純な方法では，処理区間の終了点がイベントの存在区間を横切る場合に，「検出結果の分断」や「検出結果の欠落」が発生するため，分割しない場合と，検出結果が変化してしまう (図 1)．従来研究[11)] では，検出結果を維持したまま，イベント検出を並列化するために，1 つの映像ファイルをオーバラップ区間を設けて固定長の処理区間に分割する方法を採った．しかし，オーバラップ区間を設けても，依然として，処理区間の終了点がイベントの存在区間を横切りが起こるため，分割しない場合と分割した場合とで検出結果が異なることがあった．

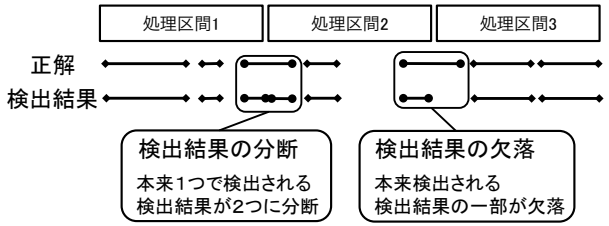

図 1 イベントの誤検出パターン
Fig. 1 Patterns of false positive events.

†1 日本電信電話株式会社 NTT サイバーソリューション研究所
NTT Cyber Solution Laboratories, NTT Corporation

Vol.2012-AVM-76 No.2
2012/2/23

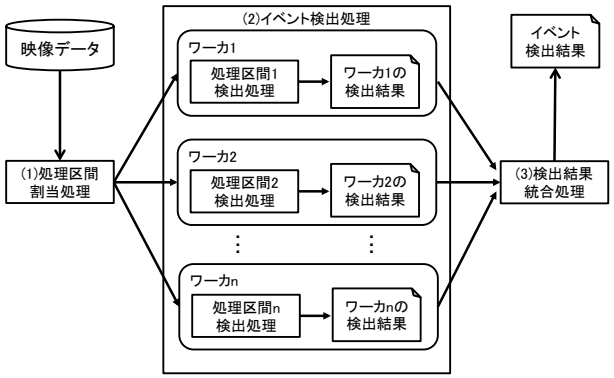

図 2 提案手法の概要
Fig. 2 Overview of the proposed method.

そこで，本研究では，ワーカに可変長の処理区間を割り当てて，イベントの存在区間に合わせて適応的に処理区間を延長し，分割しない場合と検出結果を一致させる手法を提案する．提案手法では，ワーカがイベントの検出中状態で処理区間の終了点に到達した場合に，検出中のイベントが終了するまで処理区間を延長する．さらに，イベント検出処理によっては，映像中の全てのフレームを処理対象にせず一定間隔でフレームを間引く処理や，処理区間の開始点より数フレーム前の内部バッファリングを必要とする処理がある．このような場合においても，検出結果を一致させるため，提案手法では，フレームの間引き処理に対して，分割の影響を受けず，常に先頭から整数倍のフレームが処理対象になるように処理区間の開始点を補正する．バッファリングに対しては，分割しない場合と分割した場合でイベント検出時の内部状態が一致するように，必要なバッファリング分だけ映像の先頭側 (以下，前方と呼ぶ) に処理区間の開始点を補正する．また，本稿では提案手法を用いた実験の結果，イベント検出精度を損なうことなく，高速な並列分散が実現できたことを報告する．

2. 提 案 手 法

2.1 提案手法の概要

本手法では，入力として 1 つの映像ファイルを複数のワーカで共有しながら，並列にイベント検出を実行する．本手法は，以下の順序で処理を進める (図 2)．

(1) 処理区間割当処理：処理対象の映像データをワーカ数で分割して，各ワーカに割り当てる．

(2) イベント検出処理：各ワーカが担当する処理区間に対してイベント検出処理を進め，それぞれの検出結果を出力する．

(3) 検出結果統合処理：各ワーカが担当した処理区間の検出結果を統合して，最終的な検出結果を出力する．

以降では，イベント検出処理と検出結果統合処理について詳しく述べる．

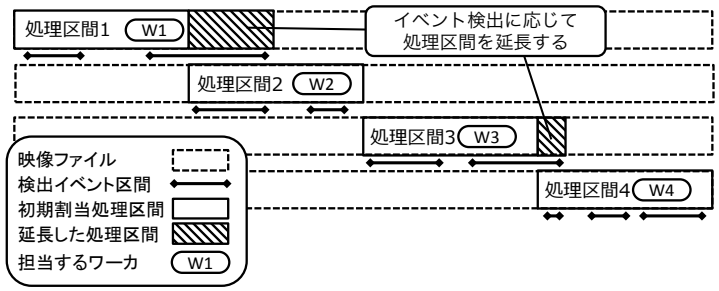

図 3 処理区間の延長
Fig. 3 Extension of the partitions.

2.2 イベント検出処理

イベント検出処理では，以下の 3 つの処理により，分割する場合にも正しくイベント区間を検出する．

2.2.1 処理区間の延長

ワーカは，割当てられた処理区間の開始点まで映像をシークし，終了点までイベント検出処理を進める．処理区間の終了点に到達したときに，自身のイベント検出状態を基にして処理区間の延長を判断する．イベント検出中ではない状態で終了点に達した場合，そこでイベント検出を終了する．しかし，イベントを検出中の状態で終了点に達した場合，現在検出中のイベントが終了するまで処理区間を延長しながら，イベント検出を続行する．このように処理終了点を延長することによって，正しくイベント区間を検出することができる (図 3)．なお，本手法では問題を単純化するため，この検出状態の判断においてはワーカ間の通信を考えない．各ワーカは個々に処理区間の延長を判断しながらイベントを検出する．

2.2.2 処理対象フレームの補正

イベント検出処理によっては，全てのフレームを処理対象にせず，一定の間隔でフレーム

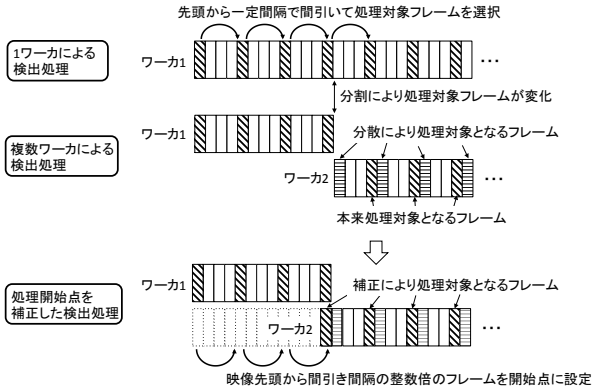

図 4 処理対象フレームの補正
Fig. 4 Correction of target frames.

を間引いて検出するものもある．このような処理の場合，間引き間隔を考慮せずに開始点を決定すると，処理対象となるフレームが 1 ワーカで実行した場合と異なることがあり，検出結果に違いを生じさせる原因になる．そこで，映像の先頭から間引き間隔の整数倍のフレームが処理対象のフレームとなるように処理区間の開始点を補正する (図 4)．この補正により，映像の分割数によらず常に同じフレームを処理対象として検出を実行できる．

2.2.3 バッファリングを考慮した開始点の補正

映像を先頭から検出する場合と途中までシークして検出する場合で，検出結果が異なることがある．これはデコーダの性質上，シーク直後のフレーム画像や音声信号が安定しないことと，イベント検出器内部のバッファリングによるものである (図 5)．そこで，分割しない場合と同じ内部状態で処理を実行できるように，本来の開始点より前方にシークして少し前のフレームからデータを読み込む．

2.3 検出結果統合処理

各ワーカが出力した検出結果を統合するとき，オーバラップする区間については前方のワーカの検出結果を優先して統合する．この理由は，前方のワーカより後方のワーカの検出結果が，検出に必要な全てのフレームを参照しないため，信頼性に欠けることがあるからである．検出結果が一部重複している場合や前方のワーカの検出結果が後方のワーカの検出結果を全て含んでいる場合には，後方のワーカの検出結果を破棄し，前方ワーカの検出結果のみを用いる．

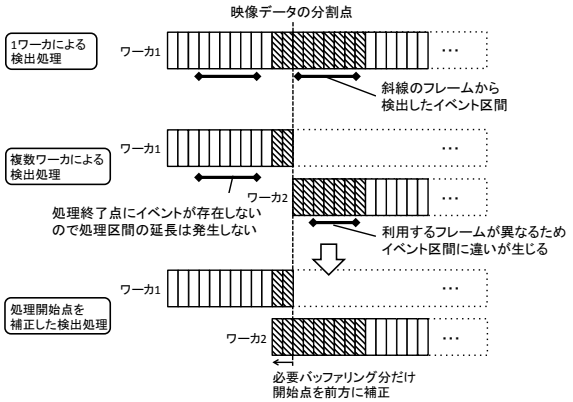

図 5 バッファリングを考慮した開始点の補正
Fig. 5 Correction of a start point for buffering.

3. プロトタイプシステム

2 節で述べた提案に基づきプロトタイプシステムを実装した．プロトタイプシステムの構成は，ネットワーク接続された 1 台の管理サーバと，5 台のイベント検出サーバ，映像ファイルを共有する 1 台の共有ストレージである．管理サーバが処理区間割当処理と検出結果統合処理を担当し，イベント検出サーバがイベント検出処理のワーカを実行する．

各サーバの仕様は，CPU : Xeon X3363 4 コア (2.83GHz)，メモリ : 16GB，OS : CentOS 5.6 である．CPU は 4 つの物理コアを持っているため，1 サーバあたり最大で 4 ワーカによってイベント検出を同時実行する．したがって，システム全体では，最大で 20 ワーカによるイベント検出が実行可能である．共有ストレージは，各サーバと NFS 共有されており，インデクシング処理時には，各サーバが同時にストレージにアクセスしながら映像データを読み出す．これらは，1Gbps のネットワークで接続されている．

4. 評 価 実 験

プロトタイプシステムにより提案手法の評価を行った．

4.1 実 験 条 件

様々な映像に提案手法が有効であることを確かめるため，ドラマ，バラエティ，ニュー

スなど複数ジャンルの映像 10 本を用いて実験した．映像の長さは 3〜10 分であり，映像フォーマットは Motion JPEG，解像度は 320 × 240，fps は 30fps，音声フォーマットは PCM である．

これらの映像に対して，1 ワーカでの実行処理時間を測定し，検出結果を正解データとした．次に，以下の 3 つの手法について処理時間と検出精度を評価した．

(1) 提案手法
イベント検出処理の状態に応じて，動的に処理区間を変更する手法である．

(2) 固定長分割手法[11)]
隣り合う処理区間に対して，固定長のオーバラップ区間を設ける手法である．ここで，オーバラップ区間の長さは 5,000ms とした．

(3) 単純分割手法
隣り合う処理区間にオーバラップする区間を設けない手法である．

処理対象のイベントはカット点とカメラワークである．カット点はイベント区間が比較的短く，頻出する．カメラワークはイベント区間がカット点よりも長いが，出現頻度は低い．このように，処理対象とするイベントによって，イベント区間の出現頻度やイベント区間の長さは異なるので，異なるイベント間で実験結果を比較することによって，提案手法の特性を調べることができる．それぞれの分散手法と検出イベントについて，ワーカ数を 2, 4, 8, 12, 16, 20 と変えながら，処理時間と検出精度の観点から評価した．

- 処理時間に関する評価指標

処理時間に関する評価指標では，2 つの観点から評価を行った．1 つは，提案手法がイベント検出処理を高速化して，処理時間の短縮が可能であるかを確認する高速化率である．もう 1 つは，その高速化が CPU リソースの稼働を最小限に抑えて実現できるかを確認する延べ実行時間割合である．映像を n 個 $(n \geq 2)$ の処理区間に分割した場合の処理区間 i を担当するワーカの処理時間 T_{ni} は，処理区間割当処理の時間 T_{nid} と，イベント検出処理の時間 T_{nie} の合計となる．また，分割時には，検出結果統合処理の時間 T_{nm} が生じる．

$$T_{ni} = T_{nid} + T_{nie} \qquad (i = 1, 2, \ldots, n) \tag{1}$$

映像を n 個の処理区間に分割した場合の高速化率 S_n は，1 ワーカによる処理時間 T_1 と T_{ni} の中で最長のものの比を求めた．

$$S_n = \frac{T_1}{\max(T_{n1}, T_{n2}, \ldots, T_{ni}) + T_{nm}} \qquad (i = 1, 2, \ldots, n) \tag{2}$$

映像を n 個の処理区間に分割した場合の延べ実行時間割合 G_n は，T_{ni} の合計値と T_1 の比により求めた．

$$G_n = \frac{1}{T_1}\left(\sum_{i=1}^{n} T_{ni} + T_{nm}\right) \qquad (i = 1, 2, \ldots, n) \tag{3}$$

- 検出精度に関する評価指標

1 ワーカにより処理した検出結果を正解データとして，複数ワーカにより処理した検出精度を，再現率と適合率により評価した．本研究では，分割数によらず検出結果が一致することを目的にしているので，検出結果が正解データに完全に一致している場合のみを正しい検出とした．つまり，正解データに含まれるイベント区間を検出できなかった場合だけではなく，検出していても，イベント区間の開始点または終了点が異なる場合は誤検出とした．

4.2 実験結果

図 6 と図 7 に，カット点検出処理とカメラワーク検出処理について，それぞれ高速化率と，延べ実行時間割合，検出結果の再現率，適合率を示す．全てのグラフは横軸がワーカ数である．また，イベント長のデータを表 1 に示す．

- カット点検出

提案手法の高速化率は，最大で 20 ワーカを用いて 1 ワーカでの処理の 15.9 倍を実現した．提案手法は全てのワーカ数において，固定長分割手法より高速であり，20 ワーカ時には 14%(提案手法：15.9 倍，固定長分割手法：13.9 倍) の高速化が図れた (図 6 左上図)．

また，提案手法の延べ実行時間割合は，全てのワーカ数において固定長分割手法と比較すると，20 ワーカ時に 15%(提案手法：1.02 倍，固定長分割手法：1.18 倍) 軽減できた．また，単純分割手法と比べてもほぼ同等であった (図 6 右上図)．

提案手法と固定長分割手法の検出精度は，ワーカ数の増加によらず再現率と適合率が常に 100%となった．それに対し，単純分割手法では，ワーカ数の増加により，再現率と適合率ともに検出精度がわずかに低下する傾向が見られた (図 6 下図)．

- カメラワーク検出

カット点検出と同様の傾向となった．すなわち提案手法では，最大で 20 ワーカで 1 ワーカの 16.2 倍の高速化率を達成した（図 7 左上図)．提案手法の延べ実行時間割合は，固定長分割手法と比較すると，20 ワーカ時に 14%(提案手法：1.03 倍，固定長分割手法：1.17 倍) 軽減できた．単純分割手法と比べてもほぼ同等であった（図 7 右上図)．

また，提案手法と固定長分割手法の検出精度は，再現率と適合率がワーカ数が変わっても 100%の精度を保った．単純分割手法では，カット点検出のときよりも大きく低下し，20

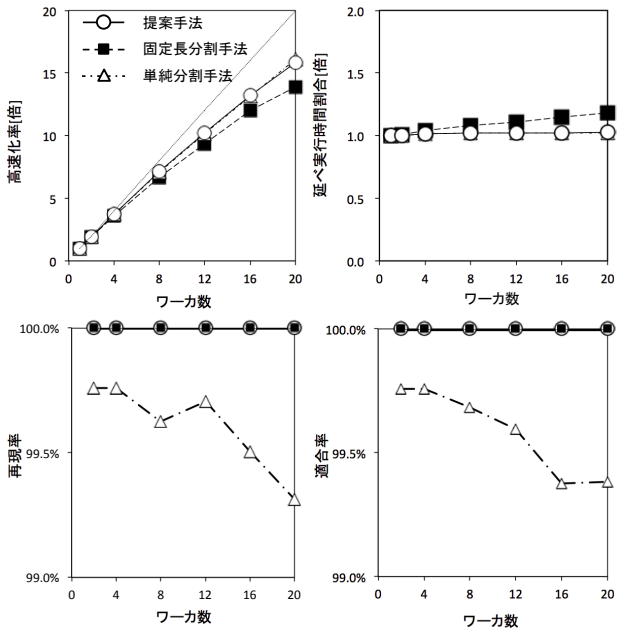

図 6 カット点検出の実験結果
Fig. 6 Experimental results of cut detection.

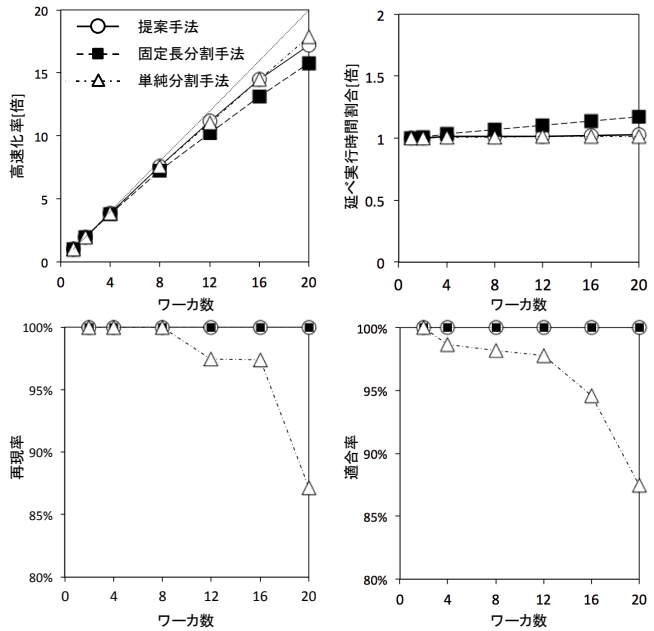

図 7 カメラワーク検出の実験結果
Fig. 7 Experimental results of camera work detection.

ワーカ時には 13%(単純分割手法：再現率・適合率 87%) 低下した (図 7 下図).

以上の実験結果から，両イベント検出において，提案手法が検出精度を維持したまま，効率的なリソース稼働によりイベント検出処理の高速化を可能であることが確認できた.

4.3 考　　察

実験結果では，3 手法の間でそれぞれの評価指標について差が生じた．この原因について高速化率と，延べ実行時間割合，検出精度のそれぞれを提案手法と他の 2 手法を比べながら考察する.

- 高速化率について

両イベント検出において，提案手法の高速化率は，イベント検出処理へのオーバヘッドを最小限に抑えられたため，単純分割手法と遜色ない結果となった．表 1 によると，カット点検出とカメラワークの平均イベント長はそれぞれ 130ms，3,511ms であり，延長が発生した場合にも，延長しなければならない区間長が短く，高速化率が高くなった.

また，提案手法と固定長分割手法の高速化率を比較すると，ワーカ数の増加に応じてその差は大きくなった．固定長分割手法では，映像長をワーカ数で分割した処理区間に 5,000ms のオーバラップ区間を加えたものが 1 ワーカの処理する区間となる．そのため，ワーカ数の増加に応じて，処理区間に対するオーバラップ区間の割合が大きくなり，高速化率が悪くなった (表 1).

- 延べ実行時間割合について

両イベント検出において，提案手法の延べ実行時間割合は，全ての処理区間において最小限の延長で効率的にイベント検出ができたため，単純分割手法とほぼ同等であった．表 1 によると，両イベント検出の映像長に対するイベント長の割合は，カット点検出で 2.2%，カメラワーク検出で 6.6%であり，全ての処理区間において延長が発生する確率は低い．そし

表 1 イベント長データ
Table 1 Statistics of the video event length.

	カット	カメラワーク
最大イベント長	2,200ms	10,100ms
最小イベント長	0ms	2,000ms
平均イベント長	130ms	3,511ms
イベント検出回数	1,208 回	135 回
映像長に対するイベント長の割合	2.2%	6.6%

て，上述した高速化率より，延長が発生しても延長しなければならない区間長が短いため，効率的に処理が行えた．

また，提案手法と固定長分割手法を比較すると，ワーカ数の増加に応じて延べ実行時間割合の差が大きくなった．固定長分割手法は，処理区間の終了点付近でのイベントの有無に関わらず，映像長に最後方のワーカを除く全てのワーカ数× 5,000ms 分を加えた区間を処理する．そのため，イベント検出が不要な区間を重複して処理することにも繋がり，リソース稼働の効率が悪くなった．

- 検出精度について

両イベント検出において，提案手法では，イベント長に応じて処理区間を延長した効果により，ワーカ数増加の影響を受けず検出精度を保つことができた．一方，単純分割手法では，ワーカ数の増加により，処理区間の終了点がイベント区間を横切る確率は高くなり，誤検出や検出漏れを引き起こした．表 1 によると，両イベント検出の映像長に対するイベント長の割合は，カット点検出で 2.2%，カメラワーク検出で 6.6%であり，カット点検出のようなイベント長の割合が少ないイベント検出においても，ワーカ数の増加が検出精度に影響した．

また，提案手法と固定長分割手法を比較すると，両イベント検出において検出精度が 100%であった．これは，固定長分割手法に設定したオーバラップ区間長 (5,000ms) が，各イベントの平均イベント長 (カット点：135ms，カメラワーク検出：3,511ms) より長かったため，検出精度が低下しなかった．

5. ま と め

本稿では，映像からのイベント検出の高速化を実現する並列分散手法を提案した．本手法は，処理区間の終了点でイベントが検出されている場合に，必要な長さだけ適応的に処理区間を延長する方式を採る．さらに，イベント検出処理によっては一定間隔でフレームを間引く処理や，処理区間の開始点より数フレーム前の内部バッファリングを必要とする処理がある．このような場合においても，分割しない場合と検出結果を一致させるため，提案手法では，処理対象のフレームやイベント検出時の内部状態が等しくなるように処理区間の開始点を補正して，正しいイベント区間の検出を可能にする．実験結果から，提案する処理区間の動的な延長は，提案手法が検出精度を維持したまま，効率的なリソース稼働によりイベント検出処理の高速化が可能であることを確認できた．

今後の課題として，平均イベント長が長く，処理区間の延長が多く発生するイベント検出処理に対して，本手法の有効性について確認していく．

参 考 文 献

1) Tonomura, Y., Akutsu, A., Taniguchi, T. and Suzuki, G.: Structured Video Computing, IEEE Multimedia, Vol.1, No.3, pp.34-43 (1994).
2) 谷口行信, 南憲一, 佐藤隆, 桑野秀豪, 児島治彦, 外村佳信：SecneCabinet：映像解析技術を統合した映像インデクシングシステム, 信学論, Vol.J.84-D-II, No.6, pp.1112-1121 (2001).
3) 谷口行信, 外村佳伸, 浜田洋：映像ショット切換え検出法とその映像アクセスインタフェースへの応用, 信学論, Vol.J79-D-II, No.4, pp.538-546 (1996).
4) Taniguchi, Y., Akutsu, A. and Tonomura, Y.: PanoramaExcerpts: Extracting and Packing Panoramas for Video Browsing, In Proc. ACM Multimedia, pp.427-436 (1997)
5) Seinstra, F.J., Geusebroek, J.-M., Koelma, D., Snoek, C.G.M., Worring, M. and Smeulders, A.W.M.: High-Performance Distributed Video Content Analysis with Parallel-Horus, IEEE Multimedia, Vol.15, No.4, pp.64-75(2007).
6) Zhang, Q., Chen, Y., Li, J., Zhang, Y. and Xu, Y.: Parallelization and Performance Analysis of Video Feature Extractions on Multi-Core Based Systems, In Proc. Parallel Processing (2007).
7) Diao, Y., Nicopoulos, C. and Jongman, K.: Large-Scale Semantic Concept Detection on Manycore Platforms for Multimedia Mining, In Proc. Parallel & Distributed Processing Symposium, pp.384-394 (2011).
8) Van de Sande, K.E.A., Gevers, T. and Snoek, C.G.M.: Empowering Visual Categorization with the GPU, Trans. IEEE Multimedia, Vol.13, No.1, pp.60-70(2011).
9) Lin, D. et al.: The Parallelization of Video Processing, Signal Processing Magazine, Vol.26, No.6, pp.103-112 (2009).
10) 赤間浩樹, 松田基弘, 毛受崇, 長谷川知洋, 内藤一兵衛, 山室雅司：メディア処理向けクラウド基盤「虹雲」, 情処全大 第 72 回平成 22 年 (3)(2010).
11) 小林昭久, 桑野秀豪：分散並列型映像インデクシング方式の提案, 信学全大 2002 年 情報・システム (2), pp.203 (2002).