ブランド露出調査のためのGPUによる局所不変特徴量の抽出

市村　直幸†

† 産業技術総合研究所
〒 305–8568 茨城県つくば市梅園 1-1-1 中央第 2
E-mail: †nic@ni.aist.go.jp

あらまし　ブランドとは，企業や各種団体，個人が，自らの有する製品，サービス，技術等を他から区別するために用いる要素の組合せである．ロゴ等のブランド要素を放送映像で露出させることへの投資は，ブランド構築のための代表的な手段である．投資効果の確認のために，放送映像上におけるブランド要素の露出面積や露出時間，露出位置等を自動的に調べる方法の開発は重要となる．本論文では，このブランド露出調査への局所不変特徴量の応用について述べる．特に，計算時間の短縮を目的とした，GPU による特徴量抽出処理の実装を検討する．局所不変特徴量の抽出では，スケールスペースピラミッドの生成や記述子の計算において，数多くの局所演算が必要となる．その局所演算を GPU の有する複数のコアで並列実行し，計算速度の向上を図る．局所領域の設定に密なエッジサンプリングによる方法と特徴点抽出による方法を併用し，記述子として SIFT 記述子を用いるアルゴリズムを実装した．720×480 画素の放送映像に対し処理を行った結果，計算時間は約 150[ms] となり，対 CPU 比で約 18 倍の高速化が実現できた．
キーワード　ブランド露出調査，物体認識，局所不変特徴量，GPU，並列処理

GPU-Accelerated Local Invariant Feature Extraction for Brand Exposure Analysis

Naoyuki ICHIMURA†

† National Institute of Advanced Industrial Science and Technology (AIST)
Tsukuba Central 2, 1-1-1, Umezono, Tsukuba, Ibraki, 305–8568 Japan
E-mail: †nic@ni.aist.go.jp

Abstract　A brand is the combination of entities used to distinguish the properties of corporations, associations and individuals such as products, services and technologies. The investment for exposing brand entities, e.g., logos, to broadcasts is a typical way to create brand equities. In order to see the effectiveness of the investment, a method to automatically calculate the sizes, duration times and locations of brand entities on broadcasts is necessary. In this paper, we show an application of local invariant features to brand exposure analysis. In particular, a GPU-based implementation of an algorithm to extract local invariant features is considered for fast computation. Since large amount of local operations are required in extracting local invariant features to calculate scale space pyramids and descriptors, the processing cores in the GPU are utilized to execute the local operations in parallel. We implement an algorithm with local region detection by both dense edge sampling and feature point extraction to calculate the SIFT descriptors. Using the GPU-based implementation, we obtain the computational time around 150 [ms] for 720×480 pixel images, which is about 18 times faster compared to the CPU counterpart.
Key words　Brand exposure analysis, object recognition, local invariant features, GPU, parallel processing

1. ま え が き

ブランドとは，企業や各種団体，個人が，自らの有する製品，サービス，技術等を他から区別するために用いる要素の組合せである [4]．ブランドの構成要素には，ブランド・ネーム，ロゴ，パッケージデザイン等があり，これらをブランド要素と呼ぶ．図 1 の上段に，ブランド要素の例として，スポーツイベントの放送におけるロゴを示す．このようにブランド要素を放送映像で露出させる

(a) (b) (c)

図 1: スポーツイベントにおけるロゴの使用例．イベントのスポンサーは，出資の見返りとして，このようなロゴを設置する権利を得る．(a) F1 の放送映像 [1]．(b) ツール・ド・フランスの放送映像 [2]．(c) MotoGP の放送映像 [3]．上段：原画像，下段：局所領域の設定結果．正方形が局所領域を表す．表示の簡明さのため，局所領域は全体の 1/4 のみを示している．

ことへの投資は，ブランド構築のための代表的な手段である．よって，投資効果確認のため，放送映像上におけるブランド要素の露出面積や露出時間，露出位置等を調査することは重要である．このような調査は，ブランド露出調査と呼ばれている．従来，この調査は人手によって行われてきた．例えば，調査員はマウスを使用し，ビデオの各フレーム毎に，目的とするロゴがどこにあるかを指定する．このような作業は調査員に多くの労力を課すと共に，調査員による結果のばらつきが混入する要因となっている．

本研究の目的は，ブランド露出調査を自動化することである．図 1 に示すように，ブランド要素は多様な見え方で映像上に表れる．よって，視点の移動や照明条件の違いにより見えに変化が生じても，ブランド要素を検出できる機能が必要とされる．また，視野逸脱や遮蔽による隠れで一部が見えない状態でも，部分的に検出を行う機能も要求される．このような機能を実現するために，局所不変特徴量 [5]〜[9] を用いる．局所不変特徴量は，(1) 局所領域の設定，(2) 局所領域の画像特徴を表す記述子の計算，の 2 段階の処理を通じて抽出される．スケールスペースに基づく特徴点抽出や局所座標系の導入等を通じ，上記 (1),(2) の処理結果が画像の幾何学的変換や輝度変化に対し不変になるようにする．また，特徴の局所性により，隠れへの対応が可能である．図 1 の下段に局所領域の例を示す．図中の大小様々な正方形が，記述子を計算する局所領域を表している．このような局所領域内で計算された記述子を用い，ブランド要素の検出を行う．

局所不変特徴量を用いたブランド要素の検出には，(i) 対応付けによる方法，(ii) 学習に基づく方法，が考えられる．前者では，検出対象のモデル画像および放送映像の両方から特徴量を抽出し，それらの対応付けを最近傍法等により行う．得られた対応点から射影変換行列の計算等を行い，検出対象の位置を得る．この方法は，検出対象のモデル画像が 1 枚しかなくても，適用可能という利点がある．実際に，不変特徴量の使用により，モデル画像が 1 枚の場合にも，ある程度の見えの変化に対応可能である [10]．また，明示的に点対応を取るため，対象の位置検出精度が高いという利点もある．しかし，対応付けのロバスト性向上を図るために，種々の見えの変化を含む複数のモデル画像を用いた場合，モデル画像から得られる特徴量に対し何らかの集約を行わないと，対応付けの計算コストが非常に高くなる．学習に基づく検出方法の 1 つとしては，Bag of Features(BoF) によるものが考えられる [11]〜[14]．この方法では，検出対象の種々の見えの変化を含むモデル画像を大量に集め，それらから局所不変特徴量を抽出する．得られた特徴量のベクトル量子化を通じ，visual words と呼ばれる代表ベクトルを求める．そして，検出を行う領域における visual words の出現頻度を特徴ベクトルとして，Support Vector Machine(SVM) 等の識別器により認識を行う．この方法には，特徴量を visual words として集約するため，種々の見えから得られる情報を容易に認識に反映できるという利点がある．その一方，visual words の作成過程で特徴量の位置情報が失われるため，対象の位置検出は sliding window で別途行う必要がある．Sliding window では，位置とスケールの探索を行うため，検出時の計算量が多くなる．識別器が線形 SVM の場合には，分岐限

定法により sliding window の計算量を削減する方法が提案されているが [14]，計算に integral image を利用するため，検出対象の回転等への対応が問題となる．また，明示的に点対応を取らない BoF，sliding window および SVM を用いる方法では，位置検出精度を高く取るのは現状では困難と思われる．

上記以外にもブランド要素の検出方法は考えられ，それぞれ一長一短はあると思われるが，いずれの場合も，局所不変特徴量が共通基盤となる得るものと考えられる．本論文では，その局所不変特徴量の高速な抽出について検討する．特徴量抽出処理では，スケールスペースピラミッドの生成や記述子の計算のために，数多くの局所演算が必要となる．よって，アルゴリズムの工夫による計算量の削減 [15]〜[19] と共に，高速な実装方法を検討することが重要である．高速な実装方法の 1 つとして，Graphics Processing Unit(GPU) の使用がある．GPU はグラフィックスパイプラインを並列処理するために，多数のコアを有している．近年，それらのコアを，汎用計算に利用することが活発に行われている [20]〜[22]．特徴量抽出処理で多用される局所演算は，画素毎や局所領域毎の並列化が容易であるため，多数のコアを用いた並列処理により高速化が期待できる典型的な例と言える．よって，本論文では GPU による特徴量抽出処理の実装を選択した．

GPU による局所不変特徴量抽出処理の実装は，SIFT や SURF，FRBD 等のアルゴリズムで行われている [23]〜[27]．各方法では，局所領域の設定方法と記述子が異なる．本研究では，局所領域の設定において，密なエッジサンプリング [28] と Laplacian of Gaussian(LoG) フィルタに基づく特徴点抽出 [19], [29] を併用する．また，記述子としては，比較実験により高い不変性を示している SIFT 記述子を使用する [9], [30]．処理の一例として，図 1 に示すような放送映像に対し，実装したプログラムを適用した．その結果，720×480 画素の解像度において処理時間は約 150[ms] となり，対 CPU 比で約 18 倍の高速化が実現できることを確認した．

以下では，まず，特徴量抽出処理の概要を示す．その後に，GPU による実装について述べる．最後に，計算時間の測定結果，および，対応付けによるロゴの検出結果を示す．

2. 局所不変特徴量抽出処理の概要

本節では，局所不変特徴量抽出処理の概要を示す [28], [29]．図 2 は，処理のフローチャートである．処理は，フィルタリング，局所領域の設定，および，記述子の計算の 3 つの部分に分けることができる．以下に，それぞれの概要を示す．

2.1 フィルタリング

まず，入力画像から，ガウスフィルタによりスケールスペースピラミッドを生成する．そして，そのスケールスペースピラミッドに対し，3 種類のフィルタを適用する．微分フィルタでは，輝度勾配の大きさと方向を計算する．ここでは，文献 [31] の 5×5 のフィルタを使用した．近似 LoG フィルタ [19], [29] とコーナー検出フィルタ [32] は，特徴点の位置と固有スケール [33] を得るために使用する．近似 LoG フィルタでは，図 3 に示すように，正規化 LoG 関数の極値点に対応する画素のみを用い，LoG フィルタの応答を近似する．基本的には，極小値に対応する中心画素の値，極大値に対応する円上の画素の平均値，それらの差に基づいてコントラストを検出する役割りを果たす．この近似 LoG フィルタを用いて多重解像度でコントラスト検出を行い，その結果からスケールスペースの極値点を探索し，特徴点の位置と固有スケールを得る．また，コーナー検出フィルタにより，スケールスペースの極値点でのコーナーらしさを計算し，極値点近傍でのエッジ形状を制約することに利用する．

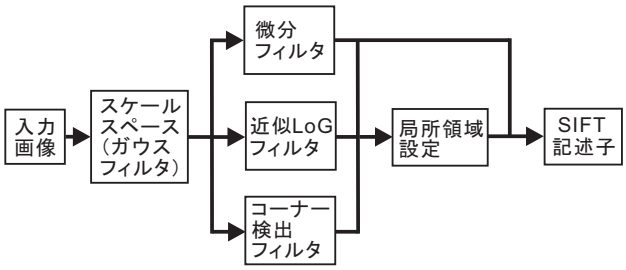

図 2: 局所不変特徴量抽出処理のフローチャート．密なエッジサンプリングによる局所領域，および，特徴点抽出による局所領域，それらを併用して記述子を計算する．

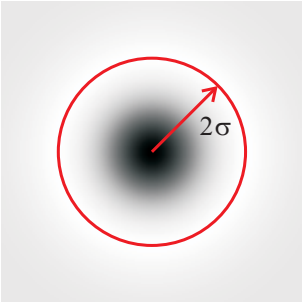

図 3: 近似 LoG フィルタ．上の画像の濃淡の変化は，正規化 LoG 関数の値の変化を表している．正規化 LoG 関数の極小値に対応する中心画素と，極大値に対応する円上の画素により，LoG フィルタの応答を近似する．フィルタの半径は 2σ となる．ここで，σ は標準偏差である．画素値とフィルタ応答の関係は，1 次多項式によりモデル化する．多項式の係数は，LoG フィルタの応答との誤差を評価関数とし，大量の事例画像を用いて決定する [19], [29].

2.2 局所領域の設定

局所領域の設定は，次の 2 つの方法で行う．1 つ目の方法は，密なエッジサンプリングによる方法 [28] である．微分フィルタの処理結果の全スケールにおいて，微分フィルタの出力がしきい値以上，かつ，空間 3×3 近

傍の極大点を探索する．そして，その条件に合致する全てのエッジ点を中心とし，エッジ点が存在するスケールに比例した大きさをもつ局所領域を設定する．2 つ目は，特徴点抽出による方法である．近似 LoG フィルタの出力を使用し，スケールスペースの 3×3×3 近傍での極値点を探索する．極値点において，コントラストを表す近似 LoG フィルタの出力，および，コーナーらしさを表すコーナー検出フィルタの出力，この 2 つの出力がしきい値以上ならば，その極値点を特徴点とする．特徴点の固有スケールは，極値点のスケールである．そして，特徴点の位置を中心として，固有スケールに比例した大きさをもつ局所領域を設定する．この特徴点抽出による方法により，主として blob を特徴とした局所領域が得られる．よって，2 つの方法を併用することにより，エッジと blob という性質の異なる特徴に基づいた局所領域の設定が行える．

2.3 記述子の計算

上記の様にして得られた局所領域内部において，輝度勾配の大きさと方向を用いて SIFT 記述子 [6], [7], [30] を計算する．局所領域を 4×4 のブロックに分割し，それぞれのブロックにおいて，輝度勾配の大きさとガウス関数で重み付けた輝度勾配の方向のヒストグラムを作成する．8 つのビンを有する各ブロックのヒストグラムを連結し，ノルムの正規化を行った後に，128 次元の記述子を得る．

上記の特徴量抽出処理では，スケールスペースピラミッドの生成やその結果に対するフィルタリング，極値点の探索，さらに，局所領域での記述子の計算において，数多くの局所演算が必要とされる．この局所演算を並列化し，処理の高速化を図るための実装について次節で述べる．

3. GPU による特徴量抽出処理の実装

本節では，GPU による特徴量抽出処理の実装について述べる．

3.1 並列処理の概要

開発環境として，NVIDIA 社の CUDA [21] を用いた．CUDA のハードウェアモデルでは，GPU 内に複数のマルチプロセッサがあり，各マルチプロセッサは複数のコアを有する．このプロセッサの階層構造に応じて，データも階層化する．まず，データをいくつかのブロックに分割する．そして，各ブロックのデータに対し，いくつかのスレッドを付随させる．ブロックはマルチプロセッサに割り当てられ，各ブロックのスレッドがコアで並列実行される．結果として，データ全体が並列処理され，処理速度の向上が図られる．

特徴量抽出処理の場合には，フィルタリングと局所領域の設定においては画素を単位とし，記述子の計算においては局所領域を単位として並列化を行う．よって，フィルタリングと局所領域の設定では，画像をブロック分割し，各ブロック内部の画素に対する処理が並列化される．また，記述子の計算では，複数の局所領域でブロックを構成し，各局所領域での処理が並列化される．CPU では，for ループにより画素および局所領域を逐次的に参照して処理を行う．この CPU による for ループでの処理を並列化していることから，本実装は Parallel-For ベースにより並列化を図っていると言える．

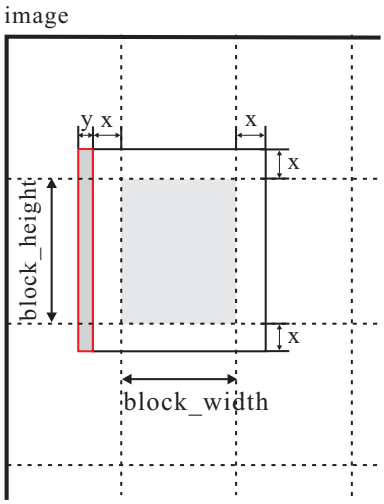

図 4: フィルタリングにおける画像のブロック分割．各ブロックは，GPU 内に複数あるマルチプロセッサに割り当てられる．そして，各ブロック内の画素に付随するスレッドが，コアにより並列実行される．ブロックとスレッドの数は execution configuration と呼ばれ，GPU で実行する関数の呼出し時に指定する．shared メモリへのコピー時に付加するデータの幅 x と y は，フィルタのスケールにより定まる．よって，スケールによりスレッド数が変化する．

3.2 GPU でのメモリの割り当て

GPU で処理を行う場合，ホストコンピュータのメインメモリにあるデータを，GPU のメモリに転送する．この転送には時間を要するため，一度転送を行った後は，可能な限り GPU のみで処理をする．その際，GPU 内部にある複数の種類のメモリを，その性質に応じて使い分けることが高速化のために重要である．

本実装では，global，constant，shared の 3 種類のメモリを使用した．これらのメモリは，記した順番に，容量が多いがアクセス速度は遅いという性質がある．メモリの使用例を，フィルタリングの場合を用いて説明する．ホストから転送された画像は，global メモリに格納される．その global メモリ上の画像を，図 4 に示すようにブロックに分割する．ブロック内の画素に対していくつかのスレッドを付随させ，画素毎にフィルタリングを並列化する．フィルタリングを行う前に，各スレッドによりブロック内の画素データを shared メモリにコピーする．これは，shared メモリがコアから高速アクセス可能であり，

処理速度が向上するためである．このコピーの際，ブロック境界部分の処理を正しく行うために，フィルタサイズの半分だけ，ブロックの外側のデータもコピーする必要がある．図 4 では，x がフィルタサイズの半分を示している．また，global メモリへのアクセス効率を上げるために (coalesced memory access を用いるために)，連続したメモリ領域の長さを，half warp size と呼ばれる数の倍数となるように調整する [34]．図 4 では，y がその調整幅である．最終的に，(block_height+2x)(block_width+2x+y) 個の画素データが shared メモリにコピーされる．フィルタリングに用いるオペレータは，全スレッドで共通かつ書き換えが不要でなので，constant メモリに格納する．このように，メモリへのアクセス効率を上げること，および，アクセス速度の速いメモリ間で演算を行うことが，高速化のために重要となる．また，ブロックとスレッドの数は execution configuration と呼ばれ，これらの数も実行速度に影響を与える．ブロックの数は，その幅と高さにより間接的に指定する場合が多い．

3.3 各処理の実装の詳細

以下に，execution configuration を含んだ GPU での実装の詳細を示す．ここでは，図 2 にはない補助的な処理も示している．

Image transfer: ホストコンピュータ (CPU) から GPU への画像データの転送.

Y Component: カラー画像からの輝度画像生成．global メモリ上で処理．ブロックサイズは 16×32．ブロックの全画素にスレッドを割り当て並列化.

Down sampling: スケールスペースピラミッド (5 オクターブ，各オクターブ 5 枚の画像) の生成に必要な画像のダウンサンプリング．global メモリ上で処理．ブロックサイズおよびスレッドの割り当ては，**Y Component** と同じ．

Gaussian filter: ガウスフィルタを用い，スケールスペースピラミッドを生成．初期スケールは 1.6．フィルタサイズは，ガウス関数の値が 10^{-3} 未満になる点で定義域を打ち切って決定．constant, shared メモリ使用．変数分離性を利用．行処理における 1 次元のブロックサイズは 128 とし，図 4 の x,y に相当するデータを付加した後に，全画素にスレッドを割り当てる．列処理でのブロックサイズは 48×16 とし，データを付加した後，ブロックを高さ 8 のサブブロックに分ける．1 つのサブブロックの全画素にスレッドを割り当て，各スレッドで他のサブブロックの同一位置にある画素に対し逐次処理を行う．このようにサブブロックを用いて処理を行うのは，1 つのブロックにおけるスレッド数に上限が存在することによる．本実装では，スレッド数の上限は 512 である．

Gradient filter: 5×5 の微分フィルタ [31] による輝度勾配の計算．constant, shared メモリ使用．ブロックサイズは 16×16 とし，データを付加した後，ブロックを高さ 8 のサブブロックに分ける．そして，**Gaussian filter** の列処理と同様にスレッドを割り当てる．

ALoG-C filter: コーナー検出フィルタを併用する近似 LoG フィルタ [19], [32] による，特徴点抽出のためのスケールスペースピラミッドの生成．constant, shared メモリ使用．ブロックサイズおよびスレッドの割り当ては，**Gradient filter** と同じ．

Edge sampling: 微分フィルタの処理結果の全スケールにおいて，微分フィルタの出力がしきい値以上，かつ，空間 3×3 近傍の極大点となる画素をサンプリング．微分フィルタの出力のしきい値は 10．global メモリ上で処理．ブロックサイズは 16×32 とし，全画素に対しスレッドを割り当てる．得られた画素の位置とスケールを，局所領域の位置とスケールとする．そして，その位置とスケールを，特徴リストとしてまとめる．GPU ではブロック内のスレッド間でのみ同期が可能であるため，共通変数である局所領域数に対し，全てのスレッド間で排他制御を行いつつ処理をすることはできない．よって，特徴リストの生成は CPU で実行する．この生成に伴い，CPU–GPU 間でデータ転送が生じる．この転送時間のため，転送を必要としない CPU での処理よりも GPU での処理が遅くなる場合もある．

Feature point: **ALoG-C filter** で得られるスケールスペースにおける 3×3×3 近傍での極値探索による，特徴点およびその固有スケールの抽出．各オクターブ毎に抽出．近似 LoG フィルタとコーナー検出フィルタの応答であるコントラストと cornerness に対するしきい値処理により，特徴点を選択．各しきい値は，10 および 100．global メモリ上で処理．ブロックサイズおよびスレッドの割り当ては，**Edge sampling** と同じ．また，特徴リストへの特徴点の位置と固有スケールの登録のために，CPU–GPU 間でのデータ転送が生じる．

Orientation: SIFT 記述子における dominant orientation [7] の計算．局所領域の大きさは，中心画素のスケールの 5 倍．global メモリ上で処理．特徴リストを長さ 16 のブロックに分割し，局所領域を単位として並列化．ブロック内の全局所領域にスレッドを割り当てる．

Descriptor: SIFT 記述子 (128 次元) の計算．局所領域の大きさは，中心画素のスケールの 15 倍．global メモリ上で処理．ブロックサイズおよびスレッドの割り当ては，**Orientation** と同じ．

次節では，上記の処理を実装し，放送映像に適用した結果について述べる．

4. 実験結果

実験には，以下の計算機環境を使用した．

ホストコンピュータ: HP xw8600 Workstation
OS: Fedora8
CPU: Intel Quadcore Xeon (3.16GHz/12MBL2)

表 1: 局所不変特徴量抽出に必要な計算時間．単位は [ms]．100 回の処理の平均値を示す．画像の解像度は，720×480 画素である．総計算時間には，ここに示したタスク以外の処理，例えばメモリアロケーション等も含まれる. タスク名に * がつく処理では，特徴リスト生成のために，CPU-GPU 間でデータ転送を必要とする．

Image	F1			Tour de France			MotoGP		
Task\Processor, Time ratio	GPU	CPU	Ratio	GPU	CPU	Ratio	GPU	CPU	Ratio
Image transfer	2.608	N/A	N/A	2.579	N/A	N/A	2.581	N/A	N/A
Y component	0.116	3.891	33.5	0.117	3.960	33.8	0.117	3.998	34.2
Down sampling	0.144	0.856	5.94	0.144	0.862	5.99	0.145	0.868	5.99
Gaussian filter	8.863	263.333	29.7	8.831	263.824	29.9	8.847	263.324	29.8
Gradient filter	4.400	219.765	49.9	4.399	219.595	49.9	4.430	219.220	49.5
ALoG-C filter	13.659	308.415	22.6	13.655	308.600	22.6	13.653	308.712	22.6
*Edge sampling	11.195	9.597	0.86	11.238	9.856	0.88	11.211	9.862	0.88
*Feature point	11.833	159.592	13.5	11.735	158.073	13.5	11.859	158.729	13.4
Orientation	7.976	169.138	21.2	7.669	170.356	22.2	8.027	170.337	21.2
Descriptor	85.530	1652.452	19.3	84.823	1483.710	17.5	87.544	1626.656	18.6
Total	150.004	2788.497	18.6	148.864	2620.296	17.6	152.096	2763.157	18.2
#feature	6405			5731			6227		

メモリ: 8GB DDR2 FBD RAM
グラフィックスカード: NVIDIA Quadro FX4600,
および，GeForce GTX 280
ディスプレイへの表示は，プライマリカードの Quadro で行った．セカンダリカードの GeForce は演算専用とした．このカードは 240 個のコアを有する．また，warp size は 32，Compute Capability 1.3 に準拠し [34]，1 つのブロックにおけるスレッド数の上限は 512 である．CPU では単一コアを使用して特徴量抽出を行い，GPU での計算時間と比較した．浮動小数点演算は，全て単精度で行った．

図 1 に示した放送映像より得た画像を用い，計算時間の計測を行った．画像の解像度は，720×480 画素である．表 1 に，これらの画像に対する GPU と CPU による計算時間とその比，および，抽出された特徴量数を示す．計算時間は，3.3 に示した各処理について計測した．**Orientation** および **Descriptor** の計算時間は，局所領域の数と大きさに依存して変化する．それ以外の処理の計算時間は，主として画像の解像度に依存する．また，計算時間の変動要因の 1 つとして，OS のリアルタイム性が保証されないことがある．この変動要因の影響を軽減するために，100 回処理を行い平均計算時間を求めた．

表 1 の結果から，CPU 使用時に比べ，GPU の使用により特徴量抽出処理が約 18 倍高速になったことがわかる．計算時間は約 150[ms] である．同様の結果を CPU で得るためには，現状では PC クラスタやプログラムの最適化が必要とされると考えられる．そのような実装と比較し，CUDA を用いた GPU での実装はより簡便であり，また，コスト面でも有利である．よって，表 1 の結果は，特徴量抽出処理における GPU の有用性を明確にするものと考えている．

図 5 および図 6 に，GPU による実装により抽出された特徴量を使用し，放送映像内のロゴを検出した結果を示す．検出は，文献 [10] の対応付けに基づくアルゴリズムで行った．このアルゴリズムは，特徴量の最近傍法による対応付けと，見えの制約を導入した RANSAC による射影変換行列の計算を基本とするものである．モデル画像は各結果の上部に示す 1 枚のみを用いた．モデル画像からの見えの変化，特にスケールと照明条件の変化，および，隠れにもかかわらず，ロゴの位置と面積の算出が可能なことがわかる．また，同一のロゴが複数存在しても，検出が可能であった．この結果から，GPU による実装で得られた特徴量により，種々のシーンでロゴの検出が可能であることが確認できた．

5. むすび

本論文では，局所不変特徴量抽出処理の GPU による実装について検討を行った．密なエッジサンプリング，特徴点抽出および SIFT 記述子を用いた処理を実装した結果，720×480 画素の放送映像において，対 CPU 比で約 18 倍の高速化が実現できた．また，抽出された特徴量により，ロゴの検出が可能なことを確認した．

本研究で示した特徴量抽出処理の GPU による実装のような，多数の計算コアを有したハードウェアによる高性能科学技術計算 (HPC) は，今後急速に発達すると予想される．その発達はコンピュータビジョンにおいて非常に有用であると考えているため，GPU 等のハードウェアを利用した特徴量抽出や物体認識等の高速化に関し，

図 5: GPU による実装により抽出された特徴量を使用した，放送映像でのロゴの検出結果．検出は，文献 [10] の対応付けに基づくアルゴリズムで行った．ロゴのモデル画像は，各結果の上部に示してある 1 枚のみを用いた．モデル画像からの見えの変化や隠れにもかかわらず，ロゴの面積と位置の算出が可能である．

今後も研究開発を進める予定である．また，今回の実装は適用対象を限定しないので，ブランド露出調査以外の応用も検討したい．

謝辞

本研究の一部は，科学研究費補助金，課題番号 18500145 の助成の下で行われた．

文　献

[1] フジテレビ 721 において放送された映像を使用している．
[2] J SPORTS において放送された映像を使用している．
[3] G+において放送された映像を使用している．
[4] ケビン・レーン・ケラー (恩蔵直人, 亀井昭宏 (訳)). 戦略的ブランド・マネジメント. 東急エージェンシー, 2000.
[5] C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. *IEEE Trans. PAMI*, Vol. 19, No. 5, pp. 530–535, 1997.
[6] D. Lowe. Object recognition from local scale-invariant features. In *Proc. Int. Conf. Comp. Vis.*, pp. 1150–1157, 1999.
[7] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comp. Vis.*, Vol. 60, No. 2, pp. 91–110, 2004.
[8] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisser-

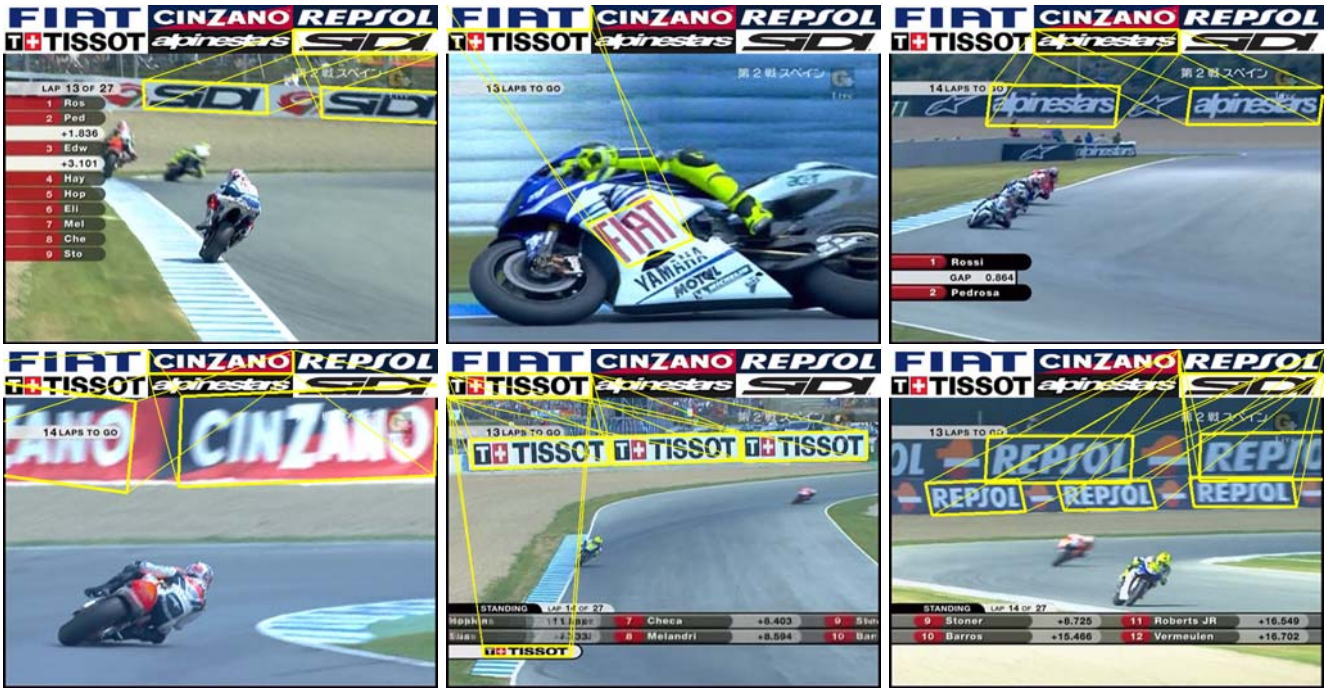

図 6: GPU による実装により抽出された特徴量を使用した，放送映像でのロゴの検出結果 (図 5 の続き).

man, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *Int. J. Comp. Vis.*, Vol. 65, No. 1/2, pp. 43–72, 2005.

[9] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. PAMI*, Vol. 27, No. 10, pp. 1615–1630, 2005.

[10] N. Ichimura. Recognizing multiple billboard advertisements in videos. In *Proc. Pacific-Rim Symp. on Image and Video Technology (PSIVT)*, pp. 463–473, 2006.

[11] C. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Proc. Workshop on Statistical Learning in Computer Vision*, pp. 1–22, 2004.

[12] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *Proc. European Conf. Comp. Vis.*, pp. 490–503, 2006.

[13] 柳井啓司. 一般物体認識の現状と今後. 情報処理学会論文誌：コンピュータビジョンとイメージメディア, Vol. 48, No. SIG 16, pp. 1–24, 2007.

[14] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: object localization by efficient subwindow search. In *Proc. Int. Conf. Comp. Vis. Patt. Recog.*, pp. 1–8, 2008.

[15] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Proc. Int. Conf. Comp. Vis.*, Vol. 2, pp. 1508–1515, 2005.

[16] V. Lepetit and P. Fua. Keypoint recognition using randomized tree. *IEEE Trans. PAMI*, Vol. 28, No. 9, pp. 1465–1479, 2006.

[17] H. Bay, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. In *Proc. European Conf. Comp. Vis.*, Vol. 1, pp. 404–417, 2006.

[18] M. Grabner, H. Grabner, and H. Bischof. Fast approximated SIFT. In *Proc. Asian Conf. Comp. Vis.*, pp. 918–927, 2006.

[19] 市村直幸. 正規化 LoG 関数の近似に基づく局所不変特徴量の抽出. 信学技報, No. PRMU2007-314, pp. 449–456, 2008.

[20] http://www.gpgpu.org/.

[21] CUDA Zone: http://www.nvidia.co.jp/object/cuda_home_jp.html.

[22] ATI Stream: http://ati.amd.com/technology/streamcomputing/.

[23] S. Heymann, K. Müller, A. Smolic, B. Fröhlich, and T. Wiegand. SIFT implementation and optimization for general-purpose GPU. In *Proc. Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pp. 317–322, 2007.

[24] SiftGPU: http://www.cs.unc.edu/~ccwu/siftgpu/.

[25] T. B. Terriberry, L. M. French, and J. Helmsen. GPU accelerating speeded-up robust features. In *Proc. Int. Symp. on 3D Data Processing, Visualization and Transmission (3DPVT)*, pp. 355–362, 2008.

[26] N. Cornelis and L. V. Gool. Fast scale invariant feature detection and matching on programmable graphics hardware. In *Proc. Workshop on Computer Vision on GPU's (in conjunction with CVPR08)*, 2008.

[27] C. Bibby and I. Reid. Fast feature detection with a graphics processing unit implementation. In *Proc. Int. Workshop on Mobile Vision*, 2006.

[28] 市村直幸. 密なエッジサンプリングに基づく局所不変特徴量による対応付け. 信学技報, 2009.

[29] 市村直幸. 近似 LoG フィルタを用いた局所不変特徴量の抽出 – GPU による実装 –. 情処研報, No. 2008-CVIM-165, pp. 243–250, 2008.

[30] 藤吉弘亘. Gradient ベースの特徴抽出 – SIFT と HOG –. 情処研報, No. 2007-CVIM-160, pp. 211–224, 2007.

[31] S. Ando. Consistent gradient operators. *IEEE Trans. PAMI*, Vol. 22, No. 3, pp. 252–265, 2000.

[32] M. Trajkovic and M. Hedley. Fast corner detection. *Image and Vision Computing*, Vol. 16, pp. 75–87, 1998.

[33] T. Lindeberg. Feature detection with automatic scale selection. *Int. J. Comp. Vis.*, Vol. 30, No. 2, pp. 79–116, 1998.

[34] *NVIDIA CUDA Programming Guide, Version 2.0, Sec.5.1.2*, 2008.