情報処理学会研究報告
IPSJ SIG Technical Report

GPU による特徴点とエッジに基づく局所不変特徴量の抽出

市　村　直　幸[†1]

近年，局所不変特徴量は，画像の対応付けや物体認識における基盤要素として幅広く用いられている．局所不変特徴量の抽出には，(1) 局所領域の設定，(2) 記述子の計算，の 2 段階の処理を必要とする．本論文では，これらの処理の GPU による高速化を検討する．特に，対応付けと物体認識の両方における適用性を重視し，特徴点とエッジの併用により局所領域を設定するアルゴリズムを実装する．フィルタリングにおける近傍領域毎，および，局所領域毎のデータ並列性を利用し，特徴量抽出に必要な数多くの局所演算を GPU の有する複数のコアで並列実行する．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．

GPU-Accelerated Local Invariant Feature Extraction Based on Feature Points and Edges

Naoyuki Ichimura[†1]

Local invariant features have been widely used as fundamental elements for image matching and object recognition. Extracting local invariant features is performed by the following steps: (1) detecting local regions, (2) computing descriptors. This paper presents a GPU-based implementation of the steps for fast computation. In particular, an algorithm using feature points and edges to detect local regions is considered for the applicability to both image matching and object recognition. Utilizing the data parallelisms of neighborhood regions in image filtering and local regions, we can execute large amount of local operations required in extracting local invariant features in parallel.The computational time of the GPU-based implementation for 720×480 pixel images is around 150 [ms], which is about 18 times faster compared to the CPU counterpart.

†1 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology (AIST)

1. まえがき

局所不変特徴量は，画像の特徴表現の一種である．この特徴量は，(1) 画像内での局所領域の設定，(2) 局所領域の画像特徴を表す記述子 (descriptor) の計算，の 2 段階の処理を通じて抽出される[1)–4)]．図 2 に局所領域の例を示す．図中の正方形が局所領域を表す．このような局所領域で計算された記述子は，局所領域内部の輝度やテクスチャ，エッジ等に基づいて画像特徴を数値化したものであり，多くの場合，ベクトルの形態をとる．

局所不変特徴量には，2 つの利点がある．1 つは，局所領域を用いることによる，視野逸脱や遮蔽による隠れへの耐性である．シーンの一部に隠れが生じても，見えている局所領域の特徴量が使用できる．もう 1 つの利点は，特徴量に不変性を付与できることである．スケールスペースピラミッドの利用や局所座標系の導入等を通じ，画像の幾何学的変換や輝度変化に対し特徴量が不変になるように，上記 (1),(2) の特徴量抽出処理を構成できる．これらの利点にから，隠れや視点の移動，照明条件等の違いにより，基準画像からの見えの変化がシーンにおいて生じたとしても，その変化の影響を軽減し，シーンから基準画像と同様の特徴量を得ることができる．

局所不変特徴量を用いた代表的なアルゴリズムとして，画像の対応付けと物体認識が挙げられる．画像の対応付けは，複数の画像間で共通部分を見出す処理である．その一例を図 4 に示す．図 4 の上部と下部の画像から特徴量を抽出し，それらの対応付けを特徴量間の距離を用いた最近傍法により行う．得られた対応点から射影変換行列の計算を行うと，図 4 の検出結果が得られる．物体認識の例としては，Bag–of–Features(BoF) によるものが挙げられる．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．この方法では，visual words の作成過程で特徴量の位置情報が失われる．よって，対象の位置検出は sliding window で別途行い，図 4 のような認識結果を得る．

上記のように局所不変特徴量は，画像の対応付けや物体認識における基盤要素として幅広く用いられている．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．その局所演算は，画素毎や局所領域毎の並列化が可能であるため，多数のコアを有する Graphics Processing Unit(GPU) を用いた並列処理[5)–7)] により高速化が期待できる. 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[8)–12)]．各方法では，局所領域の設定方法と記述子が異なる．しかし，ア

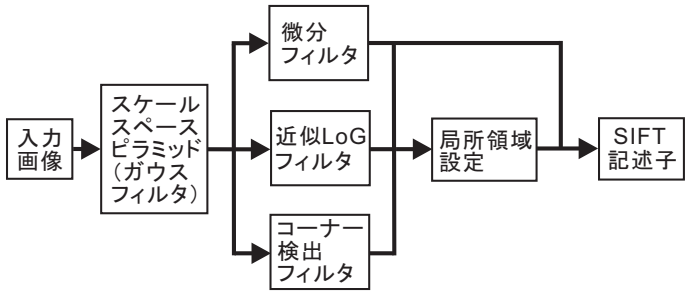


図 1 局所不変特徴量抽出処理のフローチャート．特徴点抽出による局所領域，および，密なエッジサンプリングによる局所領域，それらを併用して SIFT 記述子を計算する．

ルゴリズムに違いはあっても，基本的に，局所領域は特徴点抽出により設定されている．特徴点抽出による局所領域の設定方法は，対応付けにおいて有効性が確認されている[1)–4)]．その一方，物体認識においては，認識性能向上のために，他の局所領域の設定方法が必要との指摘がある[13)–18)]．このことから本研究では，対応付けと物体認識の両方における適用性を重視し，特徴点とエッジを併用した局所領域の設定方法を実装する．特徴点抽出による局所領域の設定には，近似 LoG フィルタ[19)] に基づく方法を用いる．また，エッジによる局所領域の設定には，スケールスペース内のエッジの極大点を全て用いる密なエッジサンプリング[20)] を用いる．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[3),21)]．処理の一例として，図 4 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，局所領域の設定と記述子の計算について説明する．その後に，それらの処理の 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，対応付けによる物体検出例を示す．

2. 局所領域の設定

図 1 に，本論文で実装を行う特徴量抽出処理のフローチャートを示す[19),20)]．処理は大きく，局所領域の設定，および，記述子の計算の 2 つの部分に分けることができる．本節では，局所領域の設定について述べる．

局所領域の設定では，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．画像のダウンサンプリングと複数のスケールをもつガウスフィルタの畳み込みを組み合わせ，スケールスペースピラミッドを生成することにより，局所領域にスケール不変性と並行移動不変性を付与することができる．その結果として，局所領域内部で計算される特徴量にもそれらの不変性が付与される．このスケールスペースピラミッドを用い，次に示す 2 つの方法で局所領域の設定を行う．

2.1 特徴点抽出による局所領域の設定

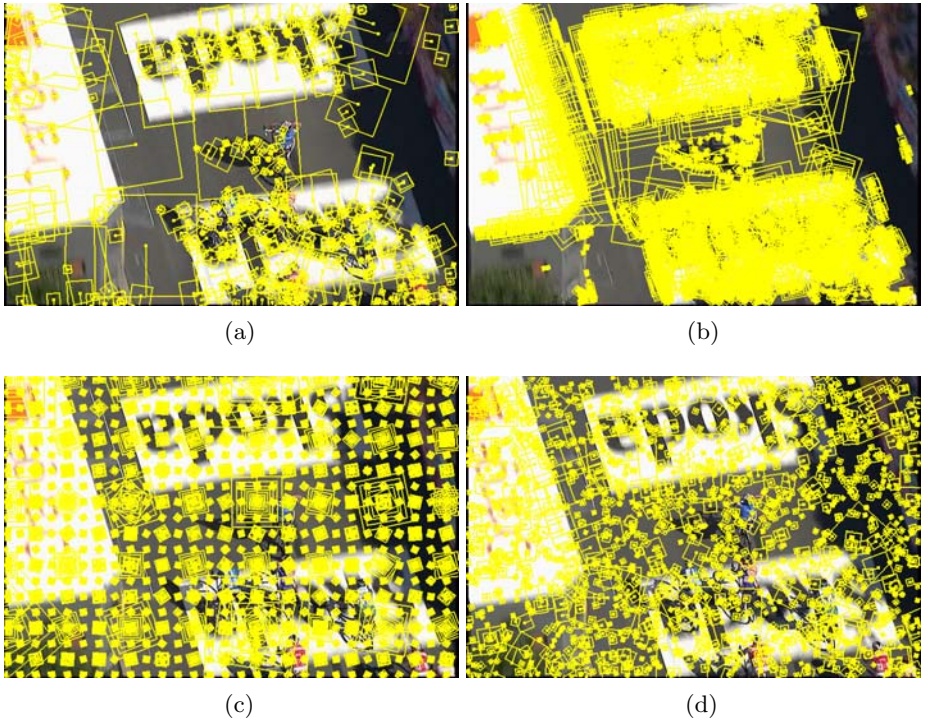
1 つ目の方法は，特徴点抽出による方法である．一般にこの方法では，正規化 Laplacian-of-Gaussian(LoG) 関数に基づくフィルタとコーナー検出フィルタを併用し処理が行われる[3),4)]．前者の正規化 LoG 関数の基本機能はコントラスト検出であり，輝度変化がある部分に局所領域を設定することを目的として導入される．また，後者のコーナー検出フィルタは，開口問題を避けるために局所領域内部のエッジ形状を制約する．このことから，特徴点抽出による方法を用いると，対応を一意に定めやすい輝度変化を有した局所領域の設定が可能となる．よって，特徴点抽出による方法は，主として対応付けにおいて用いられる．

本研究では，近似 LoG フィルタ[19)] と Trajkovic らのコーナー検出フィルタ[22)] を，スケールスペースピラミッドに適用する．そして，その結果のスケールスペースにおいて $3\times3\times3$ 近傍での極値点を探索し，その位置に特徴点があると判断する．つまり，極値点の位置が特徴点の位置とし，極値点が検出されたスケールを特徴点の固有スケール[23)] とする．局所領域は，特徴点を中心として設定され，その大きさは固有スケールの数倍から十数倍とする．なお，全極値点を用いると特徴点数が膨大になる場合もあるため，近似 LoG フィルタおよびコーナー検出フィルタの応答に対するしきい値処理により選択を行う．

2.2 密なエッジサンプリングによる局所領域の設定

2.1 節で述べた特徴点抽出による局所領域の設定方法は，対応付けにおいて多用されてきた．その一方，局所特徴に基づく物体認識では，異なる方法も採用されている．物体認識におけるスケールスペースピラミッド内での局所領域の設定方法を大別すると，エッジサンプリング，グリッドサンプリング，ランダムサンプリングとなる[13)–18)]．図 2 に，特徴点抽出とサンプリングによる方法で設定された局所領域の例を示す．Nowak らは，BoF の枠組の下で特徴点抽出とランダムサンプリングの比較を行い，ランダムサンプリングの方が認識性能が高かったと報告している[17)]．他の論文でも，認識性能向上のために，サンプリングによる方法を使用している[14)–16),18)]．これらのことは，特徴点抽出では，シーンを記述する情報を画像から十分に引き出せていないことを示唆している[20)]．よって，本論文では，サンプリングによる局所領域の設定方法を併用する．これにより，実装するアルゴリズムの対応付けと物体認識の両方への適用性が向上する．

サンプリングよる方法では，図 2(b) に示すようなエッジサンプリングを採用する．図 2(c),(d) に示すグリッドサンプリングやランダムサンプリングでは，輝度変化を全く考慮しないため，設定された局所領域に必ずしも大きな輝度変化が存在するとは言えない．物体



(a) (b)

(c) (d)

図 2 4 つの異なった方法による局所領域の設定結果．(a) 特徴点抽出による方法．(b) エッジサンプリングによる方法．(c) グリッドサンプリングによる方法．(d) ランダムサンプリングによる方法．(a) の特徴点抽出による方法は対応付けにおいて，(b),(c),(d) のサンプリングによる方法は Bag–of–Features による物体認識で主として用いられる．

の種類によっては (例えば空などでは) 輝度変化が少ないことも他の物体との識別に重要な情報となるため，物体認識ではこれらのサンプリングが有効に働くと考えられる．しかし，対応付けでは，輝度変化が少ない部分は対応を一意に定めるには不適当であり，これらのサンプリングは有効でない．よって，対応付けと物体認識の両方における適用性を考慮した場合には，輝度変化を考慮するエッジサンプリングが有効と言える．

実装を行うのは，スケールスペースピラミッドにおいて，空間 3×3 近傍でのエッジの極大点を全て用いる密なエッジサンプリング[20] である．この方法により，特徴点抽出では得られない局所領域を設定することが可能となり，また，特徴点抽出による方法のものと同様に，設定された局所領域には対応付けに有効な輝度変化が含まれることが確認されている．この密なエッジサンプリングのために，図 1 に示すように，スケールスペースピラミッドに対して微分フィルタ[24] を適用する．

3. 記述子の計算

2 節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[2),21)] 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．この際，局所領域毎に dominant orientation[2)] を基準として輝度勾配の方向を求めることにより，特徴量に回転不変性を付与できる．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．このノルムの正規化により，輝度不変性が得られる．

2 節および 3 節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．

4. GPU による特徴量抽出処理の実装

4.1 並列処理の概要

開発環境として，NVIDIA 社の CUDA[6)] 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，各局所領域内部での処理が並列化される．

4.2 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

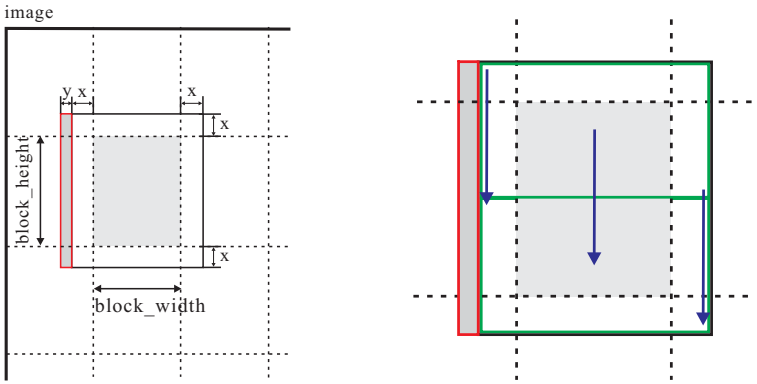


図 3 フィルタリングにおける画像のブロック分割．各ブロックは，GPU 内に複数あるマルチプロセッサに割り当てられる．そして，各ブロック内の画素に付随するスレッドが，コアにより並列実行される．ブロックとスレッドの数は execution configuration と呼ばれ，GPU で実行する関数の呼出し時に指定する．(a)shared メモリへのコピー時に行うデータの付加．付加するデータの幅 x と y は，フィルタのスケールにより定まる．よって，スケールによりスレッド数が変化する．(b) サブブロックの使用．この例では，2 つのサブブロックを示している．最上部のサブブロックにおいて全画素にスレッドを割り当て，そのスレッドによって，残りのサブブロックの同一位置にある画素を逐次処理する．これによって，ブロック内のスレッド数を制限する．

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 3(a) 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 3(a) 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[25]．図 3(a) 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．

4.3 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 1 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 3 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，図 3(b) の例のように，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．

Gradient filter: 5×5 の微分フィルタ[24] 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),22)] 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．

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．得られた特徴点の位置と固有スケールを，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．

Edge sampling: 微分フィルタの処理結果の全スケールにおいて，微分フィルタの応答がしきい値以上，かつ，空間 3×3 近傍の極大点となる画素をサンプリング．微分フィルタの応答のしきい値は 10．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，**Feature point** と同じ．また，特徴リストへ画素の位置とスケールを登録するために，CPU–GPU 間での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．

Orientation: SIFT 記述子における dominant orientation[2] 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．

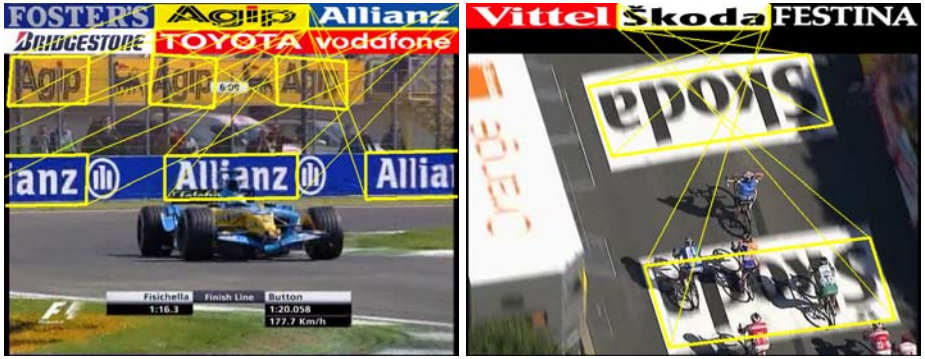
Descriptor: SIFT 記述子の計算．局所領域の大きさは，中心画素のスケールの 15 倍．global メモリ上で処理．ブロックサイズおよびスレッドの割り当ては，**Orientation** と同じ．

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．

5. 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)，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[25]，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．

放送映像[26),27)] より得た図 4 の画像を用い，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．この結果から，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等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．



(a) (b)

図 4 GPU による実装により抽出された特徴量を使用した，放送映像内のロゴの検出結果．(a)F1[26]．(b)Tour de France[27]．検出は，文献[28] の対応付けに基づくアルゴリズムで行った．

表 1 局所不変特徴量抽出に必要な計算時間．単位は [ms]．OS の非リアルタイム性を考慮し，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		
Task\Processor, Time ratio	GPU	CPU	Ratio	GPU	CPU	Ratio
Image transfer	2.608	N/A	N/A	2.579	N/A	N/A
Y component	0.116	3.891	33.5	0.117	3.960	33.8
Down sampling	0.144	0.856	5.94	0.144	0.862	5.99
Gaussian filter	8.863	263.333	29.7	8.831	263.824	29.9
Gradient filter	4.400	219.765	49.9	4.399	219.595	49.9
ALoG-C filter	13.659	308.415	22.6	13.655	308.600	22.6
*Feature point	11.833	159.592	13.5	11.735	158.073	13.5
*Edge sampling	11.195	9.597	0.86	11.238	9.856	0.88
Orientation	7.976	169.138	21.2	7.669	170.356	22.2
Descriptor	85.530	1652.452	19.3	84.823	1483.710	17.5
Total	150.004	2788.497	18.6	148.864	2620.296	17.6
#feature	6405			5731		

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図 4 に示す．検出は，文献[28] 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．ロゴのモデル画像は，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，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，シーン内の物体検出が可能であることが確認できた．BoF 等に基づく物体認識による結果は，今回得られていないので，その実装については今後の課題としたい．

6. 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．特徴点抽出，密なエッジサンプリングおよび 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，抽出された特徴量を用いた対応付けに基づくアルゴリズムにより，放送映像内のロゴの検出が可能なことを確認した．今後も，GPU 等のハードウェアを利用した特徴量抽出や対応付け，物体認識等の高速化に関し，研究開発を進める予定である．

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．

参考文献

1) Schmid, C. and Mohr, R.: Local greyvalue invariants for image retrieval, *IEEE Trans. PAMI*, Vol.19, No.5, pp.530–535 (1997).
2) Lowe, D.: Distinctive image features from scale-invariant keypoints, *Int. J. Comp. Vis.*, Vol.60, No.2, pp.91–110 (2004).
3) Mikolajczyk, K. and Schmid, C.: A performance evaluation of local descriptors, *IEEE Trans. PAMI*, Vol.27, No.10, pp.1615–1630 (2005).
4) Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T. and Gool, L.V.: A comparison of affine region detectors, *Int. J. Comp. Vis.*, Vol.65, No.1/2, pp.43–72 (2005).
5) GPGPU: http://www.gpgpu.org/.
6) CUDA Zone: http://www.nvidia.co.jp/object/cuda_home_jp.html.
7) ATI Stream: http://ati.amd.com/technology/streamcomputing/.
8) Heymann, S., Müller, K., Smolic, A., Fröhlich, B. and Wiegand, T.: SIFT implementation and optimization for general-purpose GPU, *Proc. Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pp. 317–322 (2007).
9) SiftGPU: http://www.cs.unc.edu/~ccwu/siftgpu/.
10) Terriberry, T. B., French, L. M. and Helmsen, J.: GPU accelerating speeded-up robust features, *Proc. Int. Symp. on 3D Data Processing, Visualization and Transmission (3DPVT)*, pp.355–362 (2008).
11) Cornelis, N. and Gool, L.V.: Fast scale invariant feature detection and matching on programmable graphics hardware, *Proc. Workshop on Computer Vision on GPU's (in conjunction with CVPR08)* (2008).
12) Bibby, C. and Reid, I.: Fast feature detection with a graphics processing unit implementation, *Proc. Int. Workshop on Mobile Vision* (2006).
13) Csurka, C., Dance, C.R., Fan, L., Willamowski, J. and Bray, C.: Visual categorization with bags of keypoints, *Proc. Workshop on Statistical Learning in Computer Vision*, pp.1–22 (2004).
14) Fei-Fei, L. and Perona, P.: A Bayesian hierarchical model for learning natural scene categories, *Proc. Int. Conf. Comp. Vis. Patt. Recog.*, Vol.2, pp.524–531 (2005).
15) Ma, X. and Grimson, W.E.: Edge-based rich representation for vehicle classification, *Proc. Int. Conf. Comp. Vis.*, Vol.2, pp.1185–1192 (2005).
16) Mikolajczyk, K., Leibe, B. and Schiele, B.: Multiple object class detection with a generative model, *Proc. Int. Conf. Comp. Vis. Patt. Recog.*, Vol.1, pp.26–36 (2006).
17) Nowak, E., Jurie, F. and Triggs, B.: Sampling strategies for bag-of-features image classification, *Proc. European Conf. Comp. Vis.*, pp.490–503 (2006).
18) 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).
19) 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).
20) 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， No.PRMU2009–51, pp.71–76 (2009).
21) 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).
22) Trajkovic, M. and Hedley, M.: Fast corner detection, *Image and Vision Computing*, Vol.16, pp.75–87 (1998).
23) Lindeberg, T.: Feature detection with automatic scale selection, *Int. J. Comp. Vis.*, Vol.30, No.2, pp.79–116 (1998).
24) Ando, S.: Consistent gradient operators, *IEEE Trans. PAMI*, Vol. 22, No. 3, pp. 252–265 (2000).
25) *NVIDIA CUDA Programming Guide, Version 2.0, Sec.5.1.2* (2008).
26) F1 の放送映像：フジテレビ 721 において放送された映像を使用している．
27) Tour de France の放送映像：J SPORTS において放送された映像を使用している．
28) Ichimura, N.: Recognizing multiple billboard advertisements in videos, *Proc. Pacific-Rim Symp. on Image and Video Technology (PSIVT)*, pp.463–473 (2006).