Vol.2010-CVIM-171 No.20
2010/3/19

GPUによる方向マップを用いた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．対応付けや物体認識の性能向上において，画像から密に局所特徴を抽出することの有効性が指摘されており，特徴抽出に要する計算量は増加傾向にある．本論文は，GPU(Graphics Processing Unit) による局所不変特徴量抽出の高速化を目的とする．特に，多重解像度方向マップを用いた輝度勾配の方向ヒストグラムに基づく記述子の計算方法について検討を行う．多重解像度方向マップを用いる方法では，スケールスペースの全画素におけるヒストグラムへの投票値の計算が，方向マップに対するガウシアンフィルタの適用として実行される．この投票値の計算方法は，GPU による記述子の計算において，2 つの利点をもつことを指摘する．1 つ目は，局所領域の重なり部分でのメモリアクセスの衝突が減少し，並列処理の効率が向上することである．2 つ目は，共有メモリを利用したガウシアンフィルタの実装により，多数の方向マップに対する畳み込み演算を高速に実行可能なことである．実験の結果，多重解像度方向マップの導入が特徴抽出の高速化に寄与することを確認した．

GPU Computing with Orientation Maps for Extracting Local Invariant Features

Naoyuki Ichimura[†1]

Local invariant features have been widely used as fundamental elements for image matching and object recognition. Although dense sampling of local features is useful in achieving an improved performance in image matching and object recognition, it results in increased computational costs for feature extraction. The purpose of this paper is to develop fast computational techniques for extracting local invariant features through the use of a graphics processing unit (GPU). In particular, we consider an algorithm that uses multiresolutional orientation maps to calculate local descriptors consisting of histograms of gradient orientations. By using multiresolutional orientation maps and applying Gaussian filters to them, we can obtain voting values for the histograms for all the pixels in a scale space pyramid. We point out that the method for calculating the voting values has two advantages in GPU computing. First, it improves the efficiency of parallel computing by reducing the number of memory access conflicts in the overlaps among local regions, and secondly it uses a fast implementation of Gaussian filters with shared memory for the many convolution operations required for orientation maps. We conclude with experimental results that demonstrate the usefulness of multiresolutional orientation maps for fast feature extraction.

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 1 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，主として 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．そのため，局所不変特徴量は，画像の対応付けや物体認識の基盤要素として幅広く用いられている[1)–11)]．

対応付けにおける局所領域の設定は，特徴点抽出により行われてきた．特徴点抽出により，対応付けに必要な輝度変化が存在する部分に局所領域を設定することができる．一方，物体認識では，種々の画素位置のサンプリングにより局所領域を密 (dense) に設定する方法が用いられる．そして，サンプリングに基づく方法で局所領域を設定した方が，特徴点抽出よりも高い認識性能が得られることが指摘されている[6),7)]．対応付けにおいても，テクスチャが乏しいシーンにおける性能向上や，隠れへの耐性を向上させるためには，局所領域が多いことは有利になるため，局所領域を密に設定することは望ましい[5)]．よって，対応付け

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

© 2010 Information Processing Society of Japan

Vol.2010-CVIM-171 No.20
2010/3/19

図 1 2 枚の画像，Tour de France[12] と Graffiti[13]，に対する局所領域の設定例．上：原画像．下：局所領域の設定結果．特徴点およびエッジに基づいて，局所領域を設定している[5]．対応付けや物体認識の性能向上のためには，この例のように，画像内に局所領域を密に設定する必要があることが指摘されている[5)–7)]．

と物体認識のどちらにおいても，性能向上のために局所領域を密に設定することは必要であると考えられる．

画像内に局所領域を密に設定することから，特徴抽出に要する計算量は増加傾向にある．この状況に対処するために，本論文では，GPU(Graphics Processing Unit) による局所不変特徴量抽出の高速化を目的とする．特に，輝度勾配の方向ヒストグラムに基づく記述子の計算を，多重解像度方向マップを用いて高速化することを検討する．

記述子の計算を高速化することの必要性は，特徴抽出における計算量を前述の 2 段階の処理に分けて考えると明らかとなる．局所領域の設定では，ガウシアンやラプラシアン，微分等のフィルタにおける畳み込み演算が処理の主体となるため，画像の解像度で計算量はほぼ定まる．一方，記述子の計算では，図 1 の例に示すように，局所領域を密に設定するほどそれらの重なり部分が多くなり，処理対象となる画素数が増加していく．一般に，その処理

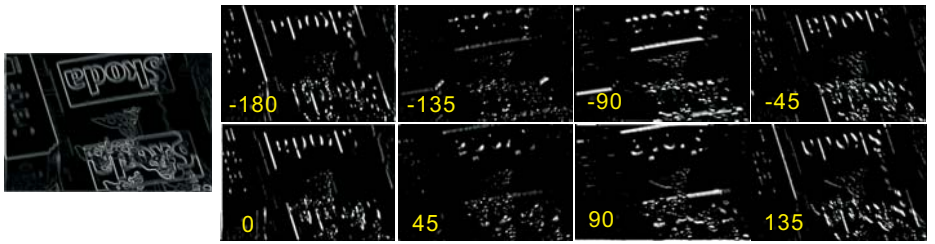

図 2 方向マップの例．この例では，左に示す 1 枚の微分フィルタの処理結果から，右に示す 8 つの輝度勾配の方向に対応する方向マップを生成している．各方向マップは，輝度勾配の大きさを保持している．

対象となる画素数は画像の解像度を大幅に越えるため，記述子に対する計算負荷が重くなるのである．

記述子に対する計算量を削減する一方法として，局所領域の重なり部分を検知し，その部分での処理の重複を省くことが考えられる．しかし，局所領域の位置と大きさはシーンに依存するため，その重なり部分を網羅的に検知する処理は容易ではない．よって，本論文では，あらかじめ全画素に対し記述子の計算に必要な処理を施す方法を用いる．この方法により，局所領域の重なり部分の検知を避け，各局所領域では全画素に対する処理結果の参照を通じて記述子を計算することができる．このような考え方に基づく記述子の計算方法として，Tola ら[10] により提案されたアルゴリズムがある．このアルゴリズムでは，輝度勾配の方向ヒストグラムに基づく記述子を計算するために，微分フィルタの出力結果から，離散的な輝度勾配の方向毎に輝度勾配の大きさを保持する方向マップ (orientation map) を生成する．図 2 に方向マップの例を示す．このような方向マップに対してガウシアンフィルタを適用し，局所的な輝度勾配の大きさの重み付き和を求める．このことは，全画素において，その近傍を用い，輝度勾配の方向ヒストグラムのビンへの投票値を計算することと等価である．そのため，畳み込み後の方向マップにおいて，局所領域内部の全画素でなく一部の画素のみを参照すれば，記述子を求められるようになる．よって，局所領域の重なり部分での計算の重複が削減される．

上記の方法では，方向マップを用いない場合の局所領域の重なり部分における計算量と方向マップに対するガウシアンフィルタの計算量を交換しているため，その 2 つの計算量のバランスに注意する必要がある．本論文で取り扱う局所不変特徴量では，大きなスケール変化に対応可能なスケール不変性を特徴量に付与するため，スケールスペースピラミッドを用いた多重解像度解析を行う．そのため，ダウンサンプリングにより解像度は減少してい

© 2010 Information Processing Society of Japan

くものの，100 枚前後の多重解像度方向マップを生成し，それに対してガウシアンフィルタを適用する必要がある．さらに，対応付けや物体認識では，Tola ら[10] のワイドベースラインステレオマッチングのように記述子を常に全画素から得る必要もない．よって，方向マップを用いずに記述子を求める場合の計算量と比較して，方向マップへの畳み込みの計算量が相対的に少なくなる可能性が低くなる．このことから，多重解像度方向マップを用いた記述子の計算方法が，必ずしも計算時間の短縮に寄与するかは明らかとは言えない．

本論文では，GPU により特徴抽出を実装し，多重解像度方向マップを用いた記述子の計算方法の有効性を示す．従来，GPU を使用して SIFT[14),15)] や SURF[16),17)] 等の局所不変特徴量の抽出が行われているが，方向マップを用いた方法については検討がなされていない．本論文では，方向マップの使用は GPU による記述子の計算において 2 つの利点があることを指摘する．1 つ目は，局所領域の重なり部分でのメモリアクセスの衝突が減少し，並列処理の効率が向上することである．2 つ目は，共有メモリを利用したガウシアンフィルタの実装により，多数の方向マップに対する畳み込み演算を高速に実行可能なことである．CPU および GPU による実装を用いた計算時間の計測を通じ，この利点を検証する．

2. 輝度勾配の方向ヒストグラムに基づく記述子

本節では，処理の高速化の対象とする，輝度勾配の方向ヒストグラムに基づく記述子について説明する．輝度勾配の方向ヒストグラムに基づく記述子には，SIFT[2)]，GLOH[3)], HOG[9)]，DAISY[10)] 等があるが，基本部分には，以下に述べる処理が含まれている．

まず，多重解像度解析を行うために，入力画像からスケールスペースピラミッドを生成する．局所領域の設定に特徴点を使用する場合には，ダウンサンプリングと LoG (Laplacian of Gaussian) フィルタ等を組合せ，スケールスペースピラミッドを生成する[2),18)]．そして，スケールスペースでの極値点を探索し，特徴点の位置と固有スケールを得る．その特徴点の位置を中心とし，固有スケールに比例する大きさを有する局所領域を画像内に設定する．局所領域の設定には，特徴点以外にエッジ等も利用される[5),19),20)]．図 1 が局所領域の設定例である．また，輝度勾配が記述子において必要とされるため，ダウンサンプリングとガウシアンフィルタ，微分フィルタを組合せて別途スケールスペースピラミッドを生成する．図 3 にその例を示す．このようなスケールスペースピラミッドにおいて，局所領域のスケールに対応するエッジ画像を用いて記述子を計算する．

図 4 を用いて，記述子の計算方法を説明する．この図では，視認性を考慮し，3 つの局所領域を最高解像度のエッジ画像上で表示している．記述子の計算では，まず，局所領域をいくつかのセルに分割する．各セル内部の画素における輝度勾配の方向を得て，そのヒストグラムを生成する．この際，輝度勾配の方向に対応するビンには，セル内部の画素のもつ輝度勾配の大きさを投票する．そして，全セルの輝度勾配の方向ヒストグラムを連結したベクトルを構成する．このベクトルに対しノルムの正規化を行ったものが，その局所領域の記述子となる．輝度勾配の方向はエッジの方向と関連があることから，この記述子が表現しているものは，局所領域内部の位置に依存した形状情報と言える．このような形状情報が，画像の対応付けや物体認識において有用であることは，多くの研究で確認されている[1)–11)]．

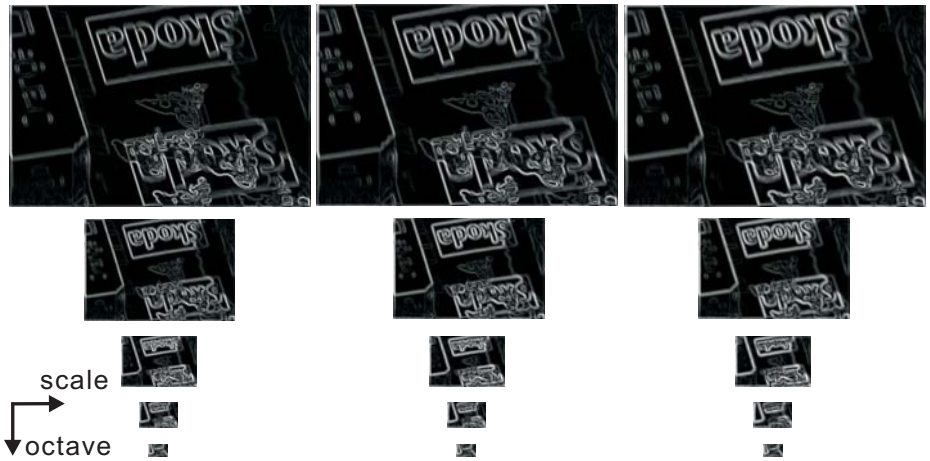

図 3 多重解像度エッジ画像の例．各画像に対し図 2 に示すような方向マップを作成し，多重解像度方向マップを得る．この図の場合，オクターブ数は 5，各オクターブに 3 つのスケールがある．よって，輝度勾配の方向を 8 つに離散化した場合，多重解像度方向マップの数は，5×3×8=120 枚となる．それら全てに対し，ガウシアンフィルタを適用する必要がある．

3. GPU による並列処理の概要

本節では，方向マップの必要性を明らかにするために，特徴抽出の実装に用いる GPU による並列処理の概要を説明する．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

© 2010 Information Processing Society of Japan

Vol.2010-CVIM-171 No.20
2010/3/19

図 4 輝度勾配の方向ヒストグラムに基づく記述子の計算．左：重なり部分をもつ局所領域の例．右：畳み込み後の方向マップにおける記述子の構成のための参照画素の位置．各セルの中心にある赤い丸が，参照画素の位置を示す．各セルの中心のみを参照することから，セル内部の全画素にアクセスする場合に比べ，局所領域の重なり部分でのメモリアクセスの衝突が減少する．

が図られる．

特徴抽出の場合には，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，局所領域毎の処理が並列化される．

CUDA で処理を行う場合，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，共有 (shared)，定数 (constant)，グローバル (global) の 3 種類のメモリを使用する．これらのメモリは，記した順番に，メモリレイテンシは短いが容量が少ないという性質がある．例えば，それぞれのメモリの容量は，16KB,64KB,1GB である．フィルタリングでは，ブロックの大きさを制限することにより，ブロック内部の画素データを全てメモリレイテンシの短い共有メモリに格納できる．フィルタリングにおける近傍領域は重なり部分を有するため，ブロック内の各画素はスレッドにより何度もアクセスされる．よって，共有メモリを通じたデータの再利用 (data reuse) は，グローバルメモリに対するアクセスを避けることによる処理の高速化に非常に有効である．さらに，近傍領域の重なり部分におけるメモリアクセスの衝突によるレイテンシを軽減することもできる．その一方，記述子の計算での共有メモリによるデータの再利用は容易ではない．何故なら，局所領域の位置と大きさはシーンに依存するため，それらの重なり部分はフィルタリングにおける近傍領域の重なり部分のように一様ではないからである．局所領域の重なり部分のデータを見出し，再利用のために容量の小さい共有メモリに格納することは困難である．よって，記述子の計算にはグローバルメモリを使用する．このため，局所領域の重なり部分において，メモリアクセスの待ちによるレイテンシが長くなる．CUDA による並列処理では，数千から数万のスレッドを同時に立ち上げることによりメモリレイテンシの隠蔽を行うが，メモリアクセスの待ちによるレイテンシが多発すると隠蔽効果が薄れる．結果として，並列処理の効率が悪化する．次節において，方向マップの使用が，この並列処理の効率悪化を防ぐ 1 つの方法であることを述べる．

4. 多重解像度方向マップを用いた記述子の計算

スケールスペースピラミッドの全画素での輝度勾配の方向と大きさは，微分フィルタを適用した時点で計算できる．よって，多重解像度エッジ画像が生成された時点で，各画素の輝度勾配の方向がヒストグラムのどのビンに投票されるかが定まる．このことから，各エッジ画像に対して，ヒストグラムのビンに対応する複数の 2 次元配列を用意し，離散化された輝度勾配の方向毎に輝度勾配の大きさを保持することができる．このヒストグラムのビン対応する 2 次元配列を方向マップと呼ぶ[10]．図 2 は，その方向マップの例である．

方向マップに対してガウシアンフィルタを適用することは，フィルタのスケールに応じた範囲で，輝度勾配の方向の発生頻度を輝度勾配の大きさとガウス関数により定まる重み付きで求めることに相当する．よって，フィルタのスケールを記述子を求める際に用いるセルの大きさに相当するように設定すれば，フィルタリングはセル内の画素によるヒストグラムへの投票値の計算と等価になる．よって，畳み込み済みの方向マップが得られれば，図 4 の右の図に示すように，そのマップにおいて各セルの中心位置での値を参照するだけで，輝度勾配の方向ヒストグラムが得られる．つまり，従来の方法のように，セルの全画素を参照する必要がなくなる．このことは，並列処理の観点から見ると，局所領域の重なり部分でのメモリアクセスが粗となり，メモリアクセスの衝突が減少することを意味する．一般に，並列処理中におけるメモリアクセスはユーザーが制御できるものではない．よって，このようにメモリアクセスの衝突が少なくなるようにアルゴリズムを構成することが，並列処理の効率向上に繋がる．

局所不変特徴量を抽出する場合には，図 3 に示すような多重解像度エッジ画像から方向マップを生成する．生成された方向マップを多重解像度方向マップと呼ぶ．多重解像度方向マップの数は，スケールスペースピラミッドのオクターブ数，各オクターブのスケール数，および，輝度勾配の方向ヒストグラムのビン数により定まる．図 3 の場合，オクターブ数が 5，スケール数が 3 であるので，ビン数を図 2 のように 8 とすると，多重解像度方向マッ

© 2010 Information Processing Society of Japan

Vol.2010-CVIM-171 No.20
2010/3/19

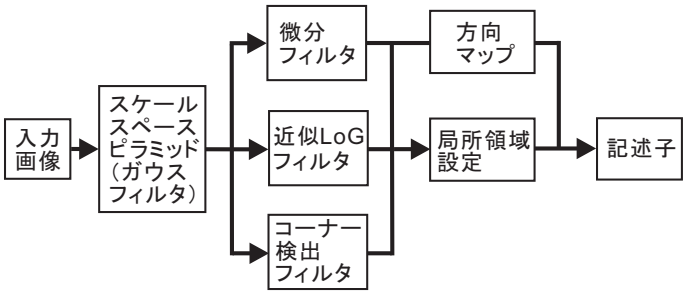

図 5 局所不変特徴量抽出処理のフローチャート．密なエッジサンプリングと特徴点抽出に基づき局所領域を設定する[5]．記述子は，多重解像度方向マップを用いて計算される．

プの数は 120 枚となる．ダウンサンプリングにより解像度は減少していくものの，多数の方向マップに対してガウシアンフィルタを適用する計算量は少ないものとは言えない．よって，このガウシアンフィルタの計算が方向マップを用いない場合の記述子の計算よりも効率良く実行できるかどうかが，方向マップの導入により特徴抽出の計算時間の短縮を図る際に重要な問題となる．

3 節で述べたように，GPU ではメモリレイテンシの短い共有メモリを使用して，効率良くガウシアンフィルタを方向マップに適用することができる．よって，多重解像度方向マップと GPU の導入は，記述子の計算時間の短縮に有効であると考えられる．次節では，多重解像度方向マップの生成を含む特徴抽出の GPU による実装を示す．

5. GPU による実装

本節では CUDA[21] を用いた特徴抽出の実装について述べる．CUDA では，処理対象となるデータをいくつかのブロックに分け，それぞれのブロックに複数のスレッドを付随させる．ブロック数とスレッド数は execution configuration(以下，EC と記す) と呼ばれる．ブロック数は，ブロックの幅や高さにより間接的に指定する場合が多い．

図 5 は，実装を行った処理のフローチャートである．以下に，この図にはない補助的な処理も含めその概要を示す．GPU のメモリの使用方法等の詳細に関しては，文献[5] も合わせて参照されたい．

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．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の生成に必要な画像のダウンサンプリング．EC は **Y Component** と同じ．

Gaussian filter: ガウシアンフィルタを用い，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．等方ガウス関数の変数分離性を利用．行処理における 1 次元のブロックサイズは 128 とし，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，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．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ためである．

Gradient filter: 5×5 の微分フィルタ[22] による輝度勾配の計算．ブロックサイズは 16×16 とし，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[23),24)] 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．EC 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．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．

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[5]．微分フィルタの応答のしきい値は 10．サンプリングし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．EC は **Feature point** と同じ．

Orientation map: 多重解像度方向マップの生成．微分フィルタの出力から，図 2 に示すように輝度勾配の方向を 8 つに離散化した方向マップを生成．EC は **Feature point** と同じ．ヒストグラムの境界効果 (boundary effect) を防ぐために，1 つの輝度勾配の方向に対し，ビンの中心値からの距離に基づいて重み付けを行い，2 つの方向マップへ輝度勾配の大きさを割り振る (付録 A.1 参照)．多重解像度方向マップへ適用するガウシアンフィルタのスケールは，**Gaussian filter** でのスケールの約 1.2 倍．これが図 4 でのセルの大きさに相当する (付録 A.2 参照)．フィルタリングの処理は，**Gaussian filter** と同じ．

Dominant orientation: 回転不変性を特徴量に付与するための dominant orientation[2)]

© 2010 Information Processing Society of Japan

の計算．局所領域の大きさは，局所領域のスケールの 5 倍．16 個の局所領域をブロックとし，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．Dominant orientation を求めるためのヒストグラムの生成には，記述子の計算と同様に多重解像度方向マップを利用した．

Descriptor: 記述子の計算．局所領域の大きさは，局所領域のスケールの 20 倍．EC は **Dominant orientation** と同じ．図 4 に示すように，局所領域を 4×4 のセルに分割．多重解像度方向マップを用いた処理では，各セルの中心位置でのみ畳み込み後の方向マップの値を参照し，輝度勾配の方向ヒストグラムを生成．方向マップを用いない場合には，各セル内部の全画素を参照．回転不変性付与のため，dominant orientation に基づきヒストグラムを並行移動 (付録 A.3 参照)．全セルのヒストグラムを連結後，ノルムの正規化を行い記述子を得る．

6. 実験結果

本節では，CPU および GPU による実装を用いた計算時間の計測結果を示す．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 Quadro で行った．演算専用とした GeForce は 240 個のコアを有し，warp size は 32，Compute Capability[25] は 1.3，1 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 および 320×240 とし，それぞれ，スケールスペースピラミッドのオクターブ数は 5 および 4 とした．また，各オクターブにおけるスケール数は 3 とした．輝度勾配の方向は 8 方向で離散化したので，120 枚もしくは 96 枚の多重解像度方向マップが生成された．

計算時間の計測結果を表 1 に示す．表 1 の CPU–C と GPU–C は，CPU と GPU による多重解像度方向マップを用いない実装を示す．また，GPU は，GPU による多重解像度方向マップ用いた実装を表す．“Orientation map”，“Dominant orientation” および “Descriptor” において GPU–C と GPU を比較することにより，多重解像度方向マップが記述子の計算時間の短縮に有用であることを確認した．また，多重解像度方向マップと GPU の導入により，対 CPU 比で 31〜34 倍程度の特徴抽出の高速化が達成できることも明らかとなった．

局所領域の大きさは，それを局所領域のスケールの何倍にするかの係数 (以下，スケール係数) によって定まる．表 1 の実験では，スケール係数は 20 であった．この係数を変化させると，局所領域の重なり部分の面積も変化する．このことは，並列処理の観点からは，スケール係数により各局所領域での計算の独立性が変化し，それに伴い並列処理の効率が変化することを意味する．図 6 の左の図は，画像 “Tour de France” におけるスケール係数に対する CPU と GPU の計算時間の比の変化を表す．実線と破線は，それぞれ，方向マップを用いる場合と用いない場合の計算時間の比を表す．アルゴリズムの計算量は CPU と GPU どちらでも同じであることから，この計算時間の比は並列処理の効率を反映している．方向マップを用いない場合には，計算時間の比はスケール係数の増加と共に減少する．つまり，局所領域の重なり部分の増加に伴い，メモリアクセスの衝突も増加し，並列処理の効率は悪化すると言える．それに対し方向マップを用いる場合には，スケール係数の変化に対する並列処理の効率の低下は見られず，逐次処理に対する優位性がより大きくなることがわかる．この優位性の要因は，局所領域の重なり部分でのメモリアクセスの衝突が減少すること，および，ガウシアンフィルタの高速な実装の効果であると考えられる．図 6 の右の図は，畳み込み済みの方向マップの生成に要する CPU と GPU の計算時間の比の変化を表す．スケール係数が大きくなるとセルの面積も増加するため，より大きなスケールを有するガウシア

表 1 局所不変特徴量抽出に必要な計算時間．単位は [ms]．OS(Linux) 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．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．

Image	Tour de France			Graffiti		
Image size	720×480			320×240		
Task/Implement	CPU–C	GPU–C	GPU	CPU–C	GPU–C	GPU
Image transfer	N/A	2.563	2.597	N/A	0.600	0.600
Y component	3.922	0.108	0.107	0.934	0.062	0.061
Down sampling	0.854	0.140	0.140	0.207	0.083	0.082
Gaussian filter	263.740	8.766	8.716	56.435	4.854	4.830
ALoG-C filter	308.480	13.639	13.639	63.758	3.812	3.817
Gradient filter	219.952	4.394	4.405	45.636	1.358	1.360
Feature point	158.199	7.538	7.546	34.718	1.957	1.948
Edge sampling	11.179	7.312	7.335	3.541	1.833	1.836
Orientation map	N/A	N/A	45.076	N/A	N/A	24.079
Dominant orientation	196.260	5.977	3.548	119.797	3.145	1.392
Descriptor	2091.589	122.483	10.772	954.991	76.171	5.119
Total	3255.619	176.641	107.536	1280.372	96.381	47.619
#feature	5731			2706		

© 2010 Information Processing Society of Japan

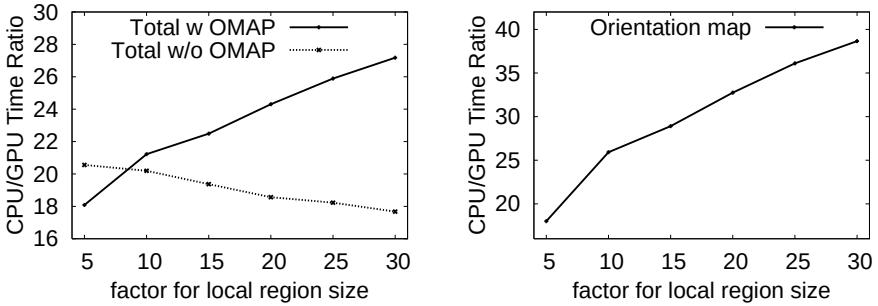

図 6 画像 Tour de France における局所領域の大きさの計算時間への影響．横軸の係数を局所領域のスケールに乗じて，局所領域の大きさを決定する．縦軸は CPU と GPU の計算時間の比を表す．左：総計算時間の比．“OMAP” は方向マップを意味する．右：畳み込み済みの方向マップの生成に要する計算時間の比．

ンフィルタを方向マップに適用する必要がある．よって，畳み込み演算の回数も多くなる．しかし，スケール係数の増加に伴う並列処理の効率の低下は生じていない．これは，共有メモリによるデータの再利用を用いた並列実装が，逐次処理と比較して非常に効率的であることを表している．以上から，局所領域の重なり部分でのメモリアクセスの衝突の減少，および，方向マップに対するガウシアンフィルタが高速に実行できること，これらが GPU による多重解像度方向マップを用いた記述子の計算方法の利点であると言える．

7. ま と め

本論文では，多重解像度方向マップを用いた局所不変特徴量抽出の高速化について検討を行った．輝度勾配の方向ヒストグラムの作成に方向マップを用いることにより，局所領域の重なり部分でのメモリアクセスの衝突を減少させることができ，GPU による並列処理の効率化が図られる．一方で，多数の多重解像度方向マップに対するガウシアンフィルタの計算量が問題となるが，フィルタリングは GPU により効率良く実行可能なため，記述子の計算の高速化が可能なことを確認した．

多重解像度方向マップの導入の効果を確認するために，計算時間の計測を行ったが，計算量を用いれば実装に依存しない比較が可能となる．それにもかかわらず計算時間を用いた理由は，方向マップを用いない場合の記述子の計算量がシーン毎に変化する局所領域の大きさに依存するため，見積もることが困難なためであった．しかし，今回実装した多重解像度方向マップを用いた方法では，局所領域内での画素の参照回数が固定されるため，計算量を見積もることは比較的容易になっている．このことは，アルゴリズムの理論的な比較において有用であると考えている．

今後は，特徴抽出と共に，対応付けや物体認識の高速化に関しても研究開発を進める予定である．

参 考 文 献

1) C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. *IEEE Trans. PAMI*, Vol.19, No.5, pp. 530–535, 1997.
2) D.Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comp. Vis.*, Vol.60, No.2, pp. 91–110, 2004.
3) K.Mikolajczyk and C.Schmid. A performance evaluation of local descriptors. *IEEE Trans. PAMI*, Vol.27, No.10, pp. 1615–1630, 2005.
4) K.Mikolajczyk, T.Tuytelaars, C.Schmid, A.Zisserman, 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.
5) 市村直幸. 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. 情処研報, No. 2009–CG–136, 2009.
6) 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.
7) 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.
8) 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.
9) N.Dalal and B.Triggs. Histograms of orientated gradients for human detection. In *Proc. Int. Conf. Comp. Vis. Patt. Recog.*, Vol.1, pp. 886–893, 2005.
10) E.Tola, V.Lepetit, and P.Fua. A fast local descriptor for dense matching. In *Proc. Int. Conf. Comp. Vis. Patt. Recog.*, 2008.
11) C.H. Lampert, M.B. Blaschko, and T.Hofmann. Efficient subwindow search: A branch and bound framework for object localization. *IEEE Trans. PAMI*, Vol.31, No.12, pp. 2129–2142, 2009.
12) 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．
13) Feature detector evaluation sequences. http://lear.inrialpes.fr/people/mikolajczyk/Database/.
14) 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.

© 2010 Information Processing Society of Japan

Vol.2010-CVIM-171 No.20
2010/3/19

317–322, 2007.
15) SiftGPU. http://www.cs.unc.edu/~ccwu/siftgpu/.
16) 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.
17) 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.
18) T.Lindeberg. Feature detection with automatic scale selection. *Int. J. Comp. Vis.*, Vol.30, No.2, pp. 79–116, 1998.
19) K.Mikolajczyk, A.Zisserman, and C.Schmid. Shape recognition with edge-based features. In *Proc. Britich Machine Vis. Conf.*, Vol.2, pp. 779–788, 2003.
20) X.Ma and W.E. Grimson. Edge-based rich representation for vehicle classification. In *Proc. Int. Conf. Comp. Vis.*, Vol.2, pp. 1185–1192, 2005.
21) CUDA Zone. http://www.nvidia.co.jp/object/cuda_home_jp.html.
22) S. Ando. Consistent gradient operators. *IEEE Trans. PAMI*, Vol. 22, No. 3, pp. 252–265, 2000.
23) M.Trajkovic and M.Hedley. Fast corner detection. *Image and Vision Computing*, Vol.16, pp. 75–87, 1998.
24) 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.
25) *NVIDIA CUDA Programming Guide, Version 2.0, Sec.5.1.2*, 2008.

付　　録

A.1　ヒストグラムの境界効果を考慮した方向マップの生成

図 7 を用いて，輝度勾配の方向 θ と大きさ g を持つ画素から，ヒストグラムへ投票を行う操作を説明する．この図で θ は，中央値が τ，幅が d であるビンに属する．このビンに対し g を加算することにより，ヒストグラムは生成される．しかし，輝度勾配の方向を離散化してヒストグラムを生成するため，θ がビンの境界に近いほど，シーンの変動の影響により属するビンが変化しやすくなる．このようなヒストグラムの境界効果は，視点位置や照明条件等の変動に対する特徴量の不変性を悪化させる一要因となる．よって，輝度勾配の方向の変動によるヒストグラムの分布の急激な変化を抑制するために，以下のような重み付き投票を行う．まず，τ を中央値とするビンには，θ との距離により得られる重み

$$w_1 = 1 - |\theta - \tau| / d \tag{1}$$

を乗じて g を投票する．そして，$\theta - \tau$ の符号の正の場合には右のビン，負の場合には左のビンに，$1 - w_1$ の重みを乗じて g を投票する．微分フィルタの出力の全画素に対し，この重み付き投票を行うことによって方向マップを生成する．

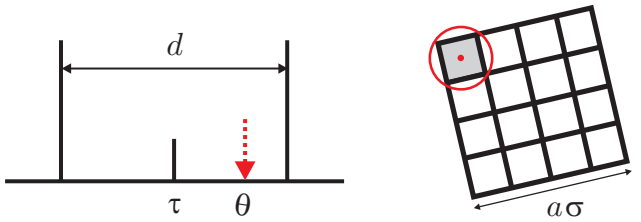

図 7　方向マップの生成方法，および，方向マップへ適用するガウシアンフィルタのスケールの決定方法．左：方向マップ生成時の離散的な輝度勾配の方向への重み付け．ヒストグラムにおける境界効果を防ぐため，輝度勾配の方向 θ とビンの中央値 τ の間の距離に基づいて，輝度勾配の大きさを 2 つの方向マップに割り振る．右：方向マップに適用するガウシアンフィルタのスケールの決定方法．セルの外接円の位置におけるガウス関数の値を制御し，スケールを決定する．

A.2　方向マップに適用するガウシアンフィルタのスケールの決定方法

図 4 に示す 4×4 のセルに局所領域を分割する場合について，方向マップに適用するガウシアンフィルタのスケールの決定方法を示す．局所領域のスケールを σ とする．このスケールに比例定数 a を乗じて，局所領域の大きさを決定する．局所領域を 4×4 のセルに分割することから，1 つのセルの幅は $a\sigma/4$ となる．

方向マップに対するガウシアンフィルタのスケールを σ_o とする．正方形のセルに対する外接円の位置で，ガウス関数がどの程度の値を保持するかを変化させ，セル内部の輝度勾配の大きさに対する重み付けを制御する．本論文では，外接円の位置でのガウス関数の定義域の値を $3\sigma_o$ とした．よって，外接円の直径は $6\sigma_o$ となる．この直径がセルの幅の $\sqrt{2}$ 倍と一致することから，σ_o は次式で求められる．

$$\sigma_o = \sqrt{2}a\sigma/24 \tag{2}$$

表 1 の計算時間の計測では $a = 20$ とし，$\sigma_o \approx 1.2\sigma$ より定まるスケールを有するガウシアンフィルタを多重解像度方向マップに適用した．

セルの形状が正方形以外の場合にも，同様の方法で方向マップに適用するガウシアンフィルタのスケールを決定することができる．

A.3　ヒストグラムの並行移動

特徴量に回転不変性を付与するために，符号を反転した dominant orientation 分だけヒストグラムを並行移動させる．この処理において，並行移動後のヒストグラムと bin の重なりに応じて重みを決定し，付録 A.1 と同様の重み付き投票を行う．

© 2010 Information Processing Society of Japan