局所特徴量抽出アルゴリズムの ハードウェアコストと精度のトレードオフ解析

水野 孝祐† 寺地 陽祐† 黒田 光彦† 川口 博† 吉本 雅彦†

†神戸大学大学院システム情報学研究科 〒657-8501 神戸市灘区六甲台町 1-1

E-mail: mi-no@cs28.cs.kobe-u.ac.jp

あらまし 近年、画像を認識するためのさまざまな局所特徴量抽出アルゴリズムが提案されており，中でも SIFT (Scale-Invariant Feature Transform)がよく用いられている．しかし SIFT 特徴量抽出には膨大な演算量とメモリ帯域が必要なため、一般的な CPU を用いてリアルタイムで処理することは難しく，ハードウェア化が必要不可欠である．そこで本稿では SIFT 特徴量抽出をハードウェア化する際に検討する必要があるコストと精度のトレードオフについて解析し，低コストかつ高精度を実現するハードウェア構成を示す．

キーワード SIFT，特徴量抽出，ハードウェア実装

Trade-Off Analysis of Local Descriptors between Hardware Cost and Accuracy

Kosuke MIZUNO†, Yosuke TERACHI†, Mitsuhiko KURODA†, Hiroshi KAWAGUCHI† and Masahiko YOSHIMOTO†

† Graduate School of System Informatics, Kobe University 1-1 Rokkodai-cho, Nada-ku, Kobe, 657-8501 Japan

E-mail: mi-no@cs28.cs.kobe-u.ac.jp

Abstract Recently, various feature descriptors for image recognition have been proposed. Especially, SIFT (Scale-Invariant Feature Transform) is major algorithm in wide-range applications. However, it is difficult for general-purpose CPU to process SIFT algorithm in real-time because SIFT algorithm requires massive processor power and memory band-width. Hardware-implementation approach is essential for real-time operation. Therefore this paper describes trade-off analysis of SIFT algorithm between hardware cost and accuracy. Then, low-cost and high-accuracy hardware configuration is introduced.

Keyword SIFT, feature extraction, hardware implementation

1. はじめに

近年，画像を認識するためのさまざまな特徴点・特徴領域検出手法や特徴量記述子が提案されている．中でも SIFT(Scale Invariant Feature Transform)[1] は検出した特徴点に対して画像の回転・スケール変化・輝度の変化などに頑健であり、画像間の対応点マッチングを高精度に行える[2] ことからイメージモザイキングや特定物体の認識などさまざまなアプリケーションに用いられている．また，カメラ付き携帯電話などのモバイル端末を用いてリアルタイムで物体を認識したいという要求が高まってきている．SIFT 特徴量をリアルタイムで抽出できるようになれば，自立移動ロボットの視覚として用いたり，自動車にカメラを載せて走りながら都市の 3 次元マップを作成したりするなどさまざまな利用法が考えられる．

しかし SIFT 特徴量抽出処理には膨大な演算量とメモリアクセス帯域が必要なため，一般的な CPU を用いてリアルタイムで処理することは現在のところ難しい．そのため SIFT を実時間処理するためにはハードウェア化が必要不可欠となる．

上記の課題を解決するためにこれまでに SIFT 特徴量抽出の GPU による実装[6]，FPGA 実装[7][8]，LSI 実装[9][10]などが提案されている．カメラの高解像度化に伴い，さらに高速な処理が要求されるためハードウェア化の重要性がますます高くなっている．

そこで本稿ではバッテリーが制限されたモバイル端末でも利用可能な低消費電力かつ高精度にリアルタイムで SIFT 特徴量抽出処理を実行できるハードウェア構成について検討する．

本稿の構成を以下に示す．第 2 章では SIFT 特徴量抽出処理の処理内容について説明し，必要演算量・メモリ帯域について述べる．第 3 章では SIFT 特徴量抽出処理を実現するためのハードウェア構成を示し，検討する必

要がある種々のパラメータについて説明する．第 4 章ではハードウェア実装において SIFT アルゴリズムやハードウェアに関するパラメータがハードウェアリソース量と特徴量のマッチング精度にどのような影響を与えるかを解析し，さまざまな条件下における比較を示す．第 5 章で本稿をまとめる．

2. SIFT 特徴量

本章では SIFT 特徴量抽出処理の概要を説明する．

2.1. 処理フロー

まず、SIFT 特徴量抽出処理の全体の流れを示す．

1. 初期ガウシアン平滑化

   入力画像 $I(u,v)$ に対するガウシアン平滑化は以下の式により定義される．

$$L(u,v,\sigma) = G(x,y,\sigma) * I(u,v) \tag{1}$$

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^{2}} \exp\left(-\frac{x^{2}+y^{2}}{2\sigma^{2}}\right) \tag{2}$$

   σ はガウシアンフィルタのスケール，x と y は注目ピクセルからの距離である．まず入力画像に対して $\sigma = 0.5$ でガウシアンフィルタを適用し，初期ガウシアン平滑化画像を得る．この前処理によりノイズの影響を低減し，精度を高めることができる．

2. ガウシアン平滑化

   初期ガウシアン平滑化画像に対して，σ を連続的に変化させながらガウシアンフィルタを適用し，複数のスケールのガウシアン平滑化画像から成るガウシアンピラミッドを構築する．

3. 特徴点抽出

   隣り合うガウシアン平滑化画像の差分を計算し DoG (Difference of Gaussian)ピラミッドを構築する．DoG ピラミッドから極値を探索し，特徴点候補の位置とスケールを得る．極値をサブピクセル補間してサブピクセル精度の位置とスケールを決定し，いくつかの閾値処理を通過したものを特徴点として出力する．

4. 特徴量記述

   まず特徴点の周辺領域の勾配情報を基に向き検出を行う．その向きに回転させた特徴量記述領域から 128 次元の特徴ベクトルを計算する．

2.2. ガウシアン平滑化

図 1 に示すように，初期ガウシアン画像に対して σ を σ_0 から一定の割合 k で連続的に増加させながらガウシアンフィルタを適用し，複数スケールの平滑化画像からなるガウシアンピラミッドを構築する．そしてガウシアン画像の差分をとり DoG ピラミッドを構築する．しかし，σ が増加し続けるとガウシアンフィルタのウィンドウサイズが大きくなり，演算コストも増大し続けてしまう．そこで SIFT では画像をダウンサンプルすることで σ の連続性を保ちながらの平滑化処理を実現している．

ダウンサンプルまでの処理 1 セットを 1 オクターブとする．1 オクターブでは σ は σ_0 から $2\sigma_0$ まで増加する．1 オクターブの分割数を s とすると，σ の増加率 k は $k = 2^{(1/s)}$となる．特徴点探索対象の画像を s 枚得るためには s + 2 枚の差分画像が必要となり，ガウシアン画像は s + 3 枚必要となる．[1] では s = 3 のとき最適な特徴点を得ることができるとしている．

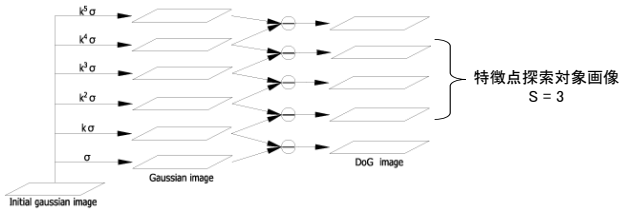

図 1 ガウシアンピラミッドと DoG ピラミッド

2.3. 特徴点抽出

まず DoG ピラミッド中の画像から極値を探索する．図 2 に示すように，注目画素（赤）を周囲の 8 画素と隣接するスケールの同じ位置を中心とする 9 画素ずつの合計 26 画素（緑）と比較し，極値かどうかを調べる．極値であればサブピクセル補間，コントラストの低い点とエッジ上の点の除去がおこなわれる．

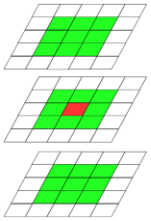

図 2 3x3x3 極値検出

2.4. 特徴量記述

まず図 3 に示すように，検出された特徴点の周辺領域から 36 方向の輝度勾配ヒストグラムを作成する．そしてヒストグラムの最大値の 0.8 倍以上の極大値を特徴点の向きとする．特徴点に複数の向きがある場合，特徴点の位置とスケールが同じで向きだけが違う別の特徴点として扱われる．

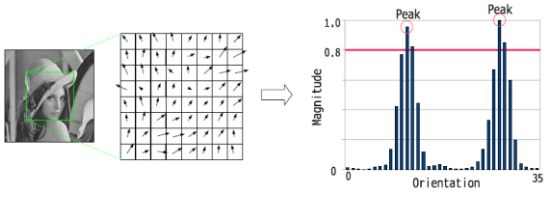

図 3 向き検出

次に図 4 に示すように，特徴量の記述領域を特徴点の向きに合わせて回転させる．これにより特徴量は回転に対して不変となる．領域を縦 4 ブロック，横 4 ブロックの 16 ブロックに区切り，ブロックごとに 8 方向の輝度勾配ヒストグラムを作成する．そして 16 ブロックのヒストグラムを繋ぎあわせ 128 要素の特徴ベクトルを作成する．最後に，特徴ベクトルの長さを正規化する．これにより特徴量は輝度の変化に対して不変となる．

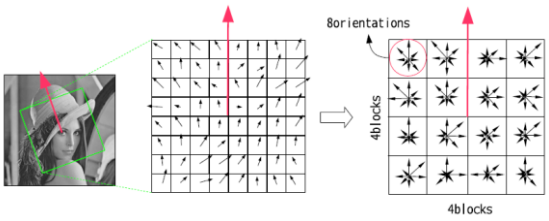

図 4 特徴量記述

2.5. 演算量・メモリ帯域の解析

図 5 の左に SIFT 特徴量抽出処理のうち主要なものの演算量を示す．ガウシアン処理が総演算量の大部分を占めている．また，画像の解像度が高くなり検出される特徴点の数が多くなってくると特徴量記述処理の演算量も無視できないレベルになってくる．特徴点探索処理は他の処理と比べると演算量は小さいが，絶対値をみると決して小さくはない．

図 5 の右に SIFT 特徴量抽出処理のうちの主要ブロックが必要とするメモリ帯域を示す．演算量と同様に，ガウシアン処理が総メモリ帯域量の大部分を占めており，特徴点探索処理がそれに続く．

以上のように，SIFT 特徴量抽出処理は演算量・メモリ帯域量ともにとても要求が高く，汎用的な CPU を用いてリアルタイムで SIFT 特徴量を抽出するのは困難である．

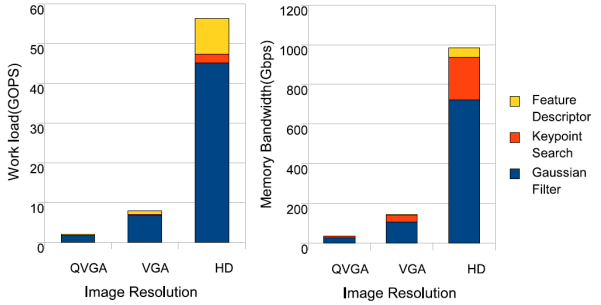

図 5 演算量解析

3. ハードウェアアーキテクチャ

本章ではハードウェアリソースを見積もるために想定するハードウェアアーキテクチャについて説明する．さらにハードウェア実装時に検討するべき種々のパラメータについて紹介する．

3.1. 全体ブロック図とパイプライン処理

図 6 に全体ブロック図を示す．また，その処理内容を以下で述べる．

1. まず外部メモリから入力画像用 SRAM に入力画像を読み込む．
2. 次に入力画像に対して初期ガウシアン処理を行い，初期ガウシアン SRAM に初期ガウシアン画像を格納する．
3. 次に初期ガウシアン画像にガウシアン処理を行い，ガウシアン SRAM にガウシアン画像を格納する．同時に，初期ガウシアン画像を縦横 1/2 ずつダウンサンプルした画像を外部メモリに書き出しておく．これは 2 オクターブ目以降の入力画像として使用される．
4. 次に特徴点探索処理を行い，結果を SRAM に格納する．
5. 次に特徴量記述処理を行い，結果を SIFT 特徴量 SRAM に格納する．
6. 最後に抽出した SIFT 特徴量を外部メモリに書きだす．

処理のスループットを上げるため，以上をパイプライン処理にて行う．図 7 に 1 オクターブ目の，図 8 に 2 オクターブ目以降のパイプラインの様子を示す．

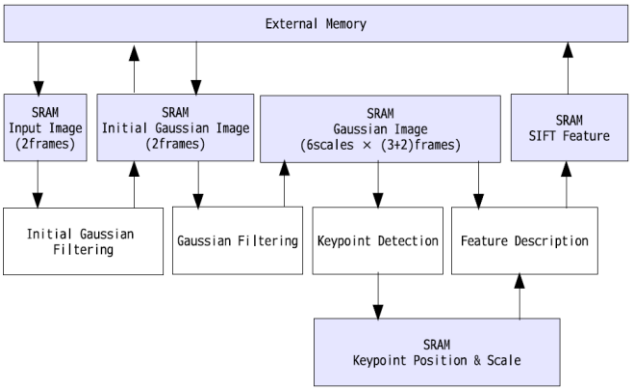

図 6 全体ブロック図

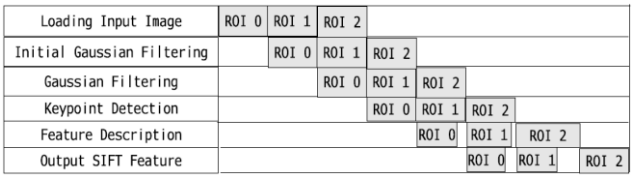

図 7 パイプライン処理（1 オクターブ目）

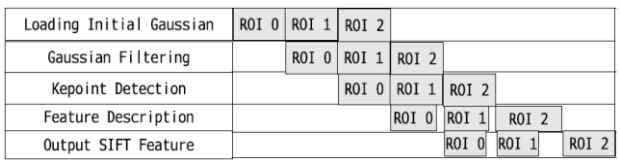

図 8 パイプライン処理（2 オクターブ目以降）

1 オクターブ目は入力画像に対して初期ガウシアン平滑化処理を行ってからガウシアン平滑化処理へと移行するが、2 オクターブ目以降は初期ガウシアン平滑化画像を外部メモリから読み込みガウシアン平滑化処理へ移行する。パイプラインステージのうち、特徴量記述処理以降のステージは検出された特徴点の数によって所要サイクル数が大きく変動する。

上記のパイプライン処理は画像から切り出した ROI (Region of Interest)単位で処理をおこなう．これはワーキングメモリの容量を削減するためである．しかし ROI に分割することにより ROI の端を処理する際に特徴点の精度劣化が生じる．精度劣化を少しでも防ぐために ROI 外の画素も同時に外部から読み込む必要がある．この ROI 外の領域を以後，拡張 ROI 領域と記述する．

3.2. ガウシアン平滑化部

図 9 にガウシアン平滑化部のブロック図を示す．画素値を格納するレジスタと乗算器をガウシアンフィルタのウィンドウサイズの分だけ用意してある．右端以外のレジスタは右隣のレジスタから，右端のレジスタは初期ガウシアン画像 SRAM またはガウシアン画像 SRAM から画素値を読み込む，シストリックアレイ構成となっている．そしてレジスタ内の画素値をガウシアン係数と乗算し，全て足し合わせることで 1 画素分の 1 次元ガウシアン平滑化画素が計算され、結果はガウシアン画像 SRAM へ格納される．1 サイクルで 1 画素分の処理が行え、シストリックアレイ構成としたことで画像 SRAM からの読み込み回数が(ガウシアンフィルタのウィンドウサイズ)分の 1 に削減される．ガウシアン平滑化パラメータ σ_0 の値にもよるが、メモリアクセス数は 6 スケール平均では 20～25 分の 1 程度に削減される．

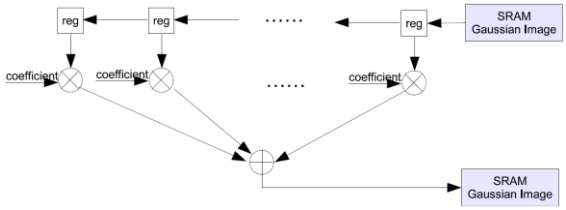

図 9 ガウシアン平滑化ブロック図

3.3. 特徴点抽出部

図 10 に 1 スケール分の特徴点抽出処理部のブロック図を示す．9 個のレジスタが用意され，ガウシアン画像の差分画素が格納される．ガウシアン処理部と同様にシストリックアレイ構成となっており，右端以外のレジスタは右隣のレジスタから，右端のレジスタは画像 SRAM から画素値を読み込む．そして両隣のスケールを同期して処理するブロックの 9 画素を含む 27 画素中で注目画素が極値かどうかを判定する．極値であった場合は特徴てをテストされ，これに通過した点を特徴点として採用し，その位置とスケールを SRAM へ格納する．

1 サイクルで 1 画素の処理が可能となっている．レジスタを 1 行（3 画素分）追加し，演算ブロックを 1 つ追加することで，並列度を高め 1 サイクルで処理できる画素を 1 画素増やすことができる．

シストリックアレイ構成と全スケールを同期して処理する方法により，メモリアクセス回数は 5/27（並列度 1 の場合），10/81（並列度 2 の場合）に削減される．

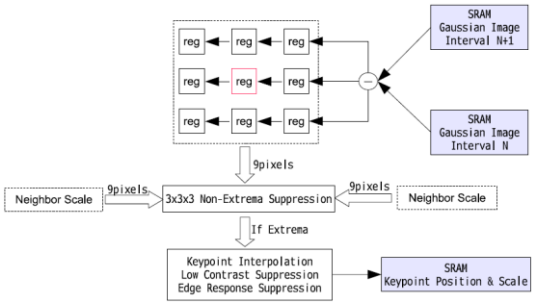

図 10 特徴点抽出ブロック図

3.4. 特徴量記述部

特徴量記述処理部は向き検出部と特徴量記述部の 2 つから構成されており，パイプライン処理で並列動作される．図 11 に向き検出部のブロック図を示す．ガウシアン画素を格納するレジスタが用意されている．右端以外のレジスタは右隣のレジスタから，右端のレジスタは画像 SRAM から画素を読み込む構成となっている．

そして輝度勾配オリエンテーション計算ブロックで処理対象画素の上下左右の画素を用いて輝度勾配ベクトルを計算，オリエンテーションヒストグラムを作成し特徴点の向きを検出する．

図 12 に特徴量記述部のブロック図を示す．基本的には向き検出部と同じで、輝度勾配ベクトルは記述領域を縦 4 ブロック・横 4 ブロックからなる 16 ブロックに区切られた中の処理対象画素の位置に対応するブロックのヒストグラムの作成に用いられる．

向き検出部と特徴量記述部は 1 サイクルで 1 画素分の処理を行えるが，特徴点探索処理部と同様にレジスタを 1 行分・演算ブロックを 1 つ追加することで並列度を高め，1 サイクルで処理できる画素を 1 画素増やすことができる．上記構成を採用することでメモリアクセス回数は 1/3（並列度 1 の場合），2/9（並列度 2 の場合）、5/27（並列度 3 の場合）に削減される．

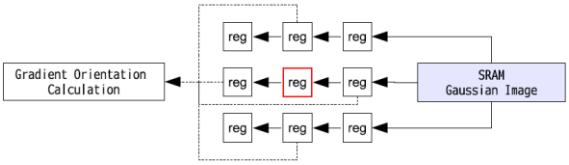

図 11 向き検出ブロック図

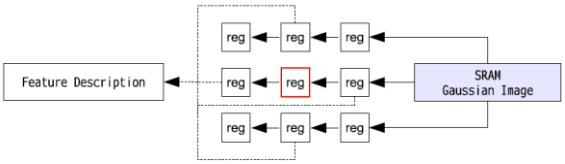

図 12 特徴量記述ブロック図

3.5. 検討すべきパラメータ

前節で述べたアーキテクチャを実装するために以下のパラメータについてシステムの制約（回路面積，消費電力，速度，精度）を満たすように慎重に検討する必要がある．

1. 1 画素に割り当てる bit 長
   1 画素を何ビットで表現するのかを決定するパラメータである．ハードウェア全体のリソース（演算器の規模，メモリ容量，メモリ帯域）と特徴点の精度に影響する．
2. ROI のサイズと拡張 ROI 領域
   画像から切り出すブロックのサイズを決定するパラメータである．ハードウェア全体のリソース（演算器の規模，メモリ容量，メモリ帯域）と特徴点の精度に影響する．
3. ガウシアン平滑化パラメータ σ_0
   ガウシアン平滑化のウィンドウサイズを決定するパラメータである．ガウシアン平滑化部の回路規模と特徴点の精度に影響する．
4. 演算モジュールの並列度
   各演算モジュール（ガウシアン平滑化，特徴点抽出，特徴量記述）の並列度を決定するパラメータである．回路規模と処理速度に影響を与える．

4. トレードオフ解析

本章では 3.2 節で述べたパラメータについてコストと精度の観点から解析したトレードオフについて説明する．

4.1. 解析方法

解析を行うためにオープンソースの SIFT ソフトウェア[11]をベースにシミュレーション環境を構築した. 画像のペア（元画像と射影変換を適用した画像）に対して特徴量抽出を行い，イメージマッチングを行うことで精度の解析をおこなった.

4.2. 精度の評価指標

特徴点探索処理の精度評価指標として2 画像間でのマッチングにおける Repeatability を、特徴量記述処理の精度評価指標として 2 画像間でのマッチングにおける Precision と Recall から求める F Measure を用いる．

4.2.1. Repeatability

Repeatability とは異なる 2 画像間でどの程度の割合で"同じ"点を特徴点として検出できたかを示す指標である．Repeatability は式 1 により求めることができる[3]。n1，n2 はそれぞれ画像 1，2 から得られた特徴点の数である．

$$\mathrm{Repeatability} = \frac{\#\mathrm{correspondences}}{\min(n1, n2)} \tag{3}$$

correspondence については図 13 を用いて説明する．画像 1 から得られた特徴点 1 を画像 2 にマップした時, 特徴点 2 との位置の誤差が閾値以下であり，記述領域が閾値以上重なっている場合を correspondence 有りとする．そして，そのような correspondence の数の合計が #correspondences である．画像 1 の記述領域は画像 1 を画像2 にマップしたときの変換パラメータによって歪んでいる．以降のシミュレーションでは，位置の閾値として $\sqrt{2}$ ，重なり率の閾値として 0.5 を用いる．

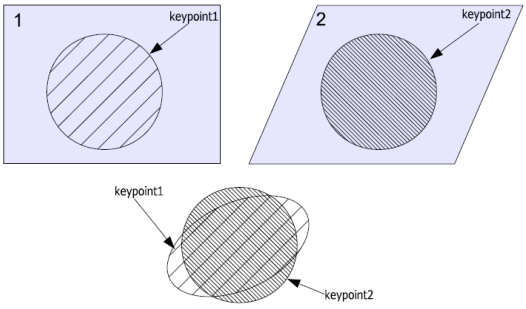

図 13 Correspondence の判定

4.2.2. Precision, Recall と F Measure

Precision と Recall は 2 画像間での特徴量のマッチングにおける精度を示す指標である．

まず，マッチングについて説明する．画像 1 から得られたある特徴点 1 に対し，画像 2 から得られた特徴点の中から特徴点 1 の特徴ベクトルとの距離が小さい順に 2 つの特徴点を検索し，その距離の比率が閾値以下である場合を match と判定する．距離の比率の閾値は 0.65 を用いる．そして，距離が最も小さかった特徴点の位置と画像 2 にマップした特徴点 1 の位置との誤差が閾値以下であった場合を correct match，そうでない場合を false match とする．座標誤差の閾値は Repeatability の場合と同様に $\sqrt{2}$ を用いる．

ここで Recall とは，2 画像間での特徴量のマッチングで correspondence 有りとされた特徴点の組からどれだけ正しくマッチングできたかを示す指標であり、式 2 で計算できる[3].

$$\mathrm{Recall} = \frac{\#\mathrm{correct\ match}}{\#\mathrm{correspondences}} \tag{4}$$

Precision とはどれだけ正しくマッチングできたかを

示す指標であり，式 3 で計算できる．

$$\text{Precision} = \frac{\#\text{correct match}}{\#\text{correct match} + \#\text{false match}} \quad (5)$$

Recall を高くしようとすると Precision は低下し，Precision を高くしようとすると Recall は低くなる傾向がある．そのため，その 2 つの調和平均である F Measure もよく用いられる．F Measure は式 4 で計算できる．F Measure が高いほど性能が良いということになる．

$$\text{F Measure} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (6)$$

4.3. テスト画像

テスト画像として以下のものを用いる．

- 標準画像 Lenna を変形した画像
- Mikolajczyk's Dataset[4]

標準画像 Lenna を変形した画像は，変形させるときのパラメータを 2 画像間の射影変換パラメータとして用いる．Mikolajczyk's Dataset はカメラ撮影された画像であるが，2 画像間の射影変換パラメータが ground truth として用意されている．表 1 でテスト画像に加えられている変形の情報を示す．

表 1 テスト画像

Number	Name	Image transform
0~9	Lenna	Zoom
10~19	Lenna	Rotation
20~24	bark	Zoom+Rotation
25~29	boat	Zoom+Rotation
30~34	bikes	Blur
35~39	trees	Blur
40~44	graf	Viewpoint change
45~49	wall	Viewpoint change
50~54	leuven	Light change
55~59	ubc	JPEG compression

4.4. 解析結果

本節ではトレードオフ解析の結果を示し，考察をおこなう．

4.4.1. 1 画素に割り当てるビット数

1 画素あたりのビット数は演算回路の規模や搭載画像メモリの量など，ハードウェア全体のリソース量に影響する．ここでは 1 画素を float，17 ビット，16 ビット，15 ビット，14 ビットで表した場合のマッチング性能を評価する．シミュレーション結果として，図 14 に Repeatability と F Measure のグラフを示す．Repeatability と F Measure の値は 60 枚のテスト画像の平均値である．1 ビットずつ割り当てるビット数を減らしていった場合，15 ビットから劣化幅が大きくなりはじめ，14 ビットで急激に性能が悪化した．ビット数が 16 以下の場合，一般的な 32 ビットバスで 1 サイクルに 2 画素ずつ外部メモリを読み書きできるため有利である．よって 1 画素に割り当てるビット数は 16 ビットまたは 15 ビットが良いと思われる．

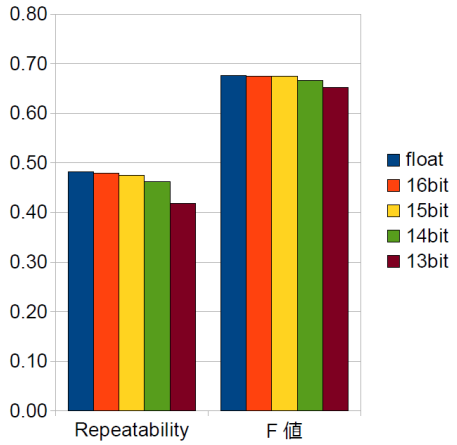

図 14 Repeatability と F Measure (1 画素に割り当てるビット数)

4.4.2. 2. ROI (Region of Interest)のサイズと拡張 ROI 領域

アーキテクチャの項目で述べたように，ROI 単位で処理する場合は拡張 ROI 領域も外部メモリから読み込み，ガウシアン処理しなければならない．ガウシアン処理部の演算量，搭載画像メモリ量と外部メモリとのメモリ帯域は拡張 ROI 領域の大小によって影響される．

ここでは拡張 ROI 領域の大きさを 1 辺 16, 12, 8, 4 画素とした場合の性能と ROI なしで一括処理する場合の性能を評価する．なお，搭載 SRAM が現実的な量となるようにするため，ROI の大きさとして 80x80 画素を用いる．

評価結果として，図 15 に Repeatability と F Measure のグラフを示す．

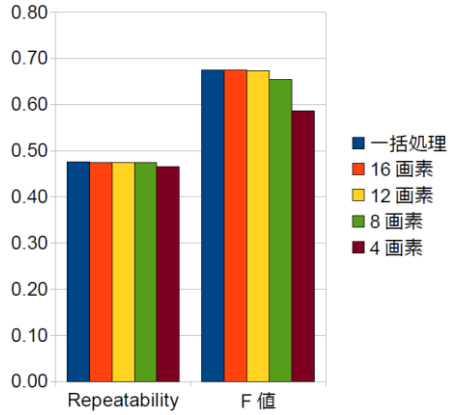

図 15 Repeatability と F Measure (拡張 ROI 領域)

Repeatability に関しては拡張 ROI 領域を小さくしていってもあまり性能劣化は起こらないが、F Measure は 8 画素で劣化が目立ちはじめ，4 画素では大きく劣化している．12 画素までは Repeatability・F Measure ともに目

立った劣化はない．しかし，一括処理と同じ結果を得るためには拡張 ROI 領域が 30 画素程度必要なことを考えると，これは注目すべき結果である．F Measure の劣化が少なかった理由は、特徴量記述処理では特徴点の中心から遠い画素ほど輝度勾配ベクトルの重みが小さくなり，拡張 ROI 領域が必要になるような画素はもともと重要度が低くなる傾向があるためだと思われる．

4.4.3. ガウシアン平滑化パラメータ σ_0

ガウシアンフィルタのウィンドウサイズは σ_0 の値によって決まり，この大小がガウシアン処理部の演算量と規模に影響する．σ_0 の値を 1.6，1.4，1.3，1.2，1.1，1.0 とする場合のマッチング性能を評価する．

評価結果として，図 16 に Repeatability，F Measure，ガウシアン処理部に含まれる乗算器の数のグラフを示す．乗算器の数は各 σ_0 において，同じ速度で処理するという条件下での値である．乗算器の数により相対的な演算量・回路規模を見積もることができる．マッチング性能と演算量･回路規模を考慮すると，σ_0 の値は 1.1～1.4 が良さそうである。

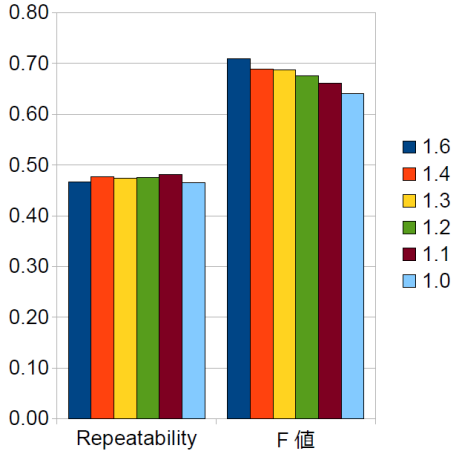

図 16 Repeatability と F Measure (ガウシアン平滑化パラメータ σ_0)

4.4.4. 演算モジュールの並列度

本節では動作周波数 100MHz で HDTV 解像度の画像を 30fps で処理する状況を想定した場合の演算モジュールの並列度について検討する．ガウシアン平滑化処理と特徴点抽出処理は解像度に依存して処理量が決まるため，適切に並列度を設定すれば精度に影響を及ぼすことなくリアルタイム処理が可能である．しかし特徴量記述処理の処理量は検出された特徴点の数や特徴点のスケールに依存し，全ての特徴点を制限時間内に処理しようとすると現実的ではない並列度が必要となる．時間内に処理が終わらない場合，処理を打ち切って次の ROI の処理にとりかかるようにしなければならないが，処理が間に合わなかった特徴点は除去され性能が劣化する．

そこで本節では精度に影響を与える特徴量記述部に絞って並列度の説明をおこなう．特徴量記述処理の並列度を 3，2，1 とした場合のマッチング性能と打ち切り処理なしの場合のマッチング性能を評価する．評価結果について，図 17 に Repeatability と F Measure のグラフを示す．並列度が下がるにつれて Repeatability は劣化している．ガウシアン処理部と比べると特徴量記述部の規模はかなり小さくなるだろうことが予想される．そのため処理打ち切りによる精度劣化を抑えることを考えると，並列度は 2 か 3 が良さそうである．

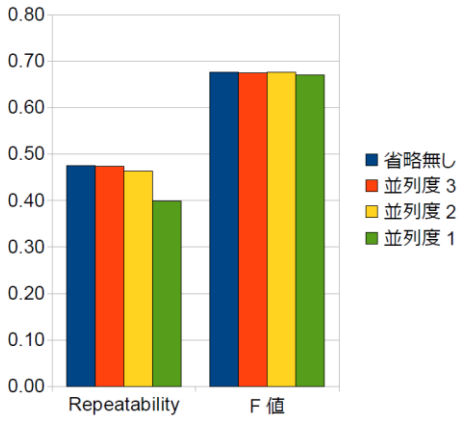

図 17 Repeatability と F Measure (特徴量記述部の並列度)

4.5. ソフトウェア実装の SIFT や他の特徴量との比較

今まで評価してきたパラメータを組み合わせて，3 つのパラメータセットを定義し，これらの性能をソフトウェア実装の SIFT・SURF と比較する．

SURF は SIFT と同じように回転・スケールや輝度の変化に不変な特徴量で、SIFT よりも演算負荷が小さいという特徴がある[5] ことから高速な処理が求められるアプリケーションで良く用いられている．

表 2 に 3 つのパラメータセットの内容を示す。set1 はマッチング性能の劣化を可能な限り抑えることを目指したパラメータ設定、set2 は性能の劣化幅が小さい領域で最もリソース量が少なくなるようなパラメータ設定、set3 はある程度の劣化を許容してリソース量の削減を狙ったパラメータ設定となっている。

表 2 パラメータ設定

Setting	Bits per pixel	σ_0	拡張 ROI	特徴量記述並列度
Set1	16bit	1.4	16pixels	3
Set2	16bit	1.2	12pixels	3
Set3	15bit	1.1	8pixels	2

表 3 に 3 つのパラメータセットのハードウェアリソースを示す。演算器の規模に関しては、SIFT の処理のうち最も演算量が大きいガウシアン処理部に含まれる乗算器の数を用いた相対的な数値となっている。

表 3 設定毎のハードウェアリソース

Setting	乗算器数	メモリ容量	外部メモリ帯域
Set1	544	3.63Mbit	2.05Gbps
Set2	472	3.13Mbit	1.81Gbps
Set3	330	2.51Mbit	1.58Gbps

3 つのパラメータセット、ソフトウェアによる SIFT と SURF の性能の評価結果について図 18 に Repeatability と F Measure を示す.

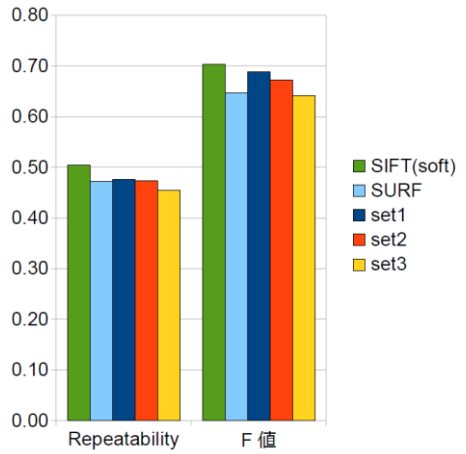

図 18 Repeatability と F Measure (ソフトウェア SIFT，SURF との比較)

ソフトウェア実装の SIFT と比べ，set1，set2，set3 は Repeatability がそれぞれ約 5.6%，約 5.9%，約 9.7%の劣化，F Measure がそれぞれ約 2.1%，約 4.3%，約 8.9%の劣化にとどまっている。

性能劣化をなるべく抑えることを狙った set1 はリソース量に見合うマッチング性能を達成できているとは言い難く、劣化幅が大きくなり始める限界のパラメータを使った set2・set3 のほうがリソース量とマッチング性能のバランスが良くなっている.

また，そのような方針で設定されたパラメータを用いることでSURF と比べても遜色のない十分実用的なマッチング性能となることが分かる.

5. まとめ

本稿では, SIFT 特徴量抽出処理を低コストかつ高精度で処理するためのハードウェアアーキテクチャを検討し，そのようなアーキテクチャにおいて SIFT アルゴリズムのパラメータがハードウェアリソースと抽出された特徴量のマッチング性能とにどのような影響を与えるかを調査した．その結果，ある程度のリソースがある場合ではリソース量の変化がマッチング性能に与える影響は小さいが，特定のリソース量を下回ると性能が急激に劣化し始める傾向があり，そのようなギリギリの領域のパラメータを用いるとリソース量とマッチング性能とが上手くバランスするということが分かった.

また，この方針で選択したパラメータを組み合わせてシミュレーションを行ったところ，ソフトウェア処理の SIFT と比較してマッチング性能の劣化は 5〜10% 程度となり，SURF と比べても遜色なく十分実用的な性能となることが分かった.

文　献

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, 60, 2, pp. 91-110, January 2004.

[2] Luo Juan, Oubong Gwun, "A Comparison of SIFT, PCA-SIFT and SURF," International Journal of Image Processing (IJIP) Volume(3), Issue(4) pp. 143-152.

[3] David Gossow, Peter Decker, Dietrich Paulus, "An Evaluation of Open Source SURF Implementations," (http://www.chrisevansdev.com/computer-vision-opensurf.html).

[4] Mikolajczyk's Dataset(http://www.robots.ox.ac.uk/ vgg/research/affine/index.html).

[5] H. Bay, A. Ess, T. Tuytelaars, L. V. Gool, "SURF: Speeded Up Robust Features", Computer Vision and Image Understanding (CVIU), Vol. 110, No. 3, pp. 346-359, 2008.

[6] S. Heymann, K. Müller, A. Smolic, B. Froehlich, and T. Wiegand, "SIFT Implementation and Optimization for General-Purpose GPU", In Proc. of the WSCG'07, pp. 317-322, January 2007.

[7] V. Bonato, E. Marques, and G. A. Constantinides, "A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection," IEEE Trans. Circuits Syst., vol. 18, no. 12, pp. 1703-1712, Dec. 2008.

[8] L. Yao, H. Feng, Y. Zhu, Z. Jiang, D. Zhao, and W. Feng, "An architecture of optimised SIFT feature detection for an FPGA implementation of an image matcher," ICFPT 2009.

[9] D. Kim, K. Kim, J. Y. Kim, S. Lee and H. Yoo, “An 81.6 GOPS Object Recognition Processor Based on NoC and Visual Image Processing Memory”, CICC, pp.443-446, Sep. 2007.

[10] J. Y. Kim, M. Kim, S. Lee, J. Oh, K. Kim, S. Oh, J.H. Woo, D. Kim and H. J. Yoo, “A 201.4GOPS 496mW real-time multi-object recognition processor with bio-inspired neural perception engine”, ISSCC Dig., pp.150-151, Feb. 2009.

[11] R. Hess, “SIFT feature detector (source code),” 2007. Available: http://web.engr.oregonstate.edu/~hess/