Vol.2016-CVIM-201 No.8
2016/3/3

Brick Partitioningと初期閾値推定を用いた高速テンプレートマッチング

町井 孝充[1,a)]　外山 史[1]　森 博志[1]　東海林 健二[1]

概要：テンプレートマッチングは検索対象画像内の各位置に対して，テンプレート画像との類似性を評価し，最も適合する位置を検出する技術である．今日まで，この技術の性能向上のために様々な手法が提案されている．本論文では，最適な位置検出の保証を維持しつつ，処理の高速化を目的とした手法を提案する．既存手法である Winner Update Algorithm(WUA) と Multilevel Successive Elimination Algorithm(MSEA) を組み合せて，各検索位置 (ウィンドウ) での照合時における計算量を大幅に削減する．提案手法では，まず，WUA を用いて少量のウィンドウから有効な初期閾値を高速に計算する．次に，MSEA を用いて本探策を行う際，この初期閾値を適用して探索することで，非常に高速なテンプレートマッチングを実現している．また，Brick Partitioning によりテンプレート及びウィンドウを領域分割することで，各ウィンドウにおいてより少ない計算量で類似性を評価することが可能となる．実験では，提案手法が大幅な探索時間削減を実現し，非常に高速な手法として知られている適応的領域分割と初期閾値推定を用いた MSEA(ABPMSE+ITE) よりも高速に探索できることが確認できた．

1. はじめに

ある既知の画像が与えられたとき，検索対象画像の中から最も類似している領域を見つける問題は，パターン認識やコンピュータビジョンにおける重要な課題の一つであり，画像データベースからの目的画像の検索，映像中の物体の追跡，検索など幅広い分野への応用がある．これらを実現するため，様々な技術が存在するが，中でもテンプレートマッチング法 [1] はよく知られている代表的な手法の一つである．テンプレートマッチング法とは，与えられた既知の画像 (テンプレート) と検索対象画像上の各位置 (ウィンドウ) における類似度を逐次評価していき，最も類似度の高いウィンドウを検出する手法である．類似度には，テンプレート及びウィンドウの各画素の差分絶対和 (絶対距離値) などが用いられる．テンプレートマッチングは検索対象画像上の最適な位置を正確に検出することができるが，その反面，計算量が多く，また検索対象画像上でウィンドウを走査して照合を行うため，検索対象画像の枚数や，テンプレートのサイズに比例して計算時間が膨大になるという問題がある．そのため，これまで多くの高速化手法が提案されてきた．

代表的な手法として，各ウィンドウに対する類似度 (距離値) の計算量を削減することで処理を高速化する残差逐次検定法 (Sequential Similarity Detection Algorithm，SSDA)[2] や，低解像度画像から高解像度画像へと順に探索していく Coarse-to-Fine 探索 [3]，色ヒストグラムを用いて，評価したウィンドウの近傍にあるウィンドウのスキップ可否を判定するアクティブ探索 [4] などが挙げられる．SSDA[2] はテンプレートとウィンドウの距離計算における，画素差分を積算していく際，随時閾値 θ との比較を行い，積算値が閾値 θ を超えた時点で，そのウィンドウでの計算を終了することで，計算量を削減している．ここで，閾値 θ はそれまでのテンプレートとウィンドウ間の距離の最小値である．Coarse-to-Fine 探索は，まず低解像度の画像で探索を開始し，その結果得られた検出位置候補の近傍のみに着目して高解像度の画像で探索する手法である．はじめに低解像度画像で探索を行っているため，探索精度を犠牲にして速度向上を図っている．そのため，最適位置の検出が保証されていないという欠点がある．アクティブ探索は色ヒストグラムによる評価値を用いた手法 [4] である．アクティブ探索では，あるウィンドウでの照合結果を利用して，その近傍ウィンドウの類似度の上限を求め，上限値が閾値を下回る場合は照合をスキップすることにより処理を高速化している．アクティブ探索は総当たり探索と同じ精度で効率よく位置検出を可能とするが，色ヒストグラムを利用していることから，形状や色情報が失われ，正確な

[1] 宇都宮大学大学院工学研究科
Graduate School of Engineering, Utsunomiya University
[a)] mt146530@cc.utsunomiya-u.ac.jp

位置を求めるのは難しいという問題がある．さらに，アクティブ探索における， 近傍ウィンドウの類似値の上限を用いた照合のスキップという考えを応用した適応的ウィンドウスキッピング法 (Adaprive Window Skiping Method, AWS)[5] がある．AWS では，色ヒストグラムの代わりにテンプレートマッチング法 [1] 等で用いられる距離を適用している．テンプレートの部分領域であるサブテンプレートとウィンドウの部分領域であるサブウィンドウを比べることで，サブウィンドウを内部に含む各ウィンドウの距離の取り得る下限値を計算し，その距離下限値と閾値 θ を比較していく．AWS はテンプレートマッチング法と同じ探索精度を保証しつつ，前述した SSDA よりも高速に処理を行うことができる．

距離の下限値という概念を利用した探索手法は他にも多く存在する．例えば Successive Elimination Algorithm(SEA)[6] は各ウィンドウにおいて距離の下限値を計算し，その値が閾値 θ を上回る場合は距離の計算をスキップすることで計算量を削減している．Multilevel SEA(MSEA)[7] は SEA を多段階に拡張した手法である．テンプレート及びウィンドウに対して多段階の分割処理を施し，より値の大きい距離下限値を獲得する．これにより，より多くのウィンドウで計算が大幅に削減され，高速に探索を行うことができる．さらに，MSEA については，適応的領域分割及び初期閾値推定を用いて，さらなる高速化を図った改良手法 [8] が提案されている．適応的領域分割は画像の複雑さに応じた領域分割を行うことで等分割時よりも効率的な距離下限値を得る．また，初期閾値推定は，より小さな閾値を本探索前に得ることで，探索開始時点から計算量を削減している．一方，Winner Update Algorithm(WUA)[9] はテンプレート及びウィンドウの多段階処理による距離下限値獲得に加え，ウィンドウの評価順を柔軟に変更して探索する手法である．常に最小の距離下限値を持つウィンドウに着目して，優先的に距離下限値の更新を行うことで，より小さな閾値 θ を高速に得ることができる．しかし，検索対象画像に対して，テンプレートのサイズが小さい，すなわちウィンドウ数が増加すると，最小距離下限値を検出するためのソート処理に多くの時間を費すという問題がある．

本論文では，WUA を用いた初期閾値の推定，さらに距離下限値計算におけるテンプレート及びウィンドウに対して，新たな領域分割手法 Brick Partitioning を提案する．提案手法は，より高速に最適位置を検出するとともに，他のウィンドウの計算スキップ率を向上させることで，テンプレートマッチング法と同じ精度を保ったまま，非常に高速な処理を行うことができる．

2. 関連研究

2.1 Full Search(FS)

T を画素サイズ $M\times N$ のテンプレート画像, I を検索対象画像とする．FS は最も基本的なテンプレートマッチング法であり, 検索対象画像上のすべての照合位置 (ウィンドウ) において, 類似度の評価を逐次行い, 最良の結果となるウィンドウを適合位置として出力する．類似度の評価値として, テンプレート, ウィンドウ間の距離を用い, これを $d(x,y)$ と表す．(x,y) は検索対象画像内におけるウィンドウの位置である (画像の横方向を x 軸, 縦方向を y 軸方向とする)．本論文では, テンプレート, ウィンドウ間の距離は式 (1) で求められる絶対値距離 (Sum of Absolute Difference:SAD) を用いる．

$$SAD: d(x,y) = \sum_{i=0}^{M-1}\sum_{j=0}^{N-1} |T(i,j) - I(x+i,y+j)| \quad (1)$$

ここで, $T(i,j)$ 及び $I(i,j)$ は, それぞれテンプレート画像及び検索対象画像の座標 (i,j) における画素値である．さらに, 検索対象画像のサイズを $X\times Y$ とすると, x, y はそれぞれ $0\le x\le X-M$, $0\le y\le Y-N$ の範囲となる．したがって, ウィンドウの総数は $(X-M+1)\times(Y-N+1)$ となり, これら全てのウィンドウに対して距離 $d(x,y)$ を計算し, 最小の $d(x,y)$ をとる座標を適合位置として出力する．

2.2 MSEA

FS では必ず最適位置を検出できる反面，すべてのウィンドウにおける距離 $d(x,y)$ を計算する必要があるため，計算量が膨大であることが欠点として挙げられる．そこで，テンプレートとウィンドウを多段階的に分割して得られる距離下限値を用いることで，計算量を削減した手法が MSEA[7] である．距離下限値とは，各ウィンドウに対して，テンプレートとの取り得る距離 $d(x,y)$ の下限の値を表しており，距離 $d(x,y)$ の計算以前に距離下限値を計算することで，距離 $d(x,y)$ の計算のスキップ可否を判定することができる．l をテンプレート及びウィンドウを分割した回数 (分割レベル) とすると，$l=1$ では $M/2\times N/2$ の部分領域に分割される．これを $M/4\times N/4$ を経て，1×1 のサイズまで分割を繰り返す．ここで各分割レベルにおけ

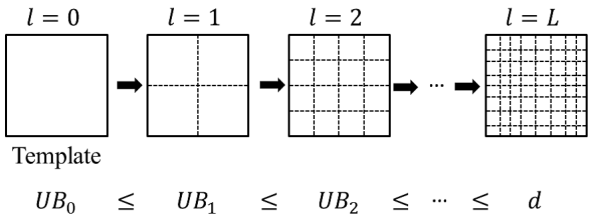

図 1 多段階的な領域の分割
Fig. 1 Multilevel partitioning process.

る領域分割の様子を図 1 に示す．UB_l は分割レベル l のときの距離下限値を表しており，分割レベルの増加に伴い，距離下限値は単調増加を示す．このとき，分割レベル l の範囲は $L = log_2 N (N < M)$ として $0 \leq l \leq L-1$ であり，また，各分割レベルにおける部分領域数は $K_l = 4^l$ である．各分割レベルにおいて，距離下限値 UB_l を計算し，閾値 θ と比較する．ここで閾値 θ とは現段階での最小の距離値 $d(x,y)$ である．分割上限まで距離下限値 UB_l を求め，閾値 θ よりも小さい場合，そのウィンドウとの距離 $d(x,y)$ を計算する．距離 $d(x,y)$ もまた閾値 θ より小さい場合，最小距離を持つウィンドウとして，閾値 θ と検出位置を更新する．分割途中で距離下限値 UB_l が閾値 θ を上回る場合，そのウィンドウとの距離 $d(x,y)$ が最小距離として更新されることはないため，以降の計算をスキップすることが可能となる．この操作を全てのウィンドウに対して順番に行っていき，最小の距離 $d(x,y)$ をとるウィンドウを検出位置として出力する．

2.2.1 距離下限値の計算

距離下限値はテンプレート及びウィンドウを分割し，部分領域に含まれる画素の積算値より求められる値である．部分領域内の画素の総和はインテグラルイメージ [10] を使って高速に計算することができる．分割レベル l のときの部分領域数は $K_l = 4^l$ であり，それぞれ k 番目の部分領域を $T_{sub(k)}$ 及び $W_{sub(k)}$ とする．このとき，分割レベル l の各 k における部分領域間の積算値の差分 $UBsub_{l,k}(x,y)$ は式 (2) のように表すことができ，これを部分距離下限値とする．ここで，$T^p_{sub(k)}$ 及び，$W^p_{sub(k)}$ はそれぞれテンプレート，ウィンドウの部分領域 $T_{sub(k)}$，$W_{sub(k)}$ に含まれる p 番目の画素値であり，P は部分領域内の画素の総数である．

$$UBsub_{l,k}(x,y) = |\sum_{p=0}^{P-1} T^p_{sub(k)} - \sum_{p=0}^{P-1} W^p_{sub(k)}| \quad (2)$$

したがって，距離下限値 $UB_l(x,y)$ は式 (3) のように各部分領域における部分距離下限値の総和から成る．

$$UB_l(x,y) = \sum_{k=0}^{K_l-1} UBsub_{l,k}(x,y) \quad (3)$$

2.2.2 距離下限値と絶対値距離

距離下限値と絶対値距離を併用している手法 MSEA では，あるウィンドウに対して，距離下限値の値によって距離計算，すなわち絶対値距離の計算のスキップ可否を判定することができる．以下に示す三角不等式

$$|a+b| \leq |a| + |b| (a, b \ni \Re) \quad (4)$$

から，次の関係が成り立つ．

$$\begin{aligned} &| \sum_{p=0}^{P-1} T^p_{sub(k)} - \sum_{p=0}^{P-1} W^p_{sub(k)}| \\ &\quad \leq \sum_{p=0}^{P-1} |T^p_{sub(k)} - W^p_{sub(k)}| \\ &\therefore \sum_{k=0}^{K_l-1} |\sum_{p=0}^{P-1} T^p_{sub(k)} - \sum_{p=0}^{P-1} W^p_{sub(k)}| \\ &\quad \leq \sum_{k=0}^{K_l-1} \sum_{p=0}^{P-1} |T^p_{sub(k)} - W^p_{sub(k)}| \\ &\therefore UB_l(x,y) \leq d(x,y) \end{aligned} \quad (5)$$

式 (5) の左辺は距離下限値を表し，右辺は絶対値距離を表す．これは，任意の分割レベルにおける距離下限値は絶対値距離以下であることを示す．また，領域数の異なる各分割レベルにおける距離下限値間においても同様である．すなわち，以下に示す式 (6) の関係が成り立つ．

$$\begin{aligned} &UB_0(x,y) \leq ... \leq UB_l(x.y) \\ &\quad \leq UB_{l+1}(x,y) \leq ... \leq d(x,y) \end{aligned} \quad (6)$$

距離下限値は分割レベルを増加させると，その値が大きくなるとともに絶対値距離に近似していく．これを利用し，MSEA では各ウィンドウについて，分割レベル 0 から $L-1$ まで順に距離下限値を計算していき，スキップ可否の判定を行う．距離下限値が閾値 θ を上回った場合は計算のスキップが可能となる．分割レベル $L-1$ までスキップが行われない場合，距離 $d(x,y)$ を計算し，再び閾値 θ との比較を行い，更新の有無を決定する．

2.2.3 適応的領域分割と初期閾値推定

文献 [8] では，MSEA に，適応的領域分割と初期閾値推定を適用して拡張した手法 (ABPMSE+ITE) を提案している．MSEA において，距離下限値は，領域を等分割することにより得ている．しかし，等分割による距離下限値の値は必ずしも効率的な値になるとは限らない．部分距離下限値の計算においては，画像が一様な部分領域からは小さい値が，複雑な部分領域からは大きな値が得られやすいことから，部分領域間で同程度の複雑さを持つように分割することで，より値の大きい距離下限値の取得を図っている．一方，初期閾値推定では，少ない処理量で値の小さな閾値を獲得し，MSEA による本探策に用いることで処理を高速化している．適応的領域分割及び初期閾値推定の具体的な手法は，文献 [8] を参照されたい．

2.3 WUA

WUA[9] は MSEA と同様，テンプレート及びウィンドウを領域分割し，距離下限値を用いて各ウィンドウを評価

していく手法であるが，最大の特徴はウィンドウの計算順序を距離下限値によって変更していく点にある．MSEA がウィンドウを原点 (左上隅) から順に走査していく手法であるのに対し，WUA では距離下限値の値が小さいウィンドウを優先的に選択し，領域分割及び距離下限値の計算を行う．WUA ではより小さい閾値を素早く得ることができるが，最小の距離下限値をもつウィンドウを検出するにあたり，距離下限値を昇順にソートする必要があるため，ウィンドウ数が増加するほどに多大な時間を費してしまうという問題点がある．以下に WUA の処理の流れを示す．

(1) テンプレートの各分割段階ごとに部分領域内の積算値を求める．
(2) $l = 0$（l：分割回数）ですべてのウィンドウに対し，テンプレートとの距離下限値を計算する．
(3) 距離下限値が最小のウィンドウを選択する．
(4) 選択されたウィンドウの領域を分割し，$l = l + 1$ とする．ウィンドウが分割上限に達した場合 5. へ．それ以外は 6. へ．
(5) テンプレートとウィンドウ間の絶対値距離を計算する．この値が他のウィンドウの距離下限値よりも小さければ，そのウィンドウの位置を検出位置として出力し，処理を終了する．それ以外は 3. へ．
(6) テンプレートと分割したウィンドウの距離下限値を計算し，新たな距離下限値で元の距離下限値を更新する．3. へ．

3. 提案手法

3.1 概要

本論文ではテンプレートマッチングのさらなる高速化を図るため，WUA による初期閾値推定及び，Brick Partitioning による領域分割法を提案する．提案手法では，まず，検索対象画像上の全ウィンドウから等間隔に選択した少量のウィンドウのみに対して WUA による探索を行う．この操作により，高速に，かつ極めて値の小さい初期閾値を獲得し，本探策に利用する．本探策では MSEA による走査で残りのウィンドウを評価していくが，初期閾値の効果により，各ウィンドウにおいて大幅に計算量を削減することができる．また，提案手法における探索の際，距離下限値を計算するための領域分割に，Brick Partitioning を用いた領域分割法を提案する．これにより，従来の等分割や，文献 [8] で用いられている適応的領域分割よりも値の大きな距離下限値を得ることができる．本手法では，初期閾値推定と Brick Partitioning による領域分割を組み合わせることによって，計算スキップ率を高め，探索全体の処理量を削減する．

3.2 WUA による初期閾値推定

従来手法である MSEA は，距離下限値を用いて効率的に探索を行うが，検索対象画像上の原点から順に走査していくため，値の小さい閾値を得る前に探索するウィンドウでは，計算効率が悪くなるという問題点がある．一方，WUA は全ウィンドウを対象に，距離下限値の小さい順に従って柔軟に探索位置を変更していく．しかしながら，最小距離下限値を検出するためにソート処理を施すため，ウィンドウ数が増加すると，多大な時間を費すという欠点が存在する．そこで提案手法では，探索を 3 つの段階に分けて処理を行う．第 1 及び第 2 段階として，WUA を用いた粗探索を実行することで，極めて値の小さな初期閾値を獲得する．さらに，第 3 段階において，未探索のウィンドウを対象に，MSEA による探索を行い，最終的な検出位置を出力する．まず，第 1 段階について説明する．第 1 段階では検索対象画像上の全ウィンドウから等間隔にウィンドウを選択し，探索を行う．ここで，検索対象画像内の任意の位置 (x, y) にあるウィンドウを $w(x, y)$ とすると，全ウィンドウの集合 $\mathbf{U}$ は式 (7) のようになる．

$$\mathbf{U} = \{w(x,y) | 0 \leq x \leq X - M, \\ 0 \leq y \leq Y - N\} \tag{7}$$

このうち，第 1 段階で対象となるウィンドウは，以下 $\mathbf{V_1}$ に含まれるウィンドウである．

$$\mathbf{V_1} = \{w(x,y) | x = Sn, y = Sn, n = 1, 2, 3...\} \tag{8}$$

ここで，S は抽出する間隔を示すパラメータであり，原点にあるウィンドウ $w(0,0)$ から x 軸方向，y 軸方向に S 画素おきに存在するウィンドウが対象となる (図 2(a))．選択したウィンドウに対してのみ，WUA を実行する．すなわち，対象ウィンドウ全てにおける距離下限値を計算し，その中で最小値をもつウィンドウの更新を繰り返す．これにより，選択されたウィンドウ内で最も距離が小さい，すなわち検出位置の候補となるウィンドウを算出する．図 2 の赤色部分は，各段階における検出位置候補として出力されるウィンドウ例である．次に第 2 段階では，図 2(b) で示すように，第 1 段階で出力した検出位置の候補となっているウィンドウの近傍を探索する．これは，既に獲得している閾値をより小さい値に更新することを目的としている．第 1 段階で得た検出位置候補のウィンドウを $w(x^1_{min}, y^1_{min})$

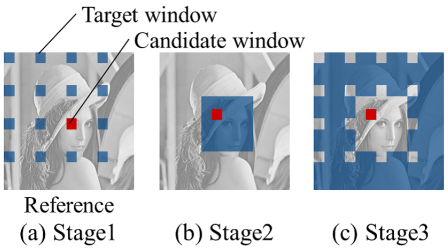

図 2 各段階における対象ウィンドウ
Fig. 2 Target window at each stage.

とすると，第 2 段階での対象ウィンドウは以下のようになる．

$$\mathbf{V_2} = \{w(x,y)|x^1_{min} - S \leq x \leq x^1_{min} + S, \\ y^1_{min} - S \leq y \leq y^1_{min} + S\} \quad (9)$$

すなわち，第 1 段階で得た検出位置候補の位置から x 軸方向に左右 S 画素，y 軸方向に上下 S 画素の範囲に存在するウィンドウに対して探索を行う．探索は第 1 段階と同様，WUA を適用し，探索したウィンドウの中で最小の距離をとるウィンドウ $w(x^2_{min}, y^2_{min})$ を出力する．以上の操作を経て得た距離値は，第 1 段階及び第 2 段階を通して最も小さい値であり，第 3 段階において，極めて有効な初期閾値として用いられる．第 3 段階では，第 1 段階及び第 2 段階では探索されなかったウィンドウを対象とする (図 2(c))．すなわち，式 (10) で表される集合 $\mathbf{V_3}$ に含まれるウィンドウである．

$$\mathbf{V_3} = \{w(x,y)|w(x,y) \notin V_1, w(x,y) \notin V_2\} \quad (10)$$

第 3 段階では，第 2 段階で得られた $w(x^2_{min}, y^2_{min})$ の距離値を初期閾値とし，対象とするウィンドウを，MSEA の探索手法を用いて，原点に近いウィンドウから順に評価していく．すなわち，ウィンドウを個々に着目していき，分割レベル 0 から順に距離下限値を計算，閾値との比較を行い，逐次検出位置かどうかを判断していく．第 3 段階においてのみ，探索手法として WUA を用いていない理由は以下の 2 点が挙げられる．1 点目は極めて有効な初期閾値が探索開始時点で既知である点である．これにより，原点に位置するウィンドウから順に評価していく MSEA 探索においても，探索開始時から走査を高速に実行することが可能となる．2 点目は，第 3 段階の対象ウィンドウ数にある．WUA の欠点として，ウィンドウが増加するに従い，最小距離下限値を持つウィンドウを検出するために行うソート処理に多大な処理時間を費す点が挙げられる．そこで，これを回避するために，対象ウィンドウ数の多い第 3 段階では，WUA よりも，単調にウィンドウを順次走査していく MSEA が適切であると考えられる．以上の処理により，最終的に閾値として保持している距離値をもつウィンドウが検出位置として出力される．提案手法により検出された位置は最適な位置を保証している．図 3 に提案手法の処理の流れを示す．第 1 段階では，閾値 ($\theta = \infty$) と対象ウィンドウ $\mathbf{V_1}$ を WUA に入力し，探索の結果として最小距離とその位置を出力する．同様に第 2 段階では，第 1 段階で得られた閾値 ($\theta = d(x^1_{min}, y^1_{min})$) と対象ウィンドウ $\mathbf{V_2}$ を WUA に入力し，最小距離とその位置を獲得する．続いて第 3 段階では，MSEA に対して閾値 ($\theta = d(x^2_{min}, y^2_{min})$) 及び対象ウィンドウ $\mathbf{V_3}$ を入力し，探索を行う．以上の処理を以て，集合 $\mathbf{U}$ に含まれる全ウィンドウの評価が完了し，最終的な検出位置 (x_{min}, y_{min}) を得る．

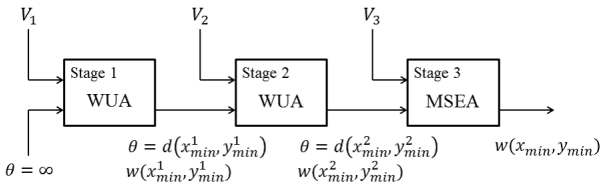

図 3 WUA による初期閾値の推定
Fig. 3 Intial Threshold Estimation by the WUA.

3.3 Brick Partitioning

従来の手法ではテンプレート, ウィンドウの分割領域は等分割によって得られているが, 等分割による領域から得られる距離下限値が必ずしも効率的な値とは限らない．そこで, 領域分割を適応的に行うことで, より厳しい距離下限値を得る手法を検討する．距離下限値を計算する際, 画像が一様な部分領域では小さな距離が得られやすいと考えられる．逆に, 画像が複雑な部分領域では, より大きな距離が得られやすい．そこで, 適応的に部分領域の大きさを設定し, 部分領域間で同程度の複雑さとなるように分割できれば, 同じ分割回数でも等分割した部分領域よりさらに厳しい距離下限値が得られると考えられる．この指針に沿った適応的領域分割 (Adaptive Block Partitioning，ABP) が文献 [8] で提案されている．本論文では適応的領域分割をさらに改良し，より，画像の複雑さの分布に忠実な領域分割方法 Brick Partitioning を提案する．まず, テンプレートから画像の複雑さを表す指標を抽出する．これには Gradient Magnitude を用いることにし, 式 (11) より算出する．

$$\begin{aligned} \| G[T(i,j)] \| &= \| \nabla T(i,j) \| \\ &= \sqrt{G_x(i,j)^2 + G_y(i,j)^2} \\ &\approx |G_x(i,j)| + |G_y(i,j)| \end{aligned} \quad (11)$$

ここで, $G[T(i,j)]$ はテンプレート画像 $T(i,j)$ の Gradient である．$G_x(i,j)$ 及び $G_y(i,j)$ は, それぞれ座標 (i,j) における x 方向及び y 方向の Gradient の値を表す．$G_x(i,j)$ 及び $G_y(i,j)$ は, 以下の式 (12), 式 (13) により算出される．

$$G_x(i,j) = T(i+1,j) - T(i,j) \quad (12)$$

$$G_y(i,j) = T(i,j+1) - T(i,j) \quad (13)$$

はじめに，y 軸方向に関して，部分領域を設定するための分割座標値を求める．Gradient Magnitude を x 軸方向に射影し，射影値 $H_y(j)$ を得る．

$$H_y(j) = \sum_{i=0}^{M-1} \| G[T(i,j)] \| \quad (14)$$

得られた射影値 $H_y(j)$ の総計 SH は以下の通りである．

$$SH = \sum_{j=0}^{N-1} H_y(j) \quad (15)$$

y 軸方向に関して，分割レベル l における部分領域を設定するための u 番目の分割座標値 $sp_y^l(u)$ は以下の式 (16) を満す値として求められる．

$$\sum_{j=sp_y^l(u-1)}^{sp_y^l(u)-1} H_y(j) = \frac{SH}{2^l} \tag{16}$$

ここで，$sp_y^l(0) = 0$，$sp_y^l(2^l) = N$，及び $0 \leq u \leq 2^l$ とする．具体的には，$j = 0(sp_y^l(0))$ から射影値を積算していき，積算値が $SH/2^l$ を満した座標値を $sp_y^l(1)$ として設定する．続いて $j = sp_y^l(1)$ から改めて射影値を積算していき，再び積算値が $SH/2^l$ を満した座標値を $sp_y^l(2)$ とする．これを $u = 2^l$ まで繰り返し，分割レベル l に対して総数 2^l の座標値を獲得する．

次に，x 軸方向に関する分割座標値を求める．なお，適応的領域分割は，上述した y 軸方向に関する操作を x 軸方向においても同様に行うことで実現される．一方で，Brick Parititioning では，x 軸方向の分割座標値は，y 軸方向に関して求めた分割座標値により設定される各部分領域ごとに計算する．すなわち，$sp_y^l(u-1)$ 及び $sp_y^l(u)$ により得られる部分領域内における Gradient Magnitude を y 軸方向に射影し，射影値 $H_x^u(i)$ を得る．

$$H_x^u(i) = \sum_{j=sp_y^l(u-1)}^{sp_y^l(u)-1} \parallel G[T(i,j)] \parallel \tag{17}$$

得られた射影値 $H_x^u(i)$ の総計 SH^u は以下の通りである．

$$SH^u = \sum_{i=0}^{M-1} H_x^u(i) \tag{18}$$

x 軸方向に関して，分割レベル l，$sp_y^l(u-1)$ 及び $sp_y^l(u)$ により設定される部分領域内における分割座標値 $sp_x^l(u,v)$ は以下の式 (19) を満す値として得られる．

$$\sum_{i=sp_x^l(u,v-1)}^{sp_x^l(u,v)-1} H_x^u(i) = \frac{SH^u}{2^l} \tag{19}$$

ここで，$sp_x^l(u,0) = 0$，$sp_x^l(u,2^l) = M$，及び $0 \leq v \leq 2^l$ とする．y 軸方向の分割座標値と同様に，射影値を積算していき，積算値が $SH^u/2^l$ を満すごとに座標値を獲得する．ある分割座標値 $sp_y^l(u-1)$ 及び $sp_y^l(u)$ により設定されている部分領域につき，x 軸方向の分割座標値は 2^l だけ得られるため，分割レベル l においては，その総数は 2^{2l} となる．$sp_x^l(u,v)$ 及び $sp_y^l(u)$ によって分割された部分領域間は同程度の Gradient Magnitude を有しており，等分割された領域や，適応的領域分割による領域から計算される距離下限値よりも大きな値が期待でき，より効率的なウィンドウの評価が可能となる．分割レベル 1 から 3 それぞれについて，等分割，適応的領域分割及び Brick Partitioning により領域分割を行った例を図 4 に示す．図 4 より，Brick Partitioning による領域分割は，等分割や適応的領域分割よりも，画像が複雑な箇所は小さい領域が，一様な箇所は大きい領域が割り当てられていることがわかる．

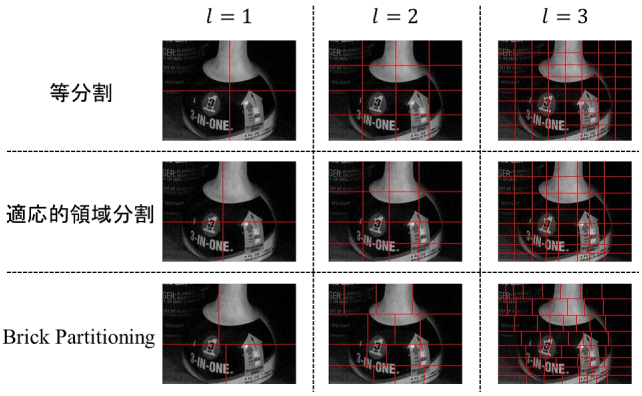

図 4 各領域分割法によるテンプレート分割例
Fig. 4 Example of template partitioning results by each method.

4. 実験

提案手法の有効性を確認するため，従来手法との比較実験を行った．本論文で比較する従来手法は FS[1], ABPMSE+ITE[8], WUA[9] とした．提案手法及び WUA については，距離下限値の計算における領域分割は，等分割 (Proposed, WUA)，適応的領域分割 (ABPProposed, ABPWUA) 及び Brick Partitioning(BrickProposed, BrickWUA) を適用した．なお，距離下限値の算出にはインテグラルイメージ [10] を用いる．提案手法におけるウィンドウ抽出パラメータ S は，すべてのテンプレートにおいて $S = 8$ とした．実験で用いたシステム及び開発環境は，CPU:Intel(R)Xeon(R)X5650 2.67GHz，Mem-

表 1 実験データ詳細
Table 1 Description of experimental data.

Template	Reference size	Template size	Correct point (x,y)
Lenna1	512×512	128×128	(231, 244)
Lenna2	512×512	64×64	(96, 128)
Object1	800×600	160×120	(240, 60)
Object2	800×600	200×150	(440, 405)
Bread1	800×600	100×75	(100, 150)
Bread2	800×600	160×120	(320, 360)
Face1	896×592	64×64	(118, 305)
Face2	896×592	400×300	(150, 515)
Cabriolet1	896×592	112×74	(303, 296)
Cabriolet2	896×592	224×148	(672, 148)
SUV1	896×592	112×74	(516, 385)
SUV2	896×592	224×148	(448, 148)
Road1	1760×1168	224×148	(775, 980)
Road2	1760×1168	400×300	(280, 575)
Yard1	1760×1168	200×150	(1065, 900)
Yard2	1760×1168	400×300	(150, 515)

表 2 各手法の処理時間 [ms] と速度比
Table 2 Computation time [ms] and rate of computation time(set FS to 100%).

Template	FS		ABPMSE+ITE		WUA		ABPWUA		BrickWUA		Proposed		ABPProposed		BrickProposed	
	Time	%	Time	%	Time	%	Time	%	Time	%	Time	%	Time	%	Time	%
Lenna1	4551	100.00	37	0.81	93	2.04	89	1.95	87	1.91	34	0.74	32	0.70	27	0.59
Lenna2	1643	100.00	68	4.13	130	7.91	122	7.42	121	7.36	40	2.43	39	2.37	34	2.06
Object1	10441	100.00	56	0.53	182	1.74	170	1.62	172	1.64	62	0.59	60	0.57	55	0.52
Object2	14009	100.00	133	0.94	172	1.22	169	1.20	168	1.19	61	0.43	60	0.42	50	0.35
Bread1	5075	100.00	143	2.81	224	4.41	211	4.15	179	3.52	97	1.91	101	1.99	88	1.73
Bread2	10270	100.00	120	1.16	204	1.98	188	1.83	182	1.77	61	0.59	54	0.52	53	0.51
Face1	3044	100.00	77	2.52	147	4.82	148	4.86	148	4.86	50	1.64	37	1.21	33	1.08
Face2	15692	100.00	65	0.41	114	0.72	105	0.66	101	0.64	36	0.22	34	0.21	33	0.20
Cabriolet1	6236	100.00	105	1.68	251	4.02	252	4.04	243	3.89	82	1.31	85	1.36	84	1.34
Cabriolet2	17333	100.00	70	0.40	197	1.13	179	1.03	165	0.95	70	0.40	56	0.32	53	0.30
SUV1	6200	100.00	108	1.74	260	4.19	236	3.80	244	3.93	80	1.29	68	1.09	62	1.00
SUV2	17032	100.00	108	0.63	189	1.10	184	1.08	178	1.04	85	0.49	64	0.37	57	0.33
Road1	87265	100.00	279	0.31	527	0.60	521	0.59	522	0.59	221	0.25	203	0.23	203	0.23
Road2	242396	100.00	265	0.10	396	0.16	402	0.16	401	0.16	130	0.05	126	0.05	121	0.04
Yard1	80851	100.00	335	0.41	650	0.80	622	0.76	607	0.75	170	0.21	169	0.20	168	0.20
Yard2	240544	100.00	285	0.11	465	0.19	445	0.18	432	0.17	177	0.07	180	0.07	166	0.06
Average	47661	100.00	141	0.29	263	0.55	253	0.53	247	0.51	91	0.19	86	0.18	81	0.16

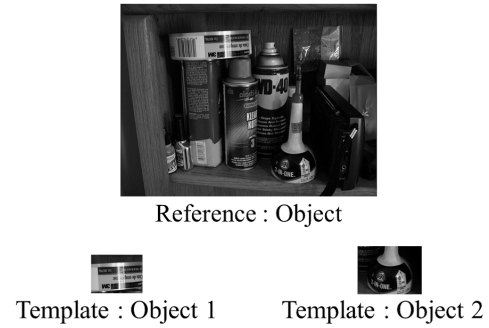

図 5 検索対象画像及びテンプレート画像
Fig. 5 Reference image and corresponding templates.

ory:64GB，Compiler:g++4.4.6(-O3 オプション使用) である．本実験では “ Lenna ” 及び Caltech の画像データセット [11] より選択した画像を検索対象画像として使用した．各画像は RGB からグレースケールに変換した．テンプレート画像は，各画像の二つの異なる位置から抽出した．また，各テンプレート画像には，-30dB のガウシアンノイズを付加した．従って，検出位置で求められる閾値も $\theta > 0$ となる．検索対象画像及び，テンプレート画像の詳細を表 1 に示す．また，図 5 に，使用した検索対象画像及びテンプレートの例を示す．

表 1 に示す実験データについて，提案手法，FS，ABPMSE+ITE 及び WUA による探索を行い，処理時間を測定した．探索処理時間及び FS との速度比を表 2 に示す．探索処理時間の単位は ms である．表 2 より，提案手法である Proposed，ABPProposed 及び BrickProposed の処理時間が大幅に削減できている．Average においては，BrickProposed が最高速であり，FS の 0.16%，続いて ABPProposed の 0.18%であった．また，従来手法で最も高速である ABPMSE+ITE と比較すると，BrickProposed は平均して ABPMSE+ITE の 57.4%の速度で処理を行えることがわかった．さらに，Road2 及び Yard2 の探索時間はそれぞれ FS の 0.04%，0.06%であった．これらは比較的テンプレート及び検索対象画像処理のサイズが大きい組み合わせであり，より多くの計算量を削減できたと考えられる．領域分割方法による比較を行うと，BrickProposed は，Proposed の 89.0%の速度，ABPProposed の 94.1%の速度で処理を行えることがわかった．また，WUA に関しては，BrickWUA は WUA の 93.9%の速度，ABPWUA の 97.6%の速度であった．

次に，提案手法の処理量削減の最大の要因と考えられる初期閾値推定に関する結果を示す．図 6 は，テンプレート Face2 における探索処理中にソート処理を実施する手法，すなわち WUA 及び提案手法の処理時間の割合である．単位は ms である．処理時間全体を計算時間 (Calculation)，整列時間 (Sorting)，その他 (Others) に分割した．計算時間とは，距離下限値及び距離値計算における加減算にかかる時間を表している．整列時間は，WUA 探索における，最小距離下限値を選び出すためのウィンドウの整列にかかる時間，その他には積分画像の計算や，領域分割を行うための分割座標値の決定などの時間が含まれている．図 6 から WUA に比べ，提案手法は大幅に整列時間を削減することによって，探索処理全体を高速化していることがわかる．WUA が一度の探索で検索対象画像上のウィンドウ全体を整列し，最小距離下限値を選び出す手法に対して，提案手法は初期閾値推定のみに目的を限定して，第 1 段階及び第

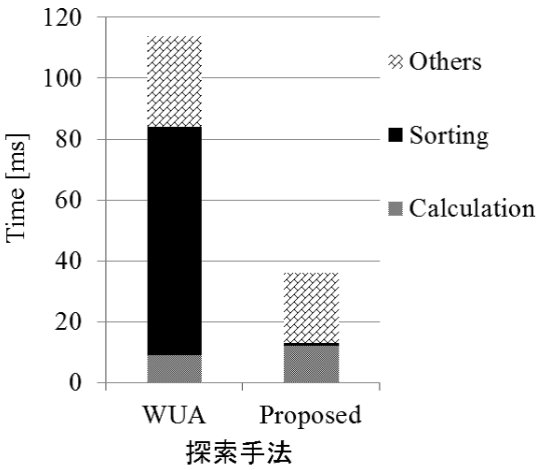

図 6 処理時間における計算内容の内訳
Fig. 6 The details of the Computation time.

2 段階で少量のウィンドウのみ整列しているため，大幅に削減できたと考えられる．

次に，WUA による初期閾値推定を行う提案手法に関して，ウィンドウを選択する間隔を示すパラメータ S とテンプレートサイズの大きさの関係を図 7 に示す．横軸をパラメータ値，縦軸を処理時間とし，単位はそれぞれ pixel，ms である．テンプレートサイズごとに，パラメータの値を変化させたときの処理時間の推移を表している．実験に用いた検索対象画像は Lenna，テンプレートは 64×64 画素から，384 画素までの異なる大きさを位置 $(96, 128)$ から抽出した．表 7 から，64×64 画素及び 384×384 画素のサイズでは $S = 8$，96×96 画素，128×128 画素及び 256×256 画素のサイズでは $S = 16$，192×192 画素では $S = 32$ で

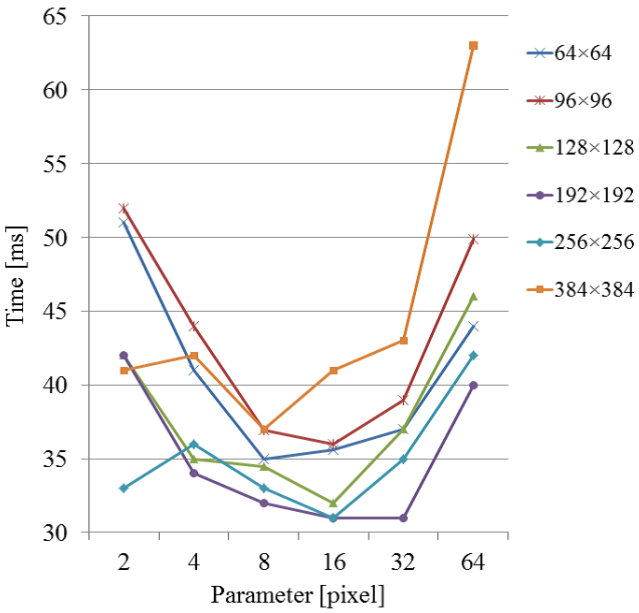

図 7 ウィンドウ選択パラメータとテンプレートサイズの関係
Fig. 7 Relationship between the window skipping parameters and template size.

最も探索時間が短い．また，全てのテンプレートサイズに関して，本実験の最小値である $S = 2$ 及び最大値 $S = 64$ では処理時間が著しく低下した．テンプレートのサイズに関わらず，ウィンドウの選択間隔 S の値は 8 ～ 32 で高速に探索処理を行えていることがわかる．

5. まとめ

本論文では，FS と同等の探索精度を保証しつつ，高速なテンプレートマッチングを行う手法として，WUA による初期閾値推定処理及び Brick Partitioning による領域分割法を提案した．検索対象画像上のウィンドウから少数を選択し，WUA による探索を行うことで，本探索を行う前に極めて小さい閾値を獲得し，処理量を削減することができた．また，Brick Partitioning による領域分割を行うことで，さらに処理時間を削減できることも確認した．実験では，最大で FS の 0.05%の速度という高速な処理結果を得られた．また，提案手法におけるウィンドウ選択間隔パラメータは，テンプレート画像のサイズに依存せず，一定間隔内で高速な探索結果が得られることがを確認できた．

参考文献

[1] 尾上守男（編）:「画像処理ハンドブック」，晃光堂 (1987).
[2] D.I.Barnea and H.F.Silverman : “A class of algorithm for fast digital image registration”, IEEE Trans.Comput.,Vol.C-21, No.2, pp.179-186 (1972).
[3] 関根優年，金丸隆志，伊藤光 : “多重照合による顔領域の特定”，信学論，Vol.J86, No.9, pp.969-973 (2003).
[4] 村瀬洋，V.V.Vinod : “局所色情報を用いた高速物体探索-アクティブ探索-”，信学論，Vol.J81，No.9,pp.2035-2042 (1998).
[5] 川西隆仁，久野和樹，木村昭吾，黒住隆行，柏野邦夫，高木茂 : “サブテンプレート間距離を用いた適応的ウィンドウスキッピングによる高速テンプレートマッチング法”，電子情報通信学会論文誌，Vol.J88, No.8, pp.1389-1397 (2005).
[6] W.Li, E.Salari : “Successive elimination algorithm for motion estimation”, IEEE Trans.ImageProcess.,Vol.4, No.1, pp.105-107 (1995).
[7] X.Q.Gao, C.J.Duanmu, C.R.Zou : “A multilevel successive elimination algorithm based on the block sum pyramid”, IEEE Trans.ImageProcess.,Vol.9, No.3, pp501-504 (2000).
[8] 森稔，柏野邦夫 : “適応的領域分割と初期しきい値推定によるテンプレートマッチングの高速化”, 信学論, Vol.J94, No.5, pp.881-892 (2011).
[9] Yong-Sheng, et al : “Fast Block Matching Algorithm Based on the Winner-Update Strategy”, IEEE Trans.ImageProcess, Vol.10, No.8, pp.1212-1222 (2001).
[10] Jik-Han Jung, et al : “A Novel Template Matching Scheme for Fast Full-Search Boosted by an Integral Image”, IEEE single processing letters, Vol.17, No.1,pp.107-110 (2010).
[11] http://www.vision.caltech.edu/archive.html