論　文

高速一般化ハフ変換——相似変換不変な任意図形検出法——

木村　彰男†　　渡辺　孝志†

Fast Generalized Hough Transform——Rotation, Scale and Translation Invariant Detection of Arbitrary Shapes——

Akio KIMURA† and Takashi WATANABE†

あらまし　入力画像からある特定の図形を相似変換不変な形で検出する問題は，画像認識における基本的な研究課題である．Ballardによって提案された一般化ハフ変換（GHT）は，この種の問題に対処するためのプロトタイプとして知られており，その耐ノイズ性，処理の柔軟さ，アルゴリズムの簡明さ，といった点で優れた手法である．しかしGHTには，輪郭点ごとに正確なこう配情報を必要とする，処理時間がかかる，といった問題点もある．そこで本論文では，これらの問題点に対処可能な新しい手法として，高速一般化ハフ変換（FGHT: Fast Generalized Hough Transform）を提案する．FGHTでは，チェック点の導入で投票そのものの信頼性を高め，より高精度な図形の検出を実現している．更に，線分近似を併用することで，従来手法に比べて大幅な処理速度の向上を実現している．提案手法の有効性を検証するために行った評価実験では，良好な検出結果が得られた．

キーワード　図形認識，一般化ハフ変換，CTT，線分近似

1. まえがき

人間のもつ視覚機能を計算機上で実現しようとする画像認識問題は，画像処理の中心的な問題として古くから注目を浴びている[1],[2]．特に，入力画像からある特定の図形を相似変換不変な形で検出する問題は，画像認識問題の中でも基本的かつ重要な研究課題であり，従来から多くの研究がなされてきた．このような相似変換不変な形での図形検出を意図した手法としては，フーリエ・メリン変換[4]，複素対数変換[5]，フーリエ記述子[6]，一般化ハフ変換（GHT）[7]，Geometric Hashing[8]などがあるが，実用上はそれぞれが問題を抱えており，満足できる状況にはない．例えば，雑音の影響や対象物同士の重なり・隠ぺいによる情報の欠落・変形の影響などによって，画像から得られる情報が不完全となった場合には，フーリエ・メリン変換，複素対数変換，フーリエ記述子はいずれも適用不可能となってしまう．

その中で，GHTやGeometric Hashingは，この種の劣化を伴う複雑な画像に対しても適用可能であり，得られる情報が不完全な場合でも，ある程度安定に図形検出を行うことが可能な手法である．特にGHTは，その耐ノイズ性，処理の柔軟さ，アルゴリズムの簡潔さ，といった面で優れた手法であり，多くの研究者の注目を集め，多数の研究がなされてきた．しかしGHTには，輪郭点ごとに正確なこう配情報を必要とする，検出対象物が相似変換を受けている場合には処理時間が爆発的に増大する，という根本的な問題点もあった．

最近，Dufresneら[9]によって，高速化の観点からGHTを改良したChord-Tangent変換（CTT）が提案されている．CTTは，GHTの問題であった処理時間を改善したばかりでなく，Geometric Hashing的な対象物の識別力も併せもった手法として構成されており，相似変換不変な図形検出問題に対する新たな展開を与えた手法であるといえる．しかしながら，CTTにおいても，検出対象物体の正確なこう配情報が画素単位に必要である，という制約条件は残されたままであり，実用面から見るとまだ問題が多い．

以上のような観点から，本論文では，GHTやCTTのもつ問題点に対処可能な新しい手法として，高速一般化ハフ変換（FGHT: Fast Generalized Hough Transform）を提案する．FGHTでは，後述するチェック点の導入で投票自体の信頼性を高め，より高精度な図形

† 岩手大学工学部情報工学科，盛岡市
Faculty of Engineering, Iwate University, 4-3-5 Ueda, Morioka-shi, 020-0066 Japan

の検出を実現している．更に，線分近似を併用することで，GHT や CTT と比べて大幅にその処理速度が改善されている．以下，2. では FGHT の原型となった CTT の基本原理とその問題点について論じ，3. で FGHT の詳細とその具体的な処理手順について述べる．そして 4. では，提案手法の有効性を検証するために行った評価実験の結果について述べる．

2. CTT の基本原理とその問題点

前述したように，CTT は GHT の高速化法として提案された手法であり，基本的な考え方は GHT を踏襲したものとなっている．以下，CTT の概略を説明するが，詳細は文献[9]を参照されたい．

2.1 C 表生成（検出対象図形の記述）

処理対象画像は 2 値の輪郭線画像とし，それぞれの輪郭点におけるこう配情報は既知であるとする．また，検出対象図形は M 個の輪郭点から構成される 2 次元図形とする（以下，この図形をテンプレート T と呼ぶ）．今，テンプレート T の輪郭線上から任意の 2 点 $a(a_x, a_y), b(b_x, b_y)$ を選び，それぞれの点におけるこう配情報から（それと直交する）接線方向を求め，それらの接線を延長した線と，選択した 2 点を結ぶ線（以下，これをコードと呼ぶ）で囲まれる三角形を考えると，この三角形の内角 δ_1, δ_2 は相似変換に関して不変な特徴となる（図 1）．

CTT では，この性質を利用して図形検出を行うが，まずは以下の手順に従って「C 表」を作成する．

[手順 1] テンプレート T の内部に，任意の参照点 R を定める．

[手順 2] 輪郭線上から任意に 2 点を選び，次の各特徴量を算出する（図 1 参照）．

- コード長 L_c

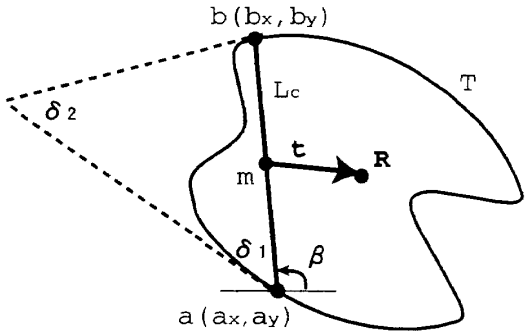

図 1 CTT で用いられる検出対象図形の不変特徴
Fig. 1 The invariant characteristics of a shape to be detected with the CTT.

- コードの角度 β．但し β は水平座標軸からの角度を表しており，$\beta \in [0, 2\pi)$ である．
- 二つの接線とコードで作られる三角形の内角 δ_1, δ_2
- コードの中点 $m = (x_m, y_m)$ から参照点 R に向かうベクトル $\mathbf{t} = (t_x, t_y)$

これらの特徴量を δ_1, δ_2 で張られる 2 次元リンクリスト（C 表）に登録する．

[手順 3] すべての 2 点の組合せ（${}_M C_2$ 個存在する）について C 表への登録を行う．但し，コード長 L_c の短いものは誤差が大きいため，適当なしきい値 L_{th} を設けて，$L_c \leqq L_{th}$ となる組合せについては表への登録処理を省略してよい． □

2.2 CTT の投票方法

CTT における投票手順は，以下のとおりである．

[手順 1] 回転，伸縮，移動量を表す 4 次元配列 $A(\theta, \kappa, x, y)$ を用意し，全配列要素を 0 にセットする．

[手順 2] 入力画像 I から新たに 2 点を選択し，この 2 点から作られるコードの特徴量，すなわち，${L_c}^I, {\beta}^I, {\delta_1}^I, {\delta_2}^I, {m}^I = ({x_m}^I, {y_m}^I)$ をそれぞれ算出する．

[手順 3] $\delta_1 = {\delta_1}^I$ かつ $\delta_2 = {\delta_2}^I$ となる δ_1, δ_2 をキーとして C 表を参照し，対応するコードのパラメータ L_c, β, t_x, t_y を読み出す．対応するコードがなかった場合は手順 2 に戻る．

[手順 4] 以下に示す式から相似変換パラメータ $(\theta, \kappa, x_r, y_r)$ を算出し，対応する配列要素 $A(\theta, \kappa, x_r, y_r)$ に 1 を加算（投票）する．

$$\theta = \beta^I - \beta \tag{1}$$

$$\kappa = {L_c}^I / L_c \tag{2}$$

$$\begin{pmatrix} x_r \\ y_r \end{pmatrix} = \kappa \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} t_x \\ t_y \end{pmatrix} + \begin{pmatrix} {x_m}^I \\ {y_m}^I \end{pmatrix} \tag{3}$$

なお，C 表内に δ_1, δ_2 で登録された特徴量が複数個ある場合には，そのすべてについて上式で相似変換パラメータを算出し，投票処理を行う．

[手順 5] I 内のすべての 2 点対に対して上記の手順を繰り返し，最後に A からしきい値以上の投票値をもつ要素を選んで検出パラメータとする． □

CTT においては，形状の異なった複数の対象図形を同時に識別・検出することも可能である（この場合，C 表中に図形を識別するための識別子を同時に登録し

ておき，投票数を識別子ごとに集計するように手順を変更すればよい)．この意味で，CTT は「対象物の検出」能力だけでなく，Geometric Hashing [8] のような「対象物の識別」能力を併せもっている，と言ってよい．

2.3 CTT の問題点

以上のように，CTT では輪郭点を二つ組み合わせて得られる特徴量を算出することで，検出したいすべての相似変換パラメータを（1 回の投票処理で）求めることができるため，GHT の課題であった「検出対象が相似変換を受けると，処理時間が爆発的に増大する」という問題点を見事に克服したと言える．

しかし，上述のアルゴリズムから明らかなように，CTT は検出対象物体の輪郭線画素ごとに正確なこう配情報を必要とするという大きな制約条件がある．つまり，図 2 のようにこう配情報が不正確な場合には，誤って δ_1, δ_2' が算出されるため，C 表参照に誤りを生じ，結果として正しい位置への投票が行われない．例えば，異なった照明条件のもとで得られた画像に CTT を適用した場合には，その検出能力は大きく変化すると考えられる．なぜなら，一般にカメラ入力された濃淡画像から正確にこう配情報を得ることは困難であると考えられるからである（後に，これを実験結果で示す)．

また，CTT の原論文中では，組み合わせる 2 点の距離を "sufficently large" にとる，すなわち，L_{th} を十分大きくとることで処理の高速化を図ったとされているが，処理対象物が「輪郭線画素ごと」である限り，雑音を含んだ複雑な実画像に対しては処理時間がまだ十分ではないと思われる．

以上，CTT の問題点を述べたが，実用レベルのパターン検出法を構成するには，検討すべき課題が多く残っていると言える．

3. 高速一般化ハフ変換（FGHT)

3.1 接線情報の不正確さに対する投票方式の改善

既に述べたように，CTT に代表される GHT 的手法における最大の問題点は，輪郭点でのこう配情報（接線情報）が一般には不正確な場合が多く，そのために必ずしも有効に働かない，という点である．そこで，本論文では，そのような接線情報の不正確さを考慮に入れた新しい投票方式を提案する．

まず，CTT で用いられている不変特徴 δ_1, δ_2 の定義を次のように変更する．すなわち，選択した二つの輪郭点における接線と，その 2 点を結ぶ線分で構成される三角形を考え，輪郭点の 2 点における三角形の内角をそれぞれ δ_a, δ_b とする（図 3)．そして，**2.1** で述べた手順に従って特徴量を算出し，あらかじめ拡張 C 表（表 1 参照）に登録しておく．

今，入力画像のある二つの輪郭点（とその接線）からなる三角形を考え，その 2 点における三角形の内角がそれぞれ ${\delta_a}^I, {\delta_b}^I$ であったとする．このとき

$$|{\delta_a}^I - \delta_a| \leqq \Delta_{th} \text{ かつ } |{\delta_b}^I - \delta_b| \leqq \Delta_{th} \qquad (4)$$

または

$$|{\delta_b}^I - \delta_a| \leqq \Delta_{th} \text{ かつ } |{\delta_a}^I - \delta_b| \leqq \Delta_{th} \qquad (5)$$

を満たすような δ_a, δ_b をキーとして拡張 C 表の参照を行い，それらの登録パラメータと式 (1)〜(3) を用いて相似変換パラメータを算出し，投票する．ここで，Δ_{th} は角度許容誤差を示すしきい値である．

これにより，角度誤差 Δ_{th} 内で可能性のあるすべての変換パラメータ（配列要素）について投票が行われるので，接線情報の不正確さを考慮したアルゴリズムとなる．

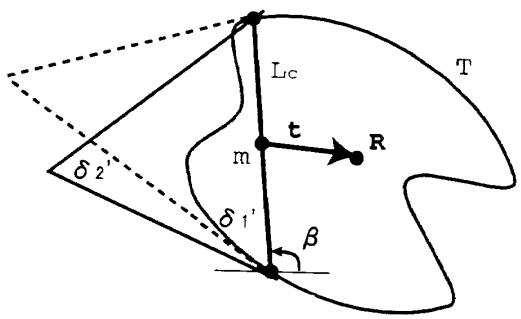

図 2 輪郭線のこう配情報が正しく求められない場合
Fig. 2 Illustration of the case that a gradient information of contour line is not exact.

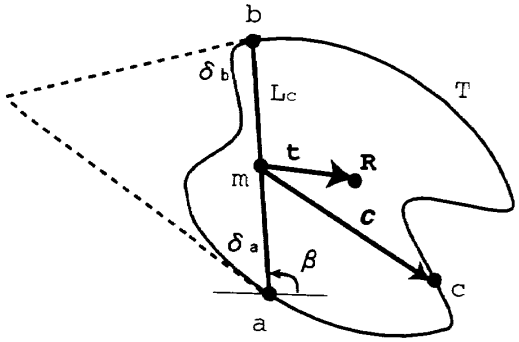

図 3 FGHT におけるテンプレートの記述
Fig. 3 Description of a template pattern in the FGHT.

表1 拡張C表の構造
Table 1 Contents of extended C-table.

番号	不変特徴	チェック点		コード		参照点へのベクトル
		方向角	ベクトル	長さ	角度	
1	$\delta_a(1), \delta_b(1)$	$\tau_c(1)$	$(c_x(1), c_y(1))$	$L_c(1)$	$\beta(1)$	$(t_x(1), t_y(1))$
2	$\delta_a(2), \delta_b(2)$	$\tau_c(2)$	$(c_x(2), c_y(2))$	$L_c(2)$	$\beta(2)$	$(t_x(2), t_y(2))$
...	...	...	...	...	...	...

3.2 チェック点による投票の制限

3.1で述べた投票方式は，可能性のあるところすべてに投票する，という考え方に基づくものであるので，「入力画像から得られた $\delta_a{}^I, \delta_b{}^I$ が拡張C表内のどの δ_a, δ_b に対応するか」を正確に求めることは困難である．つまり，式(4)，(5)を満たす δ_a, δ_b が複数存在すると，式(1)〜(3)で算出される相似変換パラメータも複数生じてしまう．このことは，余分な投票，つまり誤投票が増大するということを意味しており，投票自体の信頼性を下げてしまうことになる．

そこで，この誤投票を抑制するために，次のような工夫を施す．すなわち，各輪郭点ペア（コード）に対して，その存在の確からしさをチェックするために，第3番目の輪郭点をテンプレート上から選び，その位置ベクトル $\mathbf{c} = (c_x, c_y)$ を参照点と同様な形式で記述して拡張C表に登録しておく（図3，表1参照）．今，この第3番目の点をチェック点と呼ぶことにすると，入力画像においてもこのチェック点の存在が確認できる輪郭点ペアのみを投票対象とすればよいので，投票の信頼性は飛躍的に高まると期待できる．

テンプレート上にチェック点を1点設けるということは，そこから得られる不変情報が，単に輪郭点ペアを選択して得られる不変情報（つまり δ_a, δ_b のみ）に比べて，より大局的なものである（すなわち，テンプレート全体にわたった特徴である）ということを意味する．従って，チェック点の個数を増やしていくと，究極的にはテンプレートの形状そのものを示す特徴量が得られることになり，チェック点を多くとればとるほど投票の信頼性は高まっていく，と言える．

しかし，チェック点は多く取りすぎると輪郭線が欠落している場合に対処できず，更に投票自体の速度も落としてしまうことになるため，その選択方法が極めて重要となる．本手法では，高速化を考慮してチェック点は1点のみとし，その選択法は，「$\mathbf{c}$ のノルム $||\mathbf{c}||$ が，しきい値 c_{len} 以上となっているものからランダムにとる」と約束する．なお，投票時には，チェック点における接線情報も比較することでより信頼性を高めるものとする．

3.3 線分近似の併用による処理の高速化

ここで，処理時間について考えてみる．チェック点を設けたことで無駄な投票が行われなくなるので，投票回数そのものは削減することができる．しかし，入力画像において処理対象となる点の数は変わっていないので，このままではチェック点の計算の分だけ処理時間が余計にかかってしまうことになる．従って，高速化のためには，投票回数の削減だけではなく，投票処理全体の効率化が必要である．

このような観点からテンプレート画像と入力画像における輪郭線を区分的に線分近似して投票の効率化を図った例が，渡辺ら[10]によって示されている．本論文でもこの考え方を採用するものとした．つまり，あらかじめテンプレート画像と入力画像において輪郭線の線分近似を行い，線分の長さがしきい値 s_{th} 以上のものを選択してその中点を処理対象輪郭点とする，という考え方である．

これによって，「輪郭点ごとの接線方向」を「得られた線分ごとの傾き」で置き換えて考えることができ，処理対象とする輪郭点ペアの数を激減させることが可能となる．更に，線分近似によってある程度長い線分を抽出するということは，対象パターンにおける（相似変換に強いという意味での）安定な情報を引き出すことにつながっており，線分近似によって処理対象輪郭点を重要なものだけに絞る，という効用もある．すなわち，線分近似の併用は効率的で安定な投票処理を行うために非常に有用であると考えられる．

このためには，輪郭線を線分近似する手法が必要である．輪郭線が途切れなく連続している場合には多くの方法が知られている[3]が，不連続な劣化輪郭線画像には適用できない．そこで本研究では，直線検出手法として広く使用されている標準的ハフ変換（SHT）を使用することとした[11]．

3.4 FGHTのアルゴリズム

以上に述べたFGHTのアルゴリズムをまとめる．但し，ここでは，テンプレートの濃淡画像と入力濃淡画

像はいずれも微分・2値化・細線化の処理によって輪郭線画像となっているものとし，それらの輪郭線画像をそれぞれ T, I で表すことにする．

〈線分近似処理〉

[手順 1] T と I を適当な大きさのブロック $(B \times B)$ に分割し，各ブロックを SHT によって線分近似する(詳細手順は文献[11]によるが，ここでは局所的な線分近似が必要なので画像の階層化は取り入れない)．但し，処理を行う際は，ブロックを上下左右に ΔB だけ拡張するものとする．この結果得られた線分画像をそれぞれ線分の集合 $\{T_{0i}\}$, $\{I_{0j}\}$ で表す．

[手順 2] $\{T_{0i}\}$ と $\{I_{0j}\}$ から，長さが s_{th} 以上の線分を抽出し，それらを改めて $\{T_i | i = 1, 2, \cdots, M\}$, $\{I_j | j = 1, 2, \cdots, N\}$ とする． □

〈拡張 C 表の生成〉

あらかじめ T の内部に任意の参照点 $R = (R_x, R_y)$ を設けておく．

[手順 1] 図 3 のように，$\{T_i\}$ から，コード長 L_c が $L_c \geq L_{th}$ となるような二つの線分 T_a, T_b を選択し，以下の特徴量を求める．

- コードの角度 β $(\in [0, 2\pi))$
- 2 線分とコードで作られる三角形の内角 δ_a, δ_b
- コードの中点 $m = (x_m, y_m)$ から参照点 R へのベクトル $\mathbf{t} = (t_x, t_y)$

但し，コード長 L_c は，選択した線分 T_a, T_b の中点 $(x_a, y_a), (x_b, y_b)$ の間の距離である．

[手順 2] $\{T_i\}$ から T_a, T_b 以外にもう一つの線分 T_c を選択し，点 m から線分 T_c の中点 $c = (x_c, y_c)$ へのベクトル $\mathbf{c} = (c_x, c_y)$ を求める．ここで，T_c は，ノルム $||\mathbf{c}||$ が c_{len} 以上となるものからランダムに選ぶ．この点 c を点 a, b に関するチェック点とする．

[手順 3] 得られた特徴量 $L_c, \beta, \delta_a, \delta_b, t_x, t_y, c_x, c_y, \tau_c$ を拡張 C 表に登録しておく (表 1 参照)．ここで，τ_c はチェック点の線分 T_c の方向角である．これを T_i 内の可能なすべての線分対について繰り返す． □

〈検出手順〉

[手順 1] 回転，伸縮，移動量を表す 4 次元配列 $A(\theta, \kappa, x, y)$ を用意し，全配列要素を 0 にセットする．

[手順 2] $\{I_j\}$ からコード長 ${L_c}^I$ が ${L_c}^I \geq {L_{th}}^I$ となるような二つの線分 I_a, I_b を選択し，そのコードと 2 線分で構成される三角形から，上に述べたそれぞれの特徴量 ${\beta}^I, {\delta_a}^I, {\delta_b}^I$ を算出する．

[手順 3] 次式

$$|{\delta_a}^I - \delta_a| \leq \Delta_{th} \text{ かつ } |{\delta_b}^I - \delta_b| \leq \Delta_{th} \tag{6}$$

$$|{\delta_b}^I - \delta_a| \leq \Delta_{th} \text{ かつ } |{\delta_a}^I - \delta_b| \leq \Delta_{th} \tag{7}$$

のいずれかを満たすような δ_a, δ_b を拡張 C 表の中から抽出し，それらの表番号の集合を $\{n_k | k = 1, 2, \cdots\}$ とおく．

[手順 4] 表番号 n_k に登録されている特徴量をそれぞれ $L_c(n_k), \beta(n_k), t_x(n_k), t_y(n_k), c_x(n_k), c_y(n_k), \tau_c(n_k)$ とし，今処理対象としているコードの中点を $({x_m}^I, {y_m}^I)$ とする．すべての n_k について

$$\theta = \beta^I - \beta(n_k) \tag{8}$$

$$\kappa = {L_c}^I / L_c(n_k) \tag{9}$$

$$\begin{pmatrix} x_c(n_k) \\ y_c(n_k) \end{pmatrix} = \kappa \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} c_x(n_k) \\ c_y(n_k) \end{pmatrix} + \begin{pmatrix} {x_m}^I \\ {y_m}^I \end{pmatrix} \tag{10}$$

からチェック点 $(x_c(n_k), y_c(n_k))$ を求める．

[手順 5] I において，画素位置 $(x_c(n_k), y_c(n_k))$ の上下左右に ΔC 画素だけ広げた範囲を走査し，この範囲内に線分が存在して，その方向角 τ^I が

$$|\tau^I - \tau_c(n_k) - \theta| \leq \Delta_{th} \tag{11}$$

を満たした場合にのみ，次の手順 6 に進む．それ以外の場合は，手順 2 に戻る．

[手順 6]

$$\begin{pmatrix} x_r \\ y_r \end{pmatrix} = \kappa \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} t_x(n_k) \\ t_y(n_k) \end{pmatrix} + \begin{pmatrix} {x_m}^I \\ {y_m}^I \end{pmatrix} \tag{12}$$

から，検出対象図形の参照点が存在する推定位置 (x_r, y_r) を算出し，これと式 (8), (9) の κ, θ に対応する配列要素 $A(\theta, \kappa, x_r, y_r)$ に 1 を加算 (投票) する．

[手順 7] $\{I_j\}$ 内の可能なすべての線分対に対して上記手順を繰り返す．最後に A を走査して，しきい値以上の要素を投票値の高い順に選んで検出パラメータとする． □

4. 実験と検討

提案手法の有効性を検証するために評価実験を行った．計算には，動作クロック 150 MHz の Pentium プロセッサを使用した．

以下では，特に断りのない限り，コード長のしきい値 $L_{th} = 40$（画素），角度許容誤差 $\Delta_{th} = 10°$，投票空間の量子化幅 $\Delta\kappa = 0.1, \Delta\theta = 10°, \Delta x = \Delta y = 1$（画素）であり，線分近似の際の分割画像サイズは 32×32 画素（$B = 32$），その重なりは $\Delta B = 8$（画素）である．

なお，今回の実験では，最終的に投票空間を走査して解候補を選択する手順（3.4 の手順 7）を以下のように修正した．すなわち，選択された候補要素 (θ, κ, x, y) に対して，(θ, κ) を固定して (x, y) を中心に上下左右に ΔV 画素だけ広げた 2 次元領域中により高い投票値をもつ要素が存在している場合には，検出パラメータから除外した．これは，投票値が (x, y) 面の 2 次元空間に対して広く分布することを考慮に入れたためである．

4.1 評価実験

図 4 と図 5 に示す実画像を用い，これらの画像に対して CTT，FGHT（チェック点なし），FGHT をそれぞれ適用した場合の処理時間と検出精度を比較した．ここで，図 4 は 150×210 画素のテンプレート画像 T（図中の“＋”は参照点），図 5 は 256×256 画素の入力画像 I を表している．これらはいずれもカメラ入力された濃淡画像であるが，互いの照明条件は異なるものである．また，図 5 の入力画像は，ほぼ同じような曲率をもつ物体で構成されている．

図 6 と図 7 は，図 4 と図 5 の画像に対してそれぞれ微分・2 値化・細線化の各処理を施して得られた輪郭線画像であり，図 8 と図 9 は，図 6 と図 7 をそれぞれ FGHT で処理する際に得られた線分近似画像である．なお，図 6 と図 7 の輪郭点数はそれぞれ 592, 1368 であり，図 8 と図 9 の線分本数はそれぞれ 54,

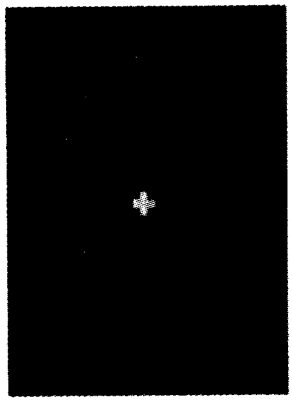

図 4　テンプレート画像
Fig. 4　Template image.

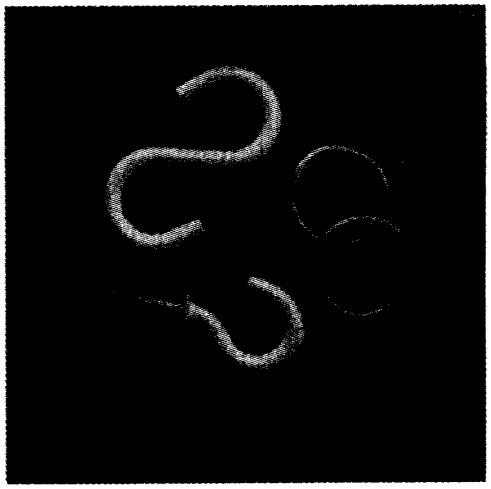

図 5　入力画像 1
Fig. 5　Tested image 1.

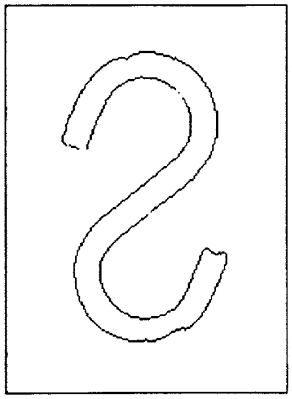

図 6　テンプレート画像（図 4）の輪郭線画像
Fig. 6　The image extracted contour lines from Fig. 4.

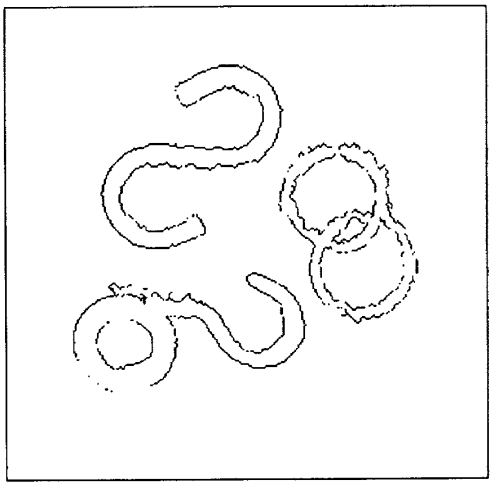

図 7　入力画像 1（図 5）の輪郭線画像
Fig. 7　The image extracted contour lines from Fig. 5.

表2 CTTとFGHTにおける投票値と処理時間の比較
Table 2 Comparison in the number of votes and CPU time for the CTT and the FGHT.

手法	総投票回数	投票値			処理時間 [s]
		1番目	2番目	正解位置	
CTT	22,089,696	16,874	15,453	10,230	246.11
FGHT チェック点なし	1,245,761	298	241(*)	241	14.87
FGHT	373,696	153(*)	61	153	20.02

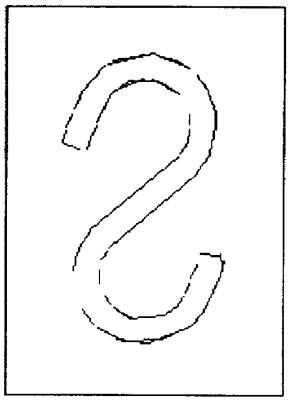

図8 テンプレート画像（図6）の線分近似画像
Fig. 8 The polygonal approximated image from Fig. 6.

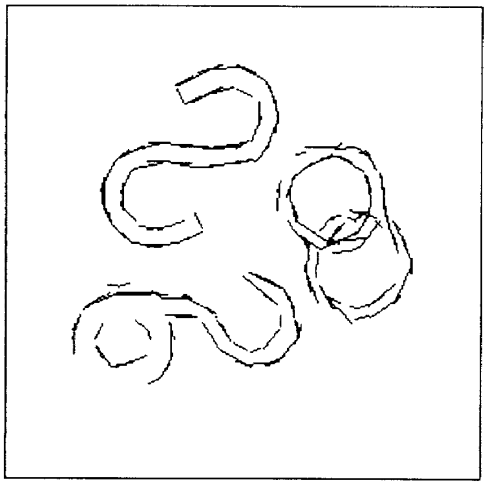

図9 入力画像1（図7）の線分近似画像
Fig. 9 The polygonal approximated image from Fig. 7.

147である．

実験に使用したパラメータは，処理対象線分を選ぶための線分長のしきい値 $s_{th} = 6$（画素），チェック点選択のための長さしきい値 $c_{len} = 20$，チェック点の存在確認範囲 $\Delta C = 3$（画素），入力画像におけるコード長のしきい値 $L_{th}{}^{I} = 10$（画素），投票空間内における解候補の確認範囲 $\Delta V = 5$（画素），検出しきい値は，CTTで3000，FGHTで50であり，CTTにおいて必要とされる輪郭点のこう配情報は濃淡画像をSobel operatorによって微分した結果から求めた．

結果を表2に示す．まず，検出に要した時間を比較すると，FGHTではチェック点のある・なしにかかわらずCTTよりも大幅に処理時間が短縮されている（約1/12）．これは，総投票回数を見ると明らかであるが，FGHTでは線分近似を用いて処理対象輪郭点を重要なものだけに絞って投票回数を削減しているため，その効果が現れたものと考えられる．

表2内の残りの部分は，各手法における検出精度を比較するために，投票空間における投票値の分布を調べたものである．ここで，表2内の「正解位置」は，A の正解位置である $(\theta, \kappa, x, y) = (220, 0.7, 100, 80)$ への投票数を示しており，「1番目」「2番目」はそれぞれ各手法で検出された A の要素の1番目と2番目の投票値を示している．CTTでは，1番目も2番目も正解とはなっておらず，正解位置への投票数とは大きな差がある．これは，2.で述べたように，照明条件が異なったために，テンプレートの輪郭点におけるこう配情報が不安定になっていることが原因と考えられる．誤って検出されてしまった結果の例を図10に示す．一方，チェック点なしのFGHT(注1)では，2番目に検出されたものは正解であった（表中では記号「*」で表している）が，1番目は誤ったものが検出されてしまっている．この原因は，検出対象物が相似変換を受けた場合に必ずしも同様の線分近似結果が得られないためと考えられる．そこで，この影響を除去するために3.で述べたチェック点を設けると，テンプレートと入力画像間における近似線分間の対応が正しく求められ，表2の最下行に示すように，1番目で正解が得られる．更に，1番目と2番目の投票数を比較すると，

(注1)：オリジナルのCTTにも線分近似を併用することで，処理時間的にはチェック点なしのFGHTとほぼ同様な時間短縮効果が期待できる．しかし，この場合はこう配情報の不確定さが考慮されていないので，厳密に言えばチェック点なしのFGHTの方が検出精度の上で良い結果を与えると考えられる．

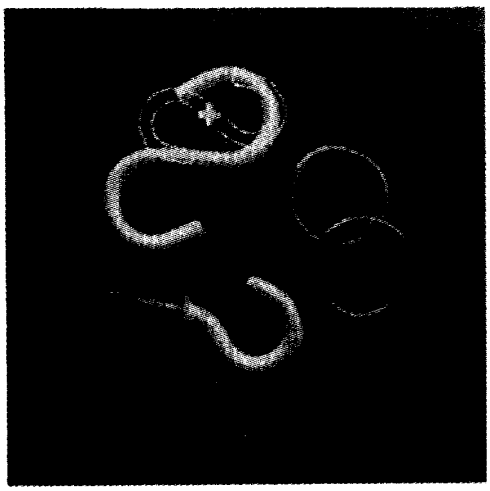

図 10　CTT による誤った検出結果例
Fig. 10　Example of the wrong detection result with the CTT.

図 12　入力画像 2
Fig. 12　Tested image 2.

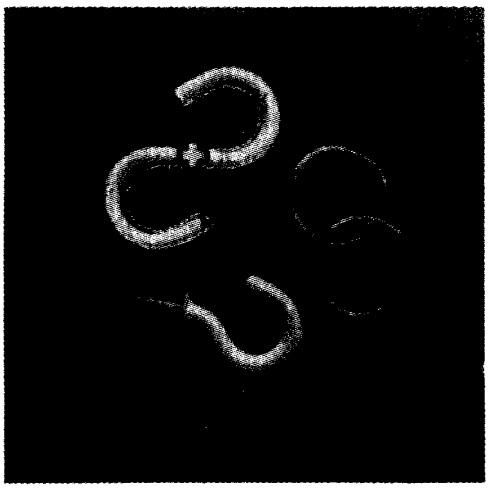

図 11　FGHT による入力画像 1（図 5）の検出結果
Fig. 11　Detection result of Fig. 5 with the FGHT.

図 13　FGHT による入力画像 2（図 12）の検出結果
Fig. 13　Detection result of Fig. 12 with the FGHT.

CTT やチェック点なしの FGHT に比べて大幅に誤投票が削減されていることがわかる．FGHT の検出結果を図 11 に示す．

最後に，図 4 のテンプレート画像を使用し，全く別の入力画像（図 12：256 × 256 画素）に対して FGHT による検出実験を行った結果を図 13 に示す．処理時間は 21.08 秒であり，検出結果は良好であった．

4.2 検　　討

実験結果より FGHT の妥当性と有効性は一応示されたものと考えるが，以下では FGHT の特徴と問題点について検討する．

FGHT における線分近似の直接的な効果は，処理対象輪郭点を重要なものに絞り込んで処理時間の短縮を果たした点であり，評価実験でもそれを確認した．更に，線分近似には，検出対象パターンをマクロ的にとらえることで輪郭線上の雑音や細かいひずみの影響を除去する，という効果もある．しかし FGHT は，アルゴリズムが輪郭線の線分近似に基づいているので，線分近似精度がパターンの検出精度に大きな影響を与える．一般に，雑音環境下では正確な線分近似を得ることはかなり難しい．そこで，この問題に対処するために FGHT ではチェック点を導入し，線分の存在を逐次確認しながら投票する方式を導入した．これは，検出対象パターンが相似変換を受けてその線分近似結果が部分的に変化してしまうような場合も有効に機能する．実験結果もそれを立証した．これとは逆に，パターンが相似変換を受けた場合でも常に安定した線分情報を抽出することが可能であれば，それらの線分だ

けを処理対象とすることでより検出精度を高めることが可能であると思われる．今回は近似線分長が s_{th} 以上のものを対象とすることで安定性を高める工夫をしたが，これらの詳細な検討は今後の課題である．また，よりロバストで高精度な線分検出法の開発も重要である．更に，チェック点の選択方法やチェック点が複数になった場合の効果も詳細に議論する必要があるが，これらについても今後の課題としたい．

5. むすび

本論文では，相似変換不変なパターン検出法として，一般化ハフ変換（GHT）を大幅に高速化した，高速一般化ハフ変換（FGHT）を提案した．FGHTの特長としては，

（1） 輪郭点におけるこう配情報の不確定さを考慮した投票方式の採用と，チェック点による投票制限によって，検出の高精度化を実現している，

（2） 線分近似の採用により，検出に要する処理時間を大幅に短縮している，

などが挙げられる．評価実験では，処理速度，検出精度，ロバスト性などの面で十分に実用的な手法となっていることが確認できた．その結果，FGHTのハードウェア化を図ることによって，リアルタイムでの相似変換不変な図形認識システムの可能性が開けたのではないかと考えている．

今後は，安定な線分情報の抽出による検出能力の向上，チェック点の効果的な選択法などについて検討を進める予定であるが，それらについては稿を改めて議論したい．

謝辞 本研究に関して御協力を頂いた本学工学部情報工学科阿部英志技官に感謝します．

文献

[1] 江尻正員監修，"画像処理産業応用一覧上・下巻，" フジ・テクノシステム，1994.
[2] 和田俊和，松山隆司，"Hough変換に基づく図形検出法の新展開，" 情報処理学会誌，vol.36, no.3, pp.253–263, 1995.
[3] 高木幹雄，鳥脇純一郎，田村秀行編，"画像処理アルゴリズムの最新動向，" 新技術コミュニケーションズ，1986.
[4] D. Casasent and D. Psaltis, "Position, rotation, and scale invariant optical correlation," Applied Optics, vol.15, no.7, pp.1795–1799, 1976.
[5] R.A. Messner and H.H. Szu, "An image processing architecture for real time generation of scale and rotation invariant patterns," Computer Vision, Graphics & Image Processing, vol.31, pp.50–66, 1985.
[6] 上坂吉則，"開曲線にも適用できる新しいフーリエ記述子，" 信学論（A），vol.J67-A, no.3, pp.166–173, March 1984.
[7] D.H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," Pattern Recognition, vol.13, no.2, pp.111–122, 1981.
[8] Y. Lamdan and H.J. Wolfson, "Geometric hashing: A general and efficient model-based recognition scheme," Proc. of ICCV, pp.238–249, 1988.
[9] T.E. Dufresne and A.P. Dhawan, "Chord-tangent transformation for object recognition," Pattern Recognition, vol.28, no.9, pp.1321–1331, 1995.
[10] 渡辺孝志，石戸橋真，"線分近似による一般化ハフ変換の高速化と任意図形検出，" 信学論（D-II），vol.J74-D-II, no.8, pp.995–1003, Aug. 1991.
[11] J. Princen, J. Illingworth, and J. Kittler, "A hierarchical approach to line extraction based on the Hough transform," Computer Vision, Graphics & Image Processing, vol.52, pp.57–77, 1990.

（平成9年7月16日受付，11月6日再受付）

木村 彰男 （正員）

平3岩手大・工・情報卒．平5同大大学院修士課程了．同年ソニー(株)入社．平7岩手大・工・情報助手．この間，画像処理，パターン認識の研究に従事．情報処理学会会員．

渡辺 孝志 （正員）

昭44東北大・工・通信卒．昭46同大大学院修士課程了．昭47（株）日立製作所入社．昭55東北大・大学院博士課程了．工博．同年岩手大・工・情報助手．同講師，同助教授を経て現在同教授．この間，パターンの学習認識，集積回路のCADシステム，セルオートマトン，画像処理の研究に従事．情報処理学会，日本リモートセンシング学会各会員．