論　文

図形検出力を向上させた高速一般化ハフ変換

木村　彰男[†]　　渡辺　孝志[†]

Fast Generalized Hough Transform that Improves its Robustness of Shape Detection

Akio KIMURA[†] and Takashi WATANABE[†]

あらまし　最近提案された高速一般化ハフ変換（FGHT）[1]には，(a) チェック点の導入によって投票の信頼性が高まり，検出の高精度化が実現される，(b) 線分近似の併用によって処理が高速化される，といった特長がある．しかしFGHTには，(1) チェック点の選び方によっては投票がうまく行われず，検出力が低下する可能性がある，(2) 投票空間が4次元配列で構成されており，検出精度を高めようとすると膨大なメモリ量を要する，といった検討課題も残されていた．そこで本論文では，これらの問題に対処し得るFGHTの改良法を新たに提案する．提案手法は，従来のFGHTに比べて検出精度がより向上するとともに，適用可能な画像の範囲が広くなっている．更に，所要メモリが大幅に削減でき，処理速度もかなり改善されている．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，良好な結果が得られた．

キーワード　図形認識，一般化ハフ変換，FGHT，最小2乗法

1. まえがき

入力画像から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[2],[3]．従来，この種の比較的ロバストな図形検出法としては，一般化ハフ変換（GHT）[4]，Geometric Hashing [5]，Chord-Tangent変換（CTT）[6]などが提案されている．しかし，これらの手法は多くの処理時間を必要とし，実用上は満足できる状況になかった．

筆者らは先に，CTTを改良した高速一般化ハフ変換（FGHT）[1]を提案した．FGHTは，上記の手法に比べて処理速度と検出精度を大幅に改善した手法であるが，(a) チェック点によって投票精度を高めているが，それでも誤投票が行われる可能性がある，(b) 処理に必要なメモリ量がかなり大きい，(c) 線分近似を併用しているため，適用可能な画像の範囲がある程度制限される，といった検討課題が残されていた．

本論文では，FGHTにおけるこのような問題に対処できる改良された図形検出法を新たに提案する．提案手法では，FGHTの投票方式が回転と伸縮の変動を吸収するように改善されている．その結果，検出対象図形の一部にひずみや変形があってもある程度対処可能となっており，適用可能な画像の範囲も広がっている．更に，パラメータの多段階投票を導入して，投票空間を4次元から2次元以下に減らしている．その結果，所要メモリが大幅に削減され，副次的効果として処理速度も大幅に改善されている．

以下，**2.**においてFGHTの概要とその検討課題について述べる．**3.**ではFGHTに施した改良点とその実現手順を述べ，提案する図形検出アルゴリズムについて説明する．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，それに基づいて検出性能をFGHTと比較する．

2. FGHTアルゴリズムとその検討課題

2.1 FGHTアルゴリズムの概要

FGHTは，GHTとCTTを大幅に高速化した手法として提案されたものである．以下にFGHTの概略を説明するが，詳細は文献[1]を参照されたい．

以下では，処理対象画像を2値の輪郭線画像とし，それぞれの輪郭点における接線方向は既知とする．また，検出対象となる図形をテンプレート T と呼ぶ．

＜拡張C表の生成＞

[†] 岩手大学工学部情報システム工学科，盛岡市
Faculty of Engineering, Iwate University, 4-3-5 Ueda, Morioka-shi, 020-8551 Japan

[手順 1] T 内に参照点 R を任意に定める（図 1）.
[手順 2] T を構成している輪郭線上から任意に二つの輪郭点 A, B を選び，以下の特徴量を算出する.

- A, B を結ぶ線分（コードと呼ぶ）の長さ L_c.
- コードの角度 β. ただし，β は水平方向の座標軸からの角度を表しており，$\beta \in [0, 2\pi)$ である.
- 2 点 A, B におけるそれぞれの接線（図 1 では点線で表示）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 へ向かうベクトル $\mathbf{t} = (t_x, t_y)^t$.

[手順 3] 2 点 A, B のほかにもう一つの輪郭点 C を選択し，コード中点 m から C へのベクトル $\mathbf{c} = (c_x, c_y)^t$ を算出する. ただし，点 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. 点 C をチェック点と呼び，点 C での接線方向を τ_c とする.
[手順 4] 得られた特徴量 $L_c, \beta, \delta_a, \delta_b, \mathbf{t}, \mathbf{c}, \tau_c$ を拡張 C 表（表 1）に登録する.
[手順 5] 手順 2〜4 を T 内の可能なすべての輪郭点対について繰り返す. □

＜投票手順＞
[手順 1] 回転 θ，伸縮 κ，移動量 (x, y) を表す 4 次元配列 $A(\theta, \kappa, x, y)$ を用意し，その配列要素をすべて 0 にセットする.
[手順 2] 入力画像 I から二つの輪郭点を選び，それをコードとみなした場合（以下，この場合もコードと称する）の特徴量 $\beta^I, {\delta_a}^I, {\delta_b}^I$ を算出する. ただし，コード長 ${L_c}^I$ が短いものは無視する.
[手順 3] 次式

$$|{\delta_a}^I - \delta_a| \leqq \delta_{th} \text{かつ} |{\delta_b}^I - \delta_b| \leqq \delta_{th} \quad (1)$$

$$|{\delta_b}^I - \delta_a| \leqq \delta_{th} \text{かつ} |{\delta_a}^I - \delta_b| \leqq \delta_{th} \quad (2)$$

のいずれかを満たす δ_a, δ_b を拡張 C 表からすべて抽出し，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中点を $({x_m}^I, {y_m}^I)$ とする. すべての n_k について，次式

$$\hat{\theta} = \beta^I - \beta(n_k) \quad (3)$$

$$\hat{\kappa} = {L_c}^I / L_c(n_k) \quad (4)$$

$$\begin{pmatrix} {x_c}^I(n_k) \\ {y_c}^I(n_k) \end{pmatrix} = \hat{\kappa} \begin{pmatrix} \cos\hat{\theta} & -\sin\hat{\theta} \\ \sin\hat{\theta} & \cos\hat{\theta} \end{pmatrix} \begin{pmatrix} c_x(n_k) \\ c_y(n_k) \end{pmatrix} + \begin{pmatrix} {x_m}^I \\ {y_m}^I \end{pmatrix} \quad (5)$$

を用いて I におけるチェック点位置 $({x_c}^I(n_k), {y_c}^I(n_k))$ を求める.
[手順 5] I において，画素 $({x_c}^I(n_k), {y_c}^I(n_k))$ からの距離 c_{th} 以内に輪郭点が存在し，その接線方向 τ^I が

$$|\tau^I - \tau_c(n_k) - \hat{\theta}| \leqq \delta_{th} \quad (6)$$

を満たした場合，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] 次式

$$\begin{pmatrix} \hat{x_r} \\ \hat{y_r} \end{pmatrix} = \hat{\kappa} \begin{pmatrix} \cos\hat{\theta} & -\sin\hat{\theta} \\ \sin\hat{\theta} & \cos\hat{\theta} \end{pmatrix} \begin{pmatrix} t_x(n_k) \\ t_y(n_k) \end{pmatrix} + \begin{pmatrix} {x_m}^I \\ {y_m}^I \end{pmatrix} \quad (7)$$

から，検出対象図形の参照点が存在する推定位置 $\hat{R} = (\hat{x_r}, \hat{y_r})$ を算出し，これと式 (3), (4) の $\hat{\theta}, \hat{\kappa}$ で定まる配列要素 $A(\hat{\theta}, \hat{\kappa}, \hat{x_r}, \hat{y_r})$ に 1 を加算（投票）する.

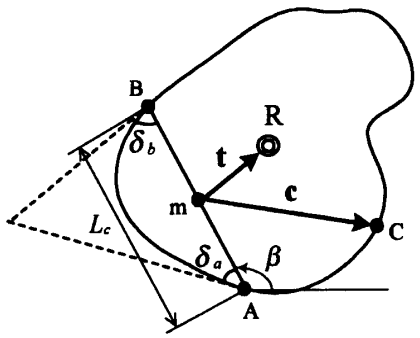


図 1 FGHT におけるテンプレートの記述
Fig. 1 Description of a template used in the FGHT.

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

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

[手順 7] I の可能なすべてのコードに対して手順 2〜6 を繰り返す．最後に，配列 A から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． □

以上が FGHT の基本原理であるが，実際には，処理の高速化のために線分近似手法を取り入れている．すなわち，上記に述べた手順のうち，処理対象となる点を線分近似後の各線分の中点で，処理対象点における接線情報を線分の傾きで，それぞれ置き換えた上で処理を実行している．

2.2 FGHT の検討課題

（1） チェック点の選び方によっては，検出力が低下する可能性がある．

FGHT では，チェック点 C によって無駄な投票を抑制し，結果として高精度な検出を実現していた．しかし，対象物の重なりや雑音等の影響によって，入力画像内でチェック点が隠されてしまうような場合には，投票そのものが行われず，結果的に検出性能が低下する可能性がある．

この問題に対する単純な対応策としては，チェック点の数を 2 点以上に増やし，それらのいずれかのチェック点が入力画像中に存在したら投票する，という対処が考えられる．しかし，それ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ので，あまり得策とはいえない．

（2） パラメータ空間上の “誤った位置” に投票が行われる可能性がある．

例えば，図 1 に示したテンプレート T を，変換パラメータ $(\bar{\theta}, \bar{\kappa}, \bar{x}, \bar{y})$ で相似変換したパターン（図 2）が入力画像 I であるとする．I における，T 上の輪郭点 A, B に対する真の対応点は，$A \leftrightarrow a, B \leftrightarrow b$ である．しかし，Iに施した線分近似の結果が T側のそれとは異なってしまった場合を想定すると，図 2 において $A \leftrightarrow a', B \leftrightarrow b'$ というように対応がずれて定まることがある（コード $\overline{a'b'}$ で定まる不変特徴 ${\delta_a}^I, {\delta_b}^I$ は式 (1), (2) を満足し，かつその際に算出されるチェック点 $({x_c}^I, {y_c}^I)$ も式 (6) を満足していることに注意されたい）．この場合，式 (3), (4) によって算出される $\hat{\theta}, \hat{\kappa}$ は $\bar{\theta}, \bar{\kappa}$ とは等しくならず，誤差を生じる．その結果，式 (7) から算出される参照点候補位置は，図のように $\bar{R}$（真）から $\hat{R}$（偽）に変化してしまい，誤った位置に投票が行われることになる(注1)．

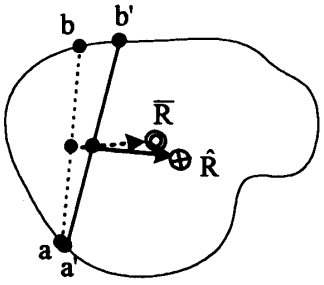


図 2 投票誤りの一例
Fig. 2 An example that voting is carried out to unfavorable position.

（3） 投票空間が 4 次元配列で構成されており，検出精度を上げようとすると所要メモリ量が膨大となって実行困難となる．

この問題は，投票値のピーク探索に費やされる時間が増大する，といった問題も発生させる．

そのほかに，画像内容が複雑で正確な線分近似結果を得ること自体が難しいような場合には FGHT の適用そのものが困難になる，などの問題もある．

以上，FGHT の問題点について述べたが，真に実用レベルのパターン検出法とするためには，これらの課題に対する対策が必要となっている．

3. FGHT の改良

3.1 投票パターンの改善

FGHT における投票位置 $(\hat{\theta}, \hat{\kappa}, \hat{x_r}, \hat{y_r})$ は式 (3), (4), (7) で与えられることは既に述べた．今，式 (7) に着目すると，$\hat{R} = (\hat{x_r}, \hat{y_r})^t$ は $(\hat{\theta}, \hat{\kappa})$ の関数となっている．これを改めて $\hat{R} = F(\theta, \kappa)$ とおき，(θ, κ) がある範囲で変動することを考える．つまり，

$$\theta_{min} \leqq \theta \leqq \theta_{max} \tag{8}$$

$$\kappa_{min} \leqq \kappa \leqq \kappa_{max} \tag{9}$$

の範囲で (θ, κ) を変化させ，そのときに算出される参照点位置 $\hat{R} = F(\theta, \kappa)$ に対応する 4 次元投票空間内の位置すべてに投票を行う．このとき，参照点位置を表す 2 次元の (x, y) 平面（4 次元投票空間の一部）を考えると，推定参照点位置 $\hat{R} = F(\theta, \kappa)$ の描く軌跡は，

$$U = F(\theta_{min}, \kappa_{min}) \tag{10}$$

$$V = F(\theta_{min}, \kappa_{max}) \tag{11}$$

$$W = F(\theta_{max}, \kappa_{min}) \tag{12}$$

$$X = F(\theta_{max}, \kappa_{max}) \tag{13}$$

(注 1)：実際に投票されるのは，4 次元空間上の 1 点である．

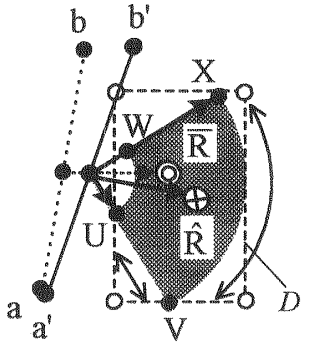


図3　投票範囲の拡大
Fig. 3　Expanded voting pattern in x-y plane.

の4点で囲まれた領域となる．これを図示すると図3のハッチ部分（扇形の部分）に示したような領域となる．投票対象範囲をこの領域に拡大することで投票範囲内に“真の”参照点位置 $\bar{R}$ を含む可能性が非常に高くなり，これによって検出精度が高まることが期待できる．しかし，このままでは投票パターン自体がかなり複雑になってしまうので，実際には図3に示すようなハッチ領域に外接する長方形領域 D を考え，これをすべて投票対象とする（高速化のために D の各辺は投票空間軸と平行にしてある）．

問題となるのは，$\theta_{min}, \theta_{max}, \kappa_{min}, \kappa_{max}$ をどう決めるか，ということである．まず，パラメータθの投票範囲 $\theta_{min}, \theta_{max}$ に関しては，角度の変動が画像内のどの方向においても同様に起こり得ると仮定して，

$$\theta_{min} = \hat{\theta} - \Delta\hat{\theta} \tag{14}$$

$$\theta_{max} = \hat{\theta} + \Delta\hat{\theta} \tag{15}$$

で与えるものとする．ただし，$\Delta\hat{\theta}$ は定数である．そして，もう一つのパラメータ κ の投票範囲$\kappa_{min}, \kappa_{max}$ に関しては

$$\kappa_{min} = \hat{\kappa} - \Delta\hat{\kappa} \tag{16}$$

$$\kappa_{max} = \hat{\kappa} + \Delta\hat{\kappa} \tag{17}$$

で与えるものとする．ただし $\Delta\hat{\kappa}$ は，一般には $\hat{\kappa}$ に依存する量である（付録参照）．

上述のように投票アルゴリズムを変更すると，確かに余分な投票も増えるが，投票パターンが真の位置 $\bar{R}$ を含む可能性が高くなり，更に，もとのFGHTで用いていたチェック点による吟味が不要になると考えられる．なぜなら，FGHTでは投票パターンが1点のみとなっているため，検出精度を上げるには正確な位置へ投票する必要があった．したがってFGHTでは，投票に利用するコードをチェック点による吟味をパスした少数のものに限定する必要があった．これに対して提案手法では，画像の変動を考慮して投票パターンを拡大している(注2)ので，投票パターン内に真のパラメータが含まれる可能性が高くなっている．このため，投票に使用するコードはチェック点による吟味を経ることなくそのまま利用できる．このことは，一つのコードから投票される情報の範囲がもとのFGHTよりも広げられたことを意味しており，結果としてパターン検出のロバスト性が向上すると期待できる．

以上のことから，ここで述べた投票範囲の拡大は，先に述べたFGHTの問題点（1），（2）に同時に対処することが可能な処置であるといえる．

3.2　投票空間の低次元化

投票パターンの拡大により，真の参照点位置への投票漏れはかなりの確率で防ぐことが可能と思われる．しかしながら，このまま単純に投票範囲の拡大を考えた場合，その投票パターンは $[\theta_{min}, \theta_{max}] \times [\kappa_{min}, \kappa_{max}] \times D$ で定まる4次元領域となってしまう．これでは，従来にも増して計算コストが掛かるばかりでなく，（真でない位置への）余分な投票もいっそう増えることになる．そこで本手法では，以下に示すように相似変換パラメータを段階的に投票して求める手法(注3)を採用する．

まず，参照点位置を求めるための2次元投票空間 H_{xy} と，回転・伸縮を求めるための1次元投票空間 H_θ, H_κ をそれぞれ用意する．最初に，H_{xy} に対する投票のみを行う，つまり，前節で述べたように投票範囲を拡大し，領域 D に対応する配列要素すべてに1を加える．その際，表2のような可変長リスト L を別途用意しておき，1回の投票を行うたびに，投票のために選択したコードの端点 E_1, E_2 のそれぞれの座標，点 U, V, W, X のそれぞれの座標，そして，使用した $\theta_{min}, \theta_{max}, \kappa_{min}, \kappa_{max}$ の値をすべて格納しておく．

H_{xy} に対する投票をすべて終えたら H_{xy} 内のピーク点 (x_p, y_p) を検出し，そのピーク点の投票に寄与したコードのみを，リスト L からすべて抽出する（その際，ピーク点 (x_p, y_p) が，リスト L に記録した4点 U, V, W, X で定められる“真の”外接長方形

（注2）：ハフ変換的手法に分布型の投票を採用した例はほかにもいくつか見受けられるが，CTT（及びその類似手法）に適用した例は，筆者らの知る限り本論文が初めてと思われる．
（注3）：パラメータを段階的に投票して検出の効率化を図った例としては，楕円検出の場合について Yoo ら [7] があるが，CTT（その類似手法）に適用した例は，本論文が初めてと思われる．

表2 コード表の構造
Table 2 Contents of chord list L.

コード端点		投票パターン				回転角		伸縮率	
E_1	E_2	U	V	W	X	θ_{min}	θ_{max}	κ_{min}	κ_{max}
${E_1}^{(1)}$	${E_2}^{(1)}$	${U}^{(1)}$	${V}^{(1)}$	${W}^{(1)}$	${X}^{(1)}$	${\theta_{min}}^{(1)}$	${\theta_{max}}^{(1)}$	${\kappa_{min}}^{(1)}$	${\kappa_{max}}^{(1)}$
${E_1}^{(2)}$	${E_2}^{(2)}$	${U}^{(2)}$	${V}^{(2)}$	${W}^{(2)}$	${X}^{(2)}$	${\theta_{min}}^{(2)}$	${\theta_{max}}^{(2)}$	${\kappa_{min}}^{(2)}$	${\kappa_{max}}^{(2)}$
...		...				...		...	

領域 D' 内にあるかどうかを高速に判定するために，文献[8]に記載されている方法を用いる）．そして，それらの抽出されたリスト内要素に記録されている $\theta_{min}, \theta_{max}, \kappa_{min}, \kappa_{max}$ の値を用いて，今度は H_κ, H_θ に対する投票を順次行う，つまり，最初に H_κ 空間内の領域 $[\kappa_{min}, \kappa_{max}]$ に対応する配列要素すべてに1を加え，ピーク点 κ_p を抽出し，次にそのピーク点の投票に寄与したコードのみを用いて H_θ 空間内の領域 $[\theta_{min}, \theta_{max}]$ に対応する配列要素すべてに1を加える．最終的にピーク点 θ_p を抽出すれば，$(\theta_p, \kappa_p, x_p, y_p)$ が解パラメータの候補となる（したがって，同一参照点位置 (x_p, y_p) に複数個の物体があっても検出可能である）．

以上のようにアルゴリズムを構成すると，余分な投票を減らすことが可能となり，更に投票空間としては最大でも2次元のものを考えればよいので，所要メモリは激減する．別途，リスト L の分のメモリが必要となるが，投票空間として4次元のものを用意するよりはずっと少なくてすむ（これについては，後に実験結果で示す）．なお，投票範囲が広い分，1回当りの投票処理時間の増大が懸念されるが，投票パターンは H_{xy} については長方形領域，H_θ, H_κ についてはある1次元範囲のみを考慮するだけでよいので，さほど影響はないと考えられる（これについても，後に実験結果で示す）．

以上により，前述した問題点（3）に対処可能となる．

3.3 最小2乗法によるパラメータ補正

前節の手順によって求められた $(\theta_p, \kappa_p, x_p, y_p)$ はあくまでも解パラメータの候補であるため，実用上はこの解の妥当性確認，精度向上のための補正，といった手順が必要である．そこで，まず，テンプレート T をパラメータ $(\theta_p, \kappa_p, x_p, y_p)$ で変換したものを改めて入力画像 I 上に描く（この際，ロバスト性を増すために，T にある幅をもたせて描くものとする）．そして，その描かれた図形と I 上の点が互いに重なるものの中から，接線方向がある許容範囲 $\Delta_{tan}°$ 内にあるものだけを選択し，T と I の間におけるそれぞれの点の対応を定める．これらの対応の定まった点列を $\{T_j = ({x_j}^T, {y_j}^T)\}, \{I_j = ({x_j}^I, {y_j}^I)\}, j = 1, 2, \cdots$ としたとき，$\{T_j\}$ と $\{I_j\}$ に対して，点パターンマッチングによる最小2乗法[9]を使って変換パラメータを補正する．すなわち，以下の点対応誤差を表す評価関数

$$\sum_j \left\{ ({x_j}'^I - {x_j}^I)^2 + ({y_j}'^I - {y_j}^I)^2 \right\} \tag{18}$$

を最小とするような (θ, κ, x, y) を求め，これを最終的な解パラメータとする．ただし，

$$\begin{pmatrix} {x_j}'^I \\ {y_j}'^I \end{pmatrix} = \kappa \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} {x_j}^T \\ {y_j}^T \end{pmatrix} + \begin{pmatrix} x \\ y \end{pmatrix} \tag{19}$$

である．

3.4 改良されたFGHTアルゴリズム

以上述べてきた改良点を取り入れると，**2.1**で述べたFGHTアルゴリズムは次のように変更される．

＜拡張C表の生成＞

2.1で述べた手順と全く同様に行うが，チェック点Cは登録しない．

＜投票手順＞

［手順1］ 参照点位置を求めるための2次元投票配列 $H_{xy}(x, y)$ を用意し，その配列要素をすべて0にセットする．また，表2の可変長リスト L を用意する．

［手順2］ 入力画像 I から二つの輪郭点を選び，それをコードとみなした場合の特徴量 ${\beta^I}, {\delta_a}^I, {\delta_b}^I$ を算出する．そのコード中点を $({x_m}^I, {y_m}^I)$ とする．

［手順3］ 式(1)または式(2)のいずれかを満たすような δ_a, δ_b を拡張C表の中からすべて抽出し，その各々について以下の手順4～6を繰り返す．

［手順4］ 手順3で選ばれた拡張C表に登録されているパラメータを ${L_c}^T, {\beta^T}$ としたとき，

$$\hat{\theta} = \beta^I - \beta^T \tag{20}$$

$$\hat{\kappa} = {L_c}^I / {L_c}^T \tag{21}$$

から $\hat{\theta}, \hat{\kappa}$ を求める．

［手順 5］ 式 (14)～(17) から，θ_{min}，θ_{max}，κ_{min}，κ_{max} を決定し，式 (10)～(13) で定まる 4 点 U, V, W, X を求める．ただし，$\Delta\hat{\theta}$ はある定数であり，$\Delta\hat{\kappa}$ は付録で述べるように適応的に定める．

［手順 6］ 4 点 U, V, W, X で定まる扇形に外接する長方形領域 D を考え，それに対応する配列 H_{xy} の要素すべてに 1 を加算（投票）する．このとき，同時に表 2 の形式で各パラメータを L に登録しておく．

［手順 7］ 画像 I の可能なすべてのコードに対して手順 2～6 を繰り返し，配列 H_{xy} に対する投票を完了させる．そして，H_{xy} において，しきい値以上の投票値をもつ配列要素を投票値の高い順に選ぶ．この集合を $\{P_i = (x_{p_i}, y_{p_i}) | i = 1, 2, \cdots\}$ とする．

［手順 8］ すべての P_i について，次の手順 9～16を繰り返す．

［手順 9］ 伸縮を求めるための 1 次元投票配列 $H_\kappa(\kappa)$ のすべての配列要素を 0 にセットする．

［手順 10］ 4 点 U, V, W, X で定まる領域 D' の内部に座標 (x_{pi}, y_{pi}) が含まれているような要素を L からすべて抽出する．そして，抽出された各要素について，そこに登録されている $\kappa_{min}, \kappa_{max}$ で定まる領域 $[\kappa_{min}, \kappa_{max}]$ に対応する配列 H_κ の要素すべてに 1 を加算（投票）する．

［手順 11］ H_κ においてしきい値以上の投票値をもつ配列要素を投票値の高い順に選び，この集合を $\{Q_j = \kappa_{p_j} | j = 1, 2, \cdots\}$ とする．

［手順 12］ すべての Q_j について，次の手順 13～16 を繰り返す．

［手順 13］ 回転を求めるための 1 次元投票配列 $H_\theta(\theta)$ のすべての配列要素を 0 にセットする．

［手順 14］ κ_{p_j} の投票に寄与したコードの $\theta_{min}, \theta_{max}$ を用いて，領域 $[\theta_{min}, \theta_{max}]$ に対応する配列 H_θ の要素すべてに 1 を加算（投票）する．

［手順 15］ H_θ においてしきい値以上の投票値をもつ配列要素 $\theta_{\kappa_{p_j}}$ を選び，パラメータ $(x_{p_i}, y_{p_i}, \theta_{\kappa_{p_j}}, \kappa_{p_j})$ でテンプレート Tを相似変換し，その図形にある幅 w をもたせて入力画像 I 上に描く．そして，**3.3** の方法に基づいて T と I でのそれぞれの点の対応を定め，それらの点列に対して最小 2 乗法を適用する．得られたパラメータを $(\hat{\theta}, \hat{\kappa}, \hat{x_r}, \hat{y_r})$ とする．

［手順 16］ 改めてパラメータ $(\hat{\theta}, \hat{\kappa}, \hat{x_r}, \hat{y_r})$ で T を変換した図形 T' を I 上に描き，互いに重なる輪郭点数をカウントする．その点数が T' を構成する総輪郭点数に対して指定された割合（例えば 50%）を超えるとき，$(\hat{\theta}, \hat{\kappa}, \hat{x_r}, \hat{y_r})$ を検出パラメータとする． □

4. 評価実験

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 400 MHz の Pentium II プロセッサを使用した．実験に使用した主なパラメータは，投票空間が $0° \leqq \theta < 360°$，$\Delta\theta = 5°$，$0.8 \leqq \kappa \leqq 1.2, \Delta\kappa = 0.02$，$0 \leqq x, y \leqq 256, \Delta x = \Delta y = 1$，角度許容誤差 $\delta_{th} = 10°$，$\Delta_{tan}{}° = 10°$，再描画幅 $w = 2$（画素），投票範囲 $\Delta\hat{\theta} = 5°$，$\Delta\hat{\kappa} = 0.05$ である．なお，この実験では求めたい κ の範囲がさほど大きくないことから，高速化を考慮して $\Delta\hat{\kappa}$ を定数としたことに注意されたい．

4.1 FGHT との比較実験

はじめに，図 4 に示す実画像を用いて実験を行った．図 4 の上側二つの画像は，いずれもカメラ入力された同一物体の異なる濃淡画像であり，下側二つの画像は，上側の画像から輪郭線を抽出し，それを線分近似した 2 値画像である．画像サイズはいずれも 256×256 画素であり，輪郭線抽出には Canny の方法 [10] を使用し，線分近似手法としては，標準的ハフ変換（SHT）[11] を使用した．ここでは，図 4 の左側の画像をテンプレート画像と考え，右側の画像を入力画像として検出実験を試みた．それぞれの画像における線分数は，テンプレートが 68，入力画像が 74 である．ここで，図 4 から明らかなように，テンプレートと入力画像の線分近似結果は大きく異なっている．

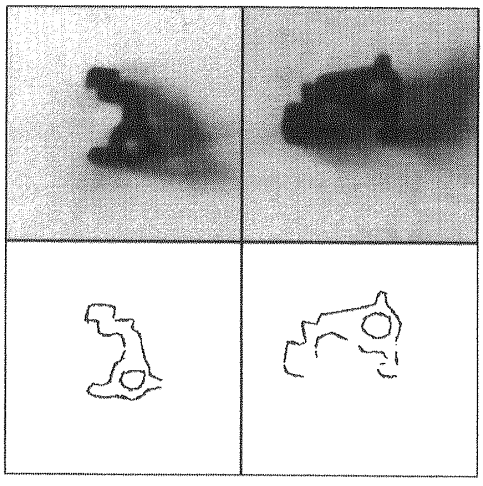


図 4 テスト画像 1
Fig. 4 Tested image 1.

まずはじめに，これらの画像に通常のチェック点を用いたFGHTを適用したところ，投票時に入力画像でチェック点そのものが見つからない場合があり，投票回数そのものが少なくなってパターン検出を行うことができなかった．そこで今回は，チェック点を使用しない形のFGHTと提案手法との比較を試みた．

検出結果を図5に示す．図では，テンプレート画像を得られた相似変換パラメータで変換し，入力画像上に重ね書きしてある．左側がFGHTによる検出結果，右側が提案手法による検出結果である．また，処理時間と所要メモリは表3にまとめた．

まず，検出精度に関してであるが，今回の実験ではテンプレートと入力画像がともにカメラ入力されたものであるため，検出精度そのものを定量的に評価することは困難である．しかしながら，見た目の検出結果（図5）としては明らかに提案手法の方がよい．これは，提案手法では，**3.1**で述べた投票範囲の拡大により（余分な投票も増えるが）真の位置にもれなく投票が行われ，真の位置の投票値が常にピークを保っていた（最大投票値113），という効果が現れたものと考えられる．逆に，FGHTでは，4次元投票空間中でのピーク点（最大投票値7）が三つ存在した．すなわち，一つの検出対象パターンに対して3種類の異なった解が検出されてしまった．図中では，最も正解に近いと思われる結果のみを示したが，これは**2.2**で述べた問題が発生し，誤投票が増えて真の解がぼやけてしまった，ということが原因であると考えられる．なお，提案手法で得られたピーク位置は $(\theta_p, \kappa_p, x_p, y_p) = (270°, 1.16, 118, 86)$ であったが，この位置におけるFGHTでの投票値は3であった．また，提案手法においては，投票のみで得られた解候補パラメータと最小2乗法で得られた補正解の間には差が見られなかった．

処理時間と所要メモリについては，表3に示したとおり，提案手法はFGHTと比べて，処理時間と所要メモリがともに約1/60に抑えられている．提案手法では，投票範囲を広げた分だけ若干投票に要する時間が増えているものの，**3.2**で述べた投票空間の低次元化によって，全体的な処理時間短縮が実現できていることが確認できる．

4.2 複雑な実画像での実験

次に，比較的入力画像が複雑な場合を想定し，図6に示したような画像に対して提案手法を適用してみた．先の実験と同様に，上側二つの画像はカメラ入力された濃淡画像であり，その左側はテンプレート，右側は入力画像を表している．また，下側二つの画像は，上

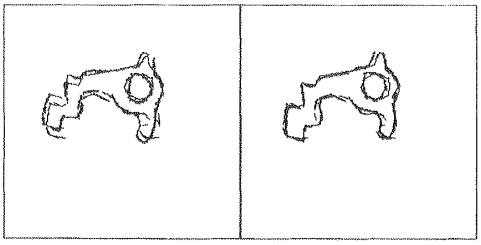


図5 (左)FGHTによる検出結果
(右) 提案手法による検出結果
Fig. 5 (Left) Detection result by the FGHT.
(Right) Detection result by the proposed method.

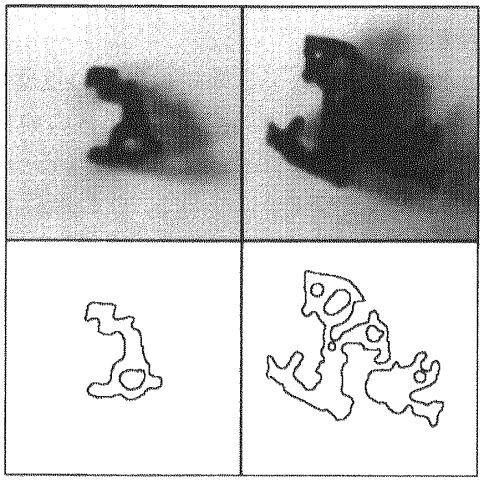


図6 テスト画像2
Fig. 6 Tested image 2.

表3 FGHTと提案手法における処理時間と所要メモリの比較
Table 3 Comparison on CPU time and requirement memory between the FGHT and the proposed method.

手法	処理時間 [s] 投票	ピーク検出	最小2乗法	計	所要メモリ [MB]
FGHT	（検出不能）				–
FGHT(チェック点なし)	0.08	10.17	–	10.25	180
提案手法	0.12	0.02	0.02	0.16	3

側二つの画像から輪郭線を抽出した2値画像であり，その輪郭点数はそれぞれ608点，1867点である．なお，この実験では伸縮率 κ に対する投票空間のサイズを $0.3 \leqq \kappa \leqq 0.8, \Delta\kappa = 0.02$ とした．

画像が複雑なため，テンプレートと入力画像における線分近似結果は大きく異なってしまった(注4)．このため，実験には輪郭線画像をそのまま使用することとし，各輪郭点における接線情報は，対応する濃淡画像の微分画像から得られるこう配情報に直交する方向としてあらかじめ算出して使用した．更に，処理の高速化を図るために，入力画像中のすべての輪郭点対を処理対象とするのではなく，文献[12]と同様の考え方で，輪郭点対をランダムに15%選択して実験を試みた．

さて，入力画像に着目すると，検出対象物には他物体との重なりによる隠ぺいがあり，かなり特徴的と思われる部分も欠落してしまっている．更に，画像を取り込む際の撮影条件が違うため，得られた輪郭線にひずみも生じている．このような意味で，検出対象としている対象物はテンプレートの純粋な相似変換にはなっていないが，現実的な応用を考えた場合はこのようなレベルでの悪条件下の画像にも対処できなければならない．従来のパターン検出法ではこの種の画像に実時間で，かつ精度良く対処することは困難と考えられるが，提案手法では，図7に示したように，ほぼ満足できる検出結果が得られた（先の実験と同様に，得られた解パラメータでテンプレートを相似変換し，それを入力画像中に重ねて表示している）．検出に要した処理時間は5.34秒であり，投票のみで得られた解候補パラメータと最小2乗法による補正解との間に差は見られなかった．

なお，図8は，本実験における投票空間 H_{xy} での投票値の分布を示したものである．この図から，真の参照点位置における投票値は唯一のピークとして定まっており，投票範囲の拡大によってピークがぼける，といった影響はかなり小さいことが確認できる．

4.3 最小2乗法の効果確認実験

前節までの実験によって，提案手法で得られるピーク点は十分真の解に近く，最小2乗法による補正を行わなくてもかなり精度の高い検出が可能であることが示された．しかし，より悪条件下での適用を想定した場合には，図8に示したような明確なピークを得るのは困難となることも予想される．そこで，そのような悪条件下でピークがぼやけてしまった場合に，提案手法で取り入れた最小2乗法による補正効果が有効に機能するかどうかを確認する実験を試みた．

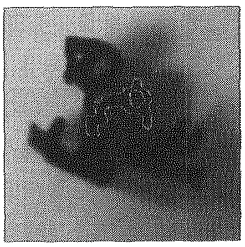


図7　図6の検出結果
Fig. 7　Detection result for Fig. 6 by the proposed method.

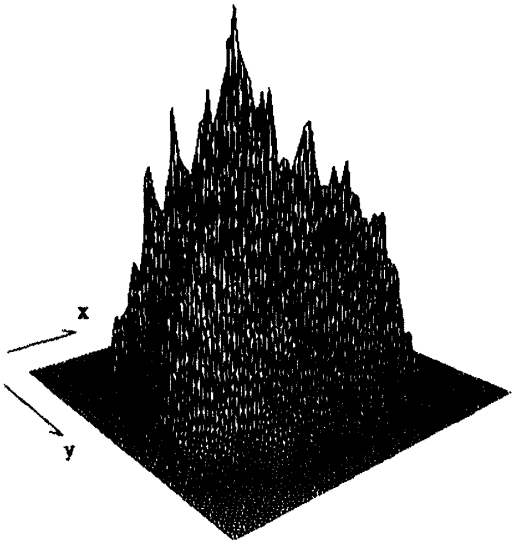


図8　投票配列 H_{xy} における投票値の分布
Fig. 8　Distributions of votes in the voting space H_{xy}.

前述したように，先の図4で示した実験において，提案手法を適用した際に投票で得られたピーク点は $(\theta_p, \kappa_p, x_p, y_p) = (270°, 1.16, 118, 86)$ であった．今，これを真の値と仮定し，それぞれの変換パラメータを真の値から故意にずらした上で最小2乗法を適用し，最終的に得られた補正解がどの程度真の値に近づくかを調べてみた．その結果，与える (θ, κ, x, y) が，$(\theta_p \pm 10°, \kappa_p \pm 0.08, x_p \pm 5, y_p \pm 5)$ の範囲であれば，最小2乗法で得られる補正解が $(\theta_p \pm 3°, \kappa_p \pm 0.02, x_p \pm 2, y_p \pm 2)$ 以内の誤差に収まることを確認した．中でも例えば，$(\theta, \kappa, x, y) = (265°, 1.08, 115, 91)$ と，真の値とはかなり異なるパラメータを与えたときにも，その補正解は真の値と完全に一致した．更に，表3の結果で示したように，全

(注4)：このため，FGHT自体は適用できなかった．

体の処理時間に対して最小2乗法が要する時間の割合は約1/8であり，最小2乗法を適用したために処理時間が大幅に遅くなる，ということもない．以上の結果は，最小2乗法が極めて効果的に働くことを示しているとともに，高速化のために投票空間の分解能をより粗くとった場合でも信頼性の高い解パラメータが得られる可能性を示唆している．

4.4 検　　討

実験結果から提案手法の妥当性と有効性は一応示されたものと考えるが，以下では主に，FGHTとの比較，という観点から提案手法について検討する．

もとのFGHTでは，その処理の一部に線分近似を採用しており，それゆえ線分近似自体が困難な場合には適用が制限される，という問題があった．更に，チェック点を導入していたために，テンプレートと入力の間で線分近似結果が異なってしまうと，(a) 正しい位置に投票されない，(b) チェック点の存在が確認できない場合には投票そのものが行われない，といった問題点も生じていた．これに対して提案手法では，「投票範囲を広げ，真の位置への投票を外さないようにする」という考え方でこれに対処しており，線分近似処理を施しても施さなくても，特に検出率が低下するということはない．チェック点に関しても，特に使用しなくても問題なく検出が可能となっている．加えて，実験で確認できたように，入力画像内に存在する検出対象図形が“テンプレートの純粋な相似変換でない”ような場合，すなわち，一部にひずみなどの変形があるような場合にも対処可能となっており，アルゴリズムの適用範囲は更に広くなっている．そして，パラメータの多段階型投票方式を導入して投票空間を2次元以下で実現し，所要メモリや処理時間についても大幅に改善を図ることができた．また，最小2乗法による検出解の補正が効果的に働いていることも確認し，その副次的効果として投票空間の分解能を粗くしても検出精度を保てる可能性を示した．これによって，更なる処理時間の改善も可能であろう．

5. むすび

本論文では，相似変換に不変なパターン検出法として，検出のロバスト性と処理の高速化という面で従来のFGHTを大幅に改善した手法を提案した．提案手法の特長としては，

（1） 投票範囲の拡大によりパターン検出のロバスト化を実現し，それによって検出対象パターンが多少のひずみを有する場合にも対処可能となった，

（2） 移動，回転，伸縮のパラメータを段階的に投票する方式を採用することで，所要メモリと処理時間の大幅な削減が可能になった，

などの点が挙げられる．一連の評価実験によって，従来のFGHTと比べて更に実用的な手法となっていることが確認できた．また，所要メモリが激減したことによって，低コストでのハードウェア化の可能性も開けたのではないかと考えている．

今後の検討課題としては，入力画像からランダムにコードを選ぶのではなく，より特徴的な点を効率的に選ぶ手法を開発することが挙げられる．

謝辞　執筆にあたり，有益なるコメントを頂いた査読者に感謝します．なお，本研究の一部は文部省科学研究費奨励研究A（No.10780215）の助成によった．

文　　献

[1] 木村彰男，渡辺孝志，“高速一般化ハフ変換 —相似変換不変な任意図形検出法，” 信学論（D-II），vol.J81-D-II, no.4, pp.726–734, April 1998.

[2] 江尻正員監修，画像処理産業応用一覧 上・下，フジ・テクノシステム，1994.

[3] R.C. Gonzalez and R.E. Woods, Digital Image Processing, Addison-Wesley Publishing, 1995.

[4] D.H. Ballard, “Generalizing the Hough transform to detect arbitrary shapes,” Pattern Recognit., vol.13, no.2, pp.111–122, 1981.

[5] Y. Lamdan and H.J. Wolfson, “Geometric hashing: A general and efficient model-based recognition scheme,” Proc. ICCV, pp.238–249, 1988.

[6] T.E. Dufresne and A.P. Dhawan, “Chord tangent transformation for object recognition,” Pattern Recognit., vol.28, no.9, pp.1321–1331, 1995.

[7] J.H. Yoo and I.K. Sethi, “An ellipse detection method from the polar and pole definition of conics,” Pattern Recognit., vol.26, no.2, pp.307–315, 1995.

[8] J. O'Rouke, Computational Geometry in C, Cambridge University Press, 1994.

[9] S.H. Chang, F.H. Cheng, W.H. Hsu, and G.Z. Wu, “Fast algorithm for point pattern matching: Invariant to translations, rotations and scale changes,” Pattern Recognit., vol.30, no.2, pp.331–338, 1997.

[10] J. Canny, “A computational approach to edge detection,” IEEE Trans. Pattern Anal. & Mach. Intell., vol.PAMI-8, no.6, pp.679–698, 1986.

[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.

[12] P. Kultancen, L. Xu, and E. Oja, “A new curve detection method: Randomized Hough Transform (RHT),” Pattern Recognit. Lett., vol.11, pp.331–338,

1990.

付　　録

パラメータκの投票範囲について

まず，$\hat{\theta},\hat{\kappa}$ によって参照点位置がどれだけ変動するのかを考えてみる．参照点位置の算出式は式 (7) であるが，$({x_m}^I, {y_m}^I)$ は入力画像より唯一に定まる点なので，ここではこれを原点にとって考える．今，$\mathbf{r}=(\hat{x_r},\hat{y_r})^t$, $R=\begin{pmatrix}\cos\hat{\theta} & -\sin\hat{\theta}\\ \sin\hat{\theta} & \cos\hat{\theta}\end{pmatrix}$, $\mathbf{t}=(t_x,t_y)^t$, とおけば，式 (7) は

$$\mathbf{r}=\hat{\kappa}R\mathbf{t} \tag{A·1}$$

となり，そのノルムは

$$\|\mathbf{r}\|=\hat{\kappa}\cdot\|R\mathbf{t}\|=\hat{\kappa}\cdot\|\mathbf{t}\| \tag{A·2}$$

である．したがって，$\hat{\kappa}$の変動$\delta\hat{\kappa}$ による $\|\mathbf{r}\|$ の変動は

$$\delta\|\mathbf{r}\|=\delta\hat{\kappa}\cdot\|\mathbf{t}\| \tag{A·3}$$

である．ここで，変動分 $\delta\hat{\kappa}$ について考えると，式 (4) より

$$\begin{aligned}\delta\hat{\kappa} &= \delta({L_c}^I/{L_c}^T)\\ &= \delta{L_c}^I\cdot(1/{L_c}^T)-{L_c}^I\cdot\frac{\delta{L_c}^T}{({L_c}^T)^2}\end{aligned} \tag{A·4}$$

である．右辺第 2 項の変動分 $\delta{L_c}^T$ は，投票時に拡張 C 表に登録済のコード特徴 ${L_c}^T$ を誤って参照する場合に起こり得る変動を表している．そこで，拡張 C 表には，不変特徴量 δ_a,δ_b の変動がそれぞれ δ_{th} 以下のときには変動 $\delta{L_c}^T$ が定数 $c_2(\ll({L_c}^T)^2)$ 以下に収まるもののみ登録するとすれば，右辺第 2 項全体が無視できる．したがって，

$$\delta\hat{\kappa}=\delta{L_c}^I/{L_c}^T=({L_c}^I-\overline{{L_c}^I})/{L_c}^T \tag{A·5}$$

と書ける．ただし，$\overline{{L_c}^I}$ は，“真の” コードから算出される特徴量である．この $\overline{{L_c}^I}$ は，入力画像に関する誤差の事前知識があれば定式化も可能と思われるが，実際には，撮影状況・照明条件・機器の違い，データ変換に伴う確率的誤差などのすべての要因を考慮する必要があり，その決定は極めて困難である．そこで，ここでは，${L_c}^I$ の変動が真の $\overline{{L_c}^I}$ に対して最大でもある小さな定数ϵ倍以内に収まっていることを仮定する (すなわち，$|{L_c}^I-\overline{{L_c}^I}|\leqq\epsilon\overline{{L_c}^I}$)．すると式 (A·5) は，

$$|\delta\hat{\kappa}|\simeq\epsilon\overline{{L_c}^I}/{L_c}^T\simeq\epsilon\cdot\hat{\kappa} \tag{A·6}$$

と書くことができる．ここで，2 番目の $\simeq$ では，$\overline{{L_c}^I}/{L_c}^T$ を $\hat{\kappa}$ で近似した．したがって，パラメータ κ 方向のずれを改めて $\Delta\hat{\kappa}\equiv|\delta\hat{\kappa}|$ とすると，パラメータ κ の投票範囲$\kappa_{min},\kappa_{max}$ は

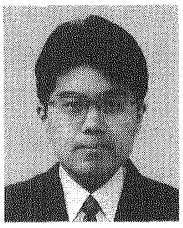
$$\kappa_{min}=\hat{\kappa}-\Delta\hat{\kappa} \tag{A·7}$$

$$\kappa_{max}=\hat{\kappa}+\Delta\hat{\kappa} \tag{A·8}$$

で定めることができる．つまり，κ の投票範囲 $[\kappa_{min},\kappa_{max}]$ は，$\hat{\kappa}$ の値が大きいときには広くとり，$\hat{\kappa}$ の値が小さいときには狭くとってよいことになる．ただし，求めたい $\hat{\kappa}$ の範囲が大きくない場合には，式 (A·6) より $\Delta\hat{\kappa}$ を定数に選んでよいことになる（本論文の実験はこの場合について行っている）．

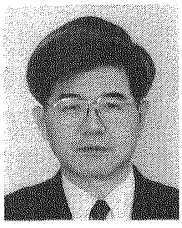
なお，κ の投票空間 H_κ の量子化にあたっては，その量子化幅 Δ の有効けたが同じになるように選べば (すなわち，$\hat{\kappa}$ の値が小さいところではΔ を小さく，$\hat{\kappa}$ が大きいところではΔ を大きくとる)，所要メモリは少なくてすむ．この場合の量子化誤差によるパラメータ検出精度の低下は，**3.3**で述べた最小 2 乗法で防ぐことが可能と考えられる．

（平成 11 年 6 月 25 日受付，10 月 26 日再受付）



木村　彰男　（正員）

平 5 岩手大大学院工学研究科情報工学専攻修士課程了．同年ソニー（株）入社．在社中は磁気記録関係の研究開発に従事．平 7 より岩手大学工学部助手．画像処理，パターン認識に関する研究に従事．情報処理学会，画像電子学会各会員．



渡辺　孝志　（正員）

昭 46 東北大大学院工学研究科修士課程了．同年（株）日立製作所入社．昭 55 東北大大学院工学研究科博士課程了．工博．同年岩手大学工学部助手．現在同教授．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，画像処理などの研究に従事．情報処理学会，日本リモートセンシング学会，地理情報システム学会各会員．