提出日　２０１５年2月６日

Summary of Bachelor's Thesis

２０１５年３月修了卒業論文概要書

Name 氏　　名	宮地諒平	ID number 学籍番号	1W110508-5
題　　目 Title (日本語の場合は英文題目も記入)		Supervisor 指導教員	渡辺 裕　　印

日本語 Japanese

Ｐ形フーリエ記述子を用いた
前髪形状からのマンガ登場人物識別

英語 English

Comic Character Recognition Using P-type Fourier Descriptor with Foretops Shape

概　　要 Summary

近年モバイル端末の発展により，電子書籍や電子コミックが普及している．電子コミックは，単にコミックを電子化するだけではなく，様々な付加効果により，ユーザーを楽しませ利便性を向上させている．しかしその作成には膨大なコストと時間がかかり，広く普及していないのが現状である．マンガにおいて髪は，外観を特徴付ける重要な要素であり，キャラクターが小さく描画される場合でも，髪は特徴的に描画される傾向にある．よって本論文では，髪をキャラクター識別に効果的な特徴であると考え，本研究では髪の輪郭から自動的で高精度なキャラクター識別を目指す．

本論文では，前髪の輪郭からＰ形フーリエ記述子を用いた識別を行った．フーリエ記述子とは，輪郭線を表現する関数を離散フーリエ変換して得られる複素数列である．特にＰ型フーリエ記述子は開曲線に対応可能で，曲線の平行移動や拡大・縮小に対して不変であり，回転や裏返しに対して元の記述子と簡単な関係が成り立つ，というような特徴があり，髪型の識別に適していると考えられる．

実験では，漫画画像からキャラクターの前髪を手動で取得し，それぞれの相違度を計算することで識別を行った．その結果，８２％で識別が可能であり，キャラクターによっては１００％の識別率が得られた．これにより，Ｐ形フーリエ記述子は，マンガキャラクターの形状の差を識別可能であり，それによるキャラクター識別がある程度可能であることが確認できた．

注意 1. 提出期限/ Deadline of Submission　2015年2月6日（金）17:00 /February 6, 2015 17:00

基幹理工学部情報理工学科

Bachelor's Thesis
卒　業　論　文

Title
論　文　題　目

Ｐ形フーリエ記述子を用いた
前髪形状からのマンガ登場人物識別

Comic Character Recognition Using P-type
Fourier Descriptor with Foretops Shape

Student ID 学籍番号	1W110508-5
Name 氏　　名	Ryohei Miyachi 宮地諒平

Supervisor 指導教員	Hiroshi Watanabe 渡辺　裕	印

２０１５年３月１５日

目次

第１章

序論

1.1　研究の背景

近年，タブレットやスマートフォンなどのモバイル端末の発展により，電子書籍，電子コミックが普及している[1]．日本でも２００２年ごろから年々市場は拡大し，２０１８年には，２０１３年度の約３．３倍の３３４０億円程度の市場規模になると予測されている．また２０２０年には電子書籍の売上げが紙の書籍を超えると予想されている[2].

しかし海外に比べると日本での電子書籍の普及率はまだまだ低く，アメリカでは 2011 年，イギリスでは２０１２年に通販サイト Amazon での電子書籍リーダーKindle 向けの電子書籍の販売部数が，紙の本の販売部数をすでに上回っている[3]．それは日本における電子書籍のコンテンツの少なさ[4]や，販売価格が中古書籍に比べて高いことが原因の一つであると考えられる．その中でも２０１４年９月２２日には集英社が国内最大のマンガ雑誌「週刊少年ジャンプ」を電子化[5]，また今年２０１５年１月には講談社が全コミック誌を電子板で配信する方針を決め[6]，今後電子コミックの普及はますます進んでいくものと考えられる．

マンガの電子書籍化は，単なる紙媒体を電子化するだけではなく，様々な付加価値を与えることができる．例えば一コマずつ順に表示することで，狭い表示領域でも快適に見ることができるようにする，キャラクターや文字などを動かすことで，紙にはない効

果を与えるなど，読み手を楽しませる取り組みが増えている．しかし作成には膨大なコストと時間がかかり，広く普及していないのが現状である．

1.2 本研究の目的

マンガにおいてキャラクターの識別が自動的に可能になれば，電子コミックでの，タップによるキャラクターの名前などの情報表示や，過去の登場場面の検索など，ユーザの電子コミックの利便性を大いに向上させる．しかし，マンガのキャラクターは実際の人物画像とは異なる特徴を持っており，今日かなり高精度になっている実際の人物の識別をそのまま使用することは難しい．また，イラストにおけるキャタクターの識別に関する研究もあまり進んでおらず，数少ない研究[7][8][9]も色情報を元にしているものが多く，白黒で描かれているマンガにおいてはそのままの応用は不可能である．

そこで本論文ではマンガにおけるキャラクターの髪に注目した．髪は外観を特徴づける重要な要素であり，キャラクターごとに特徴がある．また，キャラクターが小さく描写される場合，顔の各パーツは簡略化されて描かれることが多いが，髪は特徴的に描写される傾向がある（例を図 1 に示す）．また，横顔や後ろ姿に関しても，髪は存在しており，正面からの描写以外でも識別ができる可能性がある．以上から髪はキャラクター識別に効果的な特徴であると考え，本研究では髪情報を用いた自動的で高精度なキャラクター識別を目指す．

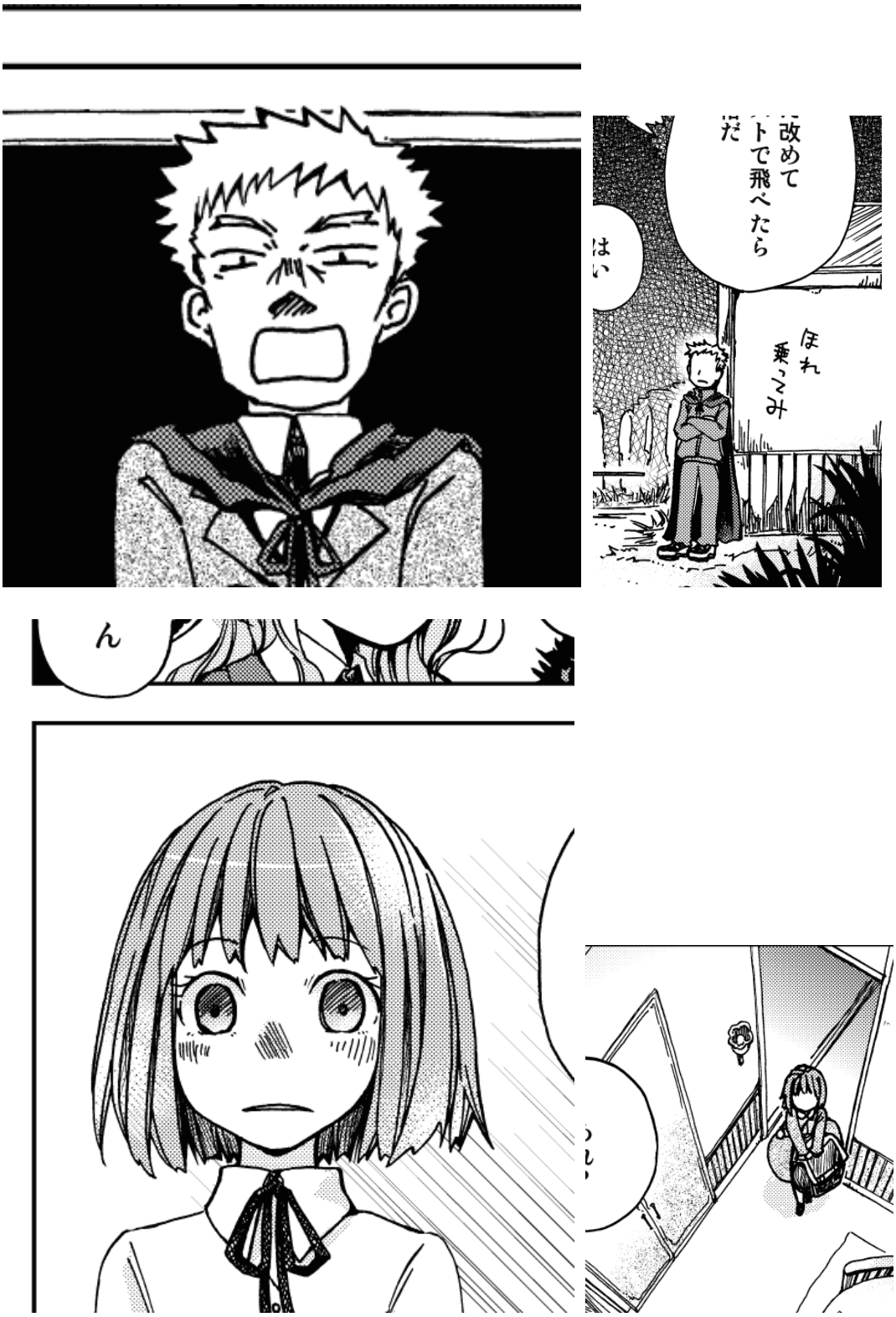

図 1　髪が特徴的に描画される例（画像は[10]より）

1.3 本論文の構成

本論文の構成と概要を以下に示す.

第１章は本章であり，本論文の研究背景と目的，構成について述べている.

第２章では，本論文で使用した２つの手法 Average Hash と P 形フーリエ記述子という手法について述べる.

第３章では，Average Hash を用いて行った実験手法と結果，その考察について示す.

第４章は, 本論文の本論であり, P 形フーリエ記述子を用いて行った実験手法と結果, その考察について示す.

第５章では本論文のまとめと，今後の課題を述べる.

第２章

Average Hash と
Ｐ形フーリエ記述子

2.1 まえがき

本章では, 本論文で使用した手法についての説明を示す. キャラクター識別に Average Hash と P 形フーリエ記述子を使用し相違度の計算を行った.

2.2 Average Hash[11]

Average Hash（以後 aHash）とは Neal Krawetz によって提案された画像識別の手法である. 画像からハッシュ値を生成し比較するもので, ハッシュ値は以下のように生成される.

1. 画像サイズを縮小
   提案手法[11]では 8*8 画素であったが, 本論文では精度向上のため 16*16 画素に圧縮
2. グレースケールに変換
3. 画像の各ピクセルから色の平均値を計算

4. 各ピクセルの濃度を元に，計算した平均値より濃い場合は０，薄い場合は１に設定
5. 16*16 画素サイズのブロック単位のビット列を生成

このビット列の生成過程を図２に示す.

２つの画像のハッシュ値のハミング距離を相違度とし, 似ている画像は近いハッシュ値が生成され相違度が小さくなる.

図 2　aHash によるハッシュ値生成例

2.3 P 形フーリエ記述子[12]

2.3.1 概要

平面上の曲線を周波数領域で記述する方法の一つとしてフーリエ記述子がある. 一般には Zahn and Roskies による Z 形フーリエ記述子[13]と，Granlund による G 形フーリエ記述子[14]がよく知られており，前者は閉曲線の一点からの長さに対する角度変化の累積，すなわち全曲率関数を１次関数で連続化したものを，後者は閉曲線をその一点からの長さの複素数値関数として見たものを, それぞれフーリエ展開して得られる係数を記述子とするものである. しかし, マンガにおける髪の輪郭では, 髪のすべてが描画され，閉曲線として捉えられるが少なく，髪の一部のみが書かれることが殆どあった. そのため以下の様な特徴を持つ，P 形フーリエ記述子を使用した.

1. 開曲線に適用できる

2. 原曲線の平行移動及び拡大・縮小に不変で，回転移動を容易に表現可能

これにより，前髪のみの輪郭を開曲線として捉え，適用することが可能である．

2.3.2 曲線のＰ表現と全曲率関数

δを十分小さい正の実数，ｎを十分大きい正の整数とした時，ある曲線Ｃを，長さδであるｎ本の線分を連接してできる折線図形Ｃに近似する．さらに各線分の端点の座標を

$$(x(i), y(i))(i = 0,1, \dots, n) \quad (1)$$

で表す．平面を，ｘ軸を実軸，ｙ軸を虚軸とする複素平面と考え，複素数

$$z(i) = x(i) + jy(i) \quad (2)$$

をを曲線Ｃと同一視する(j は虚数単位)．Ｃが開曲線の時$z(0), z(n)$は端点であり，閉曲線の時$z(0) = z(n)$となる．この時各線分の長さδは

$$\delta = |z(i+1) - z(i)| \quad (3)$$

で表せる．

図 3 に示すように，折線図形Ｃ上の２つのベクトル，$z(i) - z(i-1)$と，$z(i+1) - z(i)$のなす角を$a(i)$とする．ただし$a(0)$は図 4 の通り，ベクトル$z(1) - z(0)$とｘ軸のなす角とする．$a(i)$を$z(i)$における偏角という．

偏角関数ａを用いて，折線図形Ｃの全曲率関数を

$$\begin{cases} \theta(0) = a(0) \\ \theta(i) = \theta(i-1) + a(i)\ (i = 1, \dots n-1) \end{cases} \quad (4)$$

と定めると，θを用いて複素数置換数ｗを

$$w(i) = e^{j\theta(i)}\ (i = 0,1, \dots, n-1) \quad (5)$$

で定め，これを曲線Ｃのｐ表現と呼ぶ．

偏角と全曲率関数から，ベクトル$z(i+1) - z(i)$と，ｘ軸のなす角$\varphi(i)$(図 5)は，$\theta(i)$と２πの整数倍を除いて等しいことが分かる．よって，

$$\begin{aligned} e^{j\theta(i)} &= e^{j\varphi(i)} \\ &= (z(i+1) - z(i))/\delta \end{aligned} \quad (6)$$

となり，式（5）より，

$$w(i) = (z(i+1) - z(i))/\delta \quad (7)$$

となり，ｚから直接この式で定義することが可能となる．

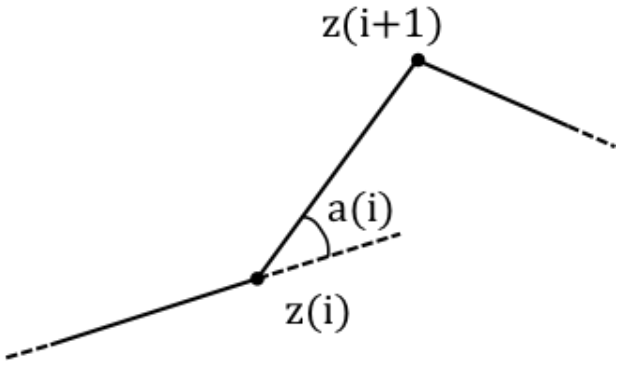

図 3　偏角 a(i) の定め方

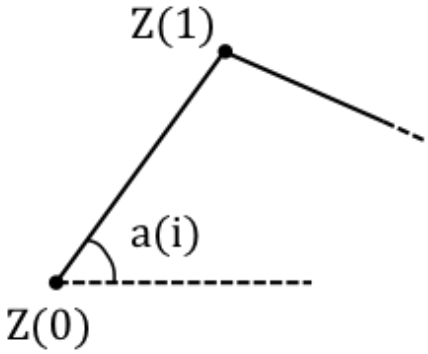

図 4　偏角 a(0) の定め方

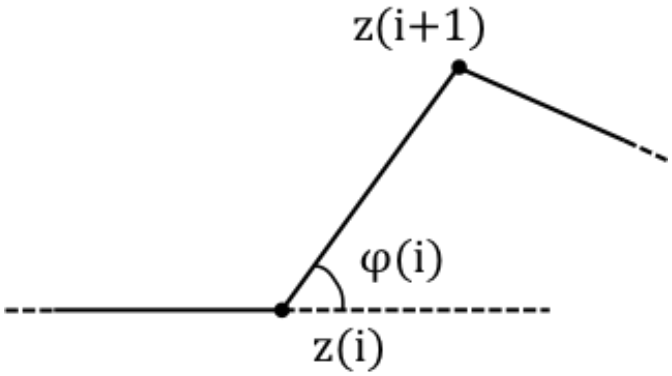

図 5　φ(i) の定め方

2.3.3 形状特徴量[15]

曲線Ｃの Ｐ表現 w から，複素関数 c を次のように求める.

$$C(k)=\frac{1}{n}\sum_{i=0}^{n-1}\omega(i)e^{-\frac{j2\pi ik}{n}} \quad (8)$$

(k=-n/2+1,...,n/2)

これを w の離散フーリエ変換といい，この C を折線図形Ｃの Ｐ形フーリエ記述子という．ここで，$C(k)$のうち，係数$k=-N,\dots,N$を開曲線の N 次の Ｐ形フーリエ記述子と呼び, これを C の形状特徴量$C_N(k)$とした. ある前髪の輪郭線を図 6 に, $n=5,25,50,100$とした時の近似した等辺多角形を以下の図 7 から図 11 に示す. N が大きくなるにつれて, 等辺多角形が元の輪郭線をよく表すことが分かる.

図 6　近似前の前髪の輪郭線

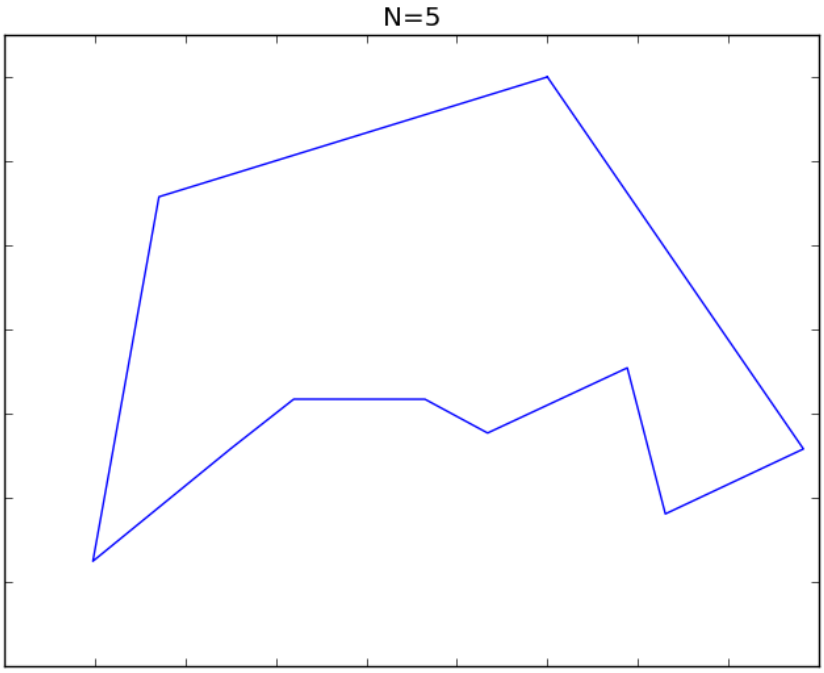

図 7　輪郭線を近似した等辺多角形（N＝５）

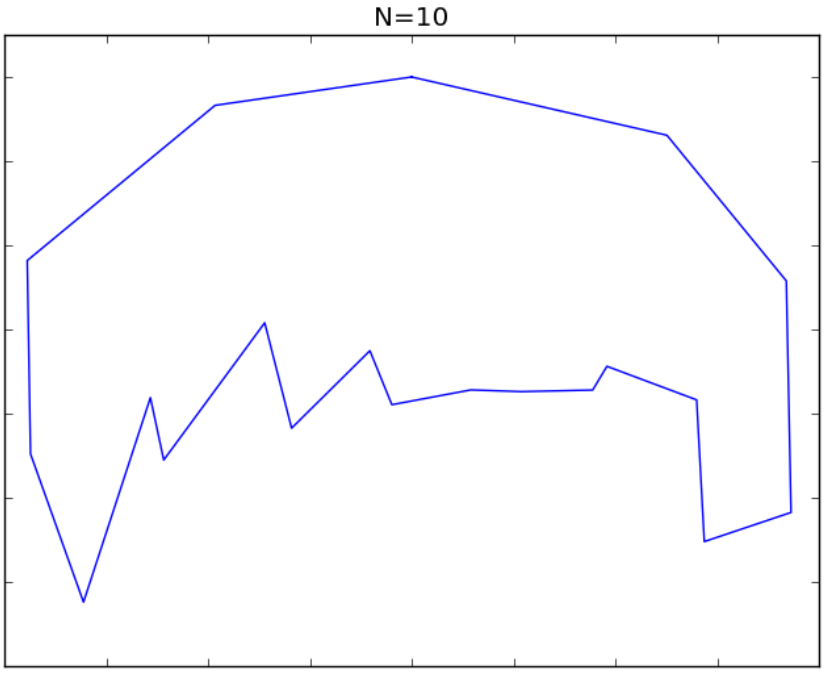

図 8　輪郭線を近似した等辺多角形（N＝１０）

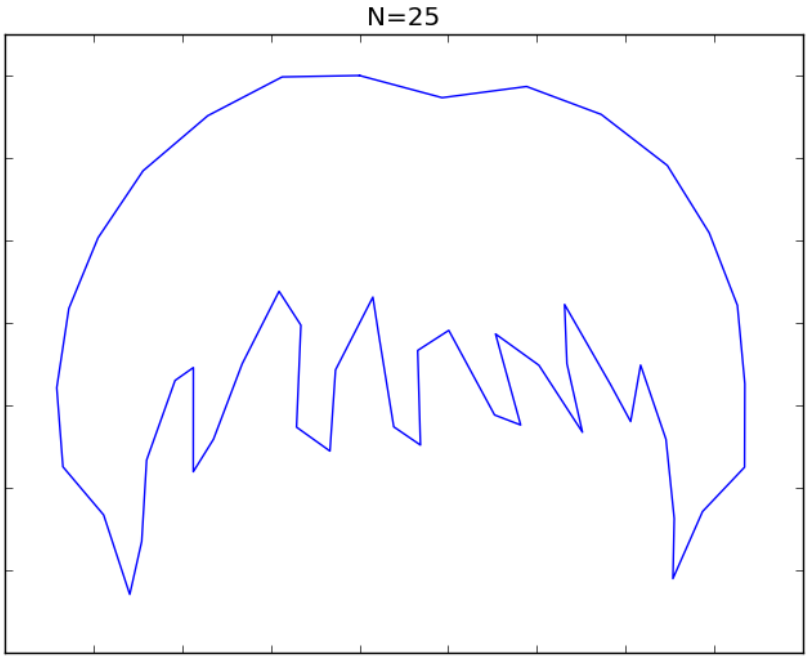

図 9　輪郭線を近似した等辺多角形（N＝２５）

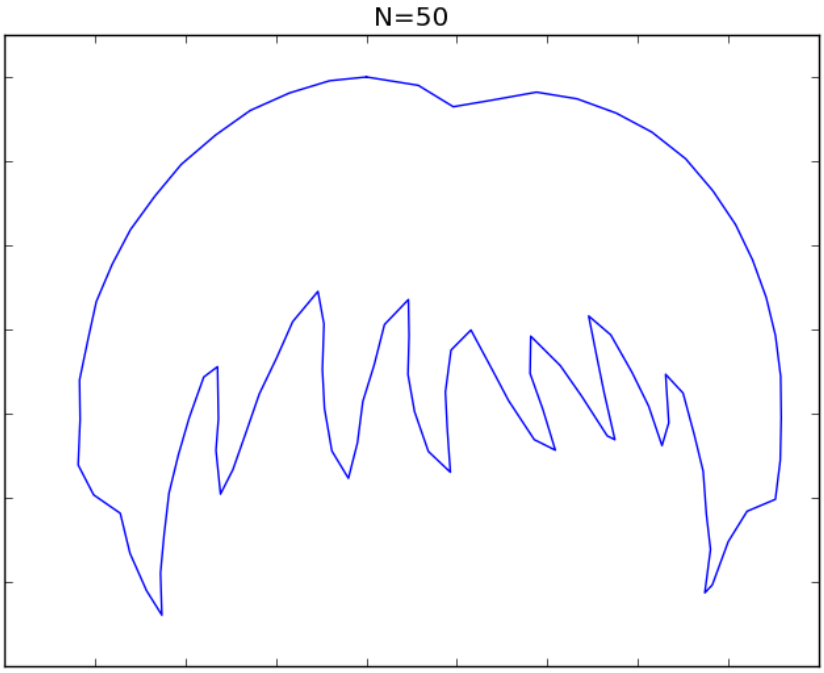

図 10　輪郭線を近似した等辺多角形（N＝５０）

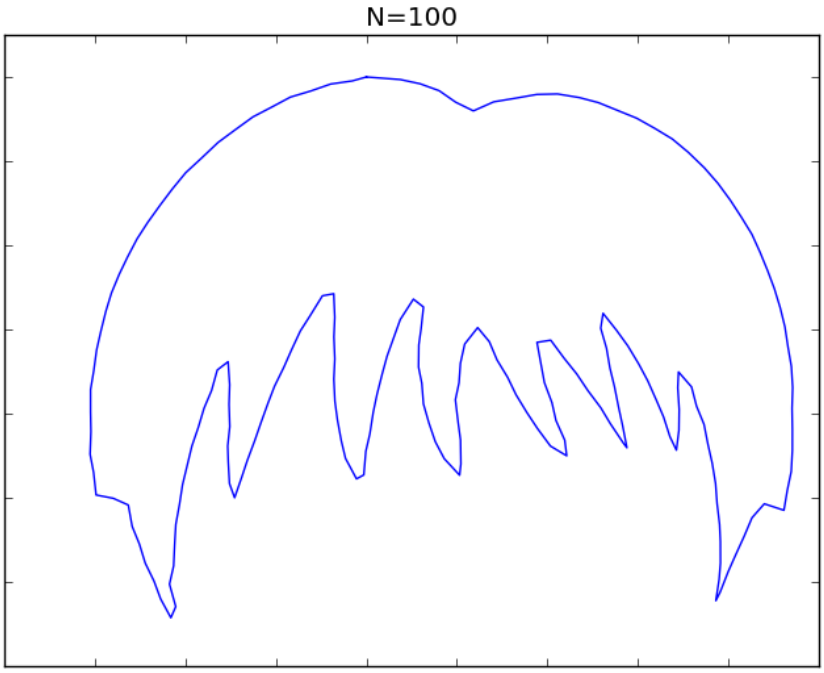

図 11　輪郭線を近似した等辺多角形（N＝１００）

2.3.4 類似度評価値[15]

開曲線a, bの相違度 $\varepsilon_{a,b}$ は，それぞれの形状特徴量$A_N(k), B_N(k)$より，以下の式を計算することで求められる．

$$\varepsilon_{a,b}(\varphi) = \sum_{k=-N}^{N} \left| A_N(k) - e^{j\varphi} B_N(k) \right|^2 \qquad (9)$$

$$(\phi = 0, \ldots, 2\pi)$$

前述の通り，Ｐ形フーリエ記述子は原曲線の平行移動及び拡大・縮小に不変で，また係数$e^{j\varphi}$をかけることで回転移動された原曲線の形状特徴量を表す．相違度$\varepsilon_{a,b}$は，曲線a, bが一致する場合０になり，曲線の形状が似ていないほど，大きい値を示す．今回はこれをそのまま，２曲線の相違度とし，識別の基準とした．

第３章

Average Hash を用いたキャラクター識別

3.1 まえがき

本章ではＰ形フーリエ記述子を用いた識別の前に行った, Average Hash を用いたキャラクター識別の実験方法とその実験結果，考察を示す.

3.2 実験方法

本実験では以下の条件で画像を用意した．例を図 12 に示す（ただしこの画像は実験に使用したものとは異なる）.

1. １つのマンガから，人物１から人物５までそれぞれ５枚ずつの２５枚
2. 髪の上と左右を若干の余白を残し画像の縁にあわせる
3. 縦横比が２：１になるよう手動で切り出し

また，本実験ではそのままの画像と，Canny フィルタによってエッジ検出を行った画像で，ハッシュ値を生成し相違度計算を行った．エッジ検出によってエッジは白で表現されるため，画像を縮小した際にエッジが多い部分がより白くなるため，前髪のきめ細

かさや，髪の輪郭の位置によって，特徴が出やすいと考えたためである．ここでエッジ検出に Canny 法を用いたのは，ノイズに強く，輪郭が綺麗に出るためである．エッジ検出にはオープンソースのコンピュータービジョン向けライブラリである OpenCV を利用した．また２つの閾値は(100,200)とした．元画像と元画像から作成したハッシュ値を図 13 に，Canny フィルタによってエッジ検出した画像とその後作成したハッシュ値を図 14 に示す．

切り出したすべての画像同士と，フィルタ適用後のすべての画像同士ハッシュ値のハミング距離を計算し，その値をそれぞれの相違度とした．

相違度が$256 * 0.35 = 89.6$以下であるものを同一キャラクターであると判断しその組み合わせの数を N，同一キャラクターを比較している組み合わせの数を C，N と C の両方に含まれている数を R としこの手法が識別に有効であるかどうかを判断した．

図 12　aHash の実験の画像切り出し例（画像は[10]より）

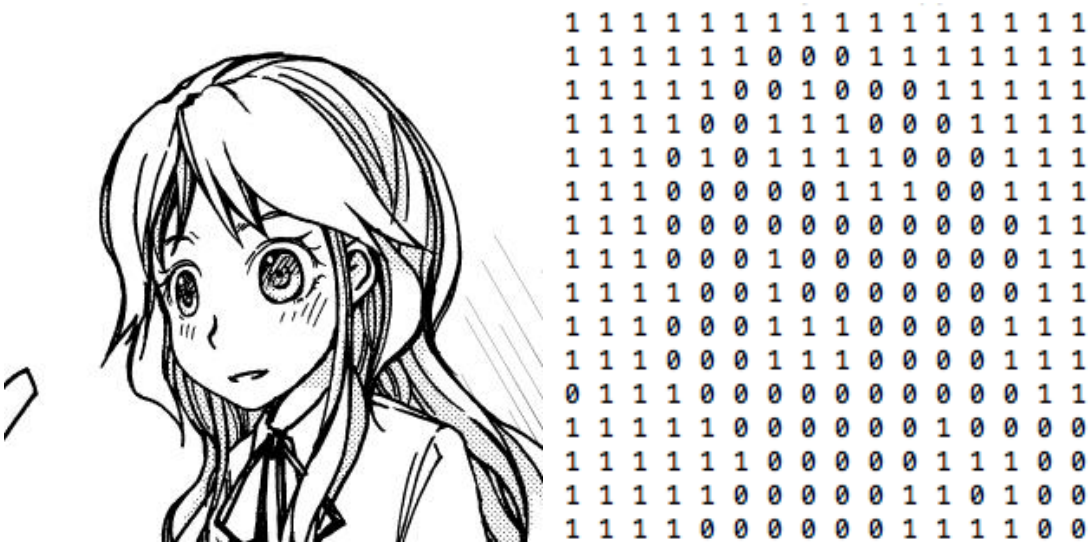

図 13　元画像とハッシュ値の作成結果（画像は[10]より）

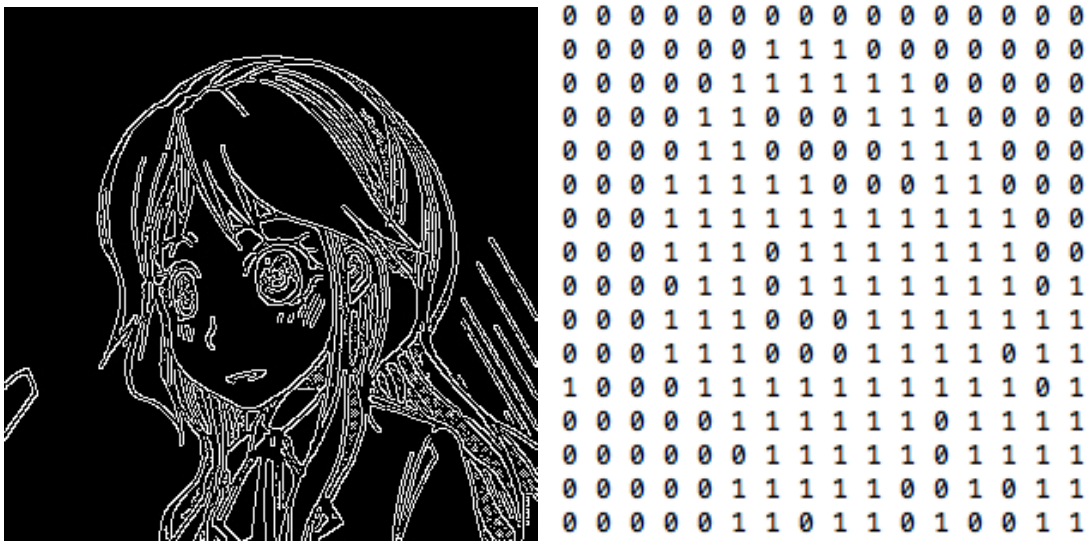

図 14　Canny フィルタ後のハッシュ値の作成結果（画像は[10]より）

3.3 実験結果

元画像の人物１から人物５それぞれと，それらの合計の C，N，R を表 1 に，Canny フィルタ適用後の人物１から人物５それぞれと，それらの合計の C，N，R を表 2 に示す.

表　1　元画像の aHash による識別結果

	人物１	人物２	人物３	人物４	人物５
N	85	65	64	70	15
C	25	25	25	25	25
R	25	15	15	19	15

表　2　フィルタ後の画像の aHash による識別結果

	人物１	人物２	人物３	人物４	人物５
N	18	5	6	36	30
C	25	25	25	25	25
R	7	5	5	25	15

3.4 考察

エッジ検出前，検出後も高い識別結果は得られなかった．しかしエッジ検出による若干の性能向上は確認でき，誤検出は大幅に減少した．これは，エッジ検出により，周りやパーツのノイズに左右されない識別となったためと考えられる．

しかし，人物５の場合は，エッジ検出後の方が誤検出多くなってしまっている．これは人物５の髪の色が白であるため，ハッシュ値に他の人物と大きな差がでているためと考えられる．この特徴を別に含ませれば識別率は更に向上すると思われる．

また，髪にスクリーントーンが貼ってあった場合，エッジ検出後の画像は図 15，図 16 のようにノイズが多くなってしまい，目的である髪の毛の輪郭の数をハッシュ値に反映させることができないため，正しい結果が得られなくなってしまう．

aHash はアルゴリズムが非常にシンプルで，実装も容易であったが，以下の図 17 に示すように，似ている画像でも１ビットのズレによって非常に大きな差が生まれてしまう．これは画像の切り出し方に依存するため，今後自動的に髪を切り出す場合においては重要な問題となってしまう可能性がある．また，回転・傾きに関しても同様で，同じ画像でも傾けた画像では全く異なるハッシュ値を示してしまう．

また，今回は頭頂部と横を基準にしたが，頭頂部まですべて描画されている数は少な

く，限定的な識別となってしまう．この限定的な条件下でも良い識別結果は得られず，この手法は効果的とはいえないだろう．この結果を受けて，P形フーリエ記述子を用いた識別を行った．

図 15　スクリーントーンを含む髪画像（画像は[10]より）

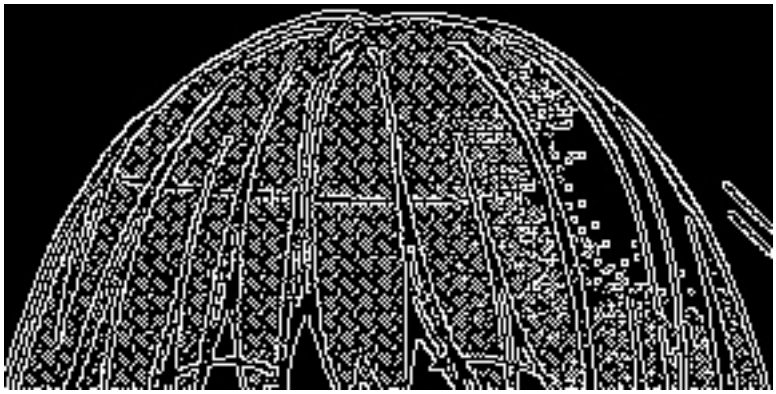

図 16　スクリーントーン有りの場合のエッジ検出結果（画像は[10]より）

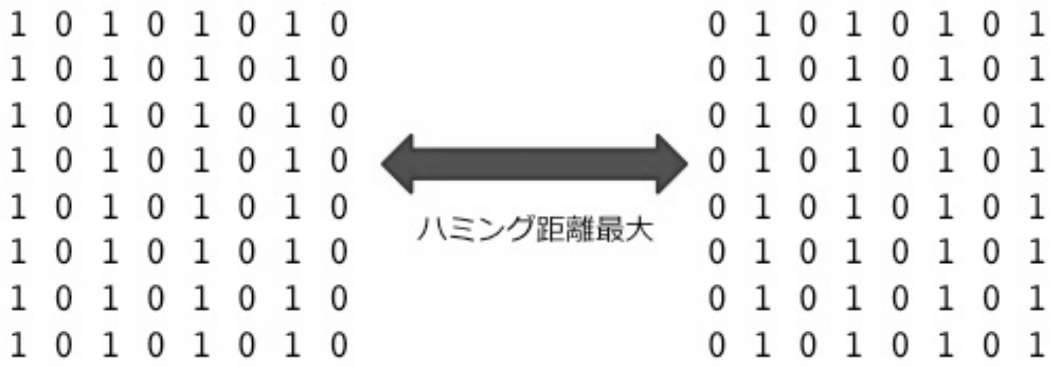

図 17　aHash における画像のずれによる差の例

第４章

Ｐ形フーリエ記述子を用いたキャラクター識別

4.1 まえがき

本章ではＰ形フーリエ記述子を用いたキャラクター識別の実験方法とその実験結果，考察を示す．

4.2 実験方法

本実験では，１つのマンガから判定用画像を人物１から人物５までそれぞれ１０枚ずつの５０枚，実験用画像を人物１から人物５までそれぞれ４枚ずつの２０枚用意した．切り出し方は以下のとおりである．

1. 前髪がすべて画像内に入るように切り出し
2. 前髪以外の部分は使用しないので，考慮しない
3. 広範囲に対応するため，横顔なども含むようにする

判定用画像と実験用画像すべてから以下のように形状特徴量を算出した．

1. 今回は耳の中央より上の部分にある髪を前髪とする
2. 前髪の輪郭を手動で取得
3. 輪郭の構成点を等距離に分割することで，曲線を等辺な多角形に近似
4. その近似した曲線から式（8）の形状特徴量を計算

手動で取り出した前髪の輪郭の例を図 18 に示す（赤線部分が前髪とした部分）．これは実験で使用した画像とは異なる．図 7 から図 11 より，N＝50 で十分髪の概形を表現できていると考え，今回の実験ではその値を使用した．また，回転移動の大きさを示すφは$\pi/36$ ごとにした．

求めた形状特徴量から，以下のとおり識別を行った

1. ある実験画像とすべての正解画像との相違度を式（9）により計算
2. その中で一番小さい値になった画像の人物であると判定
3. そこから正解・不正解を確認する

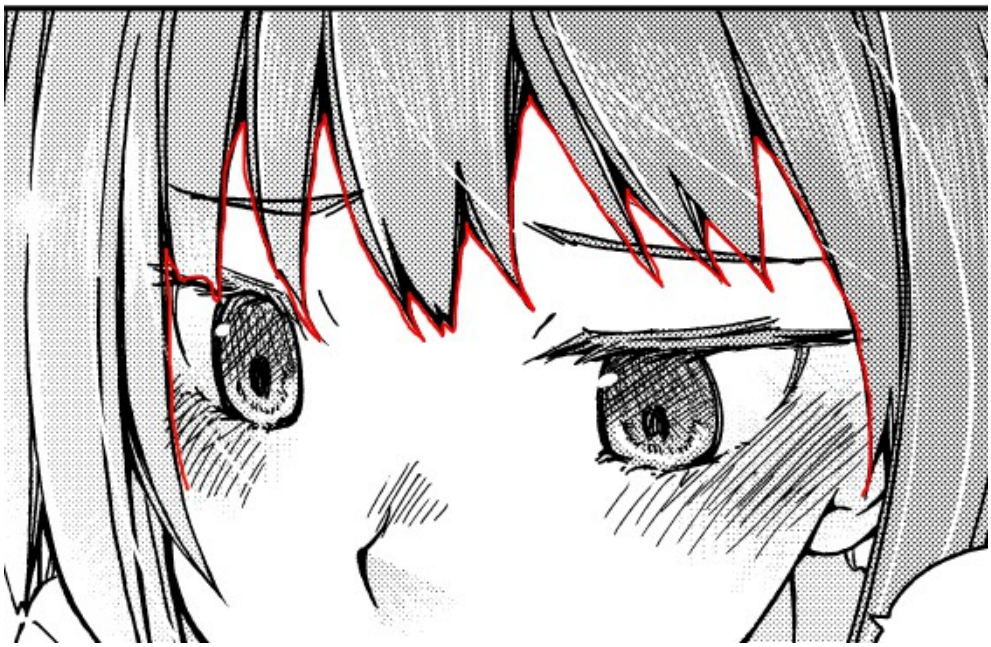

図 18　取得した前髪の例（画像は[10]より）

4.3 実験結果

人物１から人物５の識別結果を以下の表４．１に，またすべての画像における識別結果を表４．２に示す．

表 3　P形フーリエ記述子による識別結果（人物ごと）

	人物１	人物２	人物３	人物４	人物５
画像数	10	11	10	10	9
正解数	8	9	10	6	8
識別率	0.80	0.82	1.00	0.6	0.78

表 4　P形フーリエ記述子による識別結果

	すべての画像
画像数	50
正解数	41
識別率	0.82

4.4 考察

マンガキャラクターの前髪の輪郭からP形フーリエ記述子を用いて識別を行った結果，表 3 の通り，人物３の識別率が 1.00 であったのに対し，人物４が 0.60 と，キャラクターによって差はあったが，表４の通り全体では 0.82 と，比較的高い識別が得られた．よって前髪の形状はキャラクターごとに差があり，それらはP形フーリエ記述子を用いることで識別可能であることがわかった．

様々な開曲線で実験してみた結果, 曲線が複雑になればなるほど小さな形状の差で相違度が大きくなりやすいがわかった. 各頂点のずれによる各ベクトルの差が複雑な曲線であるほど大きくなり, 似た曲線を比較している場合でも全体として相違度が大きくなるためと考えられる．これを図 19 に示す．この頂点のずれはもともとの前髪の形状の差だけではなく，手動で取得した前髪の始点や終点にも依存するため，統一して取ることができれば，ずれが減り精度が向上する可能性がある．また，取得した前髪の輪郭から，それぞれ部分的に曲線の相違度を計算し，最小値を全体の相違度とすることで，特殊な形状の変化や，始点・終点のずれに対応できると考えられる．

また前髪の輪郭は手動で取得しているため，画像によって一様ではなく，その誤差によって正しい結果が出ていない場合もある．

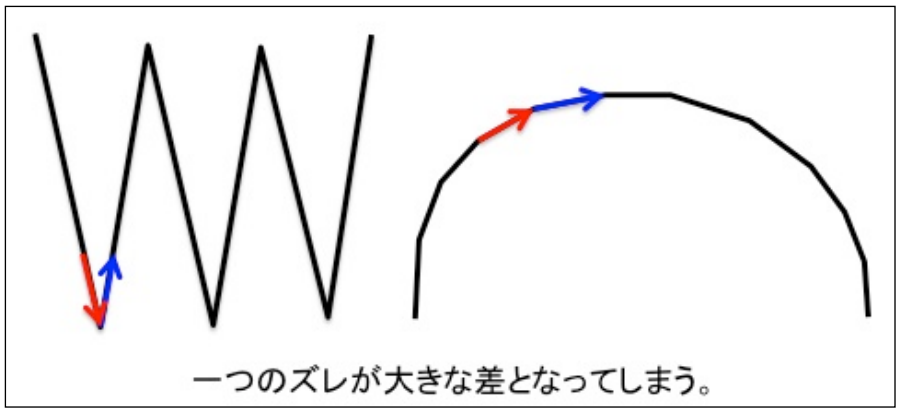

図 19　曲線の複雑さによる相違度の違いの例

第５章

結論

5.1 総括

本論文では漫画キャラクターの髪の輪郭から P 形フーリエ記述子を用いることで, キャラクター識別が８２%の識別率で行えることを示した．この結果から，P 形フーリエ記述子は，マンガキャラクターの前髪の形状の差を識別可能であり，それによるキャラクターの識別がある程度可能であることが確認できた.

5.2 課題

本論文での課題を以下に示す.

(1) 前髪輪郭の自動検出

(2) 式（9）において，N を変化させることで識別率が変化するかどうかの検証

(3) 他マンガでの検証

(4) 前髪の一部分での識別

(5) 評価方法の改善

(1)については，キャラクターをすべて自動で識別するのに不可欠である．しかし，顔のパーツとの混同や，作者によってタッチが全く異なるなど，現状では輪郭を綺麗に

取得するのは難しいだろう．(2)(3)は比較実験によって結果を得る必要がある．(4) を改良し，部分一致を探す必要がある．(5)については，本実験での評価方法は大変稚拙なもので，改善によって，識別率も向上する可能性がある．

謝辞

本研究の機会及び素晴らしい研究環境を与えて下さり，貴重な時間を割いてご指導頂きました渡辺裕教授に深謝いたします.

本研究において貴重なご意見やご指摘を頂き，様々環境を整えていただいた石井大祐助手に心から感謝致します.

日頃から貴重なご意見頂き，様々なご提案を頂きました研究室の皆様に御礼申し上げます.

本研究を行なうにあたって，コミック画像の提供および論文への掲載を許可頂いた木野陽様に心から感謝いたします.

最後に，私をここまで育ててくださった家族に心から感謝致します.

参考文献

[1] インプレス総合研究所 :“電子書籍ビジネス報告書 2014”, 株式会社インプレス, 2014/7/17.

[2] OnDeck :“書籍全体に締める電子書籍の割合は約 8%に-米国市場動向と比較しながら今後を見る”, http://on-deck.jp/archives/634, 参照 Jan,14,2015.

[3] システム論フォーラム : “どうすれば電子書籍は普及するのか”,http://www.systemicsforum.com/ja/viewtopic.php?f=33&t=146, 参照 Jan,14,2015.

[4] THE PAGE :“日本で電子書籍は普及しないのか”,http://thepage.jp/detail/20131001-00000001-wordleaf, 参照 Jan,14,2015.

[5] 集英社 : “少年ジャンププラス”, http://plus.shonenjump.com/, 参照 Jan,14,2015.

[6] 講談社 :“講談社からのお知らせ”, http://www.kodansha.co.jp/upload/pr.kodansha.co.jp/files/pdf/20150105comic.pdf.

[7] 中川雄貴, 坂本博康 :”色構造グラフを用いたアニメキャラクターの画像間の類似度測定”, 情報処理学会 火の国情報シンポジウム 2013, B4-3, pp1-3, 2013.

[8] 高山耕平,ヘンリージョハン,西田友是 : “特徴抽出によるアニメキャラクターの顔認識”, IEVC2012, 2012-11.

[9] 河谷大和 : “アニメキャラクターの特徴抽出に基づくアニメ度の評価”, 情報処理学会研究報告, vol2008, no.80, pp35-38, 2008-08.

[10] 木野陽 :”ベリーベリークリームショコラ ふたつのベリー”, 自己出版, 2010.

[11] Dr. Neal Krawetz : “The Hacke Factor Blog”, http://www.hackerfactor.com/blog/?/archives/432-Looks-Like-It.html, 参照 Jan,14,2015.

[12] 上坂吉則 : ″開曲線にも適用できる新しいフーリエ記述子″, 電子通信学会論文誌, vol. J67-A, No. 3, pp.166-173, 1984.

[13] C. T. Zahn and R. S. Roskies : "Fourier descriptors for plane closed curves", IEEE Trans.Comput., Vol. C-21, pp. 269-281, March 1972.
[14] G. H. Granlund : "Fourier preprocessing for hand print character recognition", IEEE Trans. Comput., vol. C-21, pp.195-201, 1972.
[15] 加納政芳, 加藤昇平, 伊藤英則 : "判別の難易度に基づく類似箇所検出の高速化", 情報処理学会論文誌, Vol.42, No.11, pp.2689-2698, 2001.

付録 使用したソースコード

```
#!/usr/local/bin/python
#coding: UTF-8

import sys
import cmath
import cv2
import numpy
import matplotlib.pyplot as plt

MIN_DESCRIPTOR = 100    #N*2
ANGLE = 72            #回転の分割数


def mousePaint(event,x,y,flags,param):
    global ix,iy,count,tmp,drawing,fin
    if event == cv2.EVENT_LBUTTONDOWN:
        drawing = True
        ix,iy = x,y
    elif event == cv2.EVENT_MOUSEMOVE:
        if drawing == True:
            tmp[count] = x,y
            cv2.line(img,(ix,iy),(x,y),(0,0,255,0),1)
            ix,iy = x,y
```

```
            count += 1
    elif event == cv2.EVENT_LBUTTONUP:
        drawing = False
        fin = True

# 輪郭を等辺多角形に近似
def equalIntaval(tmp):
    bangs = numpy.memmap('bangs.dat',dtype='complex',mode='w+',shape=(100000))
    l = tmp.shape[0]
    num = 0
    for n in range(0,l-1):
        mx = tmp[n+1][0]-tmp[n][0]
        my = tmp[n+1][1]-tmp[n][1]
        length = numpy.linalg.norm(tmp[n+1]-tmp[n])
        div = round(length*10)
        for m in range(0,int(div)):
            bangs.real[num] = tmp[n][0] + m*(mx/div)
            bangs.imag[num] = tmp[n][1] + m*(my/div)
            num += 1
        bangs.real[num] = tmp[n+1][0]
        bangs.imag[num] = tmp[n+1][1]
    bangs = numpy.delete(bangs,numpy.where(bangs==0)[0],0)
    return bangs

# P表現を求める
def makePRepresentation(contour):
    prep = numpy.memmap('prep.dat',
                        dtype='complex',
                        mode='w+',
                        shape=(len(contour)-1))
    length = 0.1
    for n in range(0,len(contour)-1):
        prep[n] = (contour[n+1]-contour[n])/length
```

```
    return length, prep


# 形状特徴量計算
def shapeFeature(prep,degree):
    l = len(prep)
    fourier_descriptors = numpy.fft.fft(prep) / l
    fourier_descriptors = numpy.fft.fftshift(fourier_descriptors)
    center_index = len(fourier_descriptors) / 2
    fourier_descriptors = fourier_descriptors[center_index - degree /
2:center_index + degree / 2]
    shape_feature = numpy.fft.ifftshift(fourier_descriptors)
    return shape_feature

# 回転に不変な相違度計算
def dissimilarity(shape_feature1,shape_feature2):
    tmp_diff = numpy.zeros((ANGLE,MIN_DESCRIPTOR), dtype=float)
    difference = numpy.zeros(ANGLE, dtype=float)
    shape_feature = numpy.zeros(MIN_DESCRIPTOR, dtype=complex)
    for n in range(0,ANGLE):
        coefficient = 1j*n*(2*cmath.pi/ANGLE)
        coefficient = cmath.exp(coefficient)
        shape_feature[:] = shape_feature1[:] * coefficient
        for m in range(0,MIN_DESCRIPTOR):
            tmp_diff[n][m] = numpy.abs(shape_feature2[m] - shape_feature[m])
            tmp_diff[n][m] **= 2
        difference[n] = numpy.sum(tmp_diff[n])
        index = numpy.argmin(difference)
    return index,min(difference)

#輪郭を手動で取得する部分
def getBangs(name):
    global img,ix,iy,count,tmp,drawing,fin
```

```
    tmp = numpy.memmap('tmp.dat',dtype='int',mode='w+',shape=(10000,2))
    drawing = False
    fin = False
    ix,iy = -1,-1
    count = 0
    img = cv2.imread(name)
    cv2.namedWindow('image')
    cv2.setMouseCallback('image',mousePaint)
    while(1):
        cv2.imshow('image',img)
        k = cv2.waitKey(1) & 0xFF
        if fin == True:
            break
        elif k == 27:
            break
    cv2.destroyAllWindows()
    tmp = numpy.delete(tmp,numpy.where(tmp==0)[0],0)
    return equalIntaval(tmp)

def getShapeFeature(sample):
    contour = getBangs(sample)
    length, prep = makePRepresentation(contour)
    shape_feature = shapeFeature(prep,MIN_DESCRIPTOR)
    return shape_feature

def compare(sample1,sample2):
    shape_feature1 =  getShapeFeature(sample1)
    shape_feature2 =  getShapeFeature(sample2)
    index, difference = dissimilarity(shape_feature1,shape_feature2)
    return index, difference

argv = sys.argv
if (len(argv) == 1):
```

```
        test = "test.bmp"
        index, answer = compare(test,test)
        print 360/ANGLE*index, answer
        exit()
elif (len(argv) == 2):
    sample = argv[1]
    test = "test.bmp"
    index, answer = compare(sample,test)
    print 360/ANGLE*index, answer
    exit()

sample1 = argv[1]
sample2 = argv[2]
index, answer = compare(sample1,sample2)
print 360/ANGLE*index, answer
```

図一覧

表一覧