Poisson 方程式の固有値評価を用いた画像認識

1W080154-2　菊井　知美

早稲田大学　基幹理工学部　応用数理学科

指導教授：大石　進一

2012 年 2 月 7 日

1　はじめに

本論文では、計算機によって画像の手の形を識別することを試みた。画像イメージを MATLAB 環境に読み込んで、それらの固有値分布の解析を行い手の形を識別する。今回は例として、グーとチョキとパーを識別することを考えた。

まず、Poisson 方程式の固有値問題の離散化の手法として有限要素法を利用するために有限要素法の理解に努めた。そして、人間の目で固有値分布を比較するだけではなく、計算機に正しく識別させられるようにすることを目標とし、SVM(Support vector machine) というパターン認識手法について勉強し、LibSVM を用いて計算機による手の形の識別の実験を行なった。

2　有限要素法

2.1　有限要素法とは

有限要素法 (FEM) とは、偏微分方程式で記述される現象を近似的に解くための離散化手法、計算手法として有効な数値解法である。 微分方程式を直接に扱うわけではなく、対象領域内で成り立つ方程式に対してある重み関数の積を施し、それを領域内で積分した弱形式を形成する。領域全体の解の挙動は要素内の積分の総和として表すことができるので、まず対象領域内部を微小で単純な「要素」に分割し、各要素ごとの解析を行う。各要素における係数行列の総和を取って領域全体の係数行列を作成し、解を求めることができる。

有限要素法では、解析対象を単純な要素に分割して解析を行うため、計算対象が複雑な形状や性質を持っていても、比較的簡単な解析が適用可能であるという利点がある。また、より細かい要素へと分割すればするほど、解析の近似値の精度が向上する。ただし、細かく分割する場合、多くの計算処理が必要である。

2.2　偏微分方程式の弱形式

n 次元空間 $(n = 1, 2, 3)$ を考え、点を $x = (x_1, \cdots, x_n)$ と表す。Ω を空間内の有界な領域とし、Γ はその境界で十分なめらかとする。境界 Γ を重なりがない 2 つの部分 Γ_1 と Γ_2 に分ける。Ω 内で定義された関数 f、Γ_1 上で定義された関数 g_1、Γ_2 上で定義された関数 g_2 を与える。

次の条件を満たす関数 $u = u(x)$ についての境界値問題を考える。

$$\begin{cases} -\Delta u = f \ \text{ in}\,\Omega \cdots (1.1) \\ u = g_1 \ \text{ on}\,\Gamma_1,\ \frac{\partial u}{\partial \nu} = g_2 \ \text{ on}\,\Gamma_2 \cdots (1.2) \end{cases} \tag{1}$$

v を、

$$v = 0 \ \text{ on}\,\Gamma_1 \tag{2}$$

を満たす任意の関数とする。(1.1) の両辺に v をかけ、Ω で積分する。また、(1.2) の第 2 式に v をかけ、Γ_2 上で積分し、先の式に加える。すなわち、

$$-\int_\Omega (\Delta u)\, v\, dx + \int_{\Gamma_2} \frac{\partial u}{\partial \nu}\, v d\gamma = \int_\Omega f v dx + \int_{\Gamma_2} g_2 v d\gamma \tag{3}$$

いま、Green の公式を用いて (3) の左辺を変形すれば、次式を得る。

$$\sum_{i=1}^{n} \int_\Omega \frac{\partial u}{\partial x_i} \frac{\partial v}{\partial x_i} dx - \int_\Gamma \frac{\partial u}{\partial \nu} v d\gamma + \int_{\Gamma_2} \frac{\partial u}{\partial \nu}\, v d\gamma = \int_\Omega f v dx + \int_{\Gamma_2} g_2 v d\gamma$$

ここで、$v = 0$ on Γ_1 に注意して整理すれば、次の弱形式を得る。

$$\sum_{i=1}^{n} \int_\Omega \frac{\partial u}{\partial x_i} \frac{\partial v}{\partial x_i} dx = \int_\Omega f v dx + \int_{\Gamma_2} g_2 v d\gamma \tag{4}$$

以上より、もとの問題 (1) の代わりに次の問題を考えても同じであることがわかる。

(2) を満たす任意の v に対し、弱形式 (4) を成立させ、しかも (1.2) の第 1 式を満たす関数 $u = u(x)$ を見い出せ。

これから、弱形式を用いて境界値問題を解いていくことを考える。

2.3　簡単な 1 次元の有限要素モデル

まず、次の常微分方程の境界値問題を考えた。

$$\begin{cases} -\frac{d^2 u}{dx^2} = f \ \ (0 < x < 1) \cdots (5.1) \\ u(0) = u(1) = 0 \ \ \cdots (5.2) \end{cases} \tag{5}$$

ここで、次の記号を定義する。

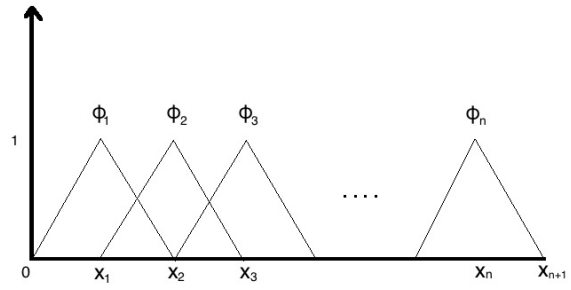
$$(u,v)=\int_\Omega uvdx \tag{6}$$

この記号を用れば、この問題の弱形式は次のように表せる。

$$(\nabla u,\nabla v)=(f,v) \tag{7}$$

u は境界値条件 (5.2) を満たす関数であり、v は $v(0)$=0 を満たす任意の関数である。
区間 $(0,1)$ を $n+1$ 個の有限要素に分割し、基底関数 $\phi_i\,(i=1,\cdots,n)$ を用いて、関数 u の近似関数 $\tilde{u}$ を作る。基底関数 ϕ_i は図 1 のような関数で定義する。

図 1　基底関数



関数 u は次の形で近似する。c_i は節点パラメータを表し、$c_i=u_i(x_i)$ とする。

$$\tilde{u}=\sum_{i=1}^{n}c_i\phi_i \tag{8}$$

ここで、近似関数 $\tilde{u}$ と $v=\phi_i$ を弱形式 (7) に代入すれば、次の式が得られる。

$$(\nabla\tilde{u},\nabla\phi_i)=(f,\phi_i)\quad(i=1,\cdots,n) \tag{9}$$

この近似関数の弱形式に (8) を代入して計算すると次を得る。

$$(f,\phi_i)=c_1(\nabla\phi_1,\nabla\phi_i)+\cdots+c_n(\nabla\phi_n,\nabla\phi_i)$$

上式より、$i=1,\cdots,n$ の全体は、

$$Ax=b \tag{10}$$

という行列の形で簡略化して表すことができる。
ただし、

$$A_{ij}=(\nabla\phi_j,\nabla\phi_i)$$

$$x=\begin{pmatrix}c_1\\ \vdots\\ c_n\end{pmatrix},b=\begin{pmatrix}(f,\phi_1)\\ \vdots\\ (f,\phi_n)\end{pmatrix}$$

である。このとき、A_{ij} は対称行列になる。
式 (10) は、まず各要素ごとのマトリックスを求め、その和をとり、拡大した近似方程式として組み立てる。この手法は直接剛性法と呼ばれていて、機械的に実行でき、コンピューターによる自動処理に適している。実際に、$f=\bar{f}$(定数関数) としたときの解を求めるプログラムを MATLAB で書いてみた。以下に示す。

```
function x = solve_bound_fem(n,f)
  m=n-1;
  A=zeros(m,m);
  A(1,1)=2*n;
  A(m,m)=2*n;
  A(1,2)=-n;
  A(m,m-1)=-n;
     for i=2:m-1;
         A(i,i)=2*n;
         A(i,i+1)=-n;
         A(i,i-1)=-n;
     end
  b=zeros(m,1);
     for i=1:m;
         b(i,1)=(f/n);
     end
     %boundary value problem (Poinsson's problem)
      x=A\b;
end
```

2.3.1　１次元の固有値問題

次に、固有値問題について考えた。
下記の方程式を満たす滑らかな関数 u と固有値 λ を求める。

$$\begin{cases}-\Delta u=\lambda u\ \ \text{in}\,\Omega\cdots(11.1)\\ u=0\ \ \text{on}\,\Gamma\cdots(11.2)\end{cases} \tag{11}$$

(11.2) 式は境界条件である。
(11.1) を弱形式で表すと、次を得る。

$$\int_\Omega\nabla u\nabla vdx=\lambda\int_\Omega uvdx \tag{12}$$

前節の記号を用いれば、

$$(\nabla u,\nabla v)=\lambda(u,v) \tag{13}$$

と表せる。
ここで、u は境界条件 (11.2) を満たす関数、v は $v=0\,\text{on}\,\Gamma$ を満たす任意の関数である。(13) 式は (7) 式と同じような形をしているといえる。前節と同様に、u の近似関数 $\tilde{u}$ と、基底関数 $\phi_i\,(i=1,\cdots,n)$ を用いて、近似方程式を (14) 式の行列の形で表現した。

$$Ax=\lambda Bx \tag{14}$$

ただし、

$$A_{ij} = (\nabla\phi_j, \nabla\phi_i)$$

$$B_{ij} = (\phi_j, \phi_i), x = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$$

である。このとき A,B は対称行列である。式 (14) は一般化固有値問題といわれる形である。MATLAB を用いて、固有値 λ と固有ベクトル x を求めるプログラムを書いた。以下に示す。

```
function [V,lambda] = solve_eig_fem(n)
  m=n-1;
  A=zeros(m,m);
  A(1,1)=2*n;
  A(m,m)=2*n;
  A(1,2)=-n;
  A(m,m-1)=-n;
    for i=2:m-1;
       A(i,i)=2*n;
       A(i,i+1)=-n;
       A(i,i-1)=-n;
    end
  B=zeros(m,m);
  B(1,1)=(2/3)*(1/n);
  B(m,m)=(2/3)*(1/n);
  B(1,2)=(1/6)*(1/n);
  B(m,m-1)=(1/6)*(1/n);
      for i=2:m-1;
      B(i,i)=(2/3)*(1/n);
      B(i,i+1)=(1/6)*(1/n);
      B(i,i-1)=(1/6)*(1/n);
     end
% eigenvalue problem
  [V,lambda]=eig(A,B);
  lambda = diag(lambda);
end
```

2.4 簡単な２次元の有限要素モデル

続いて、１次元の問題の考え方を利用して、２次元の問題について考えた。基本的な考え方は変わらないが、有限要素分割や近似関数の構成が複雑である。下の式 (15) の２次元領域の固有値問題について考えた。

$$\begin{cases} -\Delta u = \lambda u \ \text{ in}\,\Omega \cdots (15.1) \\ u = 0 \ \text{ on}\,\Gamma \cdots (15.2) \end{cases} \tag{15}$$

2.4.1 ２次元の固有値問題

１次元の問題と同様に、領域 Ω を有限要素に分割し、近似関数と基底関数を用いて近似方程式の行列を構成して解く。ただし、２次元では三角形要素によって分割を行い、基底関数はピラミッド形の関数を用いる。構成される拡大行列の形は１次元のときと同じで、

$$Ax = \lambda Bx \tag{16}$$

である。ただし、

$$A_{ij} = (\nabla\phi_j, \nabla\phi_i)$$

$$B_{ij} = (\phi_j, \phi_i), x = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$$

である。

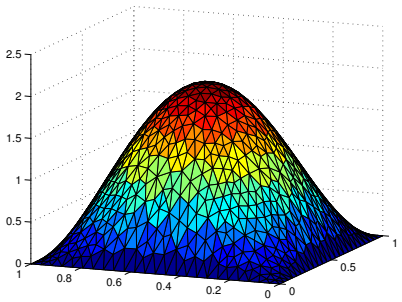
まず領域の三角形要素への分割 (メッシュ分割) を行い、節点座標を得て、それぞれの要素内での局所的な節点番号と全体的な節点番号への対応付けをすることが必要である。今回は、既存のオンライン固有値計算ができるサイト [?] を利用してそれらの情報を得た。このサイトでは、Gmsh を使っていて、任意の解析したい多角形領域を手動で入力することができる。メッシュのサイズを指定することも可能である。
このサイト [?] で得たデータを MATLAB にロードすることによって任意の多角形領域でこの固有値問題の固有値 λ と固有ベクトル x を求めるプログラムを作成した。また、求められた固有ベクトルから固有関数 (近似関数) の描画をして挙動を検証できるようにした。第１固有値による固有関数の解の挙動を図２に示した。

```
function[A,B,lambda,V,x,y,s]=fem_2D(tri,node,bdnode,w)

  n1=size(node,1);
  n2=size(bdnode,1);
  n=n1-n2;
  m=size(tri,1);
  GlobalInd=1:n1;
  InnerInd = GlobalInd;
  InnerInd(bdnode)=inf;
  %Global node to inner node
  [res,G2I] = sort(InnerInd);
  [res,G2I] = sort(G2I);
  GlobalInd(bdnode)=0;
  A=zeros(n,n);
  AE=zeros(3,3);
  B=zeros(n,n);
    %要素係数行列をつくる
    for q=1:m
      ind=[1,2,3];
      shift=[2,3,1];
        for r=1:3
         ind=ind(shift);
         li=ind(1);
         lj=ind(2);
         lk=ind(3);
         gi=tri(q,li);
         gj=tri(q,lj);
         gk=tri(q,lk);
         xi=node(gi,1);
         yi=node(gi,2);
         xj=node(gj,1);
         yj=node(gj,2);
         xk=node(gk,1);
         yk=node(gk,2);
      ss=(xi*(yj-yk)+xj*(yk-yi)+xk*(yi-yj))/2;
      s=abs(ss);
```

```
      %要素係数行列の対角成分
      AE(li,li)=((-yk+yj)^2+(xk-xj)^2)/(4*s);
      %要素係数行列の対角以外
      AE(li,lj)=((-yk+yj)*(-yi+yk)+(xk-xj)*(xi-xk))/(4*s);
      AE(lj,li)=((-yk+yj)*(-yi+yk)+(xk-xj)*(xi-xk))/(4*s);
      %拡大行列の構成
      if GlobalInd(gi) >0
        A( G2I(gi),G2I(gi) )=A( G2I(gi),G2I(gi) )+AE(li,li);
        B( G2I(gi),G2I(gi) )=B( G2I(gi),G2I(gi) )+s/6;
      end
      if GlobalInd(gi) >0 && GlobalInd(gj)>0
       A(G2I(gi),G2I(gj))=A(G2I(gi),G2I(gj))+AE(li,lj);
       A(G2I(gj),G2I(gi))=A(G2I(gj),G2I(gi))+AE(li,lj);
       B(G2I(gi),G2I(gj))=B(G2I(gi),G2I(gj))+s/12;
       B(G2I(gj),G2I(gi))=B(G2I(gj),G2I(gi))+s/12;
      end
    end
  end
    [V,lambda]=eig(A,B);
    lambda = diag(lambda);
    %第 w 固有値についての固有関数を求める
    x=node(:,1);
    y=node(:,2);
    z=V(:,w);
    n=size(node,1);
    s=ones(n,1);
    s(bdnode)=0;
    j=1;
     for i=1:length(s)
        if s(i) == 1
            s(i)=z(j);
            j=j+1;
        end
    end
  %固有関数の描画
  mytri=TriRep(tri,x,y,s);
  trisurf(mytri)
end
```

図 2　第 1 固有値の固有関数



3　有限要素法を用いた画像の固有値評価

実際に、MATLAB の imread 関数を使って手の画像ファイルを読み込み、重み関数を構成し、前章で作った二次元領域固有値問題の有限要素法のプログラムを用いて、固有値の評価を行った。
まず、解くべき問題を弱形式で表すと下の式 (17) になる。

$$\int_\Omega p(x)\nabla u\nabla v dx = \lambda \int_\Omega p(x)uvdx \qquad (17)$$

ここで、$p(x)$ は、ピクセル値から構成した重み関数を表す。今回はプログラムの簡略化のために、手のある領域で値が 1、それ以外の領域で値が 0 となるように閾値処理をした後の、各三角形有限要素内のピクセル値の平均値を $p(x)$ とした。
また、要素分割は全て合同な整列した直角三角形に限定する事にして固有値を求めるプログラムを作成した。

3.1　目視による固有値分布の比較

まず、MATLAB で作成したプログラムから求められた固有値分布を目視で比較してみた。
比較には、自分の手のグーとチョキとパーの写真をそれぞれ 10 枚ずつ撮影しそれを用いた。図 3 に示す。

図 3　実験に用いた手の画像

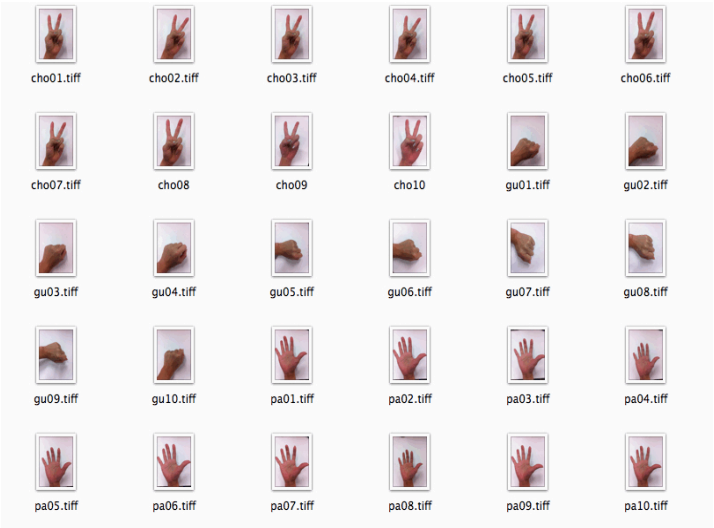


図 3 のそれぞれの画像イメージから得られる固有値のうち、比較対称はそれぞれの第 1 固有値から第 8 固有値までとした。あまり多くの固有値を求めようとするとプログラムの実行に余計な時間がかかってしまうのでなるべく少なく、かつそれぞれの形の特徴が読み取れるようにするために第八固有値までがよいと考えたからである。固有値分布を図 4 に示す。ただし、それぞれの第 1 固有値の値が 1 となるように正規化した。
それぞれの固有値分布にはおおまかな特徴があることが見て取れる。また、手の向きや大きさは固有値分布には影響を及ぼさない。

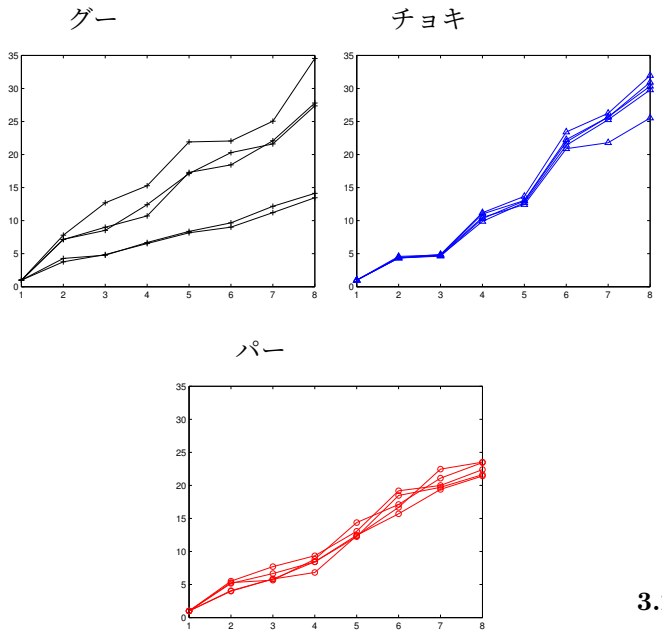
3.2　LibSVM を用いた画像の分類

次に、LibSVM を用いてより詳細に固有値分布の検証を計算機で行った。検証の対称としたのは、先ほどと同じ 30 個のデータである。

3.2.1　LibSVM とは

この実験に用いた LibSVM とは、台湾国立大学の Chih-Len Lin 氏らによって作られた教師あり学習を用いる識別手法の一つである SVM のアルゴリズムを

図 4　固有値分布



利用するためライブラリである。サポートベクタ分類器（C-SVC、nu-SVC）、回帰分析（epsilon-SVR、nu-SVR）、分布評価（1 クラス SVM）のための統合ソフトである。サポートベクターマシンは、現在知られている多くの手法の中で一番認識性能が優れた学習モデルの一つである。未学習データに対して高い識別性能を得るための工夫がされているため、すぐれた認識性能を発揮することができる。

3.2.2　LibSVM の使用方法

LibSVM は、svm-train で model を構築し、svm-predict によって予測を行う。今回の実験では MATLAB 環境に LibSVM を実装し画像の分類を試みた。その具体的な手順を以下に示す。

1. 公式サイト [?] から LIBSVM のソースコードをダウンロードし、MATLAB に展開する。

2. グーチョキパーそれぞれ 10 種類ずつの画像イメージの固有値分布を計算して得られたデータに、ランダムに 1 から 10 の index をつけた。

3. データを訓練用と評価用の 2 つのデータファイルに分ける。先ほどの index の 1 から 5 を訓練用、6 から 10 を評価用とした。

4. svm-train を以下のように関数として扱い、訓練セットから予測のための model を生成する。

```
model = svmtrain(A,B);
```

ただし、
`A=training_label_vector : m*1` ベクトル
`B=training_instance_matrix : m*n` 行列

今回は、`m=5,n=8`

5. svm-predict を以下のように関数として扱い、予測を行う。

```
C= svmpredict(D,E, model);
```

ただし、
`C=predicted_label:h*1` ベクトル
`D=testing_label_vector:h*1` ベクトル
`E= testing_instance_matrix:h*n` 行列

今回は、`h=5,n=8`

3.2.3　LibSVM による分類の実験結果と結論

表 1 に、実験の結果を示した。

	結果	成功	失敗	成功率
グー (g)	ggggg	5	0	100 %
チョキ (c)	cccgg	3	2	60 %
パー (p)	pgpgp	3	2	60 %

表 1　実験結果

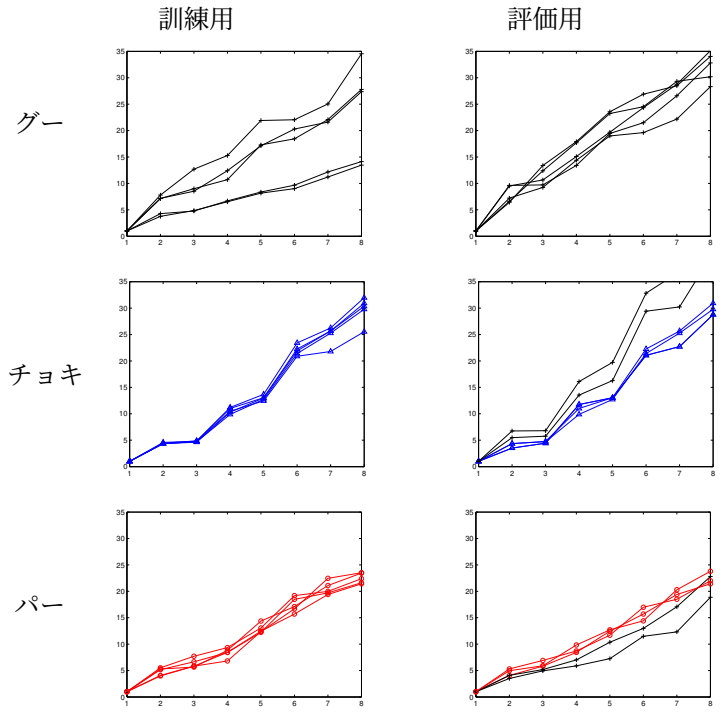
訓練用に用いた分布と評価用に用いた分布を次に示す。評価用のグラフは、グーと識別されたものを黒色 (+)、チョキと識別されたものを青色 (△)、パーと識別されたものを赤色 (○) で描画した。訓練用も同じく、グーを黒色、チョキを青色、パーを赤色としている。

表 1 の実験結果に示したように、グーは全て正しく識別されたが、パーとチョキは 2 つ失敗であった。そして、識別に失敗したものはグーとして認識された。確かに訓練用のグーのデータの中に似ているものが存在している事が見て取れる。訓練用のデータをもっと増すことによって結果が変わることが考えられる。まだプログラムの改善も必要であると考えられる。

4　結論と今後の課題

今回の検証で、MATLAB を使って画像イメージを読み込み、その固有値分布からもとの画像の識別をすることは可能であると言えそうだ。しかし、まだ成功率が低いので改善する必要がある。そのためにまず、訓練用の固有値データを増やすことが必要だと考えられる。次に、今回は整列した三角形要素分割に限定して固有値を数値計算で求めたが、より正確な固有値を

図 5　検証に用いた固有値分布



求めるには、それぞれの画像に対して最良な分割を自動で行えるようはプログラムが望ましい。ほかにも、閾値は定数としてしまったが、関数として扱えるようにすることが望ましい。閾値処理の後に、近傍処理などを行えばもっときれいに領域抽出ができる。また、今回は第一固有値から第八固有値を比較に用いたが、実際は他の固有値を用いた方が良かったかも知れないので、どのような情報を使うべきなのか検討したい。

参考文献

[1] 菊池文雄: **有限要素法概説 理工学における基礎と応用**, サイエンス社, 1999.

[2] Liu Xuefeng: **Evaluation of Laplacian Eigenvalues on Arbitrary Polygonal Domain**(http://www.xfliu.org/polyeig/)

[3] Chih-Chung Chang and Chih-Jen Lin:**LIBSVM-A Library for Support Vector Machines**(http://www.csie.ntu.edu.tw/ cjlin/libsvm/)