統計解析環境Rを活用した心理学実験 2：画像刺激の操作

安田　孝・上田　卓司・椎名　乾平

1. はじめに

心理学における知覚や記憶の実験では，刺激として画像が多く用いられる。こうした画像刺激には，線画やイラスト，写真など様々なものが用いられる。特に顔研究に焦点を当てた場合，今日は顔写真を用いるのが一般的である。使用される顔写真は，そのまま用いられる場合もあるが，様々な加工がなされることも多い。加工を行うために，Adobe Photoshopなどの画像処理ソフトウェアが用いられる。画像処理ソフトウェアは，手軽に様々な加工が行われる一方で，どういった内部処理が行われているか明らかではない場合があり，その結果として加工自体を勘に頼って行わざるをえない。これは刺激統制という観点から見た場合，第3者による再現が困難になるなど，問題が生じる原因にもなりうる。

本稿では，刺激の加工に統計処理ソフトウェアのRを使用することを試みる。Rは無料で使用することができ，様々な関数が用意されている。こうした関数には画像を扱うことができるものもある。関数で行われている処理を確認することができる上，必要に応じて関数自体を作成することも可能である。画像の素材として，顔画像[1]を用いる。顔画像は認知心理学や社会心理学などの領域で，視覚認知からコミュニケーション場面まで幅広く検討対象となっている。

Rの実行に際し，主として金（2007），RjpWiki（http://www.okada.jp.org/RWiki/），を参考にした。また，Rを操作する上で必要なデータハンドリングを中心とした内容に関しては第1部（上田・安田・椎名，2010）で詳細に解説されている。

2. 画像の読み込み・操作・保存

使用する画像を，作業ディレクトリに用意する。もしくは，画像が保存されているフォルダを作業ディレクトリに指定する。作業ディレクトリの変更はメニューの「ファイル」にある「ディレクトリの変更」で行う。以下では，作業ファイルに保存されている画像をRに読み込み，この画像に対してネガポジ反転の加工を行い，それを別のファイル名を付けて保存する，という一連の作業を行う。

本論文では以下，Rに含まれる画像処理パッケージである「biOps」を使用する。パッケージをダ

ウンロードした上で，「library(biOps)」を実行し，インストールされていることが前提となる。

2.1. 画像の読み込み

読み込む対象となる画像名を face.jpg とする。この画像を読み込むには，「readJpeg」関数を用いる。以下では画像を読み込み，face01というオブジェクト（変数）に格納する。画像ファイル名""（ダブルコーテーション"）で囲う。

```
> face01 <- readJpeg("face.jpg")
#画像「face.jpg」を読み込み，オブジェクト「face01」に格納
> face01 #「face01」を表示
size:  256 x 256
type:  grey
```

オブジェクト名「face01」を入力した時に画面に表示されるのは，画像サイズと画像形式（グレースケール）だけである。読み込んだデータを画像として R のグラフィックデバイスで表示するには，「plot.imagedata」関数を用いる。

```
> plot.imagedata(face01)　　#face01の画像を表示
```

2.2. 画像の操作

2.2.1. 関数を用いた画像の加工の例

ここでは，R に含まれる関数を用いて画像加工を行い，その結果を保存する基本的な手順を示す。例として，パッケージ biOps に含まれる画像のネガティブ反転を行う関数「imgNegative(イメージオブジェクト)」を適用し，これをオブジェクト「face02」に格納する，という一連の操作を行う。

```
> face02 <- imgNegative(face01)
#関数 imgNegative をイメージオブジェクト face に適用
> plot.imagedata(face02)　　#face02を表示
> writeJpeg("negaface.jpg", face02)　　#face02を"negaface.jpg"という名前で保存
```

writeJpeg 関数では，「writeJpeg（拡張子まで含めた保存時のファイル名，対象となるイメージオブジェクト)」の順で記す。ファイル名は，“ダブルコーテーション”で囲う必要がある。作業ディレクトリに指定したフォルダを開くと，negaface.jpg というファイルが新たに加わっていることが確認できる。

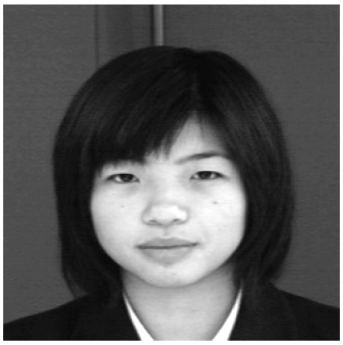

図 1. 元画像（face01）

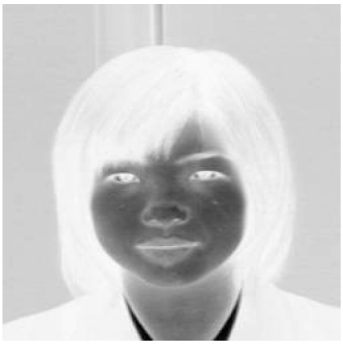

図 2. ネガポジ反転画像（face02）

3. R における画像の表現

3.1. 行列データとしての画像

読み込んだ画像データは，そのままイメージオブジェクトとして用い，様々な関数を適用することができる。しかし，ベクトル・行列・アレイ形式のデータとして扱う方が，演算処理を行う場合に適切な場合もある。以下では，行列，配列，の各形式でデータを扱う方法を示す。なお，行列の取り扱いに関する詳細な内容については，第 1 部（上田・安田・椎名, 2010）を参照されたい。

先ほど読み込んだ画像オブジェクト「face01」は，0 〜255までの256階調からなるグレースケールデータである。この画像は縦横のピクセルで構成される 2 次元データであり，行列（matrix）形式のデータである。これは「dim」コマンドで確認できる。

```
> dim(face01)
[1] 256 256
```

今度は，色情報の次元（RGB）を含んだデータを読み込んでみる。カラー画像「face_colour.jpg」を用意し，これを読み込む。

```
> face03 <- readJpeg("face_colour.jpg")　#画像を読み込み，face03に格納
> face03
size:  350 x 400
type:  rgb
> dim(face03)
[1] 400 350   3
```

画像形式が rgb データであることが確認できる。また次元は，縦横の画素数に加え，色情報次元数の 3 （rgb）が加わっている。こうした 3 次元（以上）のデータは，配列（array）と呼ばれる。なお，2 次元の配列は，行列と同義である。

3.2. 画像の正規化

ここで，画像の正規化について述べる。特に画像を行列の値として扱い何らかの演算を行うと，画素値が256を超えてしまうことがある。この時は，例えば以下のようなエラーが表示され，処理が中断してしまう事が多い。

```
>以下にエラー grey(img):
>　グレー・レベルが不正です。[0,1] の範囲でなければなりません
```

こうした場合，正規化を行い画素値を 0 ～255の間に収める必要がある。画像の正規化に用いる関数は，「imgNormalize（イメージオブジェクト)」である。この関数は以後，実行例の中で適宜用いられている。

```
> imgNormalize(face01)　#画像の正規化
```

3.3. 画像の書き出し

R では，画像が数値データとして表現されていることを確認する。そのために，行列（配列）型に変換し，テキストファイルに書き出す操作を行う。画像を行列に変換する関数は「as.matrix（配列型にするには，as.array)」を用いる[2]。

```
> face1matrix <- as.matrix(face1norm)
#画像を行列型に変換した物を face1matrix に格納
> face1matrix
size:  256 x 256
type:  grey

> write.table(face1matrix, "face1matrix.txt", sep=", ", col.names=FALSE, row.names=FALSE)
#カンマ区切りのテキストファイルとして書き出し
```

作業ディレクトリに指定したフォルダを確認すると，「face1matrix.txt」というファイルが出来て

いる。ファイルを開くと，数値データが並んでいる事が分かる。この数字が元の画像であることを確認する一つの方法として，Excel を立ち上げて，このファイルを読み込んだ後，メニューの「表示」-->「ズーム」-->「指定」で最小値の10%にしてみるとよい。Excel の一つのセルに注目すると，値の小さな数字（例えば一桁）ほど余白が多くなり，結果として白い背景部分が残る。逆に値の大きな数字であるほどセルの中に占める黒い部分が多くなり，白い背景部分が少なくなる。その結果，表示サイズを縮小するとセルの中の画素値により全体に濃淡が現れ，読み込んだ画像の輪郭が認識できるはずである（図 3 ）。

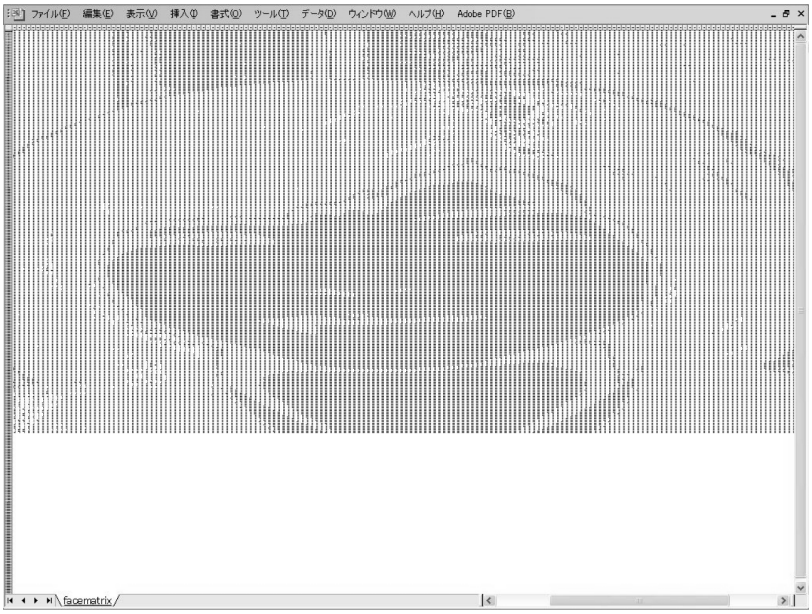

図 3.　画像データのエクセル表示

3. 4.　行列からの画像の生成

続いて，作成した行列をイメージに戻す操作を行う。行列からイメージを作成するには「imagedata（行列）」を用いる。この時に，画像データの形式を指定する必要がある。

```
> face04 <- imagedata(face1matrix, type="grey")
```

4.　画像処理

4. 1.　ランダムマスク画像の作成

画像提示の直後に視覚残像を消去するために提示するマスクを作成する。

まず，単純なマスクを作成することを考える。これは元画像と同じピクセル数の一様乱数から成る行列を作成し，それを画像にすればよい。一様乱数を発生させる関数は「runif（個数，最小値，最大値）」である。

```
observ <- 128*128              #観測値の生成個数を observ に格納
mask01 <- matrix（runif(observ, 0, 255), 128, 128）  #行列を作成し，mask01に格納
writeJpeg("mask01.jpg", imagedata(imgNormalize(mask01), type="grey"))
```

4.2. 顔画像の視認性を落とす；ガウシアンノイズ

ランダムマスクを用いて，画像の視認性を低下させることを試みる。そのためには，画像のイメージ行列に，平均 0 の正規乱数を足してやればよい。以下では，256×256の画像を読み込んだイメージオブジェクトに，平均 0，標準偏差20のガウシアンノイズを加えてみる。正規乱数を発生させるためには，関数「rnorm(個数，平均，SD)」を使う。

```
> mask2 <- matrix(rnorm(256*256, 0, 20), 256, 256) #マスクの作成
> face05 <- face01 + mask2                          #行列の和
> plot.imagedata(imgNormalize(face05))
```

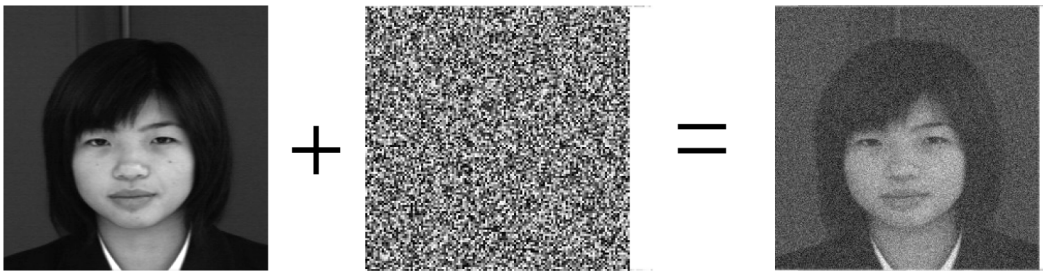

図 4. 元画像（face01）＋乱数（mask2）＝ガウシアンノイズ付加画像（face05）。ただし mask2 に関しては正規乱数の平均を 0 にすると画像に出来ないため，あくまでも比喩的な表現である。

パッケージ biOps には，上記と同様の操作を行う関数「imgGaussianNoise(イメージオブジェクト，平均，SD)」が用意されている。こちらを用いても，理論上は同じ結果となるが，出力画像は同じにはならない。これは，正規乱数を用いた場合は画素値が 0 ～255の間に収まるよう正規化を行うため，その過程で元の画素値の差が圧縮されてしまうことに拠る。

4.3. 畳み込みによる画像変換（imgConvolve 関数による「ぼかし」とエッジ検出）

顔認識においては，顔の顕著な部分特徴（目や鼻，口など）の全体的な配置や，部分特徴間の位置関係といった布置情報に基づく全体処理が重要だと指摘されている（Maurer, Le Grand, & Mondloch, 2002）。この重要性を検討する際に，画像の空間周波数成分を抽出（操作）した刺激が用いられてきた（永山, 2000; 遠藤・桐田, 2004）。代表的なものとして，ブロック化，移動平均法，フーリエ解析を用いた空間周波数成分の抽出，といった方法が用いられてきた。パッケージ biOps に含

まれる「imgConvolve」関数を主として用い，空間周波数成分の検討を行うための刺激操作として，画像に対する「ぼかし」とエッジ検出を行う。

あるピクセルの座標値を (x,y)，その画素値を $f(x,y)$ とする。画像の畳み込み（convolution）とは (x,y) のまわりのピクセルの画素値を用いて $f(x,y)$ の値を変換し，新しい画素値を定義する操作である。この操作はフィルター操作と呼ばれることもある。

尚，デジタル画像における畳み込みは，畳み込み積分という積分を離散化して得られたものであり，従って積分の近似操作という意味合いもある。

4.3.1. ぼかし（Blur）

ライブラリ biOps には，画像に対してぼかしフィルターを適応できる複数の関数が含まれている。「imgBlur（オブジェクト）」はその中の一つで，下記の行列（マスク行列と呼ばれる）を用いてイメージ行列を畳み込むことによって画像をぼかす。

	1/16	1/8	1/16
マスク行列	1/8	1/4	1/8
	1/16	1/8	1/16

```
> face06 <- imgBlur(face01)
> plot.imagedata(face06)
```

「畳み込み」は，画像上のあるセル (x,y) にマスク行列の中心（マスク行列の網掛けの位置）を対応づけ，マスク行列の値とまわりの画素値を用いて新しい $f(x,y)$ を作る。imgBlur 関数の場合は，新しい (x,y) の画素値＝

$$\frac{1}{16}f(x-1,y-1)+\frac{1}{8}f(x,y-1)+\frac{1}{16}f(x+1,y-1)$$
$$+\frac{1}{8}f(x-1,y)\quad+\frac{1}{4}f(x,y)\quad+\frac{1}{8}f(x+1,y)$$
$$+\frac{1}{16}f(x-1,y+1)+\frac{1}{8}f(x,y+1)+\frac{1}{16}f(x+1,y+1)$$

と定める。（言うまでもなく x,y を変化させ全てのピクセルを対象にこの操作を行う。）

imgBlur 関数はあらかじめマスク行列が固定されているため，ボケ度合いを変化させられない。関数「imgConvolve(イメージオブジェクト，マスク行列，バイアス)」を用いることで，もっと様々な操作が可能になる。なおオプションの「バイアス」は画像全体の画素値に係わり，デフォルトでは32が設定されている。数値を上げるほど，画像の画素値が上がる。

顔の空間周波数情報の研究で用いられた移動平均法は，注目画素に対して，その周辺にある画素を用いて平均化し，その結果を注目画素に適用する方法である。上記の「imgConvolve」関数のマス

ク行列を同一値にすることで可能となる。ただし，マスクの合計値が 1 になるよう注意する必要がある。以下の例では，注目画素の周囲 8，合計 9 画素の平均を取るマスク（mask3by3）と，周囲24（合計25）の平均を取るマスク（mask5by5）を作成している。後者の方が，ボケ度合いは大きくなる。

```
                1/9  1/9  1/9                1/25 1/25 1/25 1/25 1/25
                1/9  1/9  1/9                1/25 1/25 1/25 1/25 1/25
mask3by3 =      1/9  1/9  1/9     mask5by5 = 1/25 1/25 1/25 1/25 1/25
                1/9  1/9  1/9                1/25 1/25 1/25 1/25 1/25
                                             1/25 1/25 1/25 1/25 1/25
```

```
> mask3by3 <- matrix(c(1,1,1,1,1,1,1,1,1)/9, 3, 3, byrow = TRUE) #マスク行列の作成
> mask5by5 <- matrix(c(rep(1,25))/25, 5, 5, byrow = TRUE) #マスク行列の作成
> face07 <- imgConvolve(face01, mask3by3, 25)
> face08 <- imgConvolve(face01, mask5by5, 25)
```

なお，パッケージ biOps には，平均ではなく中央値を用いるメディアンフィルター関数「imgBlockMedianFilter(imgdata, dim)」も用意されている。ただしこの方式は，移動平均法と比較してエッジが保存されるのが利点であるため，顔の空間周波数情報の研究文脈で用いるにはあまり適さないと言える。

```
> face09 <- imgBlockMedianFilter(face01, 10)
> plot.imagedata(face09)
```

図 5. 左から，Blur (face06)，移動平均 (face07)，移動平均 (face08)，imgBlockMedianFilter 適用 (face09)

4.3.2. エッジ検出

画像からエッジ（輪郭）の抽出するのは画像工学での基本技術であるが，このようなエッジ画像が心理学の実験で直接用いられることはあまりない。しかし現代心理学に大きな影響を与えた Marr

（1982）において，Marr-Hildreth のエッジ検出アルゴリズムが視覚の心理モデル（受容野のモデル）として提案されているので，エッジ検出について簡単に述べる。尚以下の記述では「エッジ検出アルゴリズム」と「フィルター」は同義である。

4.3.3. ラプラシアンフィルター

画像の画素値の差分をうまく用いると，エッジを検出することができる。ラプラシアンフィルターとは，ピクセルの座標値を x,y，そのピクセルでの画像の画素値を $f(x,y)$ とする時，$f(x,y)$ を x 方向で 2 回微分，y 方向で 2 回微分したものを足し合わせ，その値を新たに $f(x,y)$ の値として置き直すものである。このことを記号的に $\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)f(x,y)$ と書くこともある。

y の値を定数と見なして $f(x,y)$ を x 方向で一回微分すると，微分の定義より

$$f'(x,y)=\frac{\partial f(x,y)}{\partial x}=\lim_{a\to 0}\frac{f(x+a,y)-f(x-a,y)}{2a}$$

この結果を x 方向でもう一回微分すると

$$\{f'(x,y)\}'=\frac{\partial f(x,y)'}{\partial x}=\lim_{b\to 0}\frac{f'(x+b,y)-f'(x-b,y)}{2b}$$

$$=\lim_{b\to 0}\frac{\lim_{a\to 0}\frac{f(x+a+b,y)-f(x-a+b,y)}{2a}-\lim_{a\to 0}\frac{f(x+a-b,y)-f(x-a-b,y)}{2a}}{2b}$$

$$=\lim_{b\to 0}\lim_{a\to 0}\frac{f(x+a+b,y)-f(x-a+b,y)-f(x+a-b,y)+f(x-a-b,y)}{4ab}$$

a,b は無限に 0 に近づけるべきであるが，デジタル画像の世界では小さくできる限界がある。この式の場合 $a=b=1/2$ まで小さくしても有意味な結果を得ることができ，

$=f(x+1,y)+f(x-1,y)-2f(x,y)$

となる。同様な微分操作を y 方向に 2 回行うと，$f(x,y+1)+f(x,y-1)-2f(x,y)$ となる。ラプラシアンフィルターは両者の和と定義されているので

$f(x+1,y)+f(x-1,y)-2f(x,y)+f(x,y+1)+f(x,y-1)-2f(x,y)$
$=f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1)-4f(x,y)$

が求める結果である。この式は「(x,y) の上下左右のセルの画素値の和から，(x,y) の現在の画素値の 4 倍を引いて，新しい (x,y) の画素値とせよ」と言っており，マスク行列を書くと

$$\text{masklaplace}=\begin{matrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{matrix}$$

であり，結果を見やすくするためネガポジ反転画像を出力すると

```
> masklaplace <- matrix(c(0,1,0,1,-4,1,0,1,0), 3, 3, byrow = TRUE)
> facelaplace <- imgConvolve(face01, masklaplace, 0)
> writeJpeg("facelaplace.jpg",imagedata(imgNormalize(imgNegative(facelaplace)), type="grey"))
```

となる（図 6 ）。

4.3.4. ガウス関数のラプラシアンフィルター（LOG, Laplacian Of Gaussian filter）

このフィルターは視覚心理学では Marr-Hildreth のフィルターとして有名である。またこのフィルターを近似するためにガウス関数の差分フィルター（DOG, Difference of Gaussian Filter）というものが使われることも多い。両方ともその形状からメキシカンハットフィルターと呼ばれることがある。

このフィルターは，まず元画像 $f(x,y)$ に対してガウス関数

$$\Phi(x,y) = \frac{1}{2\pi\sigma} e^{-\frac{1}{2}\left(\frac{x^2}{\sigma^2}+\frac{y^2}{\sigma^2}\right)}$$

を用いてランダムノイズ成分を消すためのぼかしの操作を行い，その後ラプラシアンフィルターを用いてエッジ検出を行う。それぞれの操作を順次，単独で行っても良いが，いっぺんに行って計算量を節約するのが「ガウス関数のラプラシアンフィルター」の眼目である。すなわち
元画像 $f(x,y)$ →ガウス関数で $f(x,y)$ をぼかす→ぼかした画像にラプラシアンをかける
という手続きの後半二つをいっぺんにやろうということである。このためにはガウス関数にラプラシアンをかけたフィルターを作り，そのフィルターを原画像に適用すればよい。

元画像 $f(x,y)$ をガウス関数によってぼかすためにはたたみ込み積分を用いて

$$b(x,y) = \int\int \frac{1}{2\pi\sigma} e^{-\frac{1}{2}\left(\frac{(x-a)^2}{\sigma^2}+\frac{(y-b)^2}{\sigma^2}\right)} f(a,b)\,da\,db$$

とする。ここで $b(x,y)$ はぼかされた画像である。この $b(x,y)$ にラプラシアンフィルターをかけるため x で 2 回微分すると

$$\int\int \left\{ -\frac{1}{2\pi\sigma^4} e^{-\frac{1}{2}\left(\frac{(x-a)^2+(y-b)^2}{\sigma^2}\right)} + \frac{(x-a)^2}{2\pi\sigma^6} e^{-\frac{1}{2}\left(\frac{(x-a)^2+(y-b)^2}{\sigma^2}\right)} \right\} f(a,b)\,da\,db$$

y で 2 回微分すると

$$\int\int \left\{ -\frac{1}{2\pi\sigma^4} e^{-\frac{1}{2}\left(\frac{(x-a)^2+(y-b)^2}{\sigma^2}\right)} + \frac{(y-b)^2}{2\pi\sigma^6} e^{-\frac{1}{2}\left(\frac{(x-a)^2+(y-b)^2}{\sigma^2}\right)} \right\} f(a,b)\,da\,db$$

となるので，両者の和は

$$\int\int \frac{(x-a)^2+(y-b)^2-2\sigma^2}{2\pi\sigma^6} e^{-\frac{1}{2}\left(\frac{(x-a)^2+(y-b)^2}{\sigma^2}\right)} f(a,b)\,da\,db$$

となる。これで理論的には終了であるが，デジタル画像の世界では x,y,a,b は整数値しか取り得ないので，この式を少ない計算量で離散的に近似できる方法を考えなければならない。

$$\frac{(x-a)^2+(y-b)^2-2\sigma^2}{2\pi\sigma^6}e^{-\frac{1}{2}\left(\frac{(x-a)^2+(y-b)^2}{\sigma^2}\right)}$$

の部分の形状を図に書いて見ると，図 7 のようになる。そこで離散近似の一つの解は

$f(x-1,y+1)+f(x-1,y)+f(x-1,y-1)+f(x,y+1)+f(x,y-1)+f(x+1,y+1)+f(x+1,y)$
$+f(x+1,y-1)-8f(x,y)$

となる。この式は「(x,y) のまわりの 8 方向のセルの画素値の和から，(x,y) の現在の画素値の 8 倍を引いて，新しい (x,y) の画素値とせよ」と言っている。マスク行列は以下の通り。

$$\text{Maskmarr} = \begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix}$$

結果を見やすくするためネガポジ反転画像を出力すると

```
> maskmarr <- matrix(c(1,1,1,1,-8,1,1,1,1), 3, 3, byrow = TRUE)
> facemarr <- imgConvolve(face01, maskmarr, 0)
>writeJpeg("facemarr.jpg",imagedata(imgNormalize(imgNegative(facemarr)), type="grey"))
```

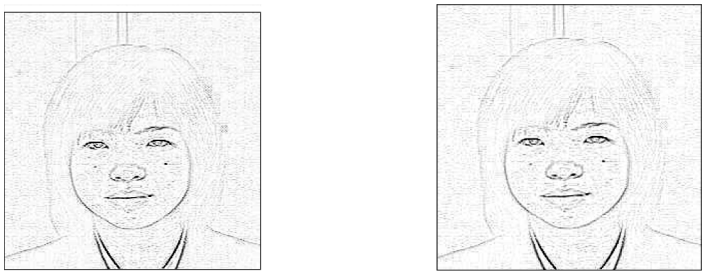

図 6.　左は Facelaplace（ラプラシアンフィルターの結果のネガポジ反転)。
右は Facemarr（Marr-Hildreth フィルターの結果のネガポジ反転)。

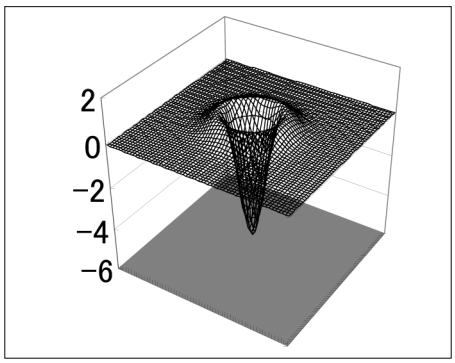

図 7.　Marr-Hildreth フィルター

4.4. バンドパスフィルター

画像にフーリエ変換を行い，空間座標を周波数に変換した上で，特定の空間周波数のみを取り出すフィルターを，バンドパスフィルターと呼ぶ。このうち，特に高周波数成分のみを抽出するものをハイパスフィルター，低周波数成分を抽出するものをローパスフィルターと呼ぶ。biOps パッケージでは，離散フーリエ変換の処理を高速に行うアルゴリズムである，高速フーリエ変換（Fast Fourier Transform: FFT）を用いた関数が用意されている。ハイパスフィルターを用いるには，関数「imgFFTHighPass(FFT の複素行列, 半径)」，ローパスフィルターを用いるには，関数「imgFFTLowPass(FFT の複素行列, 半径)」，バンドパスフィルターを用いるには関数「imgFFTBandPass(FFT の複素行列，周波数フィルターの内径，周波数フィルターの外径)」をそれぞれ用いる。画像から FFT の複素行列を計算するには，関数「imgFFT(イメージオブジェクト)」を用いる。

具体的な手続きは，フーリエ変換を行い FFT 行列を生成し，その行列に対してフィルターをかけ，最後にフーリエ逆変換を行い画像に戻す，という 3 段階の作業を経る。

```
face_fftmatrix <- imgFFT(face01)　#FFT 行列の作成
face10_fft <- imgFFTHighPass(face_fftmatrix, 25) #ハイパスフィルターの適用
face11_fft <- imgFFTLowPass(face_fftmatrix, 25)　#ローパスフィルターの適用
face12_fft <- imgFFTBandPass(face_fftmatrix, 25, 70) #バンドパスフィルターの適用
face10 <- imgFFTInv(face13_fft)　#フーリエ逆変換と画像の書き出し
face11 <- imgFFTInv(face14_fft)
face12 <- imgFFTInv(face15_fft)
```

図 8. 左から，ハイパスフィルター（Face10），ローパスフィルター（face11），バンドパスフィルター（face12）の適用例。

5. 特異値分解と顔の復元

5.1. R 関数を用いた特異値分解

画像を行列として扱うということは，これまでの例で見たとおり，画像に対して行列演算処理を施すことが出来るということである。特異値分解は，そうした処理の一つであるが，R では関数「svd

（行列）」で実行可能である。使用している行列は，「3.3 画像の書き出し」を行ったときに作成したもの（図 3 ）である。

```
> face1svd <- svd(face1matrix)
> face1svd
$d
（以下略）
```

特異値分解した結果はデータオブジェクトにまとめられ，mysvdface$d，mysvdface$u, mysvdface$v でそれぞれ，特異値ベクトル，左特異行列，右特異行列となる。

特異値行列から元の行列を計算するには

```
> fukugen <- face1svd$u %*% diag(face1svd$d)  %*% t(face1svd$v)
> fukugen
```

行列積の演算子は「*」ではなく「%*%」を使用する。また特異値分解したデータオブジェクトの mysvdface$d は特異値ベクトルなので対角行列を作成する関数「diag( )」を使う。右特異値行列を掛ける時には転置（関数「t( )」）する必要がある。こうしてできた fukugen はあくまでも単なる行列データなので，画像として扱うには「imagedata( )」関数でイメージデータ形式にしてやる。

5.2. 少ない次元数で元の顔を復元する例

特異値分解した結果のうち，少ない次元だけで復元を行い，JPEG 画像として保存する。以下は，それぞれ2，5，10，20，40の各次元で復元を行い，その結果を fukugen_2dim〜fukugen_40dim に書き出している。

```
> fukugen_2dim <- face1svd$u[,1:2] %*% diag(face1svd$d[1:2]) %*% t(face1svd$v[,1:2])
> fukugen_5dim <- face1svd$u[,1:5] %*% diag(face1svd$d[1:5]) %*% t(face1svd$v[,1:5])
> fukugen_10dim<-face1svd$u[,1:10]%*%diag(face1svd$d[1:10]) %*%t(face1svd$v[,1:10])
> fukugen_20dim<-face1svd$u[,1:20] %*%diag(face1svd$d[1:20]) %*% t(face1svd$v[,1:20])
> fukugen_40dim <- face1svd$u[,1:40] %*% diag(face1svd$d[1:40]) %*% t(face1svd$v[,1:40])
```


fukugen_2dim


fukugen_5dim


fukugen_10dim


fukugen_20dim


fukugen_40dim

図 9. 特異値分解による元の顔の復元。各画像の次元は，1 行目の左から，2，5，10，20，40次元である。各画像の下に，上記プログラムで用いたオブジェクト名を付した。

5.3. for ループによる反復処理

上記のプログラムでは，次元を変化させるたびに同じプログラムを記述した。for ループを用いて反復処理させることで，1 回の実行で済ませることができる。

```
> face1 <- readJpeg("test.jpg")　# カラー画像読み込み
> face1 <- face1[,,1]
# 単独の色成分（R）の数値だけとりだして実質的にグレースケール化された行列を作成
> face1svd <- svd(face1)
> for(i in c(2,5,10,20,40)) {　　#次元数を指定する
        fukugen_face <- paste("fukugen_", i, "dim",sep="")
        # Excel で言うところの concatenate に相当する関数
        fukugen_img <- paste("fukugen_", i, "dim", ".jpg",sep="")
        fukugen_face <- face1svd$u[,1:i] %*% diag(face1svd$d[1:i])　%*% t(face1svd$v[,1:i])
        fukugen_face <- imagedata(imgNormalize(fukugen_face), type="grey")
        writeJpeg(fukugen_img, fukugen_face)
}
```

6. 複数の画像刺激の一括処理と平均顔の作成

ここでは，複数の画像に対する処理を扱う。まず複数の画像を一括してRに取り込む方法を説明する。次いで，取り込んだ画像を用いて平均顔の作成を行う。平均顔（average face）とは，複数の顔を重ね合わせることで，文字通り「平均的な」顔を作り出すことである。顔研究においては，例えば平均顔はそうでない顔よりも魅力的に見える（Perrett, May, & Yoshikawa, 1994）といった研究をはじめ，顔認知の性質を明らかにしようとする実験で頻繁に用いられる。

6.1. 刺激データセットからの画像一括読み込み

複数のデータセットを一括して読み込むために，リストを作成しておく。この時は，MS Excel 等を使用すると良いであろう。今回は次に示す，第 1 列目に ID 番号，第 2 列目にファイル名だけのシンプルなリストとした。3 列目以降に，例えば性別や何らかの評定値などを含めることも可能である。読み込んだファイルは，139枚の jpeg 画像である。第 1 行目が変数名にあたる（ID と facefile）。R で読み込むときのため，変数名をつける時は次の 2 点に注意する。

1. 算用数字から始めない
2. 変数名にスペースを含めない

リストを作成したら，テキストファイル（タブ区切り）として保存する。以後，この刺激定義ファイルをオブジェクト名「profile」として扱う。またこのファイルは，作業ディレクトリ上に置く。

表 1. 刺激定義リスト（profile）の例

ID	Facefile
1	F001.jpg
2	F002.jpg
3	F003.jpg
⋮	⋮
139	F139.jpg

```
> profile <- read.table("facelist.txt", header=TRUE)
#プロファイルの読み込み。ヘッダ（ID と Facefile）情報を有効にするために，header=TRUE のオプションをつける
> profile$ID <- factor(profile$ID)                    #因子として，ID を付与
> facematrices <- lapply(profile$facefile, readJpeg)
#facematrices に，イメージオブジェクトを読み込む。複数のオブジェクトに同じ関数を適用するために，関数「lapply（オブジェクト）」を用いる。
> names(facematrices) <- profile$facefile
#読み込んだオブジェクトにリストの画像名を付与
```

以上の手続きで，リスト形式での顔画像の読み込みが完了した。読み込んだのがリスト形式であることは，関数「is.list(リスト)」で確認できる。

```
> is.list(facematrices)
[1] TRUE
```

また，読み込んだ画像の形式を確認するには，リスト名（facematrices）をそのまま画面に入力すれば良い。ただし数が多いので，リスト中の 1 つ目のファイルだけを取り出す形で確認を行ってみる。リストの中の要素にアクセスするには，リスト名のあとに［ブラケット］で要素の番号を指定する。画像サイズが350×400のグレースケール画像であることが分かる。

```
> facematrices[1]
$gf001.jpg
size:  350 x 400
type:  grey
```

イメージオブジェクトは，行列で表現されている。画像情報（type: grey）が表示されているのは，R では純粋な数値データ（行列）ではなく，画像情報も付加されているためである。

6.2. 平均顔の作成

パッケージ「biOps」には「imgAverage(イメージリスト)」という，画像の平均を計算する関数が用意されている。上記で作成したイメージリスト「facematrices」を使用し，平均顔を作成する。

```
> average01 <- imgAverage(facematrices)
> plot.imagedata(average01)
```

ここでは，読み込んだ139枚の画像全てを用いた平均顔を作成している。一部分のみを使用する場合は，リストの要素を指定すればよい。以下の例では，20枚目から30枚目の画像を読み込み，平均顔を作成している。

```
> average02 <- imgAverage(facematrices[20:30])
> plot.imagedata(average02)
```

読み込む画像の位置が合っていない場合，ぼけた画像になってしまう。顔の部分特徴を可能な限り保持した平均顔を作成するためには，読み込む前に，各画像の位置合わ（目の高さを揃える，等）を行っておくことが必要となる。

図10．平均顔の例。左端が10枚の画像による平均顔（average02）。以下順に，20枚，50枚。右端が全画像（139枚）による平均顔（average01）。

7．おわりに

本稿では，R による画像刺激の操作について，顔画像を対象とした活用法を紹介してきた。本稿で取り上げた顔画像などでは，仔細な部分特徴への操作は簡単とは言えず，そうした処理は画像処理ソフトウェアを用いた方が効率的な場合もある。しかし原則として関数の内部処理が確認できる R を用いることで，種々の画像加工で行われている具体的な内容が確認でき，画像処理ソフトウェアによる手作業よりも明示的に加工の足跡をたどることが出来る。特にここで取り上げた画像全体を対象とした操作においては，R は有用な選択肢の一つであると言えよう。

文献

遠藤光男・桐田隆博 2004. 顔認識における空間周波数成分の役割．琉球大学法文学部人間科学科紀要．人間科学, 13, 179–202.
金明哲 2007. R によるデータサイエンス：データ解析の基礎から最新手法まで．森北出版.
Marr, D. 1982 *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. New York: Freeman. 乾敏郎，安藤広志（訳）ビジョン：視覚の計算理論と脳内表現　産業図書（1987/10）
Maurer, D., Le Grand, R., & Mondloch, C. J. 2002 The many faces of configural processing. *Trends in Cognitive Sciences*, 6, 255–260.
永山ルツ子 2000 顔知覚の空間周波数特性．心理学評論, 43, 276–292.
Perrett, D. I., May, K. A., & Yoshikawa, S. 1994 Facial shape and judgements of female attractiveness. *Nature*, 368, 239-242.
RjpWiki　2010年10月31日取得．http://www.okada.jp.org/RWiki/.
上田卓司・安田孝・椎名乾平 2010．統計解析環境 R を活用した心理学実験：1 心理学実験における刺激統制と実験準備．学術研究（教育心理学編）59, 1-20.

注

1　本論文に使用した顔画像データは，財団法人ソフトピアジャパンから使用許諾を受けたものです。権利者に無断で複写，利用，配布等を行うことは禁じられています。

2　ここでは単に「行列」ではなく「行列型」と表現している。その理由は，R で as.matrix を用いて画像を行列に変換した場合，数値の表現に加えて画像としての情報も維持されるからである。