Article

imageProcAnal: A novel Matlab software package for image processing and analysis

WenJun Zhang
School of Life Sciences, Sun Yat-sen University, Guangzhou 510275, China; International Academy of Ecology and Environmental Sciences, Hong Kong
E-mail: zhwj@mail.sysu.edu.cn, wjzhang@iaees.org

Received 8 November 2019; *Accepted 23 December 2019*; *Published 1 June 2020*

Abstract
In present study, I developed a powerful Matlab-based software package, imageProcAnal (Version 1.0), for image processing and analysis. Several modules were available for uses. Functions to resize, crop, rotate, dilate, pixelate and watermark images are included in Basic module. Adjustment of image edge, contrast and gamma factor is available in Adjustment module. In Denoising module, four elementary denoising methods and their joint uses are provided for image filtering and denoising. In Edge and Contour Detection module, five methods for detection of image edge and contour are provided. Also three methods for image sharpening are available in Sharpening module. Segmentation module includes three methods for image segmentation. Connected components recognition can be made in Object Recognition module. In Analysis module, Pearson correlation measure was provided to align images. Pearson correlations and *p* values among images can be obtained. Mutiple methods across different modules can be sequentially and jointly used to achieve ideal image. Finally, some examples for image processing can be found in Demo module. In addition to conventional methods, I proposed two new methods for image segmentation, i.e., TDOS (Two-Dimensional Ordered Segmentation) and MWAS (Moving Windows Averaging Segmentation). In TDOS, which is based on two-dimensional ordered cluster analysis, the pixels belonging to the same segment (class) are adjoint and are not separated by any of remaining segments. MWAS is based on moving windows averaging method for transect boundary detection. Algorithms of TDOS and MWAS were given, and full Matlab codes of MWAS were given.

Keywords Matlab; software; image denoising; image adjustment; edge detection; image sharpening; image segmentation; image alignment; Pearson correlation.

Network Pharmacology
ISSN 2415-1084
URL: http://www.iaees.org/publications/journals/np/online-version.asp
RSS: http://www.iaees.org/publications/journals/np/rss.xml
E-mail: networkpharmacology@iaees.org
Editor-in-Chief: WenJun Zhang
Publisher: International Academy of Ecology and Environmental Sciences

1 Introduction

Image recognition and processing is a discipline that uses computer systems to identify and process images to achieve a human-like visual system that understands the outside world (Wu, 2006; Bhardwaj and Solanki,

2016). An image is an information carrier that conveys information to people in different patterns. Some of the patterns can be recognized and understood by computer vision systems by simulating the visual function of the human eye, but due to insufficient prior knowledge in related fields, some other patterns cannot be simulated and identified by computer vision systems. The factors that affect the development of image recognition and processing research are mainly the prior knowledge of related fields. This includes not only existing computer algorithms, but also knowledge of image acquisition, image-related databases, etc.

Image processing and analysis is a fundamental aspect for various sciences and technologies, for example, medical digonosis, gene expression, landscape recognition, etc (Dogra et al., 2018; Zhang and Qi, 2019). However, the common used software for image processing are somewhat simple and short of many functionalities. The present study aims to develop and present the powerful Matlab-based software for image processing and analysis. Various functionalities seldom found in common software are available in the software pakage. In addition, new methods are proposed also.

2 Materials and Methods

2.1 Methods

Various methods for image processing and analysis, and corresponding Matlab codes are presented in the following.

2.1.1 Rotate, crop, resize, dilate and watermark images

Rotate an image with a specified degree:

```
degr=10;
rgb=imread('geneExpression.png');
j=imrotate(rgb,degr, 'bilinear', 'crop');    %Rotate image by degr degrees
imshow(j);
```

Crop the image with a specified rectangular area:

```
rgb=imread('geneExpression.png');
rgb=imcrop(rgb);                    %Crop image interactively
imshow(rgb);
```

Resize the image to b×a pixels:

```
a=600; b=800;
rgb=imread('geneExpression.png');
j=imresize(rgb, [a b])    %Resize image to b×a pixels
imshow(j);
```

Dilate the image:

```
m=5;
rgb=imread('wjzhang.png');
imj=im2bw(rgb);
se=strel('diamond',m);
j=imdilate(imj,se)    %Dilate image with a diamond of size m
```

```
imshow(j);
```

The general process of embedding watermark in an image using discrete cosine transform is: (1) Discrete cosine transformation of the original image; (2) generation of watermark signals; (3) embed watermark in the image. The following is the procedure for the inverse of two-dimensional discrete cosine transformation to obtain a watermarked image:

```
rgb=imread('wjzhang.png');
imdim=numel(size(rgb));
if (imdim~=2) rgb=rgb2gray(rgb); end
alfa=0.01; len=1000;                                    %Set parameters
dctf=dct2(rgb);
[m,n]=size(dctf);                                       %DCT (Discrete Cosine Transformation) of the image
watermark=randn(len,1);
[y0,z0]=sort(watermark);                                %Generate watermark series and sort it
v=dctf(:);
[y1,z1]=sort(v);
mn=m*n;
k=len;
p=zeros(mn,1);                                          %Find the position for embeding watermark
for i=1:mn
if (k>=1)
p(mn)=y1(mn)*(1+alfa*y0(k));
k=k-1;
else p(mn)=y1(mn);
end
mn=mn-1;
end
y=zeros(mn,1);
mn=m*n;
for i=1:mn
y(z1(i))=p(i);
end
s=1;
for j=1:n
for i=1:m
dctf2(i,j)=y(s);
s=s+1;
end
end                     %Revise the amplitudes of s frequency components with greater amplitudes and embed watermark
idctf=idct2(dctf2);
imshow(idctf,[]);       %Inverse of DCT for obtaining the image embedded with watermark
```

2.1.2 Image adjustment

Image adjustment is a processing method that highlights certain information in an image according to specific requirements, such as improving some edge information and adjusting the contrast of the image, while weakening or removing some unwanted information. The adjusted image can often enhance the ability to recognize special information, improve the visual effect, and allow the observer to check a more direct and clear image.

2.1.2.1 Grayscale linear transformation

Insufficient brightness or non-linearity of some images will make the contrast of images less than ideal. We can improve the contrast of an image by reassigning the pixel amplitude. The grayscale of the image was expanded by linear mapping. In addition, the image compression method is adopted, or the grayscale is segmented, that is, compression is performed in a certain interval and expanded in another interval according to image characteristics and our requirements.

2.1.2.2 Gamma adjustment

Gamma value in the function 'imadjust' can determine the correction effect. Gamma value determines the grayscale mapping method of the input image to the output image, that is, determines whether to adjust low grayscale or adjust high grayscale. When Gamma equals 1, it is a linear transformation.

2.1.2.3 Two-dimensional filtering

The spatial texture information of an image can reflect the position, shape, size and other characteristics of the image. Using linear filtering technology can adjust some texture features of the image and remove other unuseful features. Linear filtering is a neighborhood operation whose response is given by the sum of the product of the filter coefficients and the corresponding pixel values of the area swept by the filtering window. Matlab provides a variety of filtering operators, such as mean filtering, Gaussian lowpass filtering, Laplacian operator, motion blur operator, edge adjustment, edge extraction, contrast enhancement filter, etc.

The following is the full Matlab codes for image adjustment:

```
rgb=imread('geneExpression.png');
imdim=numel(size(rgb));
if (imdim~=2) i=rgb2gray(rgb);
else i=rgb;
end
h=fspecial('prewitt');
j=imfilter(i,h);                    %Edge adjustment
imshow(j);

a=0.6; b=0.9; c=0.2; d=0.5;
rgb=imread('geneExpression.png');
imdim=numel(size(rgb));
if (imdim~=2) i=rgb2gray(rgb);
else i=rgb;
end
j=imadjust(i,[a b],[c d]);          %Contrast adjustment; Grayscale from [a b] to [c d]
imshow(j);

gamma=2;
rgb=imread('geneExpression.png');
imdim=numel(size(rgb));
```

```
if (imdim~=2) i=rgb2gray(rgb);
else i=rgb;
end
j=imadjust(i,[],[], gamma);                    %Gamma adjustment
imshow(j);
```

2.1.3 Image denoising

There may be a variety of noises in an image. These noises may be generated during propagation, or they may be generated during image processing such as quantization. In Matlab there are roughly five types of noises: 'gaussian' is Gaussian noise, 'localvar' is Gaussian white noise, 'poisson' is Poisson noise, 'salt & pepper' is salt and pepper noise, and 'speckle' is uniform random noise.

Restoration of noisy image is an important area of image processing. The negative effect of these noises on the image can be eliminated by smooth filtering. A good smoothing method should eliminate these noises without blurring the edges and lines of the image. There are many filtering methods, such as mean filtering, median filtering, adaptive filtering, wavelet denoising, etc. Each method has its own advantages and disadvantages.

2.1.3.1 Mean filtering

Mean filtering belongs to the spatial filtering and smoothing technology. It mainly uses the neighborhood averaging method to remove the particle noise obtained by scanning the image. The neighborhood averaging method mainly uses the average value of several neighboring pixels to replace a certain pixel.

2.1.3.2 Median filtering

Median filtering is a commonly used non-linear smoothing filtering. Its filtering principle is similar to the mean filtering. The difference between the two is that the output pixel value of the median filtering is determined by the intermediate value of the neighboring pixels instead of the average value. Median filtering is very effective for filtering the salt and pepper noise of the image, and the noise on the image is almost completely removed.

2.1.3.3 Adaptive filtering

The Matlab image processing toolbox provides ‘wiener2’ to estimate the local mean and variance of each pixel to achieve adaptive filtering of image noise.

2.1.3.4 Wavelet denoising

There are three main steps in image denoising using wavelet analysis: (1) wavelet decomposition of the image signal; (2) threshold quantization of the decomposed high-frequency coefficients, and (3) image signals are reconstructed using two-dimensional wavelets.

The following is the full Matlab codes of jointly used four methods above for image denoising:

```
rgb=imread('geneExpression.png');
imdim=numel(size(rgb));
if (imdim~=2) i=rgb2gray(rgb);
else i=rgb;
end
h=fspecial('average');
j=imfilter(i,h);                    %Mean filtering
k=medfilt2(j);                      %Median filtering
rgb=wiener2(k);                     %Adaptive filtering
rgb=im2double(rgb);
```

```
thr=0.1; sorh='s'; crit='shannon'; keepapp=0;
rgb=wpdencmp(rgb,sorh,3,'sym4',crit,thr,keepapp);    %Wavelet denoising
imshow(rgb);
```

2.1.4 Edge detection

2.1.4.1 Edge detection

Edge information is an important attribute for extracting image features. The edge detection operators include Prewitt operator, Canny operator, Sobel operator, Log operator, etc.

The edge detection of the image using several edge detection operators in MATLAB is as follows:

```
rgb=imread('geneExpression.png');
imdim=numel(size(rgb));
if (imdim~=2) i=rgb2gray(rgb);
else i=rgb;
end
bw1=edge(i,'sobel');                %Sobel method
bw2=edge(i,'roberts');              %Roberts method
bw3=edge(i,'prewitt');              %Prewitt method
bw4=edge(i,'log');                  %Laplacian of Gaussian method
bw5=edge(i,'canny');                %Canny method
subplot(2,3,1);imshow(bw1);title('Sobel edge detection');
subplot(2,3,2);imshow(bw2);title('Roberts edge detection ');
subplot(2,3,3);imshow(bw3);title('Prewitt edge detection ');
subplot(2,3,4);imshow(bw4);title('Laplacian of Gaussian edge detection ');
subplot(2,3,5);imshow(bw5);title('Canny edge detection ');
```

2.1.4.2 Contour detection

The following is the Matlab codes for contour detection of a binary image:

```
rgb=imread('wjzhang.png');
rgb=im2bw(rgb);
stru=strel('diamond',1);
imd=imdilate(rgb,stru);
ime=imerode(rgb,stru);
imed=imd-ime;
contour=1-imed;
imshow(contour);
```

2.1.5 Image sharpening

Image sharpening is mainly used to ajdust the edges and changing parts of grayscale of an image. There are two processing methods, spatial domain and transformation domain. The image sharpening operators include Roberts operator, Sobel operator, Prewitt operator, Laplacian operator, etc.

```
rgb=imread('geneExpression.png');
```

```
imdim=numel(size(rgb));
if (imdim~=2) i=rgb2gray(rgb);
else i=rgb;
end
subplot(2,2,1);imshow(i);title('Original');
h1=fspecial('sobel');
i1=filter2(h1,i);
subplot(2,2,2);imshow(i1);title('Sobel Sharpening ');
h2=fspecial('prewitt');
i2=filter2(h2,i);
subplot(2,2,3);imshow(i2);title('Prewitt Sharpening');
h3=fspecial('laplacian');
i3=filter2(h3,i);
subplot(2,2,4);imshow(i3);title('Laplacian Sharpening');
```

2.1.6 Image segmentation

Image segmentation refers to the separation of different areas with special significance in an image. These areas are disjoint from each other. Each area meets certain similarity criteria such as grayscale, texture, and color.

2.1.6.1 Maximum Inter-segment Variance Threshold

The maximum inter-segment variance threshold method proposed by Ostu is derived on the basis of least square method. The following is to segment an image using maximum inter-segment variance threshold method.

```
thr=128;
rgb=imread('geneExpression.png');
imdim=numel(size(rgb));
if (imdim~=2) c=rgb2gray(rgb);
else c=rgb;
end
subplot(1,2,1);imshow(c);
title('Original Image');
freq=imhist(c);
[r,t]=size(c);
N=r*t;
L=256;
freq=freq/N;
for i=2:L
if freq(i)~=0
st=i-1;
break
end
end
for i=L:-1:1
if freq(i)~=0;
```

```
nd=i-1;
break
end
end
f=freq(st+1:nd+1);
p=st;q=nd-st;u=0;
for i=1:q;
u=u+f(i)*(p+i-1);
ua(i)=u;
end
for i=1:q
w(i)=sum(f(1:i));
end
d=(u*w-ua).^2./(w.*(1-w));
[y,tp]=max(d);
th=tp+p;
if th<=210
th=tp+p;
else th=210;
end
rgb1=zeros(r,t);
for i=1:r
for j=1:t
X1(i,j)=double(c(i,j));
end
end
for i=1:r
for j=1:t
if (X1(i,j)>=thr)
rgb1(i,j)=X1(i,j);
else
rgb1(i,j)=0;
end
end
end
subplot(1,2,2);imshow(rgb1);title('Segmentated Image');
```

2.1.6.2 Single Threshold Histogram

If the histogram of the image shows obvious double peaks, the grayscale corresponding to the valley between the double peaks is selected as the threshold for segmentation, that is, the single threshold histogram segmentation.

```
thr=128;
rgb1=imread('geneExpression.png');
imdim=numel(size(rgb1));
```

```
if (imdim~=2) rgb=rgb2gray(rgb1)
else rgb=rgb1;
end
[m,n]=size(rgb);
for i=1:m
for j=1:n
if (rgb(i,j)>thr);
rgb(i,j)=255;
else
rgb(i,j)=0;
end
end
end
subplot(1,2,1);imshow(rgb1); title('Original Image');
subplot(1,2,2);imshow(rgb); title('Segmentated Image');
```

2.1.6.3 Two-dimensional ordered segmentation (TDOS)

Two-dimensional ordered cluster analysis (Zhang, 1993; Zhang et al., 2014) can be used for image segmentation. In this method, the pixels belonging to the same segment (class) are adjoint and are not separated by any of remaining segments. As a new method for image segmentation, here I name it as TDOS (Two-Dimensional Ordered Segmentation) method.

Assume there are p indices and q pixels ($q=m\times n$, i.e., the total number of elements (pixels) of $m\times n$ grayscale matrix). Now we have $x_i=(x_{i1}, x_{i2},\ldots, x_{ip})$, $i=1,2,...,q$. For gray image segmentation, $p=1$, i.e., grayscale value; for RGB color image segmentation, p indices can be represented by grayscale, red value, green value, blue value, etc.

In the ordered cluster analysis, the pixels in the same segment are spatially adjacent, and the lines between any two pixels that belong to different segments are not intersected.

The following is the method from Zhang (1993). Assume the two-dimensional coordinate, i.e., the row and column IDs, of pixel i is (y_{i1}, y_{i2}), $i=1,2,\ldots,q$.

(1) For pixels x_i and x_j, $i=1,2,...,q$-1; $j>i$, calculate

$$G_1(i,j)=\{x_k, x_i \mid y_{k2}>(y_{j2}-y_{i2})(y_{k1}-y_{i1})/(y_{j1}-y_{i1})+y_{i2}\}$$
$$G_2(i,j)=\{x_k, x_j \mid y_{k2}<(y_{j2}-y_{i2})(y_{k1}-y_{i1})/(y_{j1}-y_{i1})+y_{i2}\}$$
$$\bar{G}_1(i,j)=\{x_k, x_j \mid y_{k2}>(y_{j2}-y_{i2})(y_{k1}-y_{i1})/(y_{j1}-y_{i1})+y_{i2}\}$$
$$\bar{G}_2(i,j)=\{x_k, x_i \mid y_{k2}<(y_{j2}-y_{i2})(y_{k1}-y_{i1})/(y_{j1}-y_{i1})+y_{i2}\}$$

Thus there will be $q(q$-1) possible segmentaions. Calculate the error function for each segmentaion:

$$e(G(2))=\sum^{2}_{S=1}D(i_S, i_{S+1}-1)$$

where $D(i, j)=\sum^{2}_{S=1}(x_S-\bar{x}_{i-j})(x_S-\bar{x}_{i-j})'$, and

$$\bar{x}_{i-j}=(\sum^{j}_{S=i} x_S)/(j-i+1),\ i<j,\ i=1,2,...,q-1$$

Choose the segmentaions that meets *min e*(*G*(2)), the two segments can be achieved.

(2) Assume that *k* segments have been achieved. Following the procedure above, segmentate them as two segments respectively, and choose the segmentation that meets *max min e*(*G*(2)), thus *k*+1 segments is achieved.

(3) If *k*<*q*, segmentation continues, and if *k*=*q*, the calculation terminates.

Error function can be used as cluster distance level. At given level, we can use all resulatant segments as the segments of the image.

The improved two-dimensional ordered clustering method (Zhang et al., 2014) is also available for TDOS method.

2.1.6.4 Moving windows averaging segmentation (MWAS)

Moving windows averaging segmentation is based on boundary detection algorithm (Zhang and Schoenly, 1999; Zhang, 2005). Here I name it as MWAS.

For a gray image, the grayscale value, *j*, of any pixel falls in [0,255]. Calculate the frequency distribution of grayscale values. It means that there are 256 ordered grayscale values along the *x*-axis, and 1 measurement variable at each grayscale value *j* (i.e., number of the pixels with grayscale value *j*, f_j).

Consider a window with window width *Q* in calculation of moving average segmentation. This window is split into two equal halves, W_a and W_b, each half has *Q*/2 ordered grayscales. The grayscale location of this window is defined by *k*+0.5, where *k*=*Q*/2,*Q*/2+1,*Q*/2+2,...,256-*Q*/2. The average of f_j in each half window is

$$W_a(k+0.5)= \sum_{j=k-Q/2+1}^{k} f_j/(Q/2)$$
$$W_b(k+0.5)= \sum_{j=k+1}^{k+Q/2} f_j/(Q/2)$$

The distance between each of the resulting 256-*Q*+1 pairs of averages can be calculated. For a window, calculate the absolute distance

$$DS(k+0.5) =|W_a(k+0.5)-W_b(k+0.5)|$$

Monte Carlo technique is used to estimate expected mean distance of distance matrix for a window width *Q*. Each grayscale value is randomly repositioned along grayscale series 0~255 for *l*=1,2, .. ,*p* times (e.g., *p*=200), and distance for each of the reorderd data sets, producing a new distance matrix *DR*(*k*+0.5,*l*). For each *k*+0.5 window midpoint grayscale along the series (0,1,2,…,255), the mean expected distance and standard deviation are as follows

$$DSB(k+0.5)= \sum_{l=1}^{p}(DR(k+0.5,l)/p$$
$$SD(k+0.5)=(\sum_{l=1}^{p}(DR(k+0.5,l)-DSB(k+0.5))^2/(p-1))^{0.5}$$

The overal expected mean distance and average standard deviation for the series at window width Q is given by

$$DSB(\cdot)= \sum_{k=Q/2}^{256-Q/2} DSB(k+0.5)/(256-Q+1)$$
$$SDB= \sum_{k=Q/2}^{256-Q/2} SD(k+0.5)/(256-Q+1)$$

For each window midpoint location, the distance estimate can be transformed into *Z* score as follows

$$DZ(k+0.5)=(DS(k+0.5)-DSB(\cdot))/SDB$$

As done in Zhang et al. (2005), If the present ratio $SDB^2/DSB(\cdot)$ against the cumulative ratios of all window widths is less than 0.05, i.e., 95% of information has been obtained, or the number of window widths is greater than $n-Q+1$, then start to calculate pooled distance.turn to pooling procedure.

Finally, the pooled distance estimate is calculated for each window midpoint grayscale from T different window widths

$$DZB(k+0.5)= \sum_{s=1}^{T} DZ(k+0.5,s)/T$$

The overal expected mean distance and average standard deviation for the series of pooled results are $PDSB=0$ and $PSDB=1$ respectively.

Calculate lower limit of the confidential interval with different confidence degree (Zhang and Schoenly, 1999)

PDSB -2* *PSDB*	(Grain Level I)
PDSB -1.5* *PSDB*	(Grain Level II)
PDSB -1.0* *PSDB*	(Grain Level III)
PDSB -0.5* *PSDB*	(Grain Level IV)

If the pooled distance, $DZB(k+0.5)$, of window midpoint is less than the upper limit, we consider the corresponding grayscale location is a vally grayscale.

After doing these, the boundary grayscale locations (valley grayscale locations) can be determined across the series 0~255. Thus, the interval [0,255] can be segmentated into several segments or no any segment.

```
mw=3; st=9; %Initial window width=3; Increment of window width=9
percent=5;    %Cumulative information=95%
sim=200;       %Monte Carlo simulations=200
rgb=imread('brainTumorMRI.png');
imdim=numel(size(rgb));
if (imdim~=2) rgb=rgb2gray(rgb);
else rgb=rgb;
end
freq=imhist(rgb);
s=1;                    %1 variable, i.e., grayscale
a(s,:)=freq';
ts=256;
kp=1;
prop=0;su=0;
sm=0; w=mw;
while (sm>=0)
sm=sm+1;
t1=0; t2=0;
z=0;
for i=1:ts-w+1
```

```
for k=1:s
for j=i:w-1+i
if (j>(floor(w/2)+z)) t2=t2+a(k,j); else t1=t1+a(k,j); end
end
b(k,i)=t1/floor(w/2);
b(k,i+1)=t2/floor(w/2);
t1=0; t2=0;
end
md(i,sm)=floor(w/2)+z+0.5;
z=z+1;
ds2(i,sm)=0;
for l=1:s
ds2(i,sm)=ds2(i,sm)+abs(b(l,i)-b(l,i+1));
end
end
for i=1:ts-w+1
dsb(i)=0;
p(i)=0;
qq(i)=0;
end
kk=1;
while (kk<=sim)
cols=randperm(ts);
for k=1:s
for j=1:ts
cc(k,j)=a(k,cols(j));
end
end
t1=0; t2=0; z=0;
for i=1:ts-w+1
for k=1:s
for j=i:w-1+i
if (j>(floor(w/2)+z)) t2=t2+cc(k,j); else t1=t1+cc(k,j); end
end
b(k,i)=t1/floor(w/2);
b(k,i+1)=t2/floor(w/2);
t1=0; t2=0;
end
z=z+1;
ds=0;
for k=1:s
ds=ds+abs(b(k,i)-b(k,i+1));
end
dsb(i)=dsb(i)+ds/sim;
p(i)=p(i)+ds^2;
```

```
qq(i)=qq(i)+ds;
end
kk=kk+1;
end
for i=1:ts-w+1
sd(i)=(p(i)-2*qq(i)*dsb(i)+sim*dsb(i)^2)/(sim-1);
if (sd(i)>=0) sd(i)=sqrt(sd(i)); else sd(i)=0;
end
end
momn=0; mosd=0;
for i=1:ts-w+1
momn=momn+dsb(i);
mosd=mosd+sd(i);
g(i)=ds2(i,sm);
end
mnbar(kp)=momn/(ts-w+1);
sdbar(kp)=mosd/(ts-w+1);
su=su+sdbar(kp)^2/mnbar(kp);
prop=sdbar(kp)^2/mnbar(kp)/su;
for i=1:ts-w+1
dz(i,sm)=(ds2(i,sm)-mnbar(kp))/sdbar(kp);
end
kp=kp+1;
if (prop<=(percent/100.0))
q=kp-1; break;
end
if (sm>=ts-w+1)
q=kp-1; break;
end
w=w+st;
end
wc=floor(st/2)*q-floor(st/2)+ts-(mw+st*(q-1))+1;
hk=ts-(mw+st*(q-1))+1;
s2=floor(st/2)*q-floor(st/2)+1;
mt=1;
tv=0;
for i=1:ts-mw+1
if (tv>=hk) mt=mt-1; end
if ((i>=s2) && (tv<hk))
mt=q; tv=tv+1;
end
pdz(i)=0;
ap=i;
for j=1:mt
pdz(i)=pdz(i)+dz(ap,j);
```

```
ap=ap-floor(st/2);
end
pdz(i)=pdz(i)/mt;
s1=i*1.0/floor(st/2);
s3=(i-wc)*1.0/floor(st/2);
if ((floor(s1)==s1) && (i<s2)) mt=mt+1; end
if ((s3>floor(s3)) && (i>wc)) mt=mt+1; end
end
for i=1:ts-mw+1
g(i)=pdz(i);
end
mnbarr=0; sdbarr=1;
for j=1:4
th=0;
switch (j)
 case 1
      th=-2*sdbarr;            %Deep vally grayscale detection: I
 case 2
      th=-1.5*sdbarr;         %Mediate vally grayscale detection: II
 case 3
      th=-1*sdbarr;            %Mild vally grayscale detection: III
 case 4
      th=-0.5*sdbarr;         %Shallow vally grayscale detection. IV
end
on=1; off=1; ids=0;
for i=1:ts-w+1
if (g(i)<=(mnbarr+th))
if ((on==1) | (off==0))
ids=ids+1;
segs(j,ids)=i;
on=0; off=1;
end
continue;
else
if ((off==1) | (on==0))
off=0; on=1;
end
end
end
len(j)=ids;
end
segs=segs-1;
s=0;
for k=1:4
if (len(k)==0) continue; end
```

```
s=s+1;
end
[m,n]=size(rgb);
ss=0;
for k=1:4
if (len(k)==0) continue; end
rgb1=zeros(m,n);
ss=ss+1;
if (len(k)==1)
if (segs(k,1)==0)
for i=1:m
for j=1:n
rgb1(i,j)=0;
end
end
end
if (segs(k,1)>0)
for i=1:m
for j=1:n
if ((rgb(i,j)<=segs(k,1)))
rgb1(i,j)=0; end
if ((rgb(i,j)>segs(k,1)))
rgb1(i,j)=255; end
end
end
end
end
if (len(k)>=2)
if (segs(k,1)==0)
for i=1:m
for j=1:n
if ((rgb(i,j)>=segs(k,1)) && (rgb(i,j)<segs(k,2))) rgb1(i,j)=0; end
for l=2:len(k)-1
if ((rgb(i,j)>segs(k,l)) && (rgb(i,j)<=segs(k,l+1)))
rgb1(i,j)=segs(k,l); end
end
end
if ((rgb(i,j)>segs(k,len(k))) && (rgb(i,j)<=255)) rgb1(i,j)=255; end
end
end
if (segs(k,1)>0)
for i=1:m
for j=1:n
if ((rgb(i,j)>=0) && (rgb(i,j)<segs(k,1))) rgb1(i,j)=0; end
for l=1:len(k)-1
```

```
if ((rgb(i,j)>segs(k,l)) && (rgb(i,j)<=segs(k,l+1)))
rgb1(i,j)=segs(k,l); end
end
end
if ((rgb(i,j)>segs(k,len(k))) && (rgb(i,j)<=255)) rgb1(i,j)=255; end
end
end
end
switch (s)
 case 1
      subplot(1,1,1);imshow(rgb1);
 case 2
      subplot(1,2,ss);imshow(rgb1);
 case 3
      subplot(1,3,ss);imshow(rgb1);
 case 4
      subplot(2,2,ss);imshow(rgb1);
end
switch (k)
 case 1
      title('Grain Level I','FontSize',7);
 case 2
      title('Grain Level II','FontSize',7);
 case 3
      title('Grain Level III','FontSize',7);
 case 4
      title('Grain Level IV','FontSize',7);
end
end
```

2.1.7 Recognition of connected components

Connectedness and connected components are important topological property and structure in topological spaces (Zhang, 2012, 2016, 2018). Choose a certain pixel of a connected component, dialate it and intersect with the original image, and iterate until it does not change to obtain the extracted component. By doing so, all connected components can be obtained. The following Matlab codes are used to recognize connected components in a binary image:

```
rgb=imread('brainTumorMRI.png');
rgb=im2bw(rgb);
xr=bwlabel(rgb,8);              %Connectedness
u=unique(xr);                   %max(u): number of connected components
stru=strel('diamond',1);         %Diamond structure
for i=1:max(u(:))
ind=find(xr==i);                %Choose an object
lind=ind(1);
```

```
x0=zeros(size(rgb));
x0(lind)=1;
x=imdilate(x0,stru);
x1=x0;
while (~isequal(x1,x))
x1=x;
x=imdilate(x1,stru).*logical(rgb);
end
imr(:,:,i)=x;
end
for i=1:size(imr,3)
figure
im=imr(:,:,i);
imshow(im,[]);
end
```

2.1.8 Image alignment

Suppose we want to align two gray images, represented by two matrices $A=(a_{ij})$ and $B=(b_{ij})$, $i=1, 2, ..., m$; $j=1, 2, ..., n$, where m and n are the number of pixels along y-axis and x-axis, $0\leq a_{ij}\leq 1$ and $0\leq b_{ij}\leq 1$ are the standardized grayscale values at pixel (i, j) of two images respectively. First, two images are transformed to gray images. Resize the two images to the same dimension of $m\times n$, where m and n are the means of dimensions of two images respectively. The grayscale values of images are standardized as values in [0,1] (Zhang and Qi, 2019).

Matrices A and B are transformed to vectors A_1 and B_1 respectively, i.e., $(A_1, B_1)=\{(a_i, b_i) \mid i=1, 2, ..., p\}$ (Zhang, 2018b). By using Pearson correlation (Zhang, 2012, 2015, 2018a-b; Zhang and Li, 2015), we can achieve the test results for between-image correlation. A greater Pearson correlation means the higher similarity in image structure between two images.

Full Matlab codes for Pearson correlation between two images are as follows (Zhang, 2012, 2015, 2018a-b; Zhang and Qi, 2019):

```
sig=input('Input significance level(e.g., 0.01): ');
RGB1=imread('geneExpression.png'); RGB2=imread('geneExpression.png');
Gray1=rgb2gray(RGB1); Gray2=rgb2gray(RGB2);
m=round((size(Gray1,1)+size(Gray2,1))/2); n=round((size(Gray1,2)+size(Gray2,2))/2);
I=imresize(Gray1,[m n]); J=imresize(Gray2,[m n]);
A=im2double(I); B=im2double(J);
A1=reshape(A, m*n, 1);    %Transform matrix A to vector A1
B1=reshape(B, m*n, 1);    %Transform matrix B to vector B1
strr='';
strr=strcat(strr,[char(13) 'Pearson correlation between two images' char(13)]);
r=corr(A1,B1);
strr=strcat(strr,[char(13) 'Pearson correlation r=' num2str(r) char(13)]);
tvalue=abs(r)/sqrt((1-r^2)/(m*n-2));
p=(1-tcdf(tvalue,m*n-2))*2;
sigma=p<sig;
```

```
if (sigma==1) strr=strcat(strr,[char(13) 'Pearson correlation is statistically significant (p=' num2str(p) ')' char(13)]); end
if (sigma==0) strr=strcat(strr,[char(13) 'Pearson correlation is not statistically significant (p=' num2str(p) ')' char(13)]); end
fprintf(strr);
```

2.2 Software package

The software package, imageProcAnal (Version 1.0), and the future versions can be freely downloaded at: **http://www.iaees.org/publications/software/index.asp**

Before using the software, users are strongly to install the MATLAB software in your computer. After than, unrar the software package and double-click imageProcAnal.exe to open the software (Fig. 1). Detailed methods used during image processing will be recorded above the current image.

Fig. 1 The software, imageProcAnal (Version 1.0). Here an image has been loaded into the window.

2.3 Demo images

Demo images in present article are SF-RT gene expression pattern in *S. furcifera*, *geneExpression.png* (Lin et al., 2019; Fig. 2), MRI image used for experiment is infected with a tumor region in the temporal lobe, *brainTumorMRI.png* (Dogra et al., 2018; Fig. 3), and a human image, *wjzhang.png* (Fig. 4).

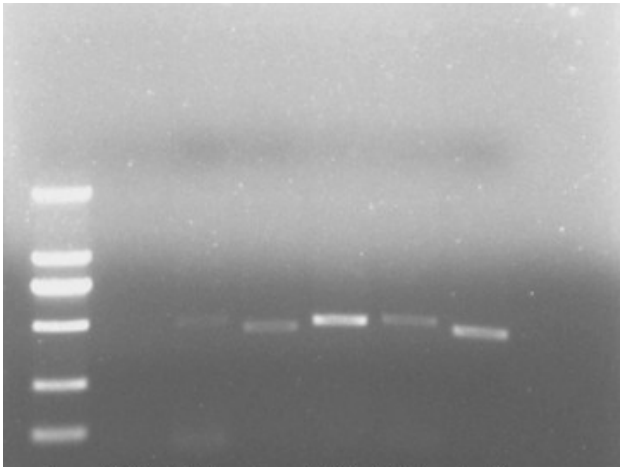


Fig. 2 SF-RT gene expression pattern in *S. furcifera*, geneExpression.png (Lin et al., 2019).

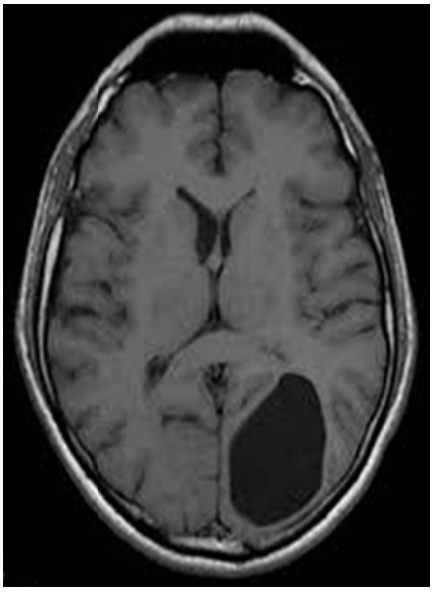


Fig. 3 MRI image used for experiment is infected with a tumor region in the temporal lobe, brainTumorMRI.png (Dogra et al., 2018).

Fig. 4 A human image, wjzhang.png.

3 Software Guide

3.1 File menu (Fig. 5)

Open: once the software starts and before making any operation on image, an image must be loaded into the window interface by using Open sub-menu.

Reopen: during any operation, Reopen can be used to re-loaded the original image initially loaded by Open sub-menu.

Save as: at any step, the current image(s) in the window can be saved using Save as sub-menu.

Save: the filename most recently used to save image can be used again to save new image for replacing the previous image.

Close: terminate and close the software.

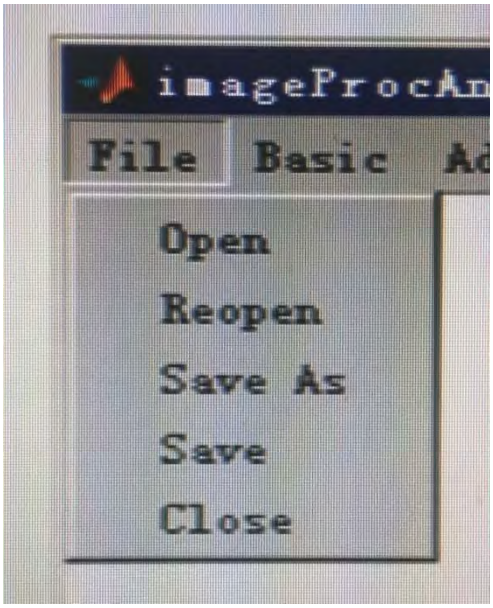


Fig. 5 File menu.

3.2 Basic module (Section 2.1.1; Fig. 6)

Resize: Resize an image to a specified size (Fig. 7a).

Rotate: Rotate an image by a specified degree (Fig. 7b).

Crop: Crop a rectangular area of the image (Fig. 8).

Zoom: Zoom on, off and out an image.

Dilate: Dilate an image with specified structure and size (Fig. 9).

Pixelate: Pixelate an image with the mouse-specified area (Fig.10). The last point is fixed with double click, and the other points are fixed with single click.

Watermark: Watermark an image with specified parameters (Fig. 11). The copyright of the image can be guaranted by adding the watermark.

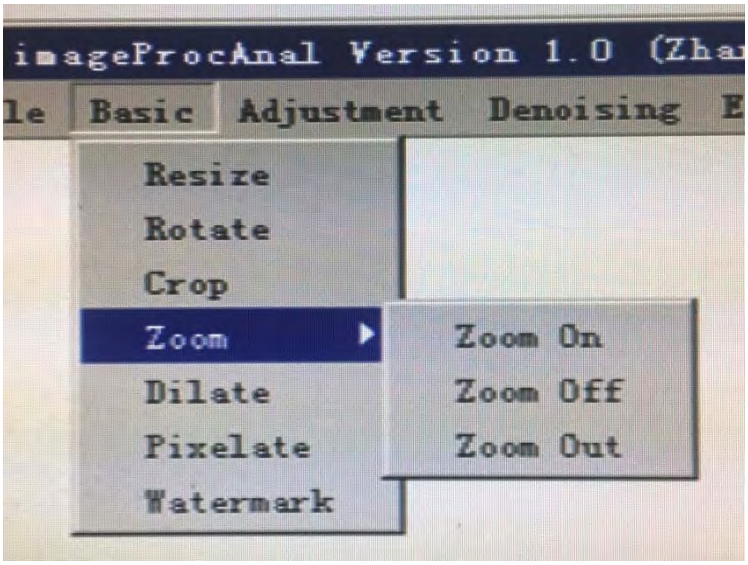
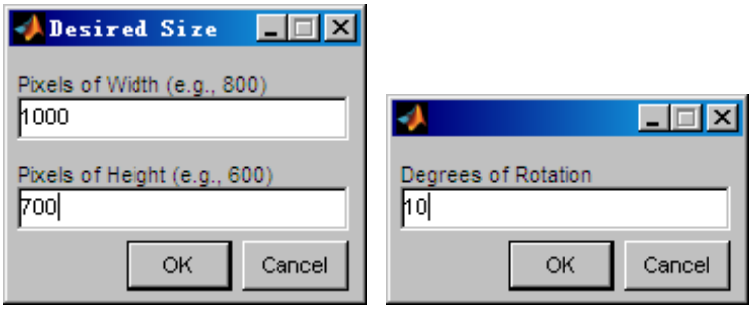


Fig. 6 Basic module.



(a) (b)

Fig. 7 Parameter input window for Resize (a) and Rotate (b).

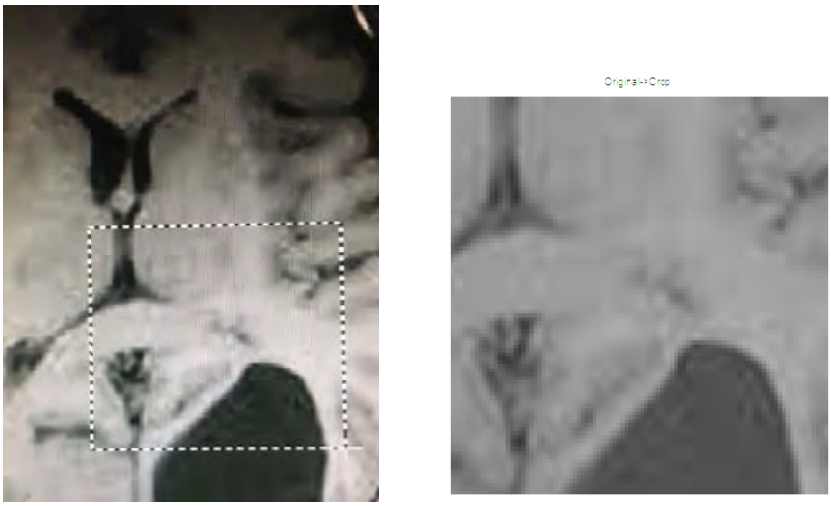


Fig. 8 Crop a rectangular area of the image.

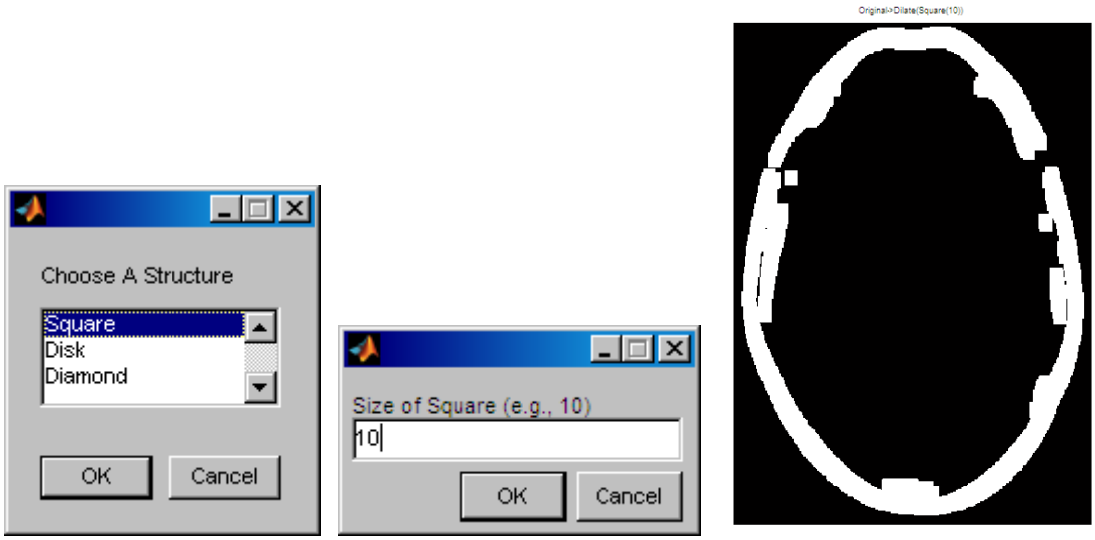


Fig. 9 Dilate an image with specified structure and size.

Fig. 10 Pixelate an image with mouse-specified area.

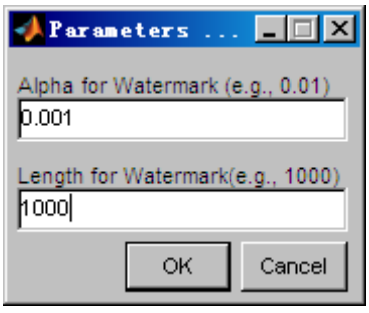


Fig. 11 Watermark an image with specified parameters.

3.3 Adjustment module (Section 2.1.2; Fig. 12)

Edge Adjustment: grayscale adjustment of image edges (Fig. 13).

Contrast Adjustment: grayscale adjustment of an image with specified contrast parameters (Fig. 14).

Gamma Adjustment: grayscale adjustment of an image with specified gamma value in the Matlab function imadjust (Fig. 15).

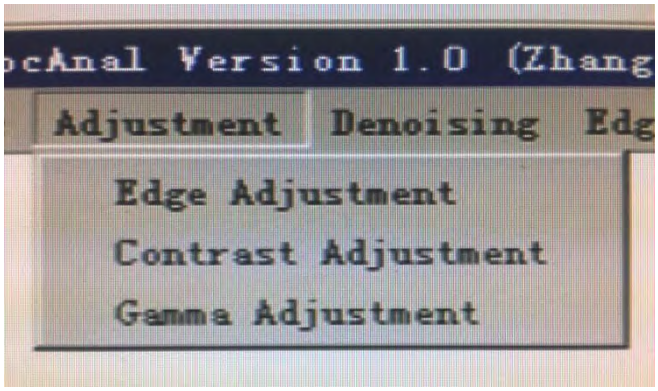


Fig. 12 Adjustment module.

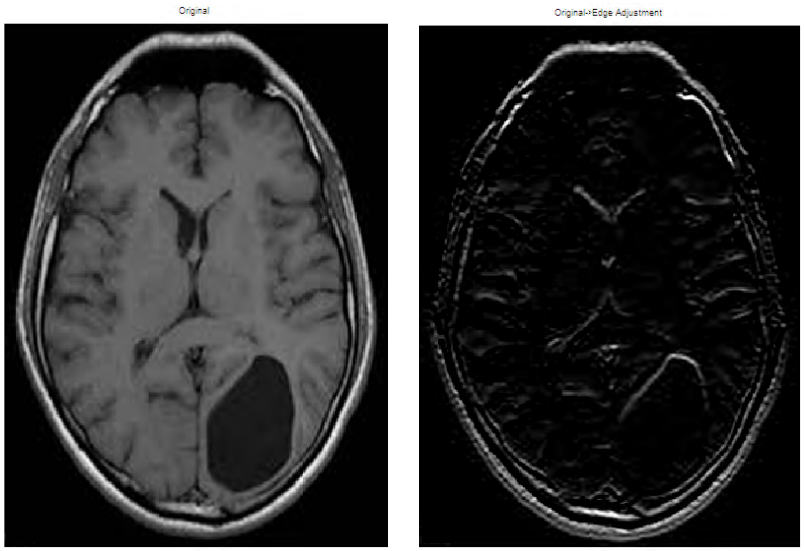


Fig. 13 Original (left) and edge adjusted (right) images.

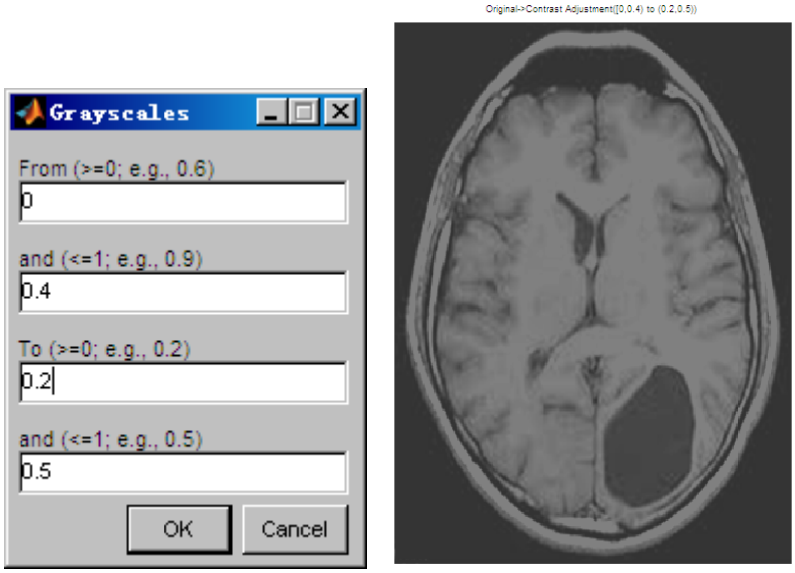


Fig. 14 Adjustment of an image with specified contrast parameters.In this example, the image was adjusted for more darkness.

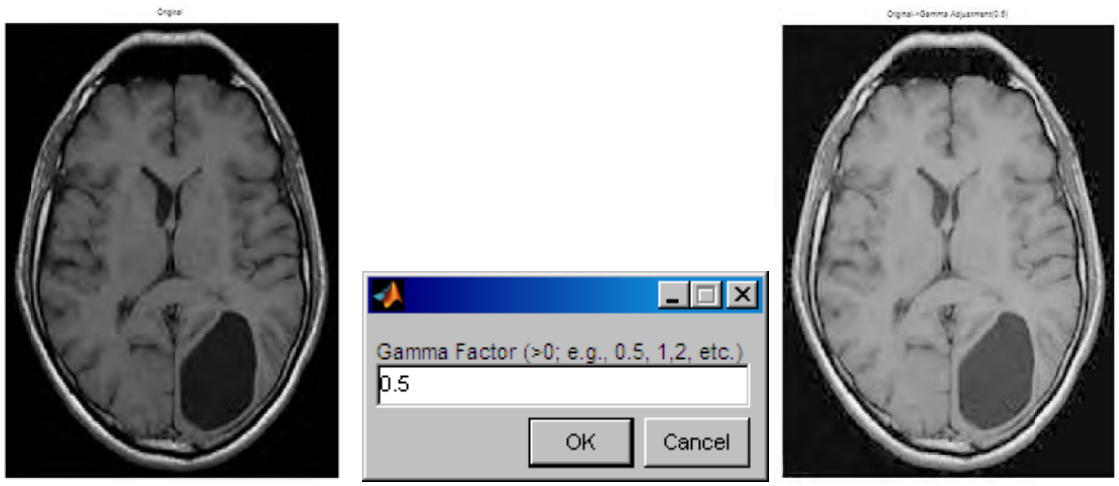


Fig. 15 Original (left) and gamma adjusted (gamma=0.5; right) images.

3.4 Denoising module (Section 2.1.3; Fig. 16)

It is necessary to use the methods in this module to remove noises in the images with background noises, e.g., the image geneExpression.png (Fig. 2). For high-quality and clean images as brainTumorMRI.png (Fig. 3) and wjzhang.png (Fig. 4), the use of this module is unnecessary.

Mean Filtering: see section 2.1.3.1.

Median Filtering: see section 2.1.3.2.

Adaptive Filtering: see section 2.1.3.3.

Wavelet Denoising: see section 2.1.3.4.

Joint (M_M_A) Filtering: Based on the original image, sequentially use Mean Filtering, Median Filtering, and Adaptive Filtering to remove noises in the image. It is one time and convenient use of the first three methods.

Joint (M_M_A_W) Filtering: Based on the original image, sequentially use Mean Filtering, Median Filtering, Adaptive Filtering and Wavelet Denoising to remove noises in the image. It is one time and convenient use of all of four methods.

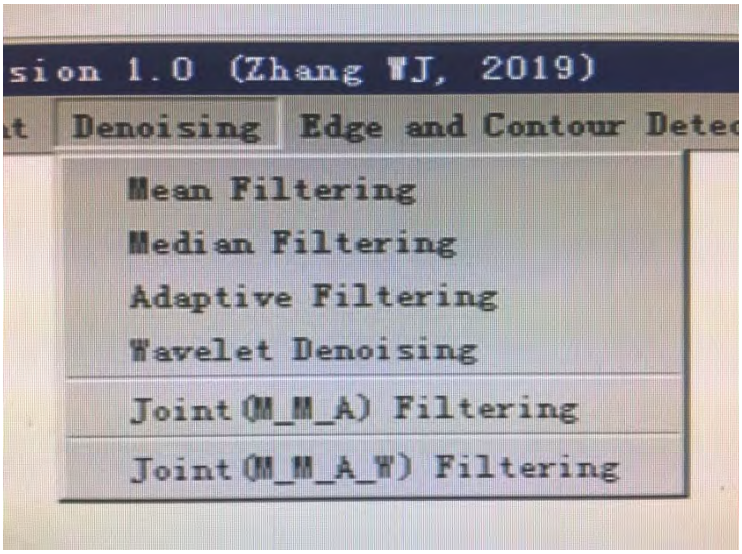
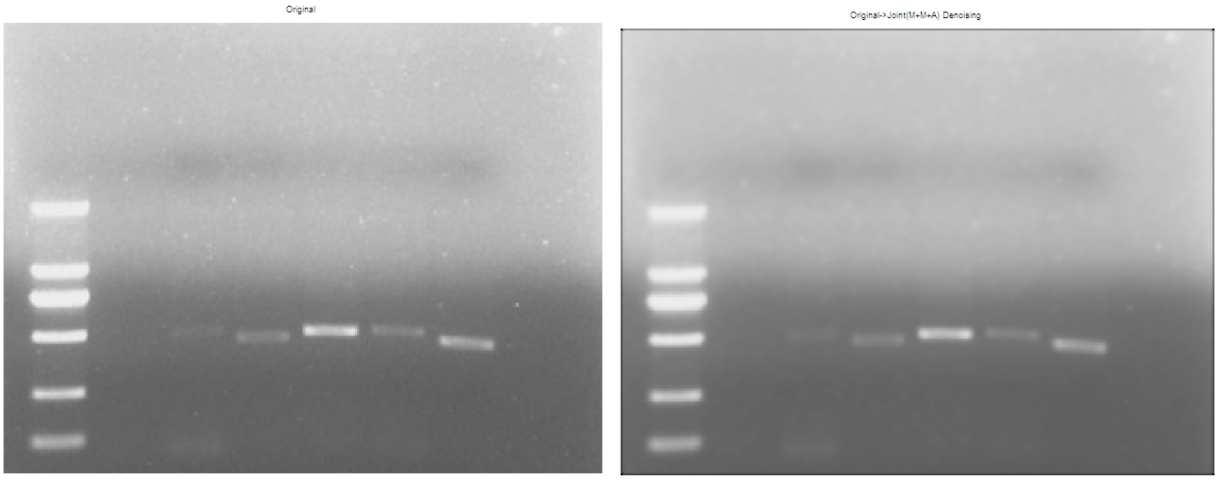


Fig. 16 Denoising module.



(a) (b)

Fig. 17 Comparison between original and noise-removed images. (a) Original image (Fig. 2); (b) Noise-removed image by using Joint (M_M_A) Filtering.

3.5 Edge and Contour Detection module (Section 2.1.4; Fig. 18)

Detection of object edges in an image and image contour can be conducted by using the methods in this module.

Sobel, Prewitt, Laplacian of Gaussian, Roberts and Canny Edge Detection: one of these methods can be chosen to detect edges (Section 2.1.4.1; Fig. 19). For an unclean image with noises, like geneExpression.png in Fig. 2, it is suggested using denoising methods to remove noises in advance, and then make edge detection.

Contour Detection: the contour of the image is detected using this method (Section 2.1.4.2; Fig. 20).

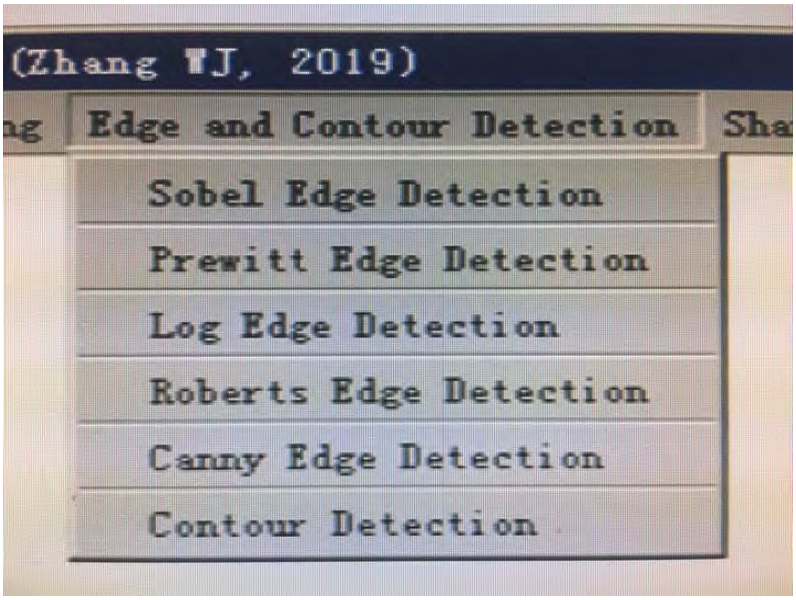


Fig. 18 Edge and Contour Detection module.

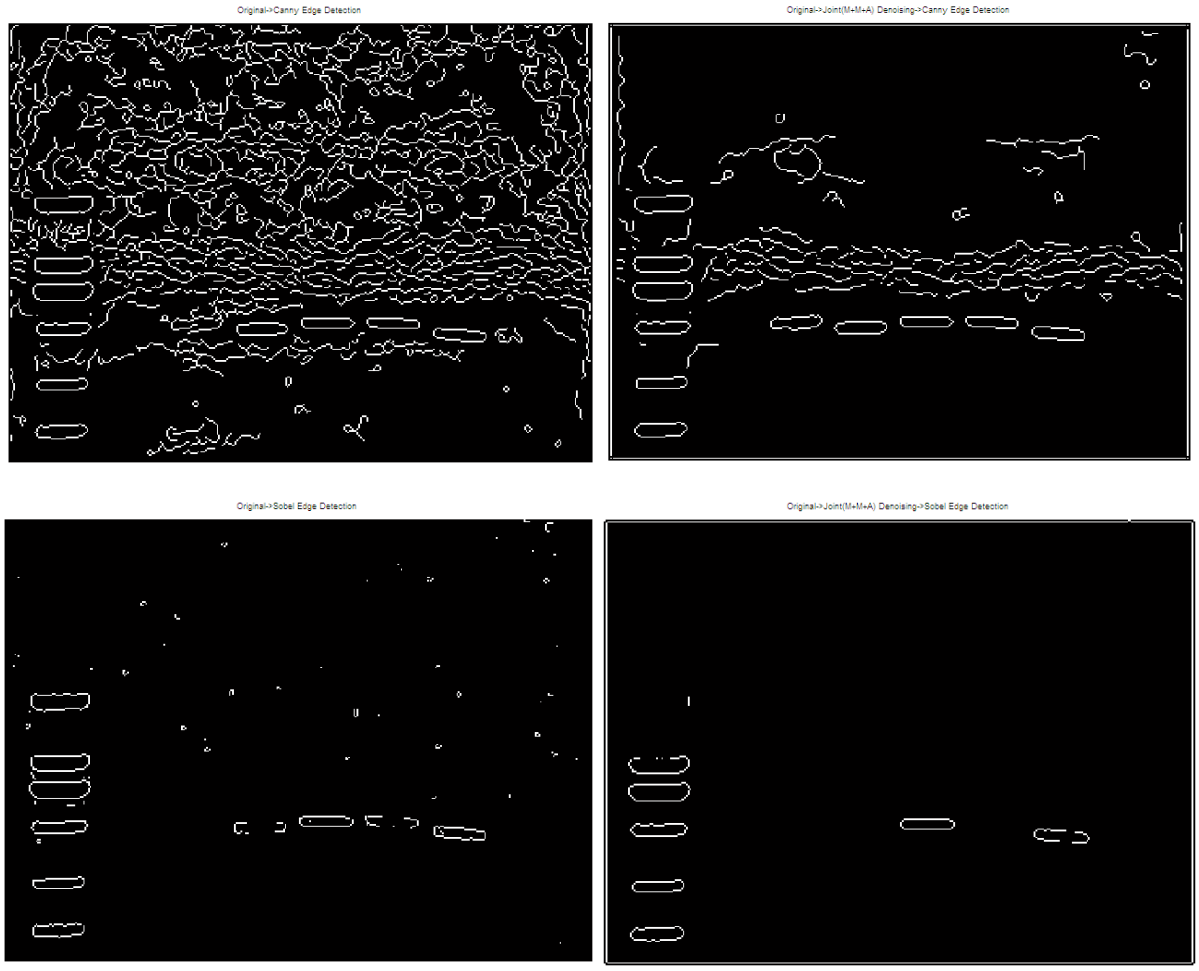


Fig. 19 Canny (upper) and Sobel (lower) edge detection on the images without (left) and with (right) removal of noises (here using Joint (M_M_A) Filtering).

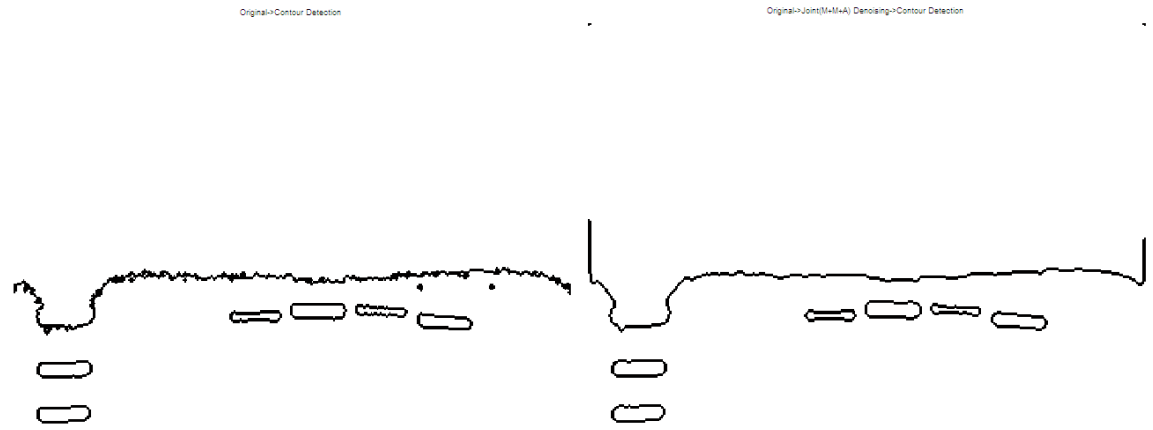


Fig. 20 Contour detection on the images without (left) and with (right) removal of noises (here using Joint (M_M_A) Filtering).

3.6 Sharpening module (Section 2.1.5; Fig. 21)

One of the methods, Sobel Sharpening, Prewitt Sharpening, and Laplacian Sharpening can be used to sharpen an image (Fig. 22). For an unclean image with noises, like geneExpression.png in Fig. 2, it is suggested using denoising methods to remove noises and then make sharpening.

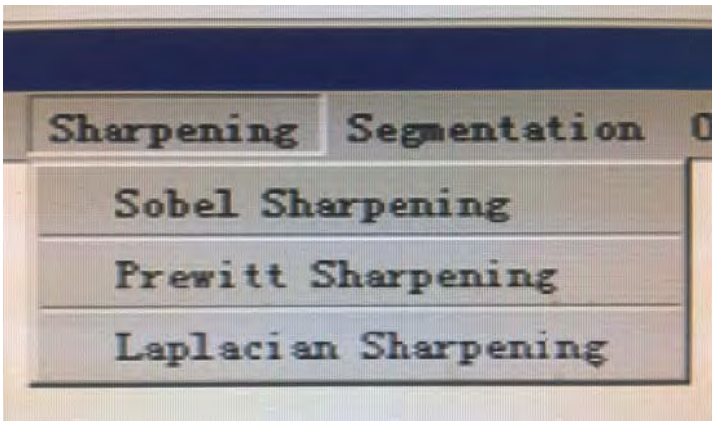


Fig. 21 Sharpening module.

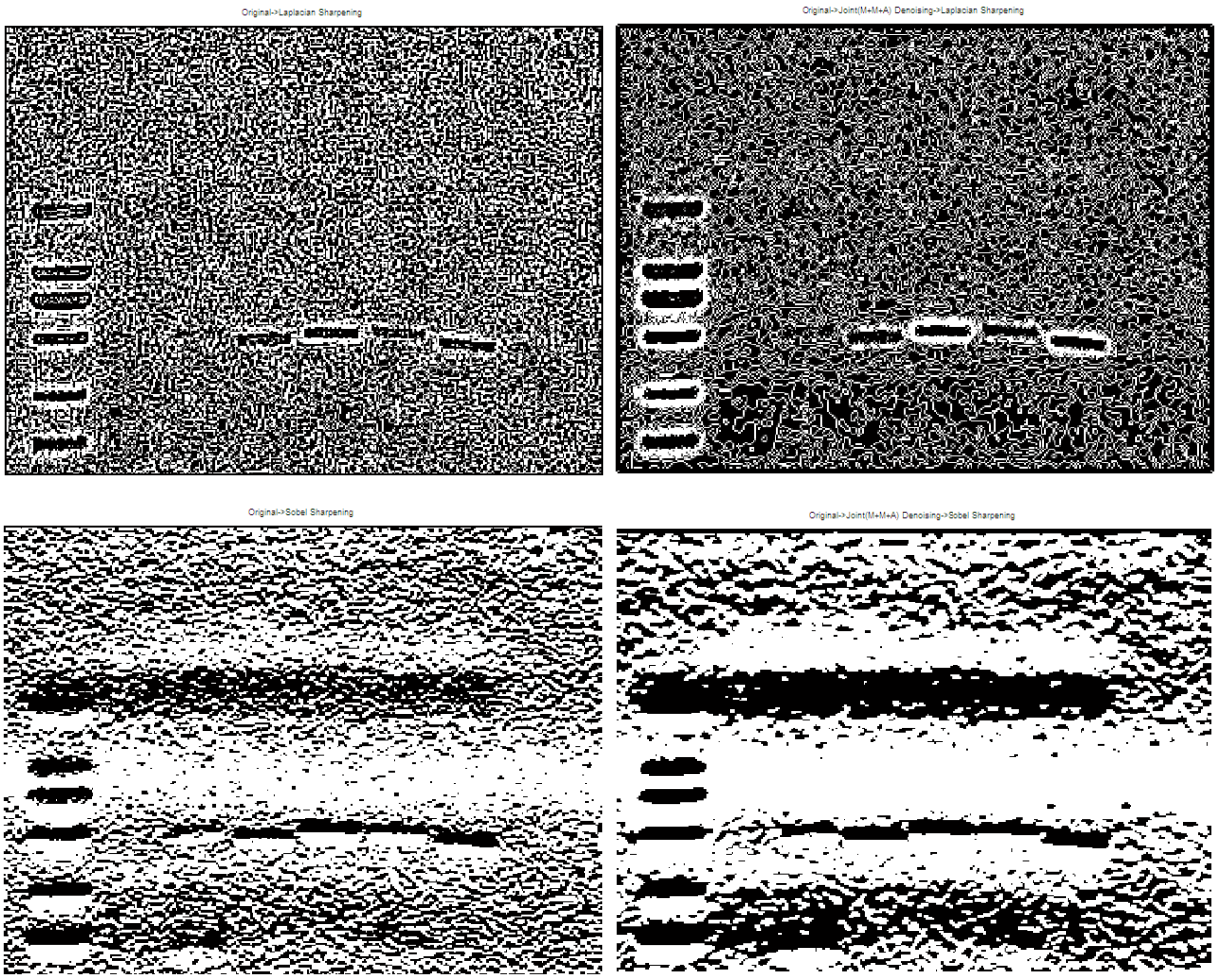


Fig. 22 Laplacian (upper) and Sobel (lower) sharpening of the images without (left) and with (right) removal of noises (here using Joint (M_M_A) Filtering).

3.7 Segmentation module (Section 2.1.6; Fig. 23)

One of the methods, Maximum Inter-segment Variance Threshold (Seection 2.1.6.1; Fig. 24), Moving Windows Averaging Segmentation (Section 2.1.6.4; Fig. 25-26), and Single Threshold Hist (Section 2.1.6.2; Fig. 27) can be used to make segmentation on an image (Fig. 23). For an unclean image with noises, like geneExpression.png in Fig. 2, it is suggested using denoising methods to remove noises and then make segmentation.

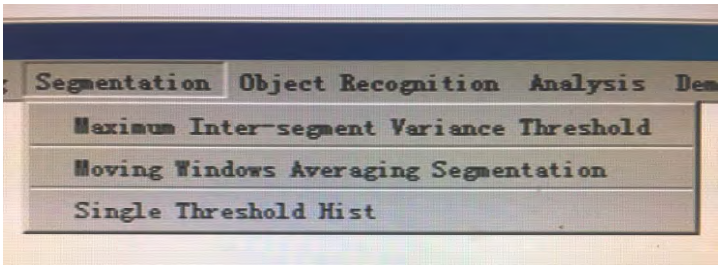


Fig. 23 Segmentation module.

Fig. 24 Maximum Inter-segment Variance Threshold segmentation method (Here the original image, geneExpression.png, has been denoised by using Joint (M_M_A) Filtering).

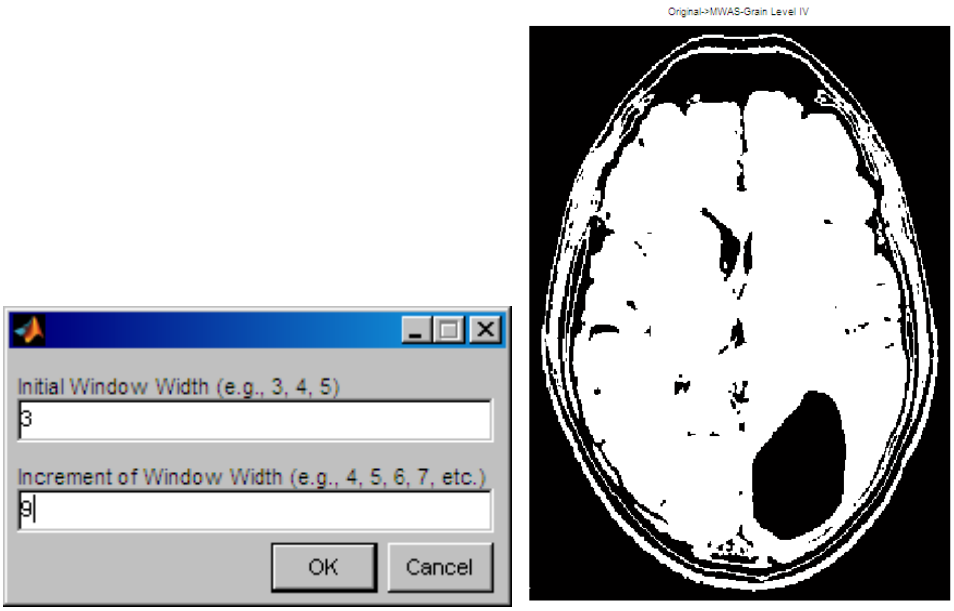


Fig. 25 Moving Windows Averaging Segmentation method (for brainTumorMRI.png).

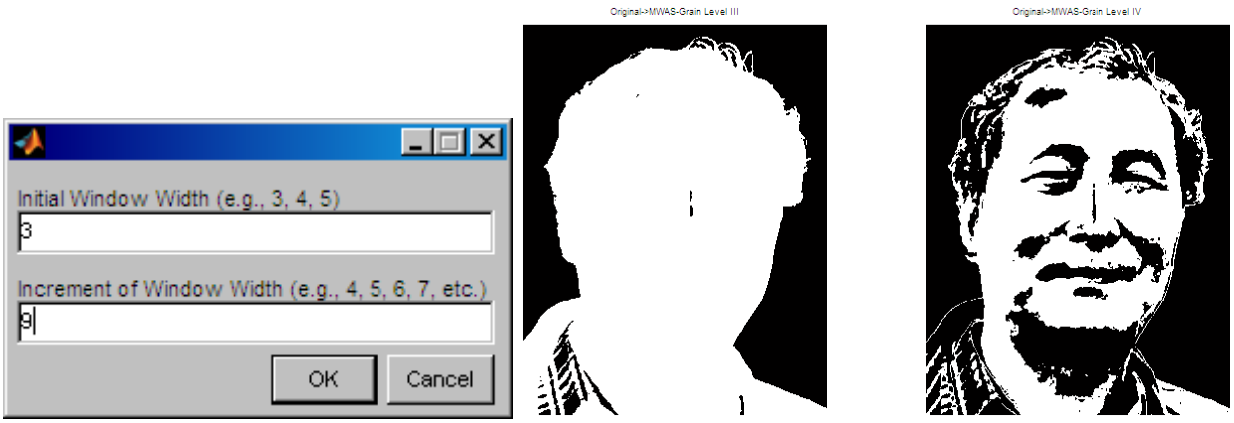


Fig. 26 Moving Windows Averaging Segmentation method (for wjzhang.png).

Fig. 27 Single Threshold Hist segmentation method (Here the original image, geneExpression.png, has been denoised by using Joint (M_M_A) Filtering).

3.8 Object Recognition module (Section 2.1.7; Fig. 28)

Once Connected Components Recognition is performed, connected components in the image will be displayed one by one in different windows (Fig. 29).

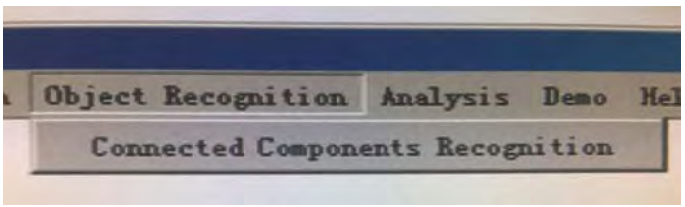


Fig. 28 Object Recognition module.

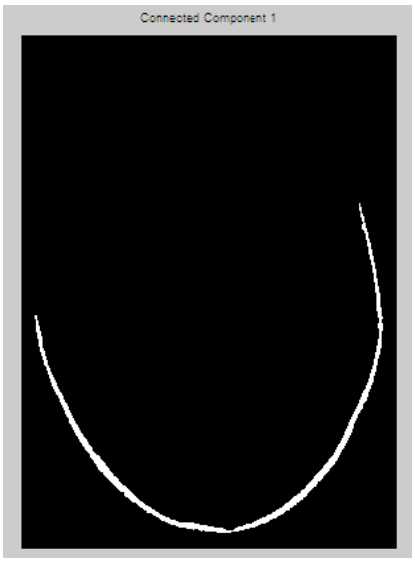


Fig. 29 One of the connected components in the image, brainTumorMRI.png (denoised by using Joint (M_M_A) Filtering).

3.9 Analysis module (Section 2.1.8; Fig. 30)

Currently, Alignment method (Zhang and Qi, 2019) for Pearson correlation between images is included in the module. By opening several images, the Pearson correlation matrix of these images can be obtained using this method (Zhang and Qi, 2019).

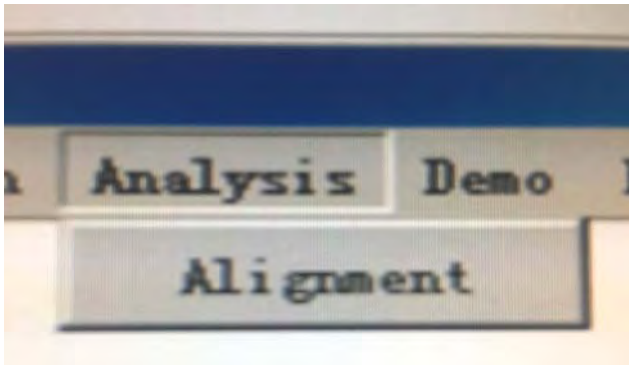


Fig. 30 Analysis module.

3.10 Demo module (Fig. 31)

In this module, we can choose to observe various demos of image processing (Fig. 32-33).

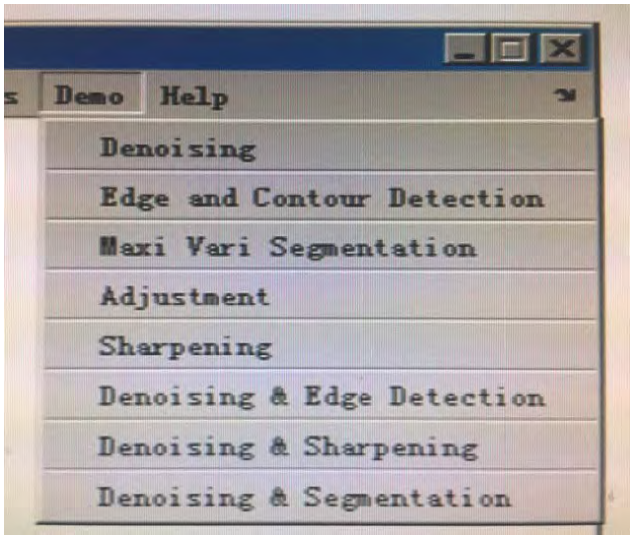


Fig. 31 Demo module.

Fig. 32 Demo for Denoising and Segmentation.

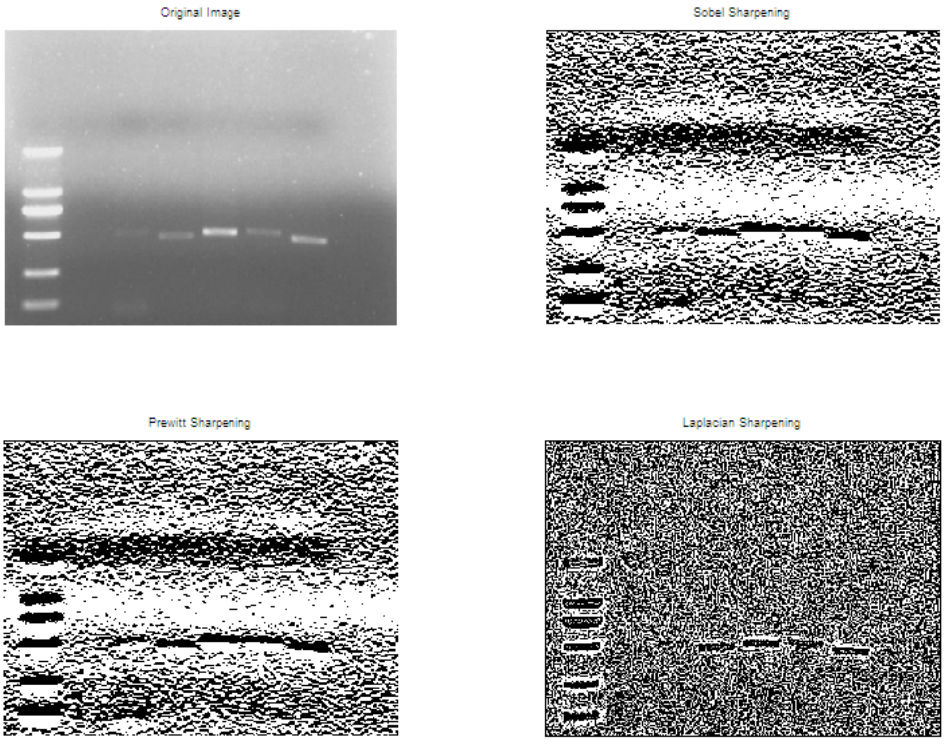


Fig. 33 Demo for Sharpening.

References

Bhardwaj N, Solanki A. 2016. An efficient algorithm for color image segmentation. Selforganizology, 3(3): 87-99

Dogra J, Prashar N, Jain S, Sood M. 2018. Improved methods for analyzing MRI brain images. Network Biology, 8(1): 1-11

Lin J, Zhang M, He JY, et al. 2019. Characterization of two non-LTR retrotransposons from Sogatella furcifera and *Nilaparvata lugens*. Arthropods, 8(1): 7-16

Wu HX. 2006. Recognition and Processing of Ecological Landscape Images. MS Dissertation, Sun Yat-sen University, Guangzhou, China

Zhang WJ. 1993. Two-dimensional order clustering and its application in the regionalization of agriculture. Bulletin of Soil and Water Conservation, 13(1): 34-41

Zhang WJ. 2005. An improved algorithm on boundary detection of ecological transect and network software. Journal of Biomathematics, 20(4): 477-486

Zhang WJ. 2012. Computational Ecology: Graphs, Networks and Agent-based Modeling. World Scientific, Singapore

Zhang WJ. 2015. Calculation and statistic test of partial correlation of general correlation measures. Selforganizology, 2(4): 65-77

Zhang WJ. 2016. Detecting connectedness of network: A Matlab program and application in tumor pathways and a phylogenic network. Selforganizology, 3(4): 117-120

Zhang WJ. 2018a. Fundamentals of Network Biology. World Scientific Europe, London, UK

Zhang WJ. 2018b. Network matrix based methods for between-network comparison. Network Biology, 8(4): 144-152

Zhang WJ, Qi YH. 2019. Matlab methods for calculation of between-image correlations and similarities. Ornamental and Medicinal Plants, 3(3-4): 6-12

Zhang WJ, Li X. 2015. General correlation and partial correlation analysis in finding interactions: with Spearman rank correlation and proportion correlation as correlation measures. Network Biology, 5(4): 163-168

Zhang WJ, Qi YH. 2019. Matlab methods for calculation of between-image correlations and similarities. Ornamental and Medicinal Plants, 3(3-4): 6-12

Zhang WJ, Qi YH, Zhang ZG. 2014. Two-dimensional ordered cluster analysis of component groups in self-organization. Selforganizology, 1(2): 62-77

Zhang WJ, Schoenly KG. 1999. BOUNDARY: A Program for Detecting Boundaries in Ecological Landscapes. IRRI Technical Bulletin No.3. International Rice Research Institute, Manila, Philippines