International Journal of Trend in Scientific Research and Development (IJTSRD)
International Open Access Journal
ISSN No: 2456 - 6470 | www.ijtsrd.com | Volume - 1 | Issue – 5

Fourier Filtering Denoising Based on Genetic Algorithms

Jiahe Shi
Faculty of Communication Engineering, Qiushi Honor College, Tianjin University

ABSTRACT

As the traditional filtering methods may cause the problem of blur when applied on stripe-like images, this paper raised a new method intending to solve the problem, and provided the process of the method development. The method, based on genetic algorithms as well as two-dimensional Fourier transform, offered a new perspective to denoise an image with abundant components of high frequency, and is of certain significance.

Keywords : *2-D Fourier transform, genetic algorithms, autocorrelations*

I. Operating Environment

This method is developed based on Matlab 2014b. Corresponding or updated version is recommended when verifying to ensure that the program runs normally. As the algorithm used a multi-cycle, all three versions of the program require much time. For the first edition of the program, when the number of individuals is only 10, running a generation requires up to 10 minutes or so, the second version of the program also takes about 6 minutes per generation, while the third edition of each generation about 2 minutes.

II. Problem background

This method is mainly applied to the repair of stripe-like images polluted by Gaussian noise. The general idea of filtering an image is to preserve its low frequency component meanwhile remove the high frequency component. For striped images (such as zebra), the frequency domain of high frequency components is fairly abundant. The problem of image blur will emerge on conditions of direct filtering. So it is necessary to adopt a new method for filtering.

In the preparation of each version of the program, ideas have undergone major changes, such as the shape of the filter. So parts of the Notes or other statements that does not affect the results may be left in the program.

III. Methods

Firstly, the characteristics of the original image spectrum and the noise image are analyzed, as shown in the following figure.

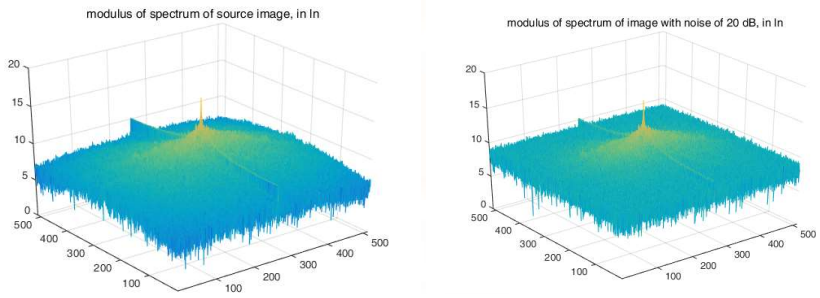

Figure 3.1 modulus of spectrum with or without noise

An image of zebra is used here. From the nature of the Fourier transform we know that the spectrum of the noise image is equal to the simple superposition of the noise spectrum and the image spectrum, i.e.

$$F(\mathbf{A}+\mathbf{n}) = F(\mathbf{A}) + F(\mathbf{n});$$

Thus, the noise spectrum "submerges" the spectrum of the image. Since the noise spectrum is random and unpredictable, we cannot completely restore the image spectrum in the frequency domain when the noise spectrum is unknown, but only have the result spectrum approached as close as possible to the spectrum of the original image. This is the general idea of this method. From the above graphs we can see that the high-frequency part of the original image is obviously subsided, and after the noise is added, the high-frequency region is obviously uplifted and relatively flat. In the low frequency part, the original image spectrum "terrain" is higher, making the "submerged" effect of noise not significant. Namely there is no obvious difference in the trend in the low frequency part. Therefore, it is desirable to find a filter function that best fits the image through a genetic algorithm so that it can properly restore the trend of the spectrum.

The following figure shows the spectrum of the image flattened:

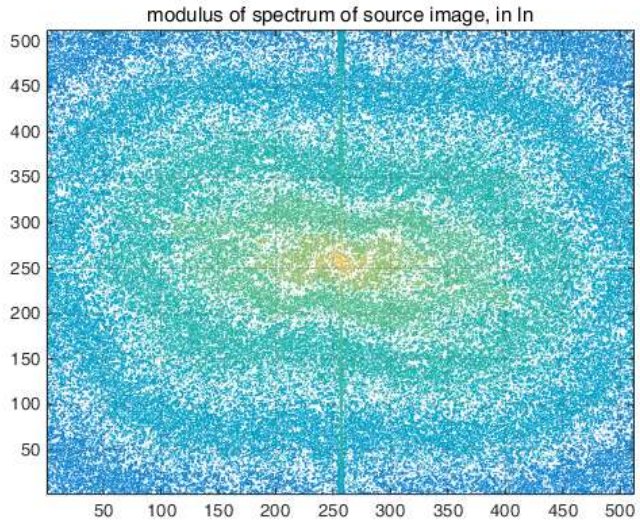

Figure 3.2 modulus of spectrum of source image, in ln

In order to make the algorithm be generic, considering the characteristics of the image spectrum, the filter is set to a series of concentric ellipses. In order to obtain similar attenuation of the four corners of the high frequency component, it is also required a gradually transition from ellipse to circle as the major axis increases. The parameters of the filter, i.e., the direction of the ellipse, the initial eccentricity, the velocity of the transition to the circle, the trend of the attenuation value, are used as the genes in the genetic algorithm, and the optimal value is given by the genetic algorithm.

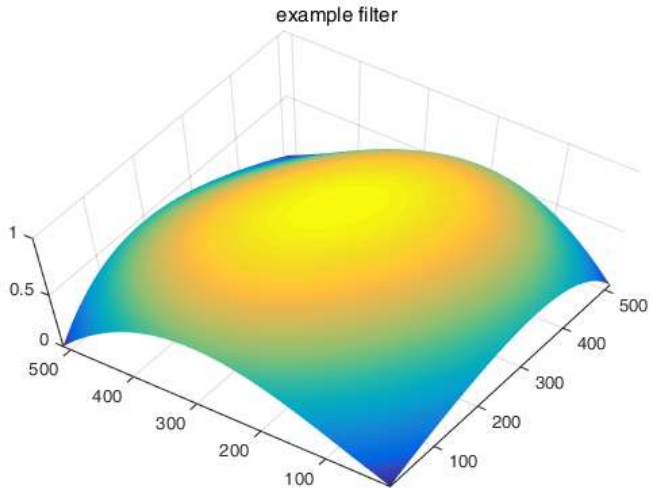

Figure 3.3 example filter

In particular, taking the characteristics of abounding high frequency components of the fringe image into account, it is supposed to restore the high frequency part of the spectral details as much as possible under the prerequisite of ensuring that the image spectrum trend approaches that of the original image. For the question of how to extract the details of high frequency components from the noise image, the method of obtaining gradient information and denoising with threshold is adopted in this method.

The gradient method is applied to extract the edge information of the image, that is, the high frequency component of the image. In this method, the gradient of the image, the derivative (the difference) of the two directions x and y, is calculated first, giving the direction of the most grayscale change at each point. The directional derivative (difference) is obtained then for this direction, so that the edge information of the image can be obtained. When the image is added 20dB Noise, the image edge information is obtained as follows:

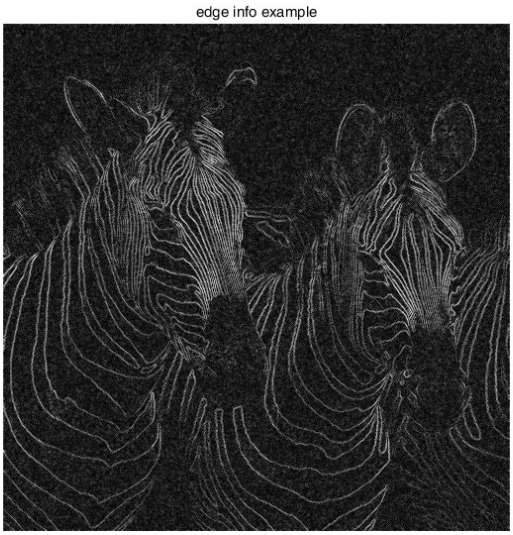

Figure 3.4 example of edge information

In order to restore as much as possible the noise-free high-frequency components from the information, considering that the magnitude of "edge information" formed by the noise tends to be low, a threshold is set thusly, ignoring the “edge information” where the gray value is less than the threshold. The process effect is as follows:

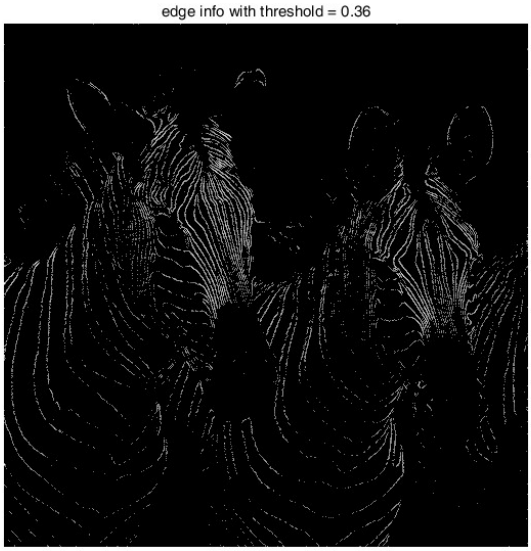

Figure 3.5 filtered edge information with threshold = 0.36

It can be seen that although the edge of the image is also lost in some degree, it still retains part of the edge information. With this part of the edge information (rather than nothing at all), it will be possible to modify the high-frequency spectrum, and thus try to maximize the restoration of the details of the image.

In the process of designing this method, there are three editions of program formed. The core idea of the three editions is to use genetic algorithm to achieve the optimal individual of filtering effect through several iterations. For genetic algorithms, there must be indicators that can be used to quantify the restoration of individuals. And the main difference among the three editions of the program is that the indicator is selected different. The versions are described here separately.

3.1 The first edition of the program

The evaluation indicator of the first edition of the program is the sharpness of the central peak of the autocorrelation function of the whole image. The following demonstrates its principles.

First the concept of two-dimensional autocorrelation function is explained. For $m \times n$ order matrix:

$$\mathbf{A}=\begin{bmatrix} a_{11} & \cdots & \cdots & \cdots & a_{1n} \\ \vdots & \ddots & & & \vdots \\ \vdots & & a_{ij} & \cdots & a_{in} \\ \vdots & & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix};$$

Define its cyclic shift as:

$$\mathbf{A}=\begin{bmatrix} a_{(m-p+1)(n-q+1)} & \cdots & a_{(m-p+1)n} & a_{(m-p+1)1} & \cdots & a_{(m-p+1)(n-q)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m(n-q+1)} & \cdots & a_{mn} & a_{m1} & \cdots & a_{m(n-q)} \\ a_{1(n-q+1)} & \cdots & a_{1n} & a_{11} & \cdots & a_{1(n-q)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{(m-p)(n-q+1)} & \cdots & a_{(m-p)n} & a_{(m-p)1} & \cdots & a_{(m-p)(n-q)} \end{bmatrix};$$

Referred to as $\mathbf{A}_{pq}$. Obviously, the following expression is true for p and q :

$$\begin{cases} |p| < m \\ |q| < n \end{cases};$$

Then the elements within its autocorrelation matrix can be given by the following equation:

$$R_{pq} = \frac{1}{mn}\mathbf{A} \cdot \mathbf{A}_{pq};$$

Where the symbol " · " represents the dot product operation for the matrix, that is, the sum after the multiplication of the corresponding term. We can see that the above equation is actually an average value. For convenience, the autocorrelation matrix is expressed as:

$$\mathbf{R}_{\mathbf{A}} = E(\mathbf{A} \cdot \mathbf{A}_{pq}); \qquad \text{(Equation 3.1.1)}$$

For white noise, that is, Gaussian noise, we know that its autocorrelation characteristics are very sharp, the value of the center peak is very large, whereas the non-center value is very small, as shown below:

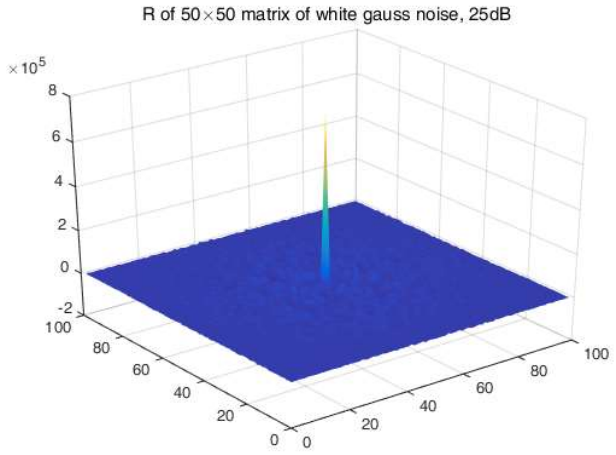

Figure 3.1.1 R of matrix of 25dB white gauss noise

For the general image, the autocorrelation matrix is relatively gentle, as shown in the following figure:

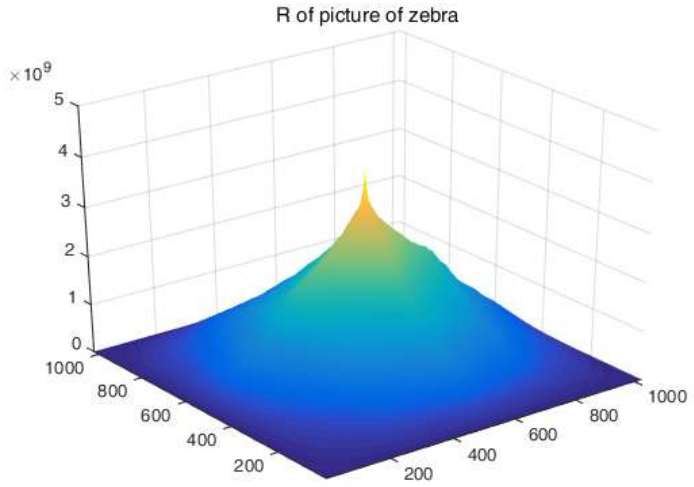

Figure 3.1.2 R of the example image

So the filtering effect can be evaluated by judging the sharpness of the matrix $\mathbf{R}$.

The next step is to prove that the autocorrelation matrix of superposition of the image matrix $\mathbf{A}$ and noise $\mathbf{n}$ is equivalent in shape to the simple superposition of the two autocorrelation matrices, i.e.

$$\mathbf{R}_{\mathbf{A+n}} = \mathbf{R}_{\mathbf{A}} + \mathbf{R}_{\mathbf{n}} + C;$$

Available from Equation 3.1.1:

$$\begin{aligned} \mathbf{R}_{\mathbf{A+n}} &= E((\mathbf{A}+\mathbf{n}) \cdot (\mathbf{A}+\mathbf{n})_{pq}) \\ &= E((\mathbf{A}+\mathbf{n}) \cdot (\mathbf{A}_{pq}+\mathbf{n}_{pq})) \\ &= E(\mathbf{A} \cdot \mathbf{A}_{pq} + \mathbf{n} \cdot \mathbf{n}_{pq} + \mathbf{A} \cdot \mathbf{n}_{pq} + \mathbf{n} \cdot \mathbf{A}_{pq}) \end{aligned}$$

Note that the image matrix and noise should be completely independent, so there are:

$$E(\mathbf{n} \cdot \mathbf{A}_{pq}) = E(\mathbf{n}_{pq} \cdot \mathbf{A}) = E(\mathbf{n}) \cdot E(\mathbf{A}) = C;$$

In this way, the autocorrelation matrix of the image matrix with noise is equal to the autocorrelation matrix of the original image, where the autocorrelation matrix of the noise is superimposed as well as the whole lifted by the height of constant C . Thus the shape is equal to the simple superposition of both.

In order to describe the sharpness of the spikes, also take the gradient method to obtain the maximum value of the directional derivative in the direction of gradient in the matrix $\mathbf{R}$. And then the ratio of it to the average difference degree (i.e., standard deviation) of matrix $\mathbf{R}$ is used as a basis for measuring the sharpness of the spikes, which is:

$$\eta = \frac{\max(\mathbf{R})}{\sigma_{\mathbf{R}}};$$

The lower the ratio, the better the filtering effect. In the genetic algorithm thusly this ratio can act as an evaluation function, ascending order.

3.2 The second edition of the program

Although the first version of the program is theoretically feasible, but in practice due to the autocorrelation function call for too much time, another alternative evaluation indicators is required as substitute. The second edition of the program therefore takes the mean of power spectrum as the evaluation index.

From the Wiener-Khinchin theorem we know that the power spectrum and the autocorrelation function are a pair of Fourier transforms, i.e.

$$\mathbf{S} = F(\mathbf{R});$$

So we can get the power spectrum of the noise image:

$$\begin{aligned} \mathbf{S}_{\mathbf{A+n}} &= F(\mathbf{R}_{\mathbf{A+n}}) \\ &= F(\mathbf{R}_{\mathbf{A}} + \mathbf{R}_{\mathbf{n}} + \mathrm{C}) \\ &= \mathbf{S}_{\mathbf{A}} + \mathbf{S}_{\mathbf{n}} + \delta \end{aligned}$$

If we ignore the central impact, we can see that the power spectrum of the noise image is the superposition of the power spectrum of the original image and the noise image. So there is:

$$\begin{aligned} E(\mathbf{S}_{\mathbf{A+n}}) &= E(F(\mathbf{R}_{\mathbf{A+n}})) \\ &= E(\mathbf{S}_{\mathbf{A}}) + E(\mathbf{S}_{\mathbf{n}}) \end{aligned}$$

And because of:

$$\mathbf{S}_{\mathbf{A+n}} = \left|F(\mathbf{A}+\mathbf{n})\right|^2;$$

There must be a non-negative expected value of the power spectrum. So $E(\mathbf{S}_{\mathbf{A+n}})$ can be regarded as the evaluation indicator. The lower value of the indicator indicates more ingredients of $E(\mathbf{S}_{\mathbf{n}})$ filtered in the image, i.e. better filtering effect. In this way, only the procedure of having the already calculated elements of the spectrum squared and averaged is needed, eliminating the need for time-consuming autocorrelation operation.

In fact, this version of the program has an erroneous principle. Since the filter is generated on the basis of the unknown contamination of the image, the filter generation statement cannot distinguish $E(\mathbf{S}_{\mathbf{n}})$ versus $E(\mathbf{S}_{\mathbf{A}})$, but only blindly pursue the minimum of $E(\mathbf{S}_{\mathbf{A+n}})$, leading ingredients of $E(\mathbf{S}_{\mathbf{A}})$ will also suffer a lot of losses. There is no way to solve this problem.

3.3 The third edition of the program

The first two versions of the program is mainly aiming at finding a filter for the image through the analysis of the image, in order to achieve optimal filtering. Although theoretically feasible, the results of the operation of the filtering effect is very poor, and takes much time. In this context, the third version of the program's aim has undergone great changes. The third edition of the program tries to find a filter that is relatively effective for all images through sample training. In order to make the filter have generality, as well as to compress required time for the program, a completely concentric model is adopted instead of the ellipse-to-round transition filter model. Since it is a sample training, the evaluation indicator can be performed under conditions where free noise images are already available. Therefore, the mean square error of the filtered image and the original image is used as the evaluation indicator:

$$MSE = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\left(A_{ij} - A_{ij}^{'}\right)^2}{mn};$$

IV. Results of the operation

4.1 The first edition of the program

The following figure is the result of the first version of the program with the number of individual population set of 10, generations of 40, mutation rate of 0.1, noise of 25dB. The program requires 71.2 minutes.

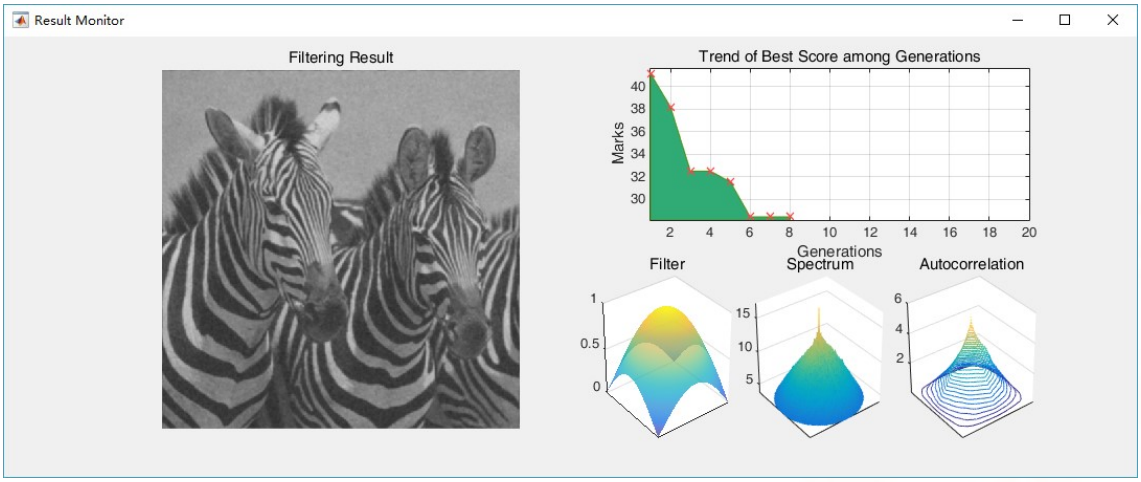

Figure 4.1 result of the first edition of the program

4.2 The second edition of the program

The following figure is the result of the second version of the program with the number of individual population set of 10, generations of 40, mutation rate of 0.1, noise of 25dB. The program requires 48.4 minutes.

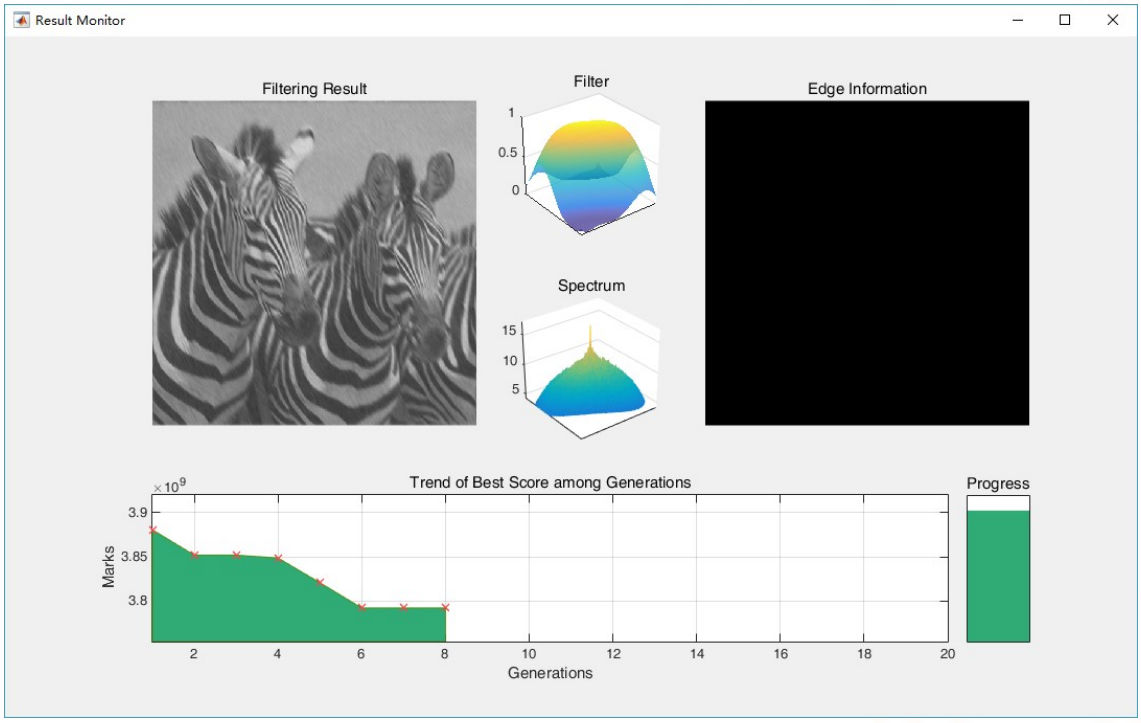

Figure 4.2 result of the second edition of the program

4.3 The third edition of the program

The following figure is the result of the third version of the program with the number of individual population set of 10, generations of 40, mutation rate of 0.1, noise of 25dB. The program requires 6 minutes.

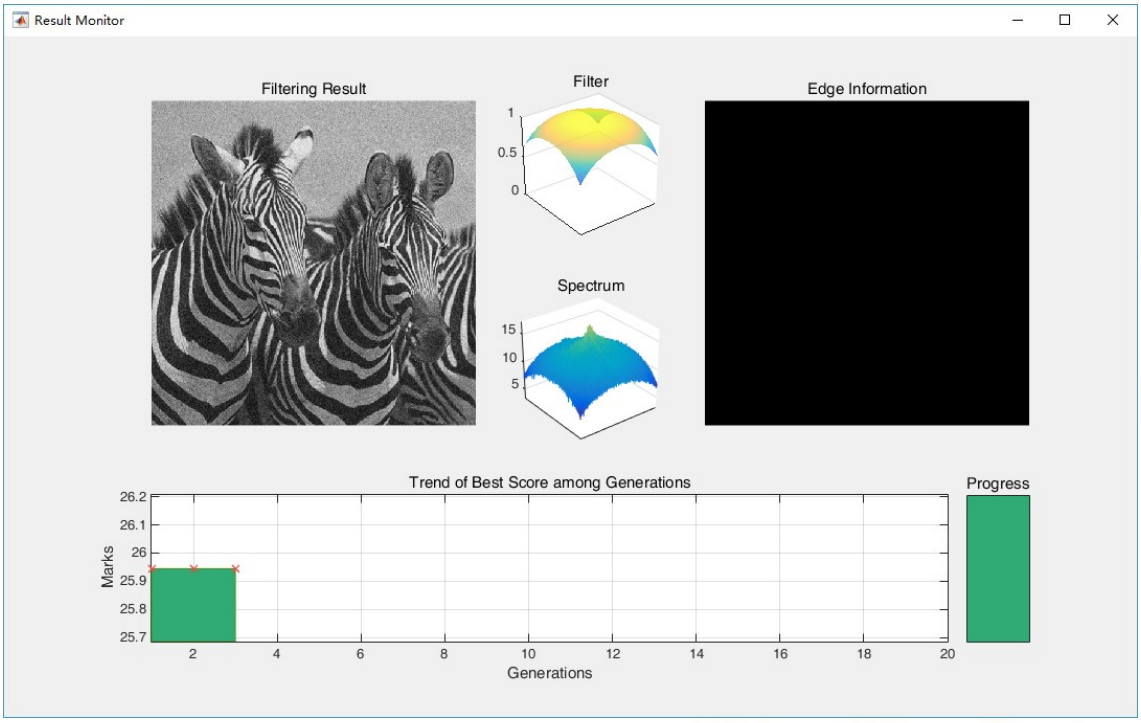

Figure 4.3 result of the third edition of the program

V Results and discussion

As can be seen from the results of the operation, all of the three editions of the program cannot get satisfying results through genetic algorithm. One possible reason is that because of the limited time, parameter of individual number is set too small, resulting in the case where genetic algorithm quickly converge to the local solution. On the other hand, the selection of the indicator in genetic algorithm selection may not be scientific. In operation, it is found that genetic algorithms tend to screen out individuals whose edge information is completely erased. This shows scientific suspicion of the measure to use the unstandardized gradient edge information as the replacement of high-frequency component.

Another point that can be further improved is the denoising of the edge information. The current simple threshold method will cause a lot of losses of edge information. My initial idea is to use the Hog algorithm, that is, to find the gradient of each point in the image. We can see that the edge of the noise is often expressed as a scatter without continuity, and the edge direction of the real image is within a certain range consistent with the direction of the adjacent edge. Therefore, we can use the method of "moving hood" in median filter measure to analyze the gradient direction in a small area, so as to realize the filtering of noise.

Although the problem of Gaussian denoising of high frequency image remained unsolved, this paper presents a new idea, which has certain significance.

VI Appendix: Program code

6.1 The first edition of the program

```
%%
%initializing
clc;
disp('GA image filtering version 1.0');
disp(' ');
disp('Initializing...');
clear;
close all;
%%
%reading the target image
disp('Loading raw image...');
source = double(imread('zebra/Zebra.png'));
[XD,YD] = size(source);
```

```
max_r = sqrt((XD/2)^2 +(YD/2)^2);

%max_S = XD*YD/4;

%%

%loading parameters

disp(' ');

disp('Loading parameters...');

generation = input('For how many generations are you expecting?: '); %overall generations of circulation in GA

 while(generation <= 1 || mod(generation, 1) ~= 0)

  disp('The parameter is supposed to be a positive integer greater than 1');

  generation = input('Please re-input the parameter: ');

 end

number = input('How many individuals are there in the flock?: '); %number of individual

 while(number <= 0 || mod(number, 2) ~= 0)

  disp('The parameter is supposed to be a positive even number');

  number = input('Please re-input the parameter: ');

 end

mutate = input('What is the probability of mutation supposed to be?: '); %probability of mutating

 while(mutate <= 0 || mutate >= 1)

  disp('The parameter is supposed to be located between 0 & 1');

  mutate = input('Please re-input the parameter: ');

 end

%%

%adding noise

disp(' ');

disp('Adding noise...');

dB = input('What intensity of noise are you expecting?: ');

noise = wgn(XD, YD, dB);

test = double(source) +noise;

cal_t = fftshift(fft2(test));

angle_t = exp(1i*angle(cal_t));

show_t = log(1 +abs(cal_t));

%%

%extracting edge information

disp('Extracting edge information...');

[TestX,TestY]=gradient(test);

bevel_t = sqrt(TestX.^2+TestY.^2);

grad_t=TestX.*(TestX./bevel_t)+TestY.*(TestY./bevel_t);

grad_t(isnan(grad_t)) = 0;

clear bevel_t TestX TestY;

%%

%GA preparation

disp(' ');

disp('Initialization accomplished, preparing for the GA circulation...');

filter = ones(size(source));

%the contour elipses are specified by the following equation:

%  (1 -c*cos^2(theta))*x^2 +(1 -c*sin^2(theta))*y^2 +c*sin(2*theta)*x*y = constant

%the filter function is as following:

%  h(x) = x^4 +para_1*x^2

%h(x) ensures value at the center of the filter equals to 0, which will contrbute to the normalization procession.

para = [0.7, pi/3, -1, -10, 1, 0.36, 0.7];

%the vector of parameter is sequenced as the following order:
```

```
% [para_e, para_theta, para_velocity, para_2, para_1, para_range, para_threshold]

%where para_range indicates the range of the filter's value, para_threshold restricts the threshold for the matrix grad to be set to zero, as well as para_velocity indicates the speed from a elipse to transform to a circle

limits = [0,  0.9999;
    0,  pi;
    -1,  0;
    -10, -2;
    0,  1;
    0,  1;
    0,  1];

limits_range = limits(:, 2) -limits(:, 1);

%the limits of values of the paras are as follows:

%      para_e  para_theta  para_velocity      para_1  para_range  para_threshold  para_coef

%  [0, 1) [0, pi)    [-1, 0]      [-10, -2]  [0,1]     [0, 1]  [0, 1]

index = ones(1, 2*number);

para = repmat(para', 1, 2*number);

best_mark = nan(1, generation);

mark_range = [0, 1];

filter = repmat(filter, 1, 1, 2*number);

result = repmat(test, 1, 1, 2*number);

show_f = repmat(show_t, 1, 1, 2*number);

R = xcorr2(test);

R = R./std(R(:));

R_range = [min(min(R(:))), max(max(R(:)))];

[DiffX, DiffY] = gradient(R);

bevel_r = sqrt(DiffX.^2 +DiffY.^2);

grad_r = DiffX.*(DiffX./bevel_r) +DiffY.*(DiffY./bevel_r);

grad_r(isnan(grad_r)) = 0;

primitive_mark = max(grad_r(:))/std(grad_r(:));

R = repmat(R, 1, 1, 2*number);

mark = primitive_mark*ones(1, 2*number);

clear primitive_mark R_estimated

disp('Preparation accomplished, Launching the GA circulation...');

disp(' ');

figure('Menubar','none','Name','Result Monitor','NumberTitle','off');

set(gcf, 'position', [239, 344, 1083, 420]);

subplot(1, 2, 1);

 handle_image = imshow(result(:, :, 1), []);

 title('Filtering Result');

subplot(2, 2, 2);

 handle_window = gca;

 area(best_mark, 'EdgeColor', [77/255, 144/255, 21/255], 'FaceColor', [49/255, 171/255, 118/255], 'YDataSource', 'best_mark');

 hold on;

 plot(best_mark, 'x', 'Color', [249/255, 71/255, 71/255], 'YDataSource', 'best_mark', 'LineWidth', 1);

 hold off;

 grid on;

 axis([1, generation, mark_range(1), mark_range(2)]);

 title('Trend of Best Score among Generations');

 xlabel('Generations');

 ylabel('Marks');

subplot(2, 6, 10);

 colormap(gca, parula);

 surf(filter(:, :, 1), 'ZDataSource', 'filter(:, :, 1)');

 shading interp;
```

```
alpha(0.7);
xlim([1, XD]);
ylim([1, YD]);
zlim([0, 1]);
set(gca, 'xtick', [], 'xticklabel', []);
set(gca, 'ytick', [], 'yticklabel', []);
camproj('perspective');
title('Filter');
subplot(2, 6, 11);
colormap(gca, parula);
mesh(show_f(:, :, 1), 'ZDataSource', 'show_f(:, :, 1)');
xlim([1, XD]);
ylim([1, YD]);
zlim([min(show_t(:)), max(show_t(:))]);
set(gca, 'xtick', [], 'xticklabel', []);
set(gca, 'ytick', [], 'yticklabel', []);
camproj('perspective');
title('Spectrum');
subplot(2, 6, 12);
  colormap(gca, parula);
  contour3(R(:, :, 1), 30,'ZDataSource', 'R(:, :, 1)');
  box off;
  xlim([1, size(R, 1)]);
  ylim([1, size(R, 2)]);
  zlim(R_range);
  set(gca, 'xtick', [], 'xticklabel', []);
  set(gca, 'ytick', [], 'yticklabel', []);
  camproj('perspective');
  title('Autocorrelation');
drawnow;
%%
%GA circulation
disp('Starting the first generation...');
tic;
for t = 1 : generation
 %generating new population
  couple = randperm(number); %free mating
  for i = 1 : 2 : number
    for node = 1 : size(para, 1)
     %swap genes
      if(round(rand))
        para(node, i +number) = para(node, i);
        para(node, i +number +1) = para(node, i +1);
      else
        para(node, i +number) = para(node, i +1);
        para(node, i +number +1) = para(node, i);
      end
     %mutation
      if(rand < mutate)
        para(node, i +number) = limits(node, 1)
+limits_range(node)*rand;
      end
      if(rand < mutate)
        para(node, i +number +1) = limits(node, 1)
+limits_range(node)*rand;
      end
    end
    clear node
  end
  clear i;
 %characterizing individuals
```

```
for i = number +1 : 2*number
  %filter generating
   for j = 1 : XD;
    for k = 1 : YD;
      ellipse = (1 -para(1, i)*cos(para(2, i))^2)*(j -XD/2)^2 +(1 -para(1, i)*sin(para(2, i))^2)*(k -YD/2)^2 +para(1, i)*sin(2*para(2, i))*(j -XD/2)*(k -YD/2);
      ellipse = pi*ellipse/sqrt(1 -(para(1, i)^2));
      circle = pi*((j -XD/2)^2 +(k -YD/2)^2);
      r = sqrt(circle/pi);
      ratio = r/max_r;
      portion = para(3, i)*ratio^2 -(para(3, i) +1)*ratio +1;
      element = (portion*ellipse +(1 -portion)*circle)/(max_r)^2/pi;
      filter(j, k, i) = element^2 +para(4, i)*element;
    end
    clear k;
   end
   clear j;
   filter(:, :, i) = filter(:, :, i)/max(max(abs(filter(:, :, i))))*para(5, i) +1;
  %profile modifying
   grad_temp = grad_t;
   grad_temp(grad_temp < (max(grad_temp(:)) +std(grad_temp(:)))*para(6, i)) = 0;
   cal_h = fftshift(fft2(grad_temp));
   show_h = para(7, i)*log(1 +abs(cal_h));
  %filtering
   show_f(:, :, i) = show_t.*filter(:, :, i) +show_h.*(1 -filter(:, :, i));
   cal_f = (exp(show_f(:, :, i)) -1).*angle_t;
   result(:, :, i) = abs(ifft2(fftshift(cal_f)));
  %scoring
   R(:, :, i) = xcorr2(result(:, :, i));
   R_temp = R(:, :, i);
   R(:, :, i) = R_temp./std(R_temp(:));
   [DiffX, DiffY] = gradient(R(:, :, i));
   bevel_r = sqrt(DiffX.^2 +DiffY.^2);
   grad_r = DiffX.*(DiffX./bevel_r) +DiffY.*(DiffY./bevel_r);
   grad_r(isnan(grad_r)) = 0;
   mark(i) = max(grad_r(:))/std(grad_r(:));
 end
 clear i;
 %natual selecting
 [mark, index] = sort(mark);
 para = para(:, index);
 filter = filter(:, :, index);
 R = R(:, :, index);
 show_f = show_f(:, :, index);
 result = result(:, :, index);
 best_mark(t) = mark(1);
 %demonstrating champion
 mark_range = minmax(best_mark);
 mark_range(1) = mark_range(1)*0.99;
 mark_range(2) = mark_range(2)/0.99;
 set(handle_image, 'CData', result(:, :, 1));
 set(handle_window, 'YLim', mark_range);
 refreshdata;
 drawnow;
 %informing
 disp(['Generation #',num2str(t),' has finished']);
 toc;
```

```
  beep;
  disp(' ');
 %terminate circle
  if(t > 2)
   if(best_mark(t) == best_mark(t -1) && best_mark(t -1) == best_mark(t -2))
     break;
   end
  end
end
%%
%display result
figure('Menubar','none','Name','Result comparasion','NumberTitle','off');
set(gcf, 'position', [239, 162, 1083, 601]);
subplot(1, 2, 1);
  imshow(test, [min(test(:)), max(test(:))]);
  title('Input Image');
subplot(1, 2, 2);
  imshow(result(:, :, 1), [min(test(:)), max(test(:))]);
  title('Filtering Result');
%clear ellipse a ratio portion element r circle square;
```

6.2 The second edition of the program

```
%%
%initializing
clc;
disp('GA image filtering version 2.0');
disp(' ');
disp('Initializing...');
clear;
close all;
%%
%reading the target image
disp('Loading raw image...');
source = double(imread('zebra/Zebra.png'));
[XD,YD] = size(source);
max_r = sqrt((XD/2)^2 +(YD/2)^2);
%%
%adding noise
disp(' ');
disp('Adding noise...');
dB = input('What intensity of noise are you expecting?: ');
  while(isempty(dB))
    dB = input('Waiting for input...');
  end
noise = wgn(XD, YD, dB);
test = double(source) +noise;
cal_t = fftshift(fft2(test));
angle_t = exp(1i*angle(cal_t));
show_t = log(1 +abs(cal_t));
%%
%loading parameters
disp(' ');
disp('Loading parameters...');
generation = input('For how many generations are you expecting?: '); %overall generations of circulation in GA
  while(generation <= 1 || mod(generation, 1) ~= 0)
    disp('The parameter is supposed to be a positive integer greater than 1');
    generation = input('Please re-input the parameter: ');
```

```
 end

number = input('How many individuals are there in the flock?: ');  %number of individual

 while(number <= 0 || mod(number, 2) ~= 0)

  disp('The parameter is supposed to be a positive even number');

  number = input('Please re-input the parameter: ');

 end

mutate = input('What is the probability of mutation supposed to be?: ');  %probability of mutating

 while(mutate <= 0 || mutate >= 1)

  disp('The parameter is supposed to be located between 0 & 1');

  mutate = input('Please re-input the parameter: ');

 end

%%

%extracting edge information

disp('Extracting edge information...');

[TestX,TestY]=gradient(test);

bevel_t = sqrt(TestX.^2+TestY.^2);

grad_t=TestX.*(TestX./bevel_t)+TestY.*(TestY./bevel_t);

grad_t(isnan(grad_t)) = 0;

clear bevel_t TestX TestY;

%%

%GA preparation

disp(' ');

disp('Initialization accomplished, preparing for the GA circulation...');

filter = ones(size(source));

%the contour elipses are specified by the following equation:

%  (1 -c*cos^2(theta))*x^2 +(1 -c*sin^2(theta))*y^2 +c*sin(2*theta)*x*y = constant

%the filter function is as following:

%  h(x) = x^4 +para_1*x^2

%h(x) ensures value at the center of the filter equals to 0, which will contrbute to the normalization procession.

para = [0.7, pi/3, -1, -10, 1, 0.36, 0.7];

%the vector of parameter is sequenced as the following order:

%  [para_e, para_theta, para_velocity, para_2, para_1, para_range, para_threshold]

%where para_range indicates the range of the filter's value, para_threshold restricts the threshold for the matrix grad to be set to zero, as well as para_velocity indicates the speed from a elipse to transform to a circle

limits = [0,  0.9999;

    0,  pi;

    -1,  0;

    -10, -2;

    0,  1;

    0,  1;

    0,  1];

limits_range = limits(:, 2) -limits(:, 1);

%the limits of values of the paras are as follows:

%      para_e  para_theta  para_velocity      para_1  para_range  para_threshold  para_coef

%  [0, 1) [0, pi)    [-1, 0]     [-10, -2]  [0,1]     [0, 1]  [0, 1]

%para_coef has currently been removed

index = ones(1, 2*number);

para = repmat(para', 1, 2*number);

best_mark = nan(1, generation);

mark_range = [0, 1];


filter = repmat(filter, 1, 1, 2*number);
```

```
result = repmat(test, 1, 1, 2*number);

show_f = repmat(show_t, 1, 1, 2*number);

grad_show = repmat(grad_t, 1, 1, 2*number);

power = (abs(fftshift(fft2(test)))).^2;

primitive_mark = mean(power(:));

mark = primitive_mark*ones(1, 2*number);

percentage = [0, 0];

clear primitive_mark R_estimated

disp('Preparation accomplished, Launching the GA circulation...');

disp(' ');

figure('Menubar','none','Name','Result Monitor','NumberTitle','off');

set(gcf, 'position', [237, 195, 1083, 650]);

subplot(3, 5, [1, 2, 6, 7]);

 handle_image = imshow(result(:, :, 1), []);

 title('Filtering Result');

subplot(3, 11, 23:32);

 handle_window = gca;

 area(best_mark, 'EdgeColor', [77/255, 144/255, 21/255], 'FaceColor', [49/255, 171/255, 118/255], 'YDataSource', 'best_mark');

 hold on;

 plot(best_mark, 'x', 'Color', [249/255, 71/255, 71/255], 'YDataSource', 'best_mark', 'LineWidth', 1);

 hold off;

 grid on;

 axis([1, generation, mark_range(1), mark_range(2)]);

 title('Trend of Best Score among Generations');

 xlabel('Generations');

 ylabel('Marks');

subplot(3, 11, 33);

 area(percentage, 'EdgeColor', 'none', 'FaceColor', [49/255, 171/255, 118/255], 'YDataSource', 'percentage');

 set(gca, 'xtick', [], 'xticklabel', []);

 set(gca, 'ytick', [], 'yticklabel', []);

 ylim([0, 1]);

 title('Progress');

subplot(3, 5, 3);

 colormap(gca, parula);

 surf(filter(:, :, 1), 'ZDataSource', 'filter(:, :, 1)');

 shading interp;

 alpha(0.7);

 xlim([1, XD]);

 ylim([1, YD]);

 zlim([0, 1]);

 set(gca, 'xtick', [], 'xticklabel', []);

 set(gca, 'ytick', [], 'yticklabel', []);

 camproj('perspective');

 title('Filter');

subplot(3, 5, 8);

 colormap(gca, parula);

 mesh(show_f(:, :, 1), 'ZDataSource', 'show_f(:, :, 1)');

 xlim([1, XD]);

 ylim([1, YD]);

 zlim([min(show_t(:)), max(show_t(:))]);

 set(gca, 'xtick', [], 'xticklabel', []);

 set(gca, 'ytick', [], 'yticklabel', []);

 camproj('perspective');

 title('Spectrum');

subplot(3, 5, [4, 5, 9, 10]);

 handle_grad = imshow(grad_show(:, :, 1), []);
```

```
 title('Edge Information');
drawnow;
%%
%GA circulation
disp('Starting the first generation...');
tic;
for t = 1 : generation
 percentage = [0, 0];refreshdata;drawnow;
%generating new population
 couple = randperm(number); %free mating
 for i = 1 : 2 : number
  for node = 1 : size(para, 1)
   %swap genes
    if(round(rand))
     para(node, i +number) = para(node, i);
     para(node, i +number +1) = para(node, i +1);
    else
     para(node, i +number) = para(node, i +1);
     para(node, i +number +1) = para(node, i);
    end
   %mutation
    if(rand < mutate)
     para(node, i +number) = limits(node, 1) +limits_range(node)*rand;
    end
    if(rand < mutate)
     para(node, i +number +1) = limits(node, 1) +limits_range(node)*rand;
    end
   end
  clear node
 end
 clear i;
%characterizing individuals
 for i = number +1 : 2*number
  percentage = [1, 1]*(i -number -1)/number;refreshdata;drawnow;
  %filter generating
   for j = 1 : XD;
    for k = 1 : YD;
     ellipse = (1 -para(1, i)*cos(para(2, i))^2)*(j -XD/2)^2 +(1 -para(1, i)*sin(para(2, i))^2)*(k -YD/2)^2 +para(1, i)*sin(2*para(2, i))*(j -XD/2)*(k -YD/2);
     ellipse = pi*ellipse/sqrt(1 -(para(1, i)^2));
     circle = pi*((j -XD/2)^2 +(k -YD/2)^2);
     %square = ((max(abs(j -XD/2), abs(k -YD/2)))^2);
     r = sqrt(circle/pi);
     ratio = r/max_r;
     %ratio = sqrt(square/max_S);
     portion = para(3, i)*ratio^2 -(para(3, i) +1)*ratio +1;
     element = (portion*ellipse +(1 -portion)*circle)/(max_r)^2/pi;
     %element = (portion*elipse/max_elipse +(1 -portion)*square/max_S);
     filter(j, k, i) = element^2 +para(4, i)*element;
    end
    clear k;
   end
   clear j;
   filter(:, :, i) = filter(:, :, i)/max(max(abs(filter(:, :, i))))*para(5, i) +1;
  %profile modifying
```

```
    grad_temp = grad_t;
    grad_temp(grad_temp < (max(grad_temp(:)) +std(grad_temp(:)))*para(6, i)) = 0;
    grad_show(:, :, i) = grad_temp;
    cal_h = fftshift(fft2(grad_temp));
    show_h = log(1 +abs(cal_h));
  %filtering
    show_f(:, :, i) = show_t.*filter(:, :, i) +show_h.*(1 - filter(:, :, i));
    cal_f = (exp(show_f(:, :, i)) -1).*angle_t;
    result(:, :, i) = abs(ifft2(fftshift(cal_f)));
  %scoring
    power = (abs(cal_f)).^2;
    mark(i) = mean(power(:));
  end
  clear i;
 %natual selecting
  [mark, index] = sort(mark);
  para = para(:, index);
  filter = filter(:, :, index);
  grad_show = grad_show(:, :, index);
  show_f = show_f(:, :, index);
  result = result(:, :, index);
  best_mark(t) = mark(1);
 %demonstrating champion
  mark_range = minmax(best_mark);
  mark_range(1) = mark_range(1)*0.99;
  mark_range(2) = mark_range(2)/0.99;
  set(handle_image, 'CData', result(:, :, 1));
  set(handle_grad, 'CData', grad_show(:, :, 1));
  set(handle_window, 'YLim', mark_range);
  refreshdata;
  drawnow;
 %informing
  disp(['Generation #',num2str(t),' has finished']);
  toc;
  beep;
  disp(' ');
 %terminate circle
  if(t > 2)
    if(best_mark(t) == best_mark(t -1) && best_mark(t -1) == best_mark(t -2))
      break;
    end
  end
end
%%
%display result
disp('Circulation terminated');
figure('Menubar','none','Name','Result comparasion','NumberTitle','off');
set(gcf, 'position', [237, 195, 1083, 650]);
subplot(1, 2, 1);
  imshow(test, [min(test(:)), max(test(:))]);
  title('Input Image');
subplot(1, 2, 2);
  imshow(result(:, :, 1), [min(test(:)), max(test(:))]);
  title('Filtering Result');
%clear ellipse a ratio portion element r circle square;
```

6.3 The third edition of the program

```
%%
%initializing
```

```
clc;
disp('GA image filtering version 3.0');
disp(' ');
disp('Initializing...');
clear;
close all;
%%
%reading the target image
disp('Loading raw image...');
source = double(imread('zebra/Zebra.png'));
[XD,YD] = size(source);
disp('Loading training sample...');
test(:, :, 1) = double(imread('zebra/ZebraG20_22.1947.png'));
test(:, :, 2) = double(imread('zebra/ZebraG30_18.8633.png'));
test(:, :, 3) = double(imread('zebra/ZebraG40_16.5752.png'));
test(:, :, 4) = double(imread('zebra/ZebraG50_14.8842.png'));
test(:, :, 5) = double(imread('zebra/ZebraG60_13.5614.png'));
sample = size(test, 3)
cal_t = test;
angle_t = test;
show_t = test;
for i = 1 : sample;
  cal_t(:, :, i) = fftshift(fft2(test(:, :, i)));
  angle_t(:, :, i) = exp(1i*angle(cal_t(:, :, i)));
  show_t(:, :, i) = log(1 +abs(cal_t(:, :, i)));
end
clear i;
%%
%loading parameters
disp(' ');
disp('Loading parameters...');
generation = input('For how many generations are you expecting?: '); %overall generations of circulation in GA
 while(generation <= 1 || mod(generation, 1) ~= 0)
   disp('The parameter is supposed to be a positive integer greater than 1');
   generation = input('Please re-input the parameter: ');
 end
number = input('How many individuals are there in the flock?: '); %number of individual
 while(number <= 0 || mod(number, 2) ~= 0)
   disp('The parameter is supposed to be a positive even number');
   number = input('Please re-input the parameter: ');
 end
mutate = input('What is the probability of mutation supposed to be?: '); %probability of mutating
 while(mutate <= 0 || mutate >= 1)
   disp('The parameter is supposed to be located between 0 & 1');
   mutate = input('Please re-input the parameter: ');
 end
%%
%extracting edge information
disp(' ');
disp('Extracting edge information...');
grad_t = test;
for i = 1 : sample;
  [TestX,TestY]=gradient(test(:, :, i));
```

```
 bevel_t = sqrt(TestX.^2+TestY.^2);

 grad_t(:, :, i)=TestX.*(TestX./bevel_t)+TestY.*(TestY./bevel_t);

end

grad_t(isnan(grad_t)) = 0;

clear bevel_t TestX TestY i;

%%

%GA preparation

disp(' ');

disp('Initialization accomplished, preparing for the GA circulation...');

filter = ones(size(source));

%the filter function is as following:

%  h(x) = x^4 +para_1*x^2

%h(x) ensures value at the center of the filter equals to 0, which will contrbute to the normalization procession.

para = [-10, 0.4, 0.36, 0.42];

%the vector of parameter is sequenced as the following order:

%          [para_velocity, para_1, para_range, para_threshold, para_coef]

%where para_range indicates the range of the filter's value, para_threshold restricts the threshold for the matrix grad to be set to zero, as well as para_velocity indicates the speed from a elipse to transform to a circle

limits = [-10, 0;
    0,  1;
    0,  1;
    0,  1];

limits_range = limits(:, 2) -limits(:, 1);

%the limits of values of the paras are as follows:

%  para_1    para_range  para_threshold  para_coef

%  [-10, -2]  [0,1]    [0, 1]      [0, 1]

index = ones(1, 2*number);

para = repmat(para', 1, 2*number);

best_mark = nan(1, generation);

mark_temp = zeros(1, sample);

mark_range = [0, 1];

filter = repmat(filter, 1, 1, 2*number);

result = repmat(test, 1, 1, 1, 2*number);

show_f = repmat(show_t, 1, 1, 1, 2*number);

grad_show = repmat(grad_t, 1, 1, 1, 2*number);

show_h = zeros(XD, YD, sample);

for j = 1 : sample

diff_temp = result(:, :, j, 1) - source;

mark_temp(j) = std(diff_temp(:));

end

clear j;

mark = mean(mark_temp)*ones(1, 2*number);

percentage = [0, 0];

disp('Preparation accomplished, Launching the GA circulation...');

disp(' ');

show_result = zeros(XD, YD);

show_grad = zeros(XD, YD);

f_show = zeros(XD, YD);

for j = 1 : sample

  show_result = show_result + result(:, :, j, 1);

  show_grad = show_grad + grad_show(:, :, j, 1);

  f_show = f_show + show_f(:, :, j, 1);

end

clear j;

show_result = show_result./sample;

show_grad = show_grad./sample;
```

```
f_show = f_show./sample;

figure('Menubar','none','Name','Result Monitor','NumberTitle','off');

set(gcf, 'position', [237, 195, 1083, 650]);

subplot(3, 5, [1, 2, 6, 7]);

 handle_image = imshow(show_result, []);

 title('Filtering Result');

subplot(3, 11, 23:32);

 handle_window = gca;

 area(best_mark, 'EdgeColor', [77/255, 144/255, 21/255], 'FaceColor', [49/255, 171/255, 118/255], 'YDataSource', 'best_mark');

 hold on;

 plot(best_mark, 'x', 'Color', [249/255, 71/255, 71/255], 'YDataSource', 'best_mark', 'LineWidth', 1);

 hold off;

 grid on;

 axis([1, generation, mark_range(1), mark_range(2)]);

 title('Trend of Best Score among Generations');

 xlabel('Generations');

 ylabel('Marks');

subplot(3, 11, 33);

 area(percentage, 'EdgeColor', 'none', 'FaceColor', [49/255, 171/255, 118/255], 'YDataSource', 'percentage');

 set(gca, 'xtick', [], 'xticklabel', []);

 set(gca, 'ytick', [], 'yticklabel', []);

 ylim([0, 1]);

 title('Progress');

subplot(3, 5, 3);

 colormap(gca, parula);

 %contour3(filter(:, :, 1), 30, 'ZDataSource', 'filter(:, :, 1)');

 %box off;

 surf(filter(:, :, 1), 'ZDataSource', 'filter(:, :, 1)');

 shading interp;

 alpha(0.7);

 xlim([1, XD]);

 ylim([1, YD]);

 zlim([0, 1]);

 set(gca, 'xtick', [], 'xticklabel', []);

 set(gca, 'ytick', [], 'yticklabel', []);

 camproj('perspective');

 title('Filter');

subplot(3, 5, 8);

 colormap(gca, parula);

 mesh(f_show, 'ZDataSource', 'f_show');

 xlim([1, XD]);

 ylim([1, YD]);

 zlim([min(show_t(:)), max(show_t(:))]);

 set(gca, 'xtick', [], 'xticklabel', []);

 set(gca, 'ytick', [], 'yticklabel', []);

 camproj('perspective');

 title('Spectrum');

subplot(3, 5, [4, 5, 9, 10]);

 handle_grad = imshow(show_grad, []);

 title('Edge Information');

drawnow;

%%

%GA circulation

disp('Starting the first generation...');

tic;

for t = 1 : generation
```

```
percentage = [0, 0];refreshdata;drawnow;
%generating new population
couple = randperm(number); %free mating
for i = 1 : 2 : number
 for node = 1 : size(para, 1)
  %swap genes
  if(round(rand))
   para(node, i +number) = para(node, i);
   para(node, i +number +1) = para(node, i +1);
  else
   para(node, i +number) = para(node, i +1);
   para(node, i +number +1) = para(node, i);
  end
  %mutation
  if(rand < mutate)
   para(node, i +number) = limits(node, 1) +limits_range(node)*rand;
  end
  if(rand < mutate)
   para(node, i +number +1) = limits(node, 1) +limits_range(node)*rand;
  end
 end
end
%characterizing individuals
for i = number +1 : 2*number
 percentage = [1, 1]*(i - number)/number;refreshdata;drawnow;
 %filter generating
 for j = 1 : XD;
  for k = 1 : YD;
   circle = pi*((j -XD/2)^2 +(k -YD/2)^2);
   filter(j, k, i) = -circle^2 +para(1, i)*circle;
  end
 end
 filter(:, :, i) = filter(:, :, i)/max(max(abs(filter(:, :, i))))*para(2, i) +1;
 %profile modifying
 for j = 1 : sample
  grad_temp = grad_t(:, :, j);
  grad_temp(grad_temp < (max(grad_temp(:)) +std(grad_temp(:)))*para(3, i)) = 0;
  grad_show(:, :, j, i) = grad_temp;
  cal_h = fftshift(fft2(grad_temp));
  show_h(:, :, j) = log(1 +abs(cal_h));
 end
 %filtering
 for j = 1 : sample
  show_f(:, :, j, i) = show_t(:, :, j).*filter(:, :, i) +show_h(:, :, j).*(1 -filter(:, :, i)).*para(4, i);
  cal_f = (exp(show_f(:, :, j, i)) -1).*angle_t(:, :, j);
  result(:, :, j, i) = abs(ifft2(fftshift(cal_f)));
 end
 %scoring
 for j = 1 : sample
  diff_temp = result(:, :, j, i) - source;
  mark_temp(j) = std(diff_temp(:));
 end
 mark(i) = mean(mark_temp);
end
%natual selecting
[mark, index] = sort(mark);
para = para(:, index);
filter = filter(:, :, index);
```

```
grad_show = grad_show(:, :, :, index);
show_f = show_f(:, :, :, index);
result = result(:, :, :, index);
best_mark(t) = mark(1);
%demonstrating champion
mark_range = minmax(best_mark);
mark_range(1) = mark_range(1)*0.99;
mark_range(2) = mark_range(2)/0.99;
show_result = zeros(XD, YD);
show_grad = zeros(XD, YD);
for j = 1 : sample
  show_result = show_result + result(:, :, j, 1);
  show_grad = show_grad + grad_show(:, :, j, 1);
  f_show = f_show + show_f(:, :, j, 1);
end
show_result = show_result./sample;
show_grad = show_grad./sample;
f_show = f_show./sample;
set(handle_image, 'CData', show_result);
set(handle_grad, 'CData', show_grad);
set(handle_window, 'YLim', mark_range);
refreshdata;
drawnow;
%informing
disp(['Generation #',num2str(t),' has finished']);
toc;
beep;
disp(' ');
%terminate circle
if(t > 2)
  if(best_mark(t) == best_mark(t -1) && best_mark(t -1) == best_mark(t -2))
    break;
  end
 end
end
%%
%display result
disp('Circulation terminated');
figure('Menubar','none','Name','Result comparasion','NumberTitle','off');
set(gcf, 'position', [237, 195, 1083, 650]);
subplot(1, 2, 1);
 imshow(source, []);
 title('Target Image');
subplot(1, 2, 2);
 imshow(show_result, []);
 title('Filtering Result');
%clear ellipse a ratio portion element r circle square;
```