杭州电子科技大学计算机学院

课程名称：《人工智能导论》

选课题目：《基于遗传算法的图像阈值分割》

任课教师：孔万增

学生姓名：黄继升

学生班级：16052317

学生学号：16041321

所在专业：计算机科学与技术

目录

一.选题目的

在现今的人工智能领域，计算机视觉和模式识别技术已经成为人工智能课题不可或缺的部分，为丰富以及推动人工智能的发展做出了巨大的贡献。然而，在我们在进行这两大人工智能相关技术的学习之前，我们必须掌握传统的对图像方面处理的知识，以及各类对图像处理的算法，从而为深入学习计算机视觉和对图像进行模式识别打下坚实的基础。目前我们常用的进行计算机视觉和图像处理的工具有 OpenCV 以及 matlab 等等。而目前比较流行，且多于开发的一般是 OpenCV，它最大的特点是执行速度很快，非常适合大数量、大规模数据的实验仿真。相比之下，matlab 只能对数量有限的样本进行仿真，且速度上不及 OpencV，但它也有 OpenCV 所不具备的优点，比如对于初学者而言，matlab 的代码学习起来比较简单，而 OpenCV 则需要较为深厚的 C/C++功底；并且 matlab 更适合进行算法验证和小规模数据的仿真。

本次实验是将指定的样本图像进行预处理后，再结合本学期课程所学，运用人工智能算法——“遗传算法”对样本图像进行图像阈值分割，最后进行图像结果的输出并且对结果进行分析，比较新型的遗传算法与传统的图像阈值分割算法优劣程度，从而对图像处理技术有更加深入的理解。再加上本次实验处理的样本数量极少，因此选用 matlab 软件作为本次实验的操作软件，无疑是一个最好的选择。

二. 实验原理

本次实验是用 matlab 对样本图像进行代码编写，并最后输出结果图像。本次所得到的样本图像是随机从互联网上下载得到的，确定适应度函数的 OTSU 算法对噪声比较敏感，因此为了得到更加准确的 T 值，减少图像噪声对 T 值的确定以及输出图像效果的影响，我们需要先对原始图像进行噪声滤除的预处理。具体实验原理如下：

（1）对原始图像进行噪声滤除，采用的滤波器是巴特沃斯低通滤波器，公式如下：

$$H(u,v)=\frac{1}{1+[\frac{D(u,v)}{D0}]^{2n}}$$

其中 D0 是截止频率，n 是巴特沃斯滤波器阶数。

（2）用 OTSU（最大类间方差法）算法得到遗传算法的适应度函数，OTSU 公式如下：

设图像的阈值为 t ,w0 为前景图所占比例 ,w1=1-w0 为背景点所占比例，u0 为前景点灰度均值，u1 点为背景点灰度均值，则有：

全局灰度均值：u=w0*u0+w1*u1

otsu 公式：g=w0(u0-u)^2+w1(u1-u)^2

（3）得到适应度函数后，利用遗传算法得到图像的阈值分割点 T

（注：在 matlab 中使用遗传算法相关的函数，必须在 matlab 中安装英国谢菲尔德(Sheffield)大学提供的 gatbx 遗传算法工具箱。）

（4）将图像按照阈值分割点 T 进行分割，最后得到结果图像并进行输出。

三.实验思路

1.将三维彩色图像转化为二维的灰度图像；

2.用巴特沃斯低通滤波器对原始图像进行预处理，其中截止频率 D0=300，阶数 n=3；

3.用最大类间方差法确定适应度函数：

二维图像的灰度值在[0,255]之间，假设分割阈值为 T，将图像灰度值>T 的灰度值赋值为 255（黑），将灰度值<T 的灰度值赋值为 0（白），最终得到的图像将是基于阈值 T 分割的黑白二值图像。

设图像的阈值为 t，w0 为前景图所占比例；

w1=1-w0 为背景点所占比例，s0 为前景点灰度均值，s1 点为背景点灰度均值，则有：

全局灰度均值：s=w0*s0+w1*s1

适应度函数：max(T)= w0(s0-s)^2+w1(s1-s)^2

4.将适应度函数 max(T)作为遗传算法的适应度函数，并得到最后的 T 值。

（1）编码染色体，将所有在[0,255]区间的灰度值编码为 8 位二进制

（2）设定种群规模，产生初始种群。我们将种群规模设定为 3，随机编码染色体组成初始种群；

（3）计算各代种群中的各染色体的适应度，并进行遗传操作，直到适应度最高的染色体出现为止，即得到最佳的 T 值；

（4）用最佳 T 值对二维灰度图进行阈值分割，得到最终的结果图像并进行输出。

四. 代码实现

1. 对原始图像进行滤波预处理：

```
imga=imread('D:\Documents\Desktop\cxsj\hjs\sample1.png'); %读出样本图像

img0=rgb2gray(imga); %将彩色图像转化为二维灰度图像

img1=im2double(img0); %将图像数据类型转化为 double 类型

x=2*size(img1,1); %获取矩阵 img1 的行数并且加倍

y=2*size(img1,2); %获取矩阵 img1 的列数并且加倍

a=-x/2:(x/2-1); %将横坐标左移以原点对称

b=-y/2:(y/2-1); %将纵坐标下移以原点对称

[A,B]=meshgrid(a,b);  %产生以向量 a 为行，向量 b 为列的两个大小相同的矩阵，
并根据这两个矩阵形成二维坐标矩阵
fd=sqrt(A.^2+B.^2);
d0=300; %截止频率为 300

n=3; %阶数为 3

LP=1./(1+(fd./d0).^(2*n)); %根据公式设置一个巴特沃斯低通滤波器

J=fftshift(fft2(img1,size(LP,1),size(LP,2))); %对 img1 进行傅里叶变换到频域后 ,再将
经过 fft 变换后的图像的[fs/2,fs]部分移动到[-fs/2,0]这个范围内形成中心对称。

K=J.*LP; %将频域图像 img1 用巴特沃斯低通滤波器进行滤波

LI=ifft2(ifftshift(K)); %将经过滤波后的的频域图像 K 的[-fs/2,0]部分恢复到正半轴，
再进行傅里叶反变换恢复到实域
filter_photo=LI(1:size(img1,1),1:size(img1,2));
imshow(filter_photo); %将滤波后的图片进行输出
```

2. 用最大类间方差法得到适应度函数

```
function otsu=OTSU(X) %定义适应度函数进行图像调用
col=size(X,1);

I= imread('D:\Documents\Desktop\cxsj\hjs\sample1.png');
K = rgb2gray(I);
x = size(K,1);
y = size(K,2);
h=imhist(K);   %提取灰度直方图信息
for k = 0:255
    P(k+1,1) = h(k+1,1)/(x*y);%计算个体概率
end

for i=1:col
      n =X(i,1);
      s = 0.0;
      w0 = 0.0;
      for j = 0:n
            s =s + j * P(j+1,1);
            w0 = w0+P(j+1,1);
      end
      s0 = s/w0;
      w1 = 1-w0;
      s3 = 0.0;
      for l = 0:255
            s3 = s3 + l*P(l+1,1);
      end
      s1 = (s3-s)/w1;
      obj(i,1) = w0*w1*(s1-s0)^2;
end
```

3. 多种群的遗传算法（根据课外参考书籍，可得知该算法比单种群普通遗传算法效果更优）

（1）定义移民算子：

```
function [Chrom,ObjV]=immigrant(Chrom,ObjV)
MP=length(Chrom);
for i=1:MP;
      [MaxO,maxI]=max(ObjV{i});
      next_i=i+1;
      if next_i>MP
            next_i=i+1;
```

```
        if next_i>MP
            next_i=mod(next_i,MP);
        end
        [MinO,minI]=min(ObjV{next_i});
        Chrom{next_i}(minI,:)=Chrom{i}(maxI,:);
        ObjV{next_i}(minI)=ObjV{i}(maxI);
    end
end
```

（2）定义人工选择算子：

```
function[MaxObjV,MaxChrom]=EliteInduvidual(Chrom,ObjV,MaxObjV,MaxChrom)
MP=length(Chrom);
for i=1:MP
    [MaxO,maxI]=max(ObjV{i});
    if MaxO>MaxObjV(i)
        MaxObjV(i)=MaxO;
        MaxChrom(i,:)=Chrom{i}(maxI,:);
    end
end
```

（3）多种群遗传算法：

```
clear all;
clc;
NIND = 4;        %定义个体数目
NVAR = 1;         %定义变量的维数
PRECI = 8;       %定义变量的二进制位数，由之前推导可知 0-255 对应八位二进制数
GGAP = 0.6;      %设定交叉概率值为 0.6
MP = 5;          %种群数目
FieldD =[PRECI;1;256;1;0;1;1];    %设置译码矩阵
for i = 1:MP
    Chrom{i} = crtbp(NIND, NVAR*PRECI);    %创建初始种群
end
pc=0.6+(0.8-0.6)*rand(MP,1);          %在【0.6,0.8】范围 i 内随机产生交叉概率
pm=0.001+(0.05-0.001)*rand(MP,1); %%在【0.001,0.05】范围内随机产生变异概率
gen = 0;        %初始遗传代数
gen0 = 0;       %初始保持代数
MAXGEN = 10; %最优个体最少保持代数
maxY = 0;       %最优值
for i =1:MP
    ObjV{i}=mysecond(bs2rv(Chrom{i},FieldD));   %计算各初始种群个体的目标函数值
```

```
end
MaxObjV = zeros(MP,1);     %记录精华种群
MaxChrom = zeros(MP,PRECI*NVAR);     %记录精华种群的编码
while gen0 <= MAXGEN
      gen = gen + 1;         %遗传代数加 1
      for i = 1:MP
            FitnV{i}=ranking(-ObjV{i});                                  % 各种群的适应度
            SelCh{i}=select('sus', Chrom{i}, FitnV{i},GGAP); % 选择操作
            SelCh{i}=recombin('xovsp',SelCh{i}, pc(i));          % 交叉操作
            SelCh{i}=mut(SelCh{i},pm(i));                                % 变异操作
            ObjVSel=mysecond(bs2rv(SelCh{i},  FieldD));          % 计算子代目标函数
值
            [Chrom{i},ObjV{i}]=reins(Chrom{i},SelCh{i},1,1,ObjV{i},ObjVSel);       %重插
入操作
      end
      [Chrom,ObjV]=immigrant(Chrom,ObjV);        % 调用移民函数进行移民操作

[MaxObjV,MaxChrom]=EliteInduvidual(Chrom,ObjV,MaxObjV,MaxChrom);              %
人工选择精华种群
      YY(gen)=max(MaxObjV);       %找出精华种群中最优的个体
      if YY(gen)>maxY     %判断当前优化值是否与前一次优化值相同
            maxY=YY(gen); %更新最优值
            gen0=0;
      else
            gen0=gen0+1; %最优值保持次数加 1
      end
end
%% 进化过程图
plot(1:gen,YY)
xlabel('进化代数')
ylabel('最优解变化')
title('进化过程')
xlim([1,gen])
%% 输出最优解
[Y,I]=max(MaxObjV);       %找出精华种群中最优的个体
X=(bs2rv(MaxChrom(I,:), FieldD));     %最优个体的解码解
disp(['最优值为：',num2str(Y)])
disp(['对应的自变量取值：',num2str(X)])
```

（4）阈值分割函数

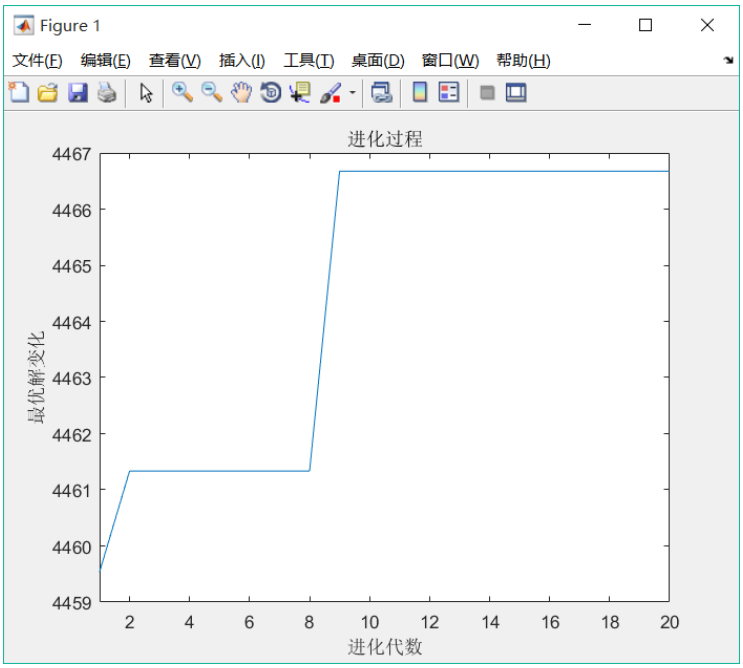
```
I = imread('D:\Documents\Desktop\cxsj\hjs\sample1.png');
% I = imread('D:\Documents\Desktop\cxsj\hjs\sample2.png');
% I = imread('D:\Documents\Desktop\cxsj\hjs\sample3.jpg');
```

```
% I = imread('D:\Documents\Desktop\cxsj\hjs\sample4.jpg');
% I = imread('D:\Documents\Desktop\cxsj\hjs\sample5.jpg');
%I = imread('D:\Documents\Desktop\cxsj\hjs\sample6.jpg');
%I = imread('D:\Documents\Desktop\cxsj\hjs\sample7.jpg');
%I = imread('D:\Documents\Desktop\cxsj\hjs\sample8.jpg');
img = rgb2gray(I);
x = size(K,1);
y = size(img,2);
for i = 1:x
    for j = 1:y
        if img(i,j) < 126 %注意：在这里必须要用遗传算法得到的各张图像分割
的最优值代入进行分割，因为每张图片有不同的最优分割阈值。我在处理后面的
图片的时候就忘了修改代码中的阈值，而导致结果出现偏差，必须重头开始处理。
            img(i,j) = 0;
        else
            img(i,j) = 255;
        end
    end
end
figure
imshow(img);
```

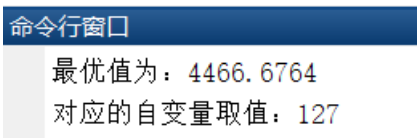
五. 结果输出及调试

（1）先以图像 sample1.png 为例进行结果输出：

最优值变化折线：

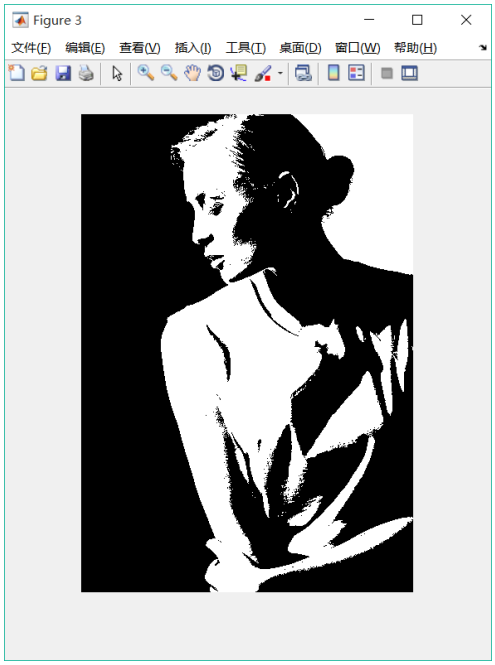


最优值以及对应的自变量取值（分割阈值）：



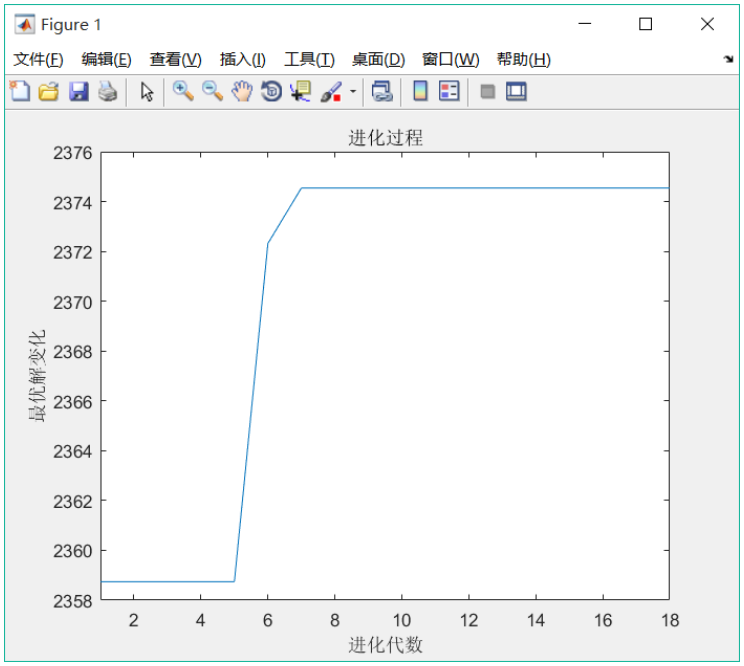
滤波后图像：

遗传算法分割后的图像：

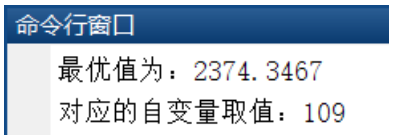


以 Sample2.jpg 为例进行再次检验：

最优值变化曲线



最优值以及对应的自变量取值：

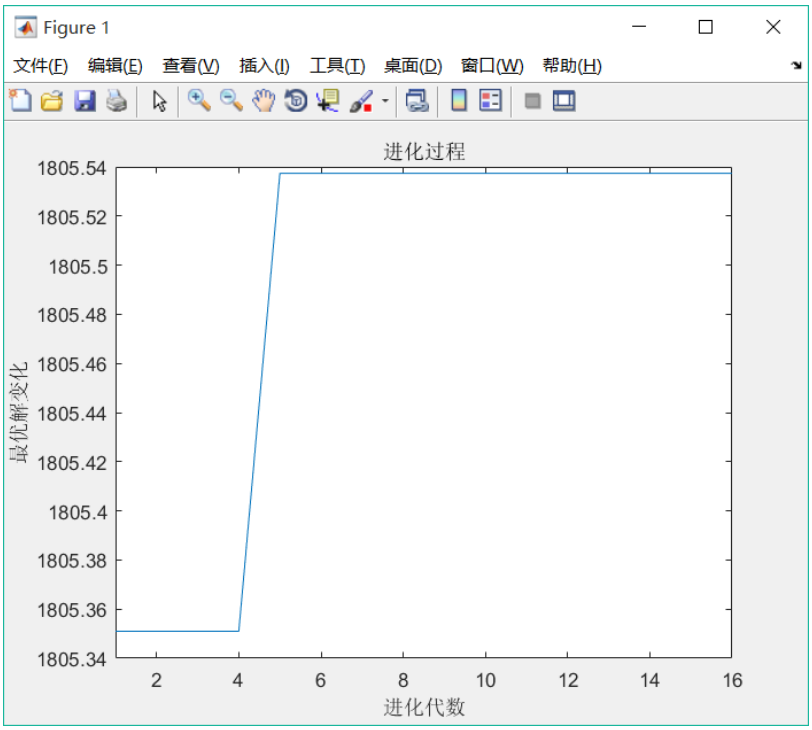


滤波后图像：

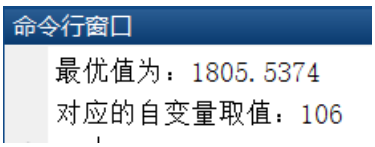
遗传算法分割后的图像：

以 sample3.jpg 为例进行再次检验：

最优值变化曲线

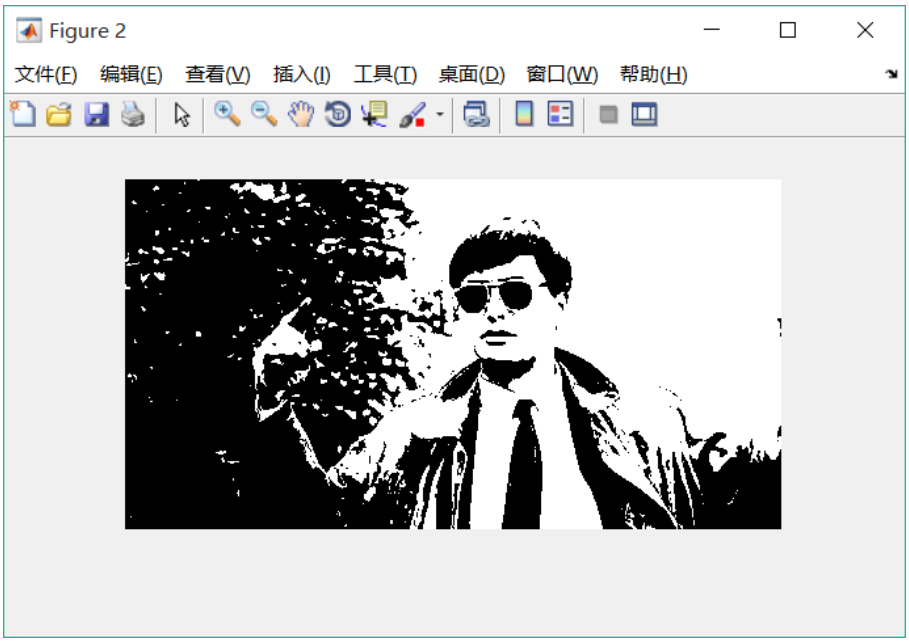


最优值以及对应的自变量取值：



滤波后图像

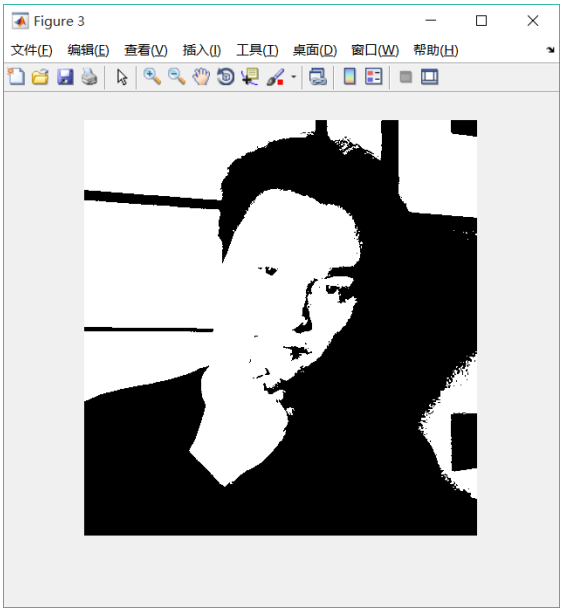
遗传算法分割后的图像



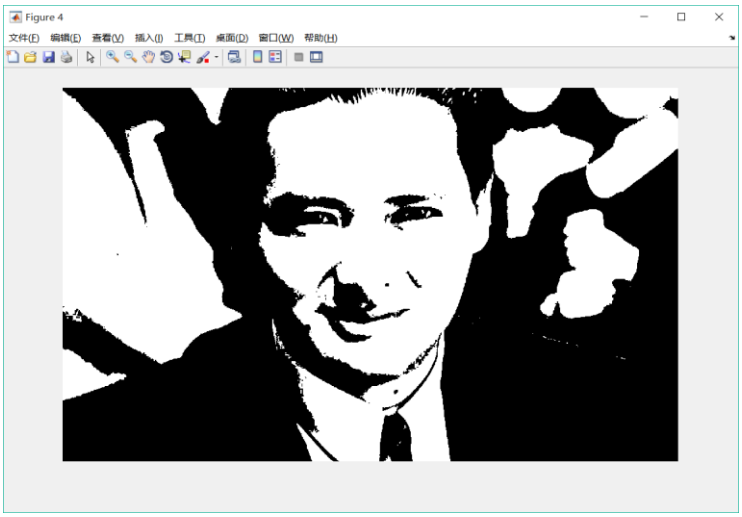
下面将剩下的测试集图像：sample4.jpg、sample5.jpg、sample6.jpg、sample7.jpg、sample8.jpg 直接进行测试输出：

sample4.png（最优值为 90）：

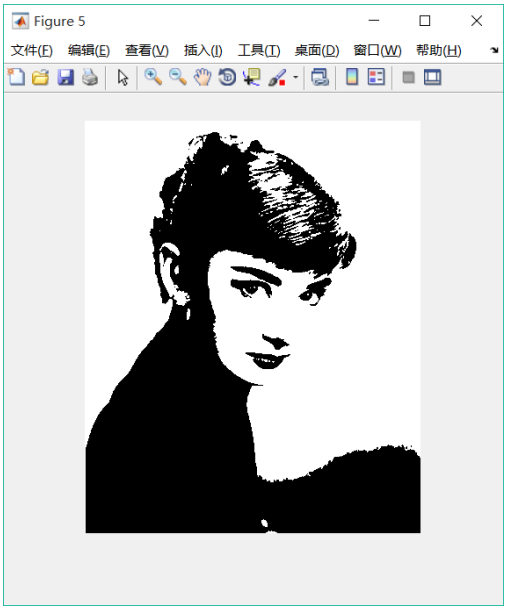
sample5.jpg（最优值为 101）：



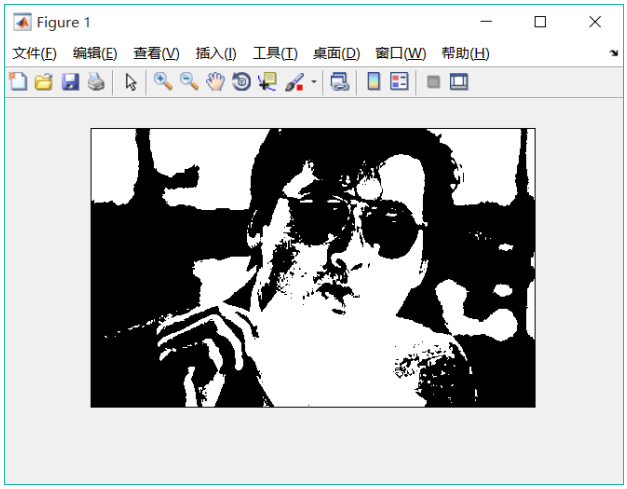
sample6.jpg（最优值为 62）：



sample7. png（最优值为 120）：



sample8.jpg（最优值为 113）：



六. 个人总结

在做这次用遗传算法进行图像阈值分割的大作业之前，我也接触过数字图像处理以及一些计算机视觉方面的知识。我之前用 matlab 处理图像经常用的是一些常见的分割模板和一些算子，如 Roberts 算子、Prewitt 算子、Sobel 算子，也曾直接用 OTSU 或迭代式阈值分割的方法对图像直接进行分割，效果并不能得到有效的保证。

这次大作业采用的算法是人工智能课程所学的算法——遗传算法，并且直接把 OTSU 算法函数作为目标函数去不断进行遗传操作，从而得到阈值分割的最优值，从效果上看，比之前直接用函数命令处理的方式更好，更明显。这充分体现了人工智能算法在处理如图像数据等非结构化数据的优势所在。为了完成这次大作业，我还从图书馆借阅资料进行参考，发现多种群遗传算法在处理效果上比传统的单种群遗传算法效果更好，因此我将多种群遗传算法作为本次处理图像的算法，效果也显而易见。