基于 MVO-SA 优化的二维最大熵图像阈值分割

唐川淇*

江南大学 信息与计算科学 1901 班

【摘　要】 本文基于传统的多元宇宙优化算法和模拟退火算法，结合两种算法的优点提出了多元宇宙-模拟退火 (MVO-SA) 算法，该算法相较原始的多元宇宙优化算法有更好的全局搜索能力以及更快的收敛速度。本文将 MVO-SA 算法应用于二维熵最大阈值图像分割，试验结果表明该算法可以实现快速、准确图像分割。

【关键词】 **模拟退火**，**多元宇宙优化**，MVO-SA，**图像阈值分割**，**智能算法**

1　引言

图像分割就是把图像中具有不同含义的区域区分开来，这些区域是互不相交的，每一个区域都满足这一区域的一致性[1]。图像分割是图像处理中的重要问题，也是计算机视觉研究中的一个经典难题。计算机视觉中的图像理解包括目标检测、特征提取和目标识别等，都依赖于图像分割的质量。尽管研究人员提出了许多分割方法[2-4]，但是到目前为止还不存在一种通用的方法．也不存在一个判断分割是否成功的客观标准。阈值图像分割分为两种：全局阈值分割和局部阈值分割[5]。全局阈值法利用全局信息，比如图像的灰度直方图，对图像做出最优的分割。本文依据全局阈值分割法结合熵，通过提出的 MVO-SA 方法实现了对灰度图像的分割。

2　MVO-SA 算法

本文结合多元宇宙优化算法和模拟退火算法提出了 MVO-SA 算法，最终将 MVO-SA 算法应用于图像分割。下面将详细介绍多元宇宙算法、模拟退火算法和结合之后的 MVO-SA 算法。

2.1　MVO 算法

2.1.1　背景介绍

2015 年，S Mirjalili，SM Mirjalili 和 A Hatamlou[6] 三人创造性提出了一种基于物理学中多元宇宙理论的群智能优化算法——多元宇宙优化算法，该算法的主要灵感来自宇宙学中的三个概念：黑洞、白洞和虫洞。黑洞、白洞和虫洞示意图如图1。

图 1　白洞　黑洞　虫洞

这三个概念的数学模型分别用于执行勘探、开发和局部搜索。并成功将其应用于函数优化和工程设计。文章将 MVO 与智能算法中非常著名的四种算法进行了比较：灰太狼优化算法、粒子群优化算法、遗传算法和引力搜索算法。结果表明，该算法具有非常优秀的性能，并在大多数实验中优于上述四种算法。由于其具有参数少、结构简单、效率高、易于理解等优点，多元宇宙优化算法的研究有着重要的理论意义和应用价值。

在理论研究方面，有许多学者对多元宇宙算法提出了改进的多元宇宙优化算法[7]。另外还有许多学者将多元宇宙优化算法结合其他算法提出了新的算法，聂颖[8] 等人针对支持向量机参数难以选择和确定的问题，采用多元宇宙优化算法，并在传统多元宇宙优化算法的基础上针对 TDR 值下降速度慢而导致旅行距离增加的问题，提出改进多元宇宙优化算法。任丽莉[9] 等人为提高多元宇宙优化算法求解实际问题的能力，提出了一种黏菌觅食的多元宇宙优化算法，该算法利用黏菌觅食行为在局部最优和全局最优之间寻求最优解，相较传统的多元宇宙优化算法具有更好的求解能力和优化性能。

多元宇宙优化也被广泛应用于函数优化与工业设计中。2017 年，A Fathy 和 A Darwish 将多元宇宙优化算法应用于求解质子交换膜燃料电池的参数识别问题。[10-11]2019 年，来文豪等人[12]将多元宇宙优化算法应用于煤矿突水的检测。2020 年，马骏[13]等人提出一种基于混合改进的多元宇宙优化算法优化支持向量机的短期光伏发电功率预测模型。同年，黄元[14]等人针对林区道路图像检测难的问题, 提出一种基于多元宇宙优化算法的林区道路图像检测方法。2021 年，任丽莉[9]等人基于多元宇宙优化算法, 结合时域仿真模型, 对 DFIG 变流器的控制参数进行智能优化。

2.1.2 算法介绍

多元宇宙优化算法利用白洞和虫洞的概念来探索搜索空间，相反，虫洞在多元宇宙优化算法中协助开发搜索空间，其对应关系如表1。

表 1 对应关系

宇宙名词	对应名词
黑洞	勘探
白洞	开发
虫洞	搜索

假设每个解都是一个宇宙，每个解中的变量都是一个宇宙中的对象。在解决优化问题过程中，以下规则被应用到多元宇宙优化算法中: 物体是可以通过白洞、黑洞道在不同的宇宙之间移动。当白洞，黑洞间建立两个宇宙，宇宙有一个膨胀率，利用具有较高膨胀率的宇宙有白洞，具有较低膨胀率的宇宙被认为有黑洞，宇宙中的粒子通过虫洞从白洞向黑洞位置转移的原理进行搜索的模拟。转移如图2所示，将宇宙按照膨胀率从大到小排列，在最上端的宇宙 1 的膨胀率最大，在底部的宇宙 N 膨胀率最小，彩色的圆代表宇宙中的粒子。自然膨胀率大的宇宙会向膨胀率小的宇宙传递粒子，在图中可以看到有宇宙 1 到其他宇宙的黑洞隧道。同时，除了黑洞隧道与白洞隧道之外还有虫洞隧道，在图中可以看到所有的宇宙都会建立虫洞隧道到最优宇宙，而这允许了最优宇宙把粒子传递给其他宇宙，需要注意的是这里的虫洞隧道并不是双向的，而是建立在虫洞宇宙到其他宇宙的单单项隧道。

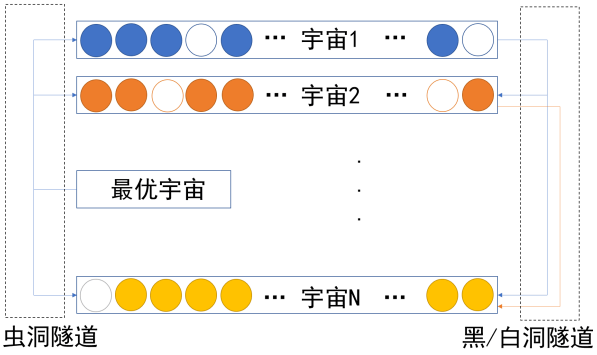

图 2 粒子转移

在整个宇宙中的迭代次数，可以确保改善平均膨胀率。在每一次迭代过程中，根据他们的膨胀率大小的分类，依靠轮盘赌机制来产生个白洞。

算法的个体构成如下:

$$u=\begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^d \\ x_2^1 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \dots & x_n^d \end{bmatrix} \tag{1}$$

公式1中 d 是变量的数目，n 是宇宙的数目。

$$x_i^j=\begin{cases} x_i^k & r<NI(U_i) \\ x_i^j & r\geq NI(U_i) \end{cases} \tag{2}$$

公式2中 x_i^j 是第 i 个宇宙第 j 个位置的黑洞，$NI(U_i)$ 代表第 i 个宇宙的膨胀率，r 为一个 $[0,1]$ 的随机数。如果随机数字 r 大于膨胀率，那么会根据轮盘赌产生到第 k 个宇宙的黑洞, 反之则不产生黑洞。最优宇宙黑洞的位置更新如公式3:

$$x_{i+1}^j=\begin{cases} X^j+TDR*((ub_j-lb_j)*r+lb_j) & (i) \\ X^j-TDR*((ub_j-lb_j)*r+lb_j) & (ii) \\ x_i^j & (iii) \end{cases} \tag{3}$$

i

在公式3中，(i) 代表着情况 $r_1<H\ and\ r_2<WEP$，(ii) 代表着情况 $r_1\geq H\ and\ r_2<WEP$，(iii) 代表着情况 $r_2\geq WEP$。其中 H 为阈值，lb_i 和 ub_i 代表着 x 的下界与上界，r、r_1、r_2 都是产生在 $[0,1]$ 的随机数字。TDR(Travelling distance rate) 为旅行距离率，WEP(Wormhole Existance Probality) 为虫洞存在概率。WEP 和 TDR 的更新公式分别如公式4和公式5所示。

$$WEP=WEP_{min}+l\times\frac{WEP_{max}-WEP_{min}}{L} \tag{4}$$

$$TDR = 1 - \frac{l^{1/P}}{L^{1/p}} \tag{5}$$

其中 l 是当前迭代次数，而 L 是最大迭代次数 WEP_{min} 与 WEP_{max} 是设定好的参数，p 是开采度。

多元宇宙优化算法的算法流程图如图4所示。

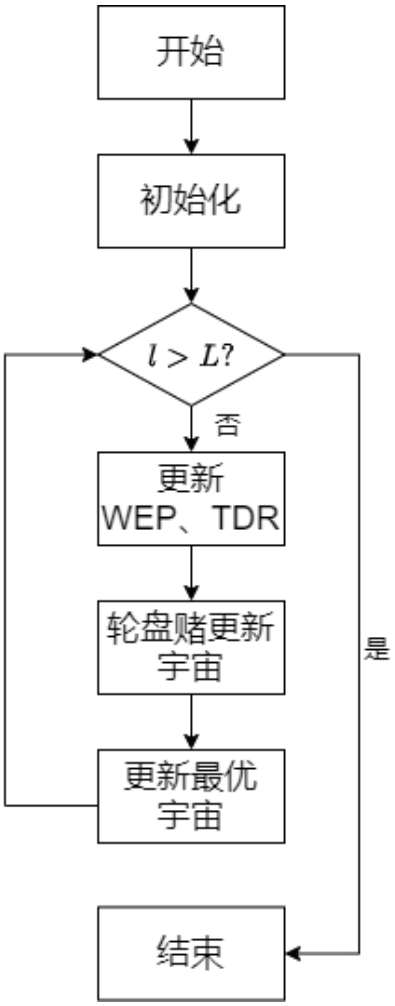

图 3 MVO 算法流程图

2.1.3 算法性能测试

为测试多元宇宙优化算法的性能，按照表2选取一些函数进行测试。设定参数之后使用算法对函数的极值进行计算。

表 2 测试函数

序号	函数	维度	范围
1	$F_1(x) = x_1^2 + x_2^2$	*2*	*[-10,10]*
2	$F_2(x) = \sum_{i=1}^{10}(-x_i \cdot sin(\sqrt{\lvert i \rvert}))$	*10*	*[-500,500]*
3	$F_3(x) = max(\lvert x_i \rvert)$	*10*	*[-100,100]*

函数 1 为抛物面函数，其在限定区间的极值为 0，实验结果如4所示，左侧图片为函数在三维空间的示意图，右侧图片为最佳宇宙膨胀率（适应值）的变化曲线，注意到图片的 y 周并不是等分的，实际上曲线随着迭代次数的变化最终已经趋于收敛。

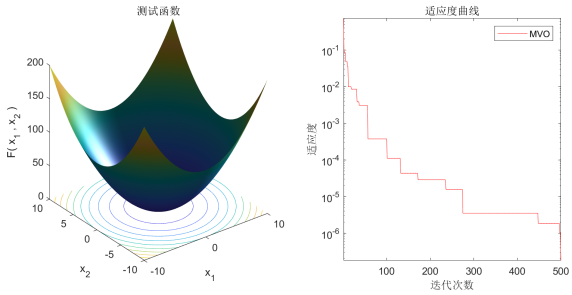

图 4 函数 1 测试效果

另外还测试了其他两个函数，实验结果分别如图5和6所示，函数均随着迭代次数的变化趋于了收敛。

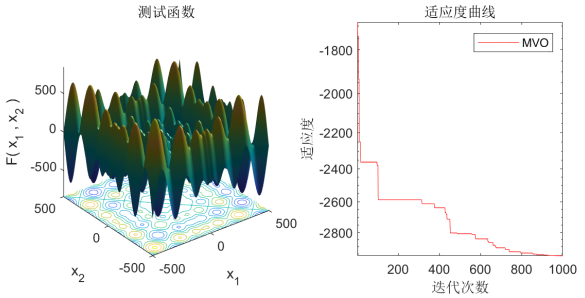

图 5 函数 2 测试效果

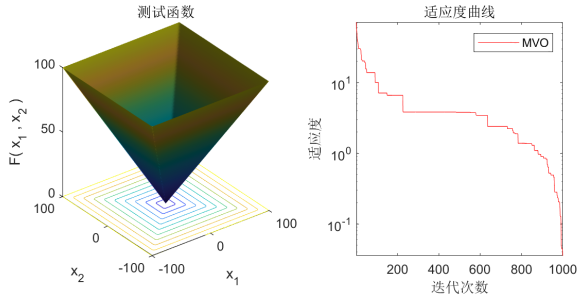

图 6 函数 3 测试效果

上述三个测试实验表明，该算法在解决单目标的约束优化问题上面有较好的鲁棒性和收敛精度。

2.2 SA 算法

2.2.1 背景介绍

模拟退火算法最早的思想是由 N. Metropolis[15] 等人于 1953 年提出。它是基于蒙特卡洛迭

代求解策略的一种随机寻优算法，其出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性。模拟退火算法有一设定初始高温，随着温度的不断降低结合随机概率在解空间中寻找全局最优。因为模拟退火独特的优化机制，其对问题信息依赖少，通用性、灵活性较强，在优化领域得到了广泛的应用。

2019 年，褚鼎立[16]等人针对鲸鱼优化算法容易陷入局部极值和收敛速度慢的问题, 提出了一种结合自适应权重和模拟退火的鲸鱼优化算法。2021 年，闫群民[17]等人为了提高粒子群算法的寻优速度和精度, 避免陷入局部优解, 提出一种自适应模拟退火粒子群优化算法。同年，尚正阳[18]等人为高效求解带能力约束的车辆路径优化问题, 提出一种带有回火操作的改进模拟退火算法。另外还有许多学者对模拟退火算法做出的若干改进[19-21]。

2.2.2 算法介绍

模拟退火算法会结合概率突跳特性在解空间中随机寻找目标函数的全局最优解，具体策略为：对原始解施加一定的随机扰动得到新的解，如果新解比当前解更优，则接受新解；如果新的解并不优于原来的姐，则基于 Metropolis 准则判断是否接受新解。具体如公式6。

$$P=\begin{cases}1 & E_{t+1}<E_t\\ e^{\frac{-(E_{t+1}-E_t)}{kT}} & E_{t+1}\geq E_t\end{cases} \tag{6}$$

如果当前的解为 x_t，这个解对应的能量为 E_t，对该解随机产生一个扰动 Δ_t，将让扰动添加到原来的解得到新的解为 x_{t+1}，如果新解的能量低于原来解的能量那么就转化为新解，如果能量相较原来更高，则按照公式6判断是否接受新解，由于有随机扰动的存在，算法具有跳出局部最优解的能力，当某一点处于局部最优解时其仍然有一定概率跳出该局部最优解，随着温度的降低这种随机扰动的影响变得越来越弱，系统也越来越趋于稳定。由于模拟退火算法具有这样的特性，其相较于其他智能算法有着更好的全局搜索能力。

模拟退火算法流程图如图7所示。

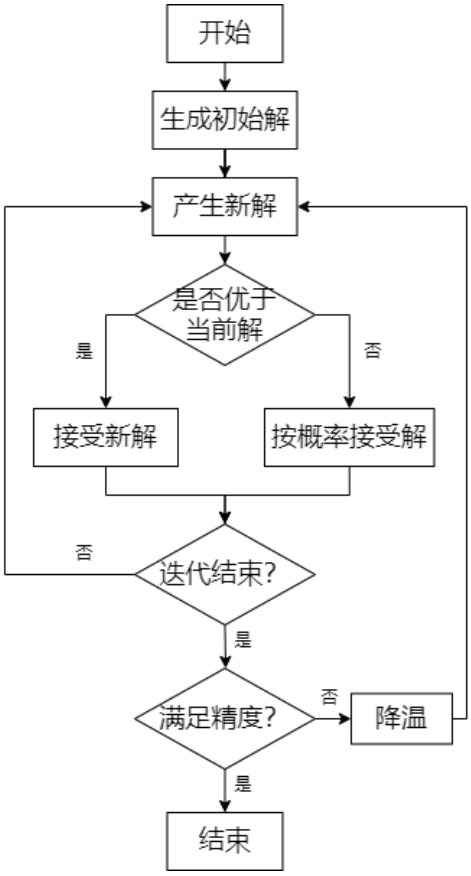

图 7 模拟退火算法流程

2.2.3 算法性能测试

可以用非常经典的 0-1 背包问题对模拟退火算法进行测试。0-1 背包问题描述如下：给定 n 种物品和一背包。物品 i 的重量为 w_i，其价值为 v_i，背包的容量为 c。问应该如何选择装入背包中的物品，使得装入背包中物品的总价值最大? 假定物体重量以及价值如表2所示。

表 3 物体重量价值

序号	重量	价值
1	*2*	*2*
2	*3*	*5*
3	*5*	*8*
4	*1*	*3*
5	*4*	*6*

在 MATLAB 中编程，首先设定好参数，包括物品价值、背包重量等参数，产生初始解后进入循环，不断产生随机扰动，根据公式判断是否接受新解，随着温度的降低最终得到问题的答案。最终得到问题的答案为 $[0,0,1,1,1]$，0 代表不选择该物体，

1 代表选择该物体，物体总价值为 17。同时使用了搜索算法和动态规划算法求解 0-1 背包问题，得到相同的答案。上述实验表明，该算法可以解决单目标的约束优化问题。

2.3 MVO-SA 算法

由于多元宇宙优化算法中虫洞只存在与最优宇宙与所有宇宙之间，这导致的所有宇宙都会趋于最优宇宙而变化而缺少对于空间的探索能力。而模拟退火算法中对于每一个温度态都会对当前解产生随机扰动，并且以一定的概率随即变换，这样就增强了算法的全局搜索能力。由于模拟退火算法具有的良好特性，可以考虑在多元宇宙算法中加入模拟退火算法来增强其全局搜索能力而避免陷入局部最优解。

2.3.1 算法改进

考虑在某一次迭代中，宇宙会向其附近的空间建立虫洞，示意图如图8。

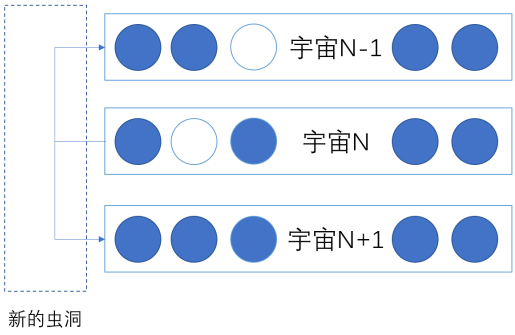

图 8 新的虫洞

宇宙 $_{N-1}$ 和宇宙 $_{N+1}$ 并不是真正存在的宇宙，而是对于宇宙 $_{N}$ 产生随机扰动之后产生的新宇宙，宇宙 $_{N}$ 会以类似模拟退火中的 Metropolis 准则向新宇宙转移。准则定义如下：

$$P=\begin{cases}1 & E_{U_{t+1}^N}<E_{U_t^N}\\ e^{\frac{-(E_{t+1}-E_t)L}{p}} & E_{U_{t+1}^N}\geq E_{U_t^N}\end{cases} \tag{7}$$

公式中 $E_{U_t^N}$ 代表宇宙 U^N 在第 t 次迭代中的膨胀率，如果新产生的宇宙的膨胀率低于原先宇宙的膨胀率，那么原来的宇宙就会通过虫洞将新宇宙的粒子转移给自己。反之，如果新产生的宇宙的膨胀率高于原来的宇宙并不会直接将概率定为 0，为了增加全局搜索能力，宇宙会按照概率7建立虫洞。

由此 MVO-SA 算法建立完毕，得到新的算法的算法流程图如图9所示。

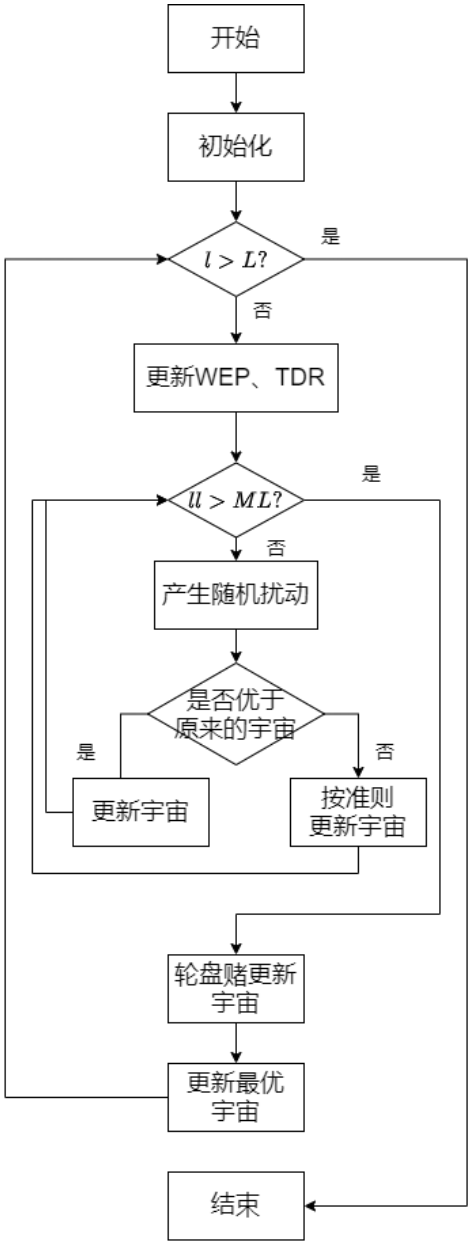

图 9 MVO-SA 算法流程

2.3.2 实验测试

首先设定函数如8，函数是对每个分量的平方求和。

$$F(x)=\sum_{i=1}^{50}x_i^2 \tag{8}$$

函数维度为 50，对于每一维的范围均为 $[-600,600]$，函数的第一维与第二维在三维空间中可以展示如图10。有图像可以轻易发现，该函数在选定空间的极值为 0，当 x 取为 $[0,0,0,\ldots,0]$ 时

可以达到极值，下面分别使用传统的多元宇宙优化算法和 MVO-SA 算法求解该函数的极值。

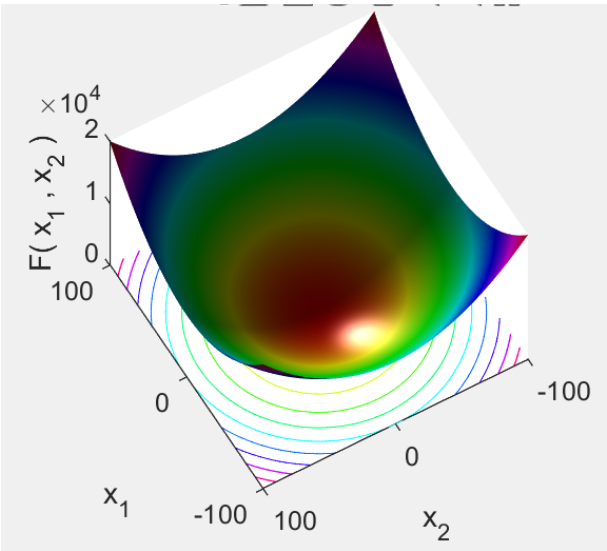

图 10 测试函数

首先使用原始多元宇宙优化算法，设定宇宙数目为 60，最大迭代次数为 500，使用传统的多元宇宙优化算法对函数求极值，得到的膨胀率变化曲线如图11。膨胀率随着迭代次数的增加逐渐下降，最终趋于稳定。

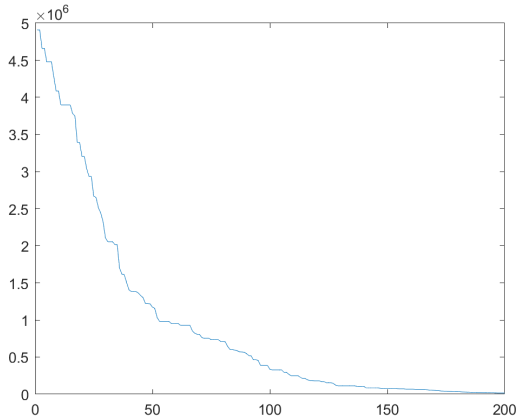

图 11 多元宇宙优化算法

同样设定宇宙数目为 60，最大迭代次数为 500，模拟退火的循环次数为 10，p 为 10，使用 MVO-SA 算法计算上述函数的极值，得到的膨胀率变化曲线如图12所示。

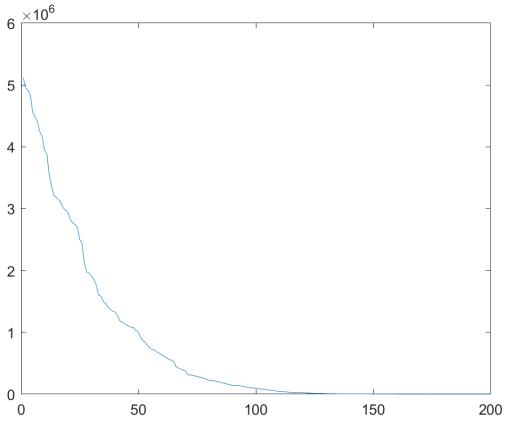

图 12 MVO-SA 算法

发现 MVO-SA 算法相较于传统的多元宇宙优化算法收敛速度更快，由于 MVO-SA 算法增加了对于宇宙周边空间的搜索能力，所以其膨胀率曲线更加平滑。MVO-SA 算法在保留多元宇宙优化算法的优良性的同时，增强了其全局搜索能力，同时具有更好的收敛速度，在实际应用中也非常有价值。

3 二维最大熵阈值分割

图像阈值分割[22-23]中的阈值通常是指灰度级阈值，而图像的灰度级直方图很好地反映了一幅图像中的灰度分布信息，是阈值选取的重要参考依据。现有的阈值选取方法大多利用了灰度级直方图信息。

3.1 直方图区域划分

图 13 Lena

对于图像13，先将其转化为灰度图像，然后统计每一灰度值的频次以及其领域平均灰度级的频次。可以画出一维灰度级直方图如图14所示。

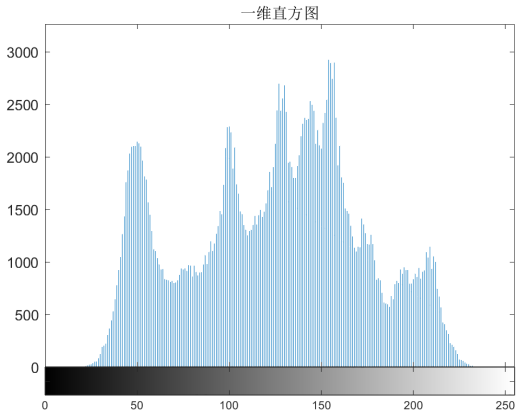

图 14 一维灰度级直方图

在传统二位灰度直方图的情况下，记灰度级为 i、领域平均灰度级为 j 的像素数目为 h_{ij}，则相应的像素出现概率为：

$$p_{ij} = \frac{h_{ij}}{M * N}, p_{ij} \geq 0 \tag{9}$$

整幅图像的灰度级均值为：

$$u_T = (\sum_{i=0}^{L-1}\sum_{j=0}^{L-1} ip_{ij}, \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} jp_{ij}) \tag{10}$$

根据公式，统计 Lena 图片的信息，画出的二维灰度级直方图如图15所示。

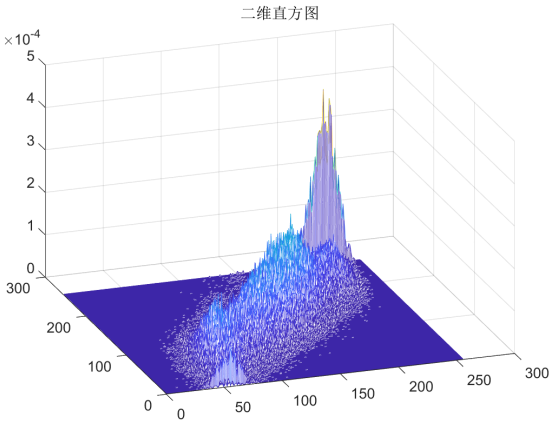

图 15 二维灰度级直方图

对于一维直方图的划分较为简单，即选择阈值划分直方图，将直方图划分为物体和背景两部分。对于二位直方图的划分稍复杂，目前常用两种方法，第一种方法是选取两条相互垂直的直线划分二位灰度直方图。还有一种方法是利用一条斜线划分二维直方图，划分如图16所示。

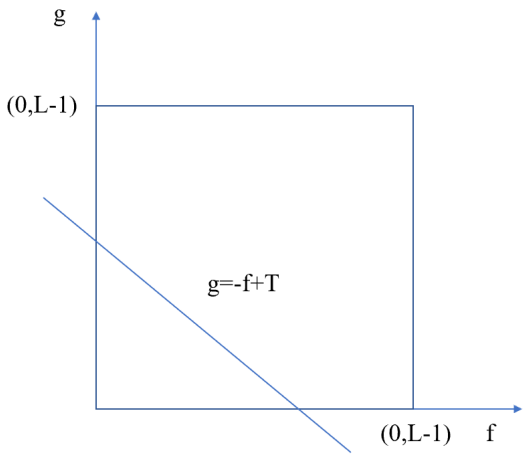

图 16 二维灰度级直方图划分

该方法认为目标像素和背景像素应该分布在直方图区域的对角线附近，而直分法显然没有遵循这一原则，存在将目标点和背景点当作边缘点和噪声点以及相反的情况。该划分方式以垂直于对角线的直线进行区域划分，对目标和背景像素的划分更为准确。

3.2 二维最大熵图像阈值分割

Kapur 等人[24]于 1985 年提出的最大熵法是另一种广受关注的阈值选取方法，其是在 Pun 等人所做工作的启发下，经过一些修正而得出。该方法以形式简单、意义明确的特点成为关注度最高、使用最多的基于熵的阈值选取方法。最大熵法的阈值选取准则是分割后的目标类和背景类的总熵值最大，即信息量最大。二维直方图情况下的最大熵计算公式为：

$$H(t,s) = -\sum_{i=0}^{t}\sum_{j=0}^{s} \frac{p_{ij}}{w_0(t,s)} ln(\frac{p_{ij}}{w_0(t,s)}) - \sum_{i=t+1}^{L-1}\sum_{j=s+1}^{L-1} \frac{p_{ij}}{w_b(t,s)} ln(\frac{p_{ij}}{w_b(t,s)}) \tag{11}$$

相应最佳的阈值为：

$$(t^*, s^*) = argmax_{0\leq t\leq L-1, 0\leq s\leq L-1} H(t,s) \tag{12}$$

4 MVO-SA 优化的图像分割算法

由上述二维最大熵阈值分割法的原理可知，要得到最终的阈值，需要去寻找阈值（t,s），使得熵值最大。于是可以利用智能优化算法进行阈值的寻优，使得获得最佳阈值。于是优化的适应度函数就是：

$$fun(t,s) = argmax_{0\le t\le L-1, 0\le s\le L-1} H(t,s) \quad (13)$$

算法由如下步骤完成：

(a) 读取图片将图片转为灰度图像。

(b) 统计灰度级频次以及领域平均灰度级，做出二位灰度直方图。

(c) 根据公式13，利用 MVO-SA 算法寻找到最优的 s 和 t 划分图片。

(d) 根据 s、t 划分图片并展示结果。

5 实验结果

本实验的环境为 Inteli5 处理器,16GB 内存, 编程语言为 MATLAB。使用图像13作为实验对象。首先统计灰度级频次以及领域平均灰度级。设定 MVO-SA 算法中宇宙数目为 10，迭代次数 100，模拟退火循环次数 10，对公式13进行优化。最终得到 $s = 76$，$t = 120$ 使得整体的熵最大，为 14.9477。熵的变化曲线如图17所示。

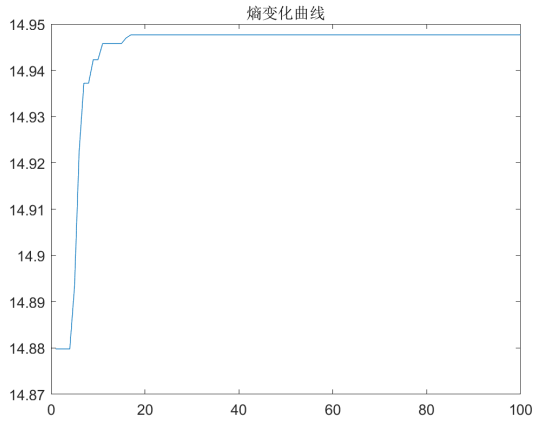

图 17 熵变化曲线

根据得到的结果，设定当灰度级小于 s 并且其领域平均灰度级小于 t 时为背景，同时其他部分为前景，按此方法划分图像，最终得到的结果如图18所示。

图 18 分割后的 Lena

6 结束语

图像的二维直方图反映了图像的点灰度分布信息，还反映了邻域空间相关信息，本文基于二维直方图使用熵最大的思想，通过 MVO-SA 算法对图像进行分割。相较于普通的分割方法该方法效率更高，分割更加准确，可以高效的实现图像阈值分割。

参考文献

[1] 韩思奇, 王蕾. 图像分割的阈值法综述[J]. 系统工程与电子技术, 2002, 24(6): 5.

[2] 李俊, 杨新, 施鹏飞. 基于 Mumford—Shah 模型的快速水平集图像分割方法[J]. 计算机学报, 2002, 25(11): 9.

[3] 潘㊣ Pan Zhe, 吴一全 Wu Yiquan. 二维 Otsu 图像分割的人工鱼群算法”,”The Two-dimensional Otsu Thresholding Based on Fish-swarm Algorithm[J]. Acta Optica Sinica, 2009.

[4] 景晓军, 蔡安妮, 孙景鳌. 一种基于二维最大类间方差的图像分割算法[J]. 通信学报, 2001, 22(4): 6.

[5] 陈修桥, 胡以华, 黄友锐. 基于二维最大相关准则的图像阈值分割[J]. 红外与毫米波学报, 2005, 24(5): 4.

[6] MIRJALILI S, MIRJALILI S M, HATAMLOU A. Multi-verse optimizer: a nature-inspired algorithm for global optimization[J]. Neural Computing and Applications, 2015, 27(2): 495-513.

[7] EWEES A A, AZIZ M, HASSANIEN A E. Chaotic multi-verse optimizer-based feature selection[J]. Neural Computing and Applications, 2019, 31(1): 1-16.

[8] 聂颖, 任楚苏, 赵杨峰. 多元宇宙优化算法改进 SVM 参数[J]. 辽宁工程技术大学学报：自然科学版, 2016(12): 5.

[9] 任丽莉, 王志军, 闫冬梅. 结合黏菌觅食行为的改进多元宇宙算法[J]. 吉林大学学报 (工学版), 2021, 51(06): 2190-2197.

[10] FATHY A, REZK H. Multi-verse optimizer for identifying the optimal parameters of pemfc model[J]. Energy, 2017, 143(jan.15): 634-644.

[11] 赵世杰, 高雷阜, 徒君, 等. 耦合横纵向个体更新策略的改进 MVO 算法[J]. 控制与决策, 2018, 33(8): 7.

[12] 来文豪, 周孟然, 李大同, 等. 无监督学习 AE 和 MVO-DBSCAN 结合 LIF 在煤矿突水识别中的应用[J]. 光谱学与光谱分析, 2019, 39(8): 6.

[13] 马骏, 江锐, 丁倩, 等. 基于多元宇宙优化支持向量机的短期光伏发电功率预测[J]. 热力发电, 2020, 49(4): 6.

[14] 黄元, 付义, 康益㊣, 等. 多元宇宙优化的林区道路图像检测方法[J]. 林业机械与木工设备, 2020, 48(2): 4.

[15] MICHAEL S, GUIDO M, ALFONS K. Heuristic and randomized optimization for the join ordering problem[J]. The VLDB Journal, 1997, 6(3): 191-208.

[16] 褚鼎立, 陈红, 王旭光. 基于自适应权重和模拟退火的鲸鱼优化算法[J]. 电子学报, 2019, 47(5): 992-999.

[17] 闫群民, 马瑞卿, 马永翔, 等. 一种自适应模拟退火粒子群优化算法[J]. 西安电子科技大学学报（自然科学版）, 2021, 48(4): 120-127.

[18] 尚正阳, 顾寄南, 王建平. 求解带能力约束车辆路径优化问题的改进模拟退火算法[J]. 计算机集成制造系统, 2021, 27(8): 2260-2269.

[19] 姚明海, 王娜, 赵连朋. 改进的模拟退火和遗传算法求解 TSP 问题[J]. 计算机工程与应用, 2013(14): 64-69.

[20] 顾汉明, 江涛, 王家映. 改进快速模拟退火方法进行 AVO 岩性参数反演[J]. 地球科学, 1999.

[21] 胡森森, 谢云. 一种改进的模拟退火算法[J]. 石油天然气学报, 2004, 026(0z3): 515-516.

[22] 罗文村, 郭伟斌. 图像阈值分割方法的比较与分析[J]. 现代计算机: 下半月版, 2000(11): 22-25.

[23] 谢鹏鹤. 图像阈值分割算法研究[D]. 湘潭大学.

[24] PUN T. A new method for gray-level picture thresholding using the entropy of the histogram[J]. Signal Processing, 1985, 2(3): 223-237.

7 附录

7.1 MVO 代码

7.1.1 mvo_main.m

```
clear;
close all;
clc;
% 多元宇宙优化
Universes_no=60;                % 宇宙数量
Max_iteration=50;               % 最大迭代数
[lb,ub,dim,fobj]=mvo_int();  % 初始化参数
[Best_score,Best_pos,cg_curve]=...
MVO(Universes_no,Max_iteration,lb,ub,dim,fobj); %MVO优化算法
plot(cg_curve);
```

7.1.2 mvo_int.m

```
function [lb,ub,dim,fobj]=mvo_int()
% 设置参数
lb=[-5,-5];  % 下界
ub=[5,5];    % 上界
dim=2;       % 维数
fobj=@F;     % 函数
    function O=F(x)
        O=x(1)^2+x(2)^2;
    end
end
```

7.1.3 MVO.m

```
function [Best_universe_Inflation_rate,Best_universe,Convergence_curve]=
MVO(N,Max_time,lb,ub,dim,fobj)
% 虫洞存在的最大与最小值
WEP_Max=1;
WEP_Min=0.2;

Best_universe=zeros(1,dim);          % 最佳宇宙位置
Best_universe_Inflation_rate=inf; % 最佳宇宙适应值
Universes=zeros(N,dim);              % 初始化宇宙
for i=1:dim
        ub_i=ub(i);
        lb_i=lb(i);
        Universes(:,i)=rand(N,1).*(ub_i-lb_i)+lb_i;
end
```

```
Convergence_curve=zeros(1,Max_time);  % 收敛曲线
Time=1;                               % 迭代器

while Time<Max_time+1
    WEP=WEP_Min+Time*((WEP_Max-WEP_Min)/Max_time); % 更新WEP
    TDR=1-((Time)^(1/6)/(Max_time)^(1/6));         % 计算TDR
    Inflation_rates=zeros(1,size(Universes,1));    % 膨胀率初始化

    for i=1:size(Universes,1)
        % 边界处理
        Flag4ub=Universes(i,:)>ub;      % 超出上界
        Flag4lb=Universes(i,:)<lb;      % 超出下界
        Universes(i,:)=(Universes(i,:).*(~(Flag4ub+Flag4lb)))+
        ub.*Flag4ub+lb.*Flag4lb;
        % 计算膨胀率
        Inflation_rates(1,i)=fobj(Universes(i,:));
        % 更行最优宇宙
        if Inflation_rates(1,i)<Best_universe_Inflation_rate
            Best_universe_Inflation_rate=Inflation_rates(1,i);
            Best_universe=Universes(i,:);
        end
    end
    % 宇宙按膨胀率排序（升序）
    [sorted_Inflation_rates,sorted_indexes]=sort(Inflation_rates);
    Sorted_universes=Universes;
    for newindex=1:N
        Sorted_universes(newindex,:)=Universes(sorted_indexes(newindex),:);
    end
    % 标准化为单位向量或矩阵，规范化矩阵行
    normalized_sorted_Inflation_rates=normr(sorted_Inflation_rates);
    % 这里有问题
    Universes(1,:)= Sorted_universes(1,:);

    for i=2:size(Universes,1)
        Back_hole_index=i;
        for j=1:size(Universes,2)  % Universes列数
            r1=rand();
            if r1<normalized_sorted_Inflation_rates(i)
                % 轮盘赌，若随机数小于膨胀率，则白变为黑
                White_hole_index=RouletteWheelSelection(-sorted_Inflation_rates);
                if White_hole_index==-1
                    White_hole_index=1;
```

```
                end
                Universes(Back_hole_index,j)=
                Sorted_universes(White_hole_index,j);
            end
            r2=rand();
            if r2<WEP
                r3=rand();
                if r3<0.5
                    Universes(i,j)=Best_universe(1,j)+
                    TDR*((ub(j)-lb(j))*rand+lb(j));
                end
                if r3>0.5
                    Universes(i,j)=Best_universe(1,j)-
                    TDR*((ub(j)-lb(j))*rand+lb(j));
                end
            end

        end
    end
    Convergence_curve(Time)=Best_universe_Inflation_rate;
    Time=Time+1;
end
end
```

7.1.4 RouletteWheelSelection.m

```
function choice = RouletteWheelSelection(weights)
  % 轮盘赌
  accumulation = cumsum(weights);
  p = rand() * accumulation(end);
  chosen_index = -1;
  for index = 1 : length(accumulation)
    if (accumulation(index) > p)
      chosen_index = index;
      break;
    end
  end
  choice = chosen_index;
end
```

7.2 SA 代码

```
clc
clear all
close all
```

```
a=0.95;%温度衰减速度
mar_length=1000;%马氏链长度
d=[2 3 5 1 4];%物品价值
k=[2 5 8 3 6];%物品重量
restriction=10;%背包能够承受的最大重量
num=length(k);%物品数量
sol_new=round(rand(1,num));%随机生成初始解
E_current=inf;E_best=inf;
%E_current是当前解对应的目标函数，E_best是最优解，E_new是新解的目标函数值
t0=97;tf=3;t=t0;
while t>tf
    for i=1:mar_length
        %产生随机扰动
        temp1=ceil(rand*num);
        sol_new(1,temp1)=~sol_new(1,temp1);
        %检查是否满足约束
        while (1)
            s=(sol_new*d'>restriction);
            if s
                %如果不满足约束随机放弃一个物品
                temp2=find(sol_new==1);
                temp3=ceil(rand*length(temp2));
                sol_new(temp2(temp3))=~sol_new(temp2(temp3));
            else
                break
            end
        end
        %计算背包中物品的价值模拟退火算法只能求最小值，所以价值取负
        E_new=sol_new*(-k');
        if E_new<E_current
            E_current=E_new;
            sol_current=sol_new;
            if E_new<E_best
                E_best=E_new;
                sol_best=sol_new;
            end
        else
            if (rand<exp(-(E_new-E_current)./t))
                E_current=E_new;
                sol_current=sol_new;
            else
                sol_new=sol_current;
```

```
            end
        end
    end
    t=t*a;
end
disp('最优解为')
sol_best
disp('物品总价值为')
-E_best
disp('背包中物品总重量')
sol_best*d'
```

7.3 MVO-SA 代码

7.3.1 mvo_main.m

```
clear;
close all;
clc;
% 多元宇宙优化
Universes_no=10;                % 宇宙数量
Max_iteration=200;               % 最大迭代数
[lb,ub,dim,fobj]=mvo_int(); % 初始化参数
[Best_score,Best_pos,cg_curve]=...
MVO(Universes_no,Max_iteration,lb,ub,dim,fobj); %MVO优化算法
plot(cg_curve);
```

7.3.2 mvo_inbt.m

```
function [lb,ub,dim,fobj]=mvo_int()
% 设置参数
lb=zeros(1,50)-600;  % 下界
ub=zeros(1,50)+600;     % 上界
dim=50;          % 维数
fobj=@F;       % 函数
    function O=F(x)
        O=0;
        for i=1:dim
            O=O+x(i)^2;
        end
    end
end
```

7.3.3 MVO.m

```
function [Best_universe_Inflation_rate,Best_universe,Convergence_curve]=
MVO(N,Max_time,lb,ub,dim,fobj)
```

```
% 虫洞存在的最大与最小值
WEP_Max=1;
WEP_Min=0.2;
% 模拟退火循环次数
ML=10;
Best_universe=zeros(1,dim);          % 最佳宇宙位置
Best_universe_Inflation_rate=inf; % 最佳宇宙适应值
Universes=zeros(N,dim);              % 初始化宇宙
for i=1:dim
        ub_i=ub(i);
        lb_i=lb(i);
        Universes(:,i)=rand(N,1).*(ub_i-lb_i)+lb_i;
end
Convergence_curve=zeros(1,Max_time);  % 收敛曲线
Time=1;                                          % 迭代器
while Time<Max_time+1
    WEP=WEP_Min+Time*((WEP_Max-WEP_Min)/Max_time); % 更新WEP
    TDR=1-((Time)^(1/6)/(Max_time)^(1/6));          % 计算TDR
    Inflation_rates=zeros(1,size(Universes,1));     % 膨胀率初始化

    for i=1:size(Universes,1)
        % 边界处理
        Flag4ub=Universes(i,:)>ub;      % 超出上界
        Flag4lb=Universes(i,:)<lb;      % 超出下界
        Universes(i,:)=(Universes(i,:).*(~(Flag4ub+Flag4lb)))+
        ub.*Flag4ub+lb.*Flag4lb;
        % 计算膨胀率
        Inflation_rates(1,i)=fobj(Universes(i,:));
        % 更行最优宇宙
        if Inflation_rates(1,i)<Best_universe_Inflation_rate
            Best_universe_Inflation_rate=Inflation_rates(1,i);
            Best_universe=Universes(i,:);
        end
    end
    % 模拟退火
     for kk=1:ML
         for pp=1:N
            rr=rand(1,50)*10-5;
            newUiverse=Universes(pp,:)+rr;

             Flag4ub=newUiverse>ub;      % 超出上界
             Flag4lb=newUiverse<lb;      % 超出下界
```

```
            newUiverse=(newUiverse.*(~(Flag4ub+Flag4lb)))+
            ub.*Flag4ub+lb.*Flag4lb;

            new_rate=fobj(newUiverse);
            if new_rate < Inflation_rates(pp)
                Universes(pp,:)=newUiverse;
                Inflation_rates(pp)=new_rate;
            else
                r2=rand();
                Met=exp(-(new_rate - Inflation_rates(pp))*Time/10);
                if r2<Met
                    Universes(pp,:)=newUiverse;
                        Inflation_rates(pp)=new_rate;
                end
            end

        end
    end

% 宇宙按膨胀率排序（升序）
[sorted_Inflation_rates,sorted_indexes]=sort(Inflation_rates);
Sorted_universes=Universes;
for newindex=1:N
    Sorted_universes(newindex,:)=Universes(sorted_indexes(newindex),:);
end
% 标准化为单位向量或矩阵，规范化矩阵行
normalized_sorted_Inflation_rates=normr(sorted_Inflation_rates);
% 这里有问题
Universes(1,:)= Sorted_universes(1,:);

for i=2:size(Universes,1)
    Back_hole_index=i;
    for j=1:size(Universes,2)  % Universes列数
        r1=rand();
        if r1<normalized_sorted_Inflation_rates(i)
            % 轮盘赌，若随机数小于膨胀率，则白变为黑
            White_hole_index=RouletteWheelSelection(-sorted_Inflation_rates);
            if White_hole_index==-1
                White_hole_index=1;
            end
            Universes(Back_hole_index,j)=Sorted_universes(White_hole_index,j);
        end
```

```
                r2=rand();
                if r2<WEP
                    r3=rand();
                    if r3<0.5
                        Universes(i,j)=Best_universe(1,j)+
                        TDR*((ub(j)-lb(j))*rand+lb(j));
                    end
                    if r3>0.5
                        Universes(i,j)=Best_universe(1,j)-
                        TDR*((ub(j)-lb(j))*rand+lb(j));
                    end
                end

            end
        end
        Convergence_curve(Time)=Best_universe_Inflation_rate;
        Time=Time+1;
    end
end
```

7.3.4 RouletteWheelSelection.m

```
function choice = RouletteWheelSelection(weights)
  % 轮盘赌
  accumulation = cumsum(weights);
  p = rand() * accumulation(end);
  chosen_index = -1;
  for index = 1 : length(accumulation)
    if (accumulation(index) > p)
      chosen_index = index;
      break;
    end
  end
  choice = chosen_index;
end
```

8 图像分割

```
function f2=readpic()
clc;
clear all;
close all;
tic
I=imread('lena30.jpg');
```

```
% I=double(I);
f=rgb2gray(I);
f2=zeros(256,256);
pp=zeros(512,512);
for i=2:511
    for j=2:511
        for ii=i-1:1:i+1
            for jj=j-1:1:j+1
                f2(f(i,j),f(ii,jj))=f2(f(i,j),f(ii,jj))+1;
                pp(i,j)=pp(i,j)+uint16(f(ii,jj));
            end
        end
    end
end
pp=round(pp./9);
%
x1=76;
x2=120;
outim=f;
for i=1:511
    for j=1:511
        if f(i,j)<x1 && pp(i,j)<x2
            outim(i,j)=0;
        else
            outim(i,j)=255;
        end
    end
end
```