一种改进的三维Otsu图像分割算法

仇国庆 熊耕耘 赵文铭
(重庆邮电大学自动化学院 重庆 400065)

摘 要 针对传统的三维Otsu分割算法计算量大、运算时间长等问题,提出一种利用一维Otsu来减小迭代空间和搜索空间,并用布谷鸟搜索算法进行寻优的算法。仿真实验表明,该算法能够有效减少运算时间。同时针对传统的三维Otsu算法因忽略2-7区域而导致错分的问题,提出了一种处理方法。该方法将2-7区域的像素点分为噪声点和非噪声点,分别对其进行处理,对2-7区域内的所有点都进行分配。仿真实验表明,由于该方法考虑了所有像素点,分割结果要优于传统的三维Otsu分割算法。

关键词 三维Otsu,布谷鸟搜索算法,阈值,图像分割
中图法分类号 TP391.41 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.08.044

Improved Three-dimensional Otsu Image Segmentation Algorithm

QIU Guo-qing XIONG Geng-yun ZHAO Wen-ming
(College of Automation,Chongqing University of Posts and Telecommunications,Chongqing 400065,China)

Abstract Aiming at the problem of large calculation and long running time of the three-dimensional Otsu image segmentation algorithm,an algorithm was proposed to reduce the iterative space and search space by using one-dimensional Otsu and cuckoo search algorithm in this paper. The simulation shows that the algorithm can effectively reduce the computing time. At the same time,aiming at the problem of the error segmentation of traditional three-dimensional Otsu image algorithm due to neglecting the region of 2 to 7,a processing method was proposed. This method divides the pixels in the region of 2 to 7 into noise and non-noise points,and assigns all the pixels. The simulation result shows that the segmentation of this method is superior to that of traditional three-dimensional Otsu segmentation algorithm.

Keywords Three-dimensional Otsu,Cuckoo search algorithm,Threshold,Image segmentation

图像分割对于图像识别有着至关重要的作用,分割结果的优劣直接影响着识别的结果[1]。而阈值分割因为计算简单、分割速度快而被广泛采用。一维的Otsu[2-5]是一种阈值分割算法,其是由日本学者于1979年提出的一种使类间方差最大以求解分割阈值的算法。但是一维Otsu的抗噪性能不佳,因此刘健庄等[6]将一维的Otsu推广到二维,选用灰度值和邻域均值构建了二维直方图,并利用最大类间方差的思想来求解。然而二维Otsu假设对象区域和背景区域上的概率近似为1,该假设普适性不够。基于此,景晓军等[7]将二维的Otsu推广到三维,在二维的基础上加入了第三维邻域中值。但是该算法的运算复杂度高,因此范九伦等[8]于2007年给出了新的递推公式,大幅缩短了运算时间。赵凤等[9]用非局部空间灰度信息代替第三维邻域中值,提高了传统Otsu的鲁棒性。为了进一步提高传统Otsu的运算效率,曾业战等[10]将粒子群优化算法[11-12]用于三维Otsu的寻优。范加武等[13]在求解空间上进行了优化,提高了运算效率,同时考虑了错分的问题,使得分割的结果优于传统的三维Otsu。本文为了提高三维Otsu的运算效率,提出了一种通过限定迭代空间和搜索空间,用布谷鸟搜索算法[14-15]来进行寻优的算法,并对分割的结果进行了后续处理,使得分割的效果更好。

1 三维Otsu图像分割算法

三维Otsu图像分割算法的思想与一维Otsu图像分割算法思想类似。首先,假设一幅图像的尺寸为$M\times N$,$g(x,y)$的定义为(x,y)处像素点的$k\times k$(k为奇数)邻域内所有像素的平均值,可用式(1)表示:

$$g(x,y)=\frac{1}{k\times k}\sum_{i=-(k-1)/2}^{(k-1)/2}\sum_{j=-(k-1)/2}^{(k-1)/2}f(x+i,y+j) \tag{1}$$

其中,$f(x,y)$为(x,y)处的像素值。$h(x,y)$的定义为(x,y)处像素点的$k\times k$邻域内所有像素的中值,可用式(2)表示:

$$h(x,y)=Med\{f(x+i,y+j),i=-(k-1)/2,\cdots,(k-1)/2;j=-(k-1)/2,\cdots,(k-1)/2\} \tag{2}$$

其中,$f(x,y)$为(x,y)处的像素值,$g(x,y)$和$h(x,y)$的灰度变化范围均为$(0,L-1)$。

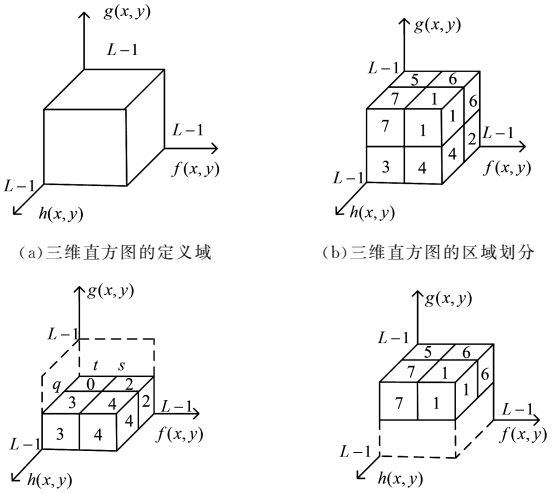
到稿日期:2017-06-25 返修日期:2017-09-13 本文受国家自然科学基金(61673079),重庆市基础科学与前沿技术研究项目(cstc2016jcyjA1919)资助。

仇国庆(1963-),男,副教授,主要研究方向为智能仪器仪表及控制装置、运动控制系统,E-mail:cquptqgq@163.com(通信作者);熊耕耘(1993-),男,硕士生,主要研究方向为图像处理、机器视觉;赵文铭(1994-),男,硕士生,主要研究方向为嵌入式。

然后将灰度值、邻域均值和邻域中值作为 3 个坐标轴，建立一个三维坐标系，如图 1 所示。本文把它定义为图像的三维直方图，由于 $f(x,y)$，$g(x,y)$和 $h(x,y)$的灰度变化范围都为$[0,L-1]$，因此该三维直方图其实是在一个 $L\times L\times L$ 大小的正方体内。用(i,j,k)表示图像中的任意一点$(f(x,y),g(x,y),h(x,y))$；直方图上任意一点的值定义为(i,j,k)在图像中出现的概率，用 p_{ijk} 来表示；用 c_{ijk} 来表示(i,j,k)在图像中出现的频数，那么 p_{ijk} 和 c_{ijk} 之间的关系可表示为：

$$p_{ijk}=\frac{c_{ijk}}{M\times N} \tag{3}$$

其中，M 和 N 分别为图像的高度和宽度，$M\times N$ 即为图像像素点的总个数，而且$\sum_{i=0}^{L-1}\sum_{j=0}^{L-1}\sum_{k=0}^{L-1}p_{ijk}=1(0\leqslant i,j,k<L)$。



(a)三维直方图的定义域 (b)三维直方图的区域划分

(c)区域 0,2,3,4 的划分 (d)区域 1,5,6,7 的划分

图 1 三维直方图的区域划分

Fig. 1 Area division of three-dimensional histogram

本文将 s,t,q 定义为灰度值、邻域均值和邻域中值 3 个平面的分割阈值，那么三维直方图将被分成 8 个长方体区域。三维 Otsu 算法使用 0～7 为每个区域编号，具体的划分方式如图 1 所示。三维 Otsu 认为由于目标内部或背景内部的像素点之间的相似性很强，因此它们的灰度值、邻域均值和邻域中值应该非常接近，故在三维直方图中区域 0 和背景对应，而区域 1 和目标对应。由于图像的边缘以及噪声等像素点的灰度值、邻域均值和邻域中值 3 个值之间的差异明显，因此区域 2－区域 7 对应边缘上的像素点和噪声点。三维 Otsu 认为在大多数情况下，与整幅图像的像素点相比，这些像素点非常少，因此假设区域 2－区域 7 上所有的点加起来的概率为 0，即：

$$\sum_{\text{区域}B}p_{ijk}\approx 0,0\leqslant i,j,k<L,B=2,3,4,5,6,7 \tag{4}$$

当用(s,t,q)分割图像时，背景(C_0)和目标区域(C_1)出现的概率分别为：

$$P_r(C_0)=\sum_{i=0}^{s}\sum_{j=0}^{t}\sum_{k=0}^{q}p_{ijk}=\omega_0(s,t,q) \tag{5}$$

$$P_r(C_1)=\sum_{i=s+1}^{L-1}\sum_{j=t+1}^{L-1}\sum_{k=q+1}^{L-1}p_{ijk}=\omega_1(s,t,q) \tag{6}$$

背景和目标区域相应的均值为：

$$\mu_{0i}(s,t,q)=(\sum_{i=0}^{s}\sum_{j=0}^{t}\sum_{k=0}^{q}ip_{ijk})/\omega_0(s,t,q) \tag{7}$$

$$\mu_{0j}(s,t,q)=(\sum_{i=0}^{s}\sum_{j=0}^{t}\sum_{k=0}^{q}jp_{ijk})/\omega_0(s,t,q) \tag{8}$$

$$\mu_{0k}(s,t,q)=(\sum_{i=0}^{s}\sum_{j=0}^{t}\sum_{k=0}^{q}kp_{ijk})/\omega_0(s,t,q) \tag{9}$$

$$\mu_{1i}(s,t,q)=(\sum_{i=s+1}^{L-1}\sum_{j=t+1}^{L-1}\sum_{k=q+1}^{L-1}ip_{ijk})/\omega_1(s,t,q) \tag{10}$$

$$\mu_{1j}(s,t,q)=(\sum_{i=s+1}^{L-1}\sum_{j=t+1}^{L-1}\sum_{k=q+1}^{L-1}jp_{ijk})/\omega_1(s,t,q) \tag{11}$$

$$\mu_{1k}(s,t,q)=(\sum_{i=s+1}^{L-1}\sum_{j=t+1}^{L-1}\sum_{k=q+1}^{L-1}kp_{ijk})/\omega_1(s,t,q) \tag{12}$$

背景和目标类间的距离测度函数为：

$$\begin{aligned}t_r\sigma_B(s,t,q)=&\omega_0(s,t,q)[(\mu_{0i}(s,t,q)-\mu_{Ti})^2+(\mu_{0j}(s,t,q)-\mu_{Tj})^2+(\mu_{0k}(s,t,q)-\mu_{Tk})^2]+\omega_1(s,t,q)[(\mu_{1i}(s,t,q)-\mu_{Ti})^2+(\mu_{1j}(s,t,q)-\mu_{Tj})^2+(\mu_{1k}(s,t,q)-\mu_{Tk})^2)]\end{aligned} \tag{13}$$

其中，$\mu_{Ti}=\sum_{i=0}^{L-1}\sum_{j=0}^{L-1}\sum_{k=0}^{L-1}ip_{ijk}$，$\mu_{Tj}=\sum_{i=0}^{L-1}\sum_{j=0}^{L-1}\sum_{k=0}^{L-1}jp_{ijk}$，$\mu_{Tk}=\sum_{i=0}^{L-1}\sum_{j=0}^{L-1}\sum_{k=0}^{L-1}kp_{ijk}$。当距离测度函数值最大时，$(s_0,t_0,q_0)$为最佳的阈值，即：

$$t_r\sigma_B(s_0,t_0,q_0)=\max_{0\leqslant s,t,q<L}\{t_r\sigma_B(s,t,q)\} \tag{14}$$

文献[7]给出了递推公式，如式(15)－式(17)所示。推导过程可参考文献[7]。

$$\omega_0(s,t,q)=\omega_0(s,t,q-1)+\omega_0(s-1,t,q)-\omega_0(s-1,t,q-1)+\psi_{stq} \tag{15}$$

$$\mu_i(s,t,q)=\mu_i(s,t,q-1)+\mu_i(s-1,t,q)-\mu_i(s-1,t,q-1)+s\psi_{stq} \tag{16}$$

$$\mu_k(s,t,q)=\mu_k(s,t,q-1)+\mu_k(s-1,t,q)-\mu_k(s-1,t,q-1)+q\psi_{stq} \tag{17}$$

其中，$\psi_{stq}=\sum_{j=0}^{t}P_{sjq}$，$P_{sjq}$ 为(s,j,q)在图像中出现的概率，可由式(3)求出，但文献[7]中 $\mu_j(s,t,q)$的公式存在错误，文献[8]已经对其进行了修正，改正后的递推公式为：

$$\mu_j(s,t,q)=\mu_j(s,t,q-1)+\mu_j(s-1,t,q)-\mu_j(s-1,t,q-1)+\sum_{j=0}^{t}jp_{sjq} \tag{18}$$

由于该递推公式的运算时间较长，不满足实时性需求，因此本文使用文献[8]中的递推公式，该递推公式的运算时间优于文献[7]给出的递推公式，其递推公式如下：

$$\begin{aligned}\omega_0(s,t,q)=&\omega_0(s-1,t,q)+\omega_0(s,t-1,q)+\omega_0(s,t,q-1)-\omega_0(s-1,t-1,q)-\omega_0(s,t-1,q-1)-\omega_0(s-1,t,q-1)+\omega_0(s-1,t-1,q-1)+P_{stq}\end{aligned} \tag{19}$$

$$\begin{aligned}\mu_i(s,t,q)=&\mu_i(s-1,t,q)+\mu_i(s,t-1,q)+\mu_i(s,t,q-1)-\mu_i(s-1,t-1,q)-\mu_i(s,t-1,q-1)-\mu_i(s-1,t,q-1)+\mu_i(s-1,t-1,q-1)+sP_{stq}\end{aligned} \tag{20}$$

$$\begin{aligned}\mu_j(s,t,q)=&\mu_j(s-1,t,q)+\mu_j(s,t-1,q)+\mu_j(s,t,q-1)-\mu_j(s-1,t-1,q)-\mu_j(s,t-1,q-1)-\mu_j(s-1,t,q-1)+\mu_j(s-1,t-1,q-1)+tP_{stq}\end{aligned} \tag{21}$$

$$\begin{aligned}\mu_k(s,t,q)=&\mu_k(s-1,t,q)+\mu_k(s,t-1,q)+\mu_k(s,t,q-1)-\mu_k(s-1,t-1,q)-\mu_k(s,t-1,q-1)-\mu_k(s-1,t,q-1)+\mu_k(s-1,t-1,q-1)+qP_{stq}\end{aligned} \tag{22}$$

其中，P_{stq} 为(s,t,q)在图像中出现的概率，可由式(3)求出。

2　布谷鸟搜索算法(CS)

布谷鸟搜索算法[16-18]是 Yang 等于 2009 年提出的一种仿生智能算法，它通过模仿布谷鸟的一些生活习性进行寻优。布谷鸟繁育下一代的方式是寄生育雏，对于这种行为，该算法提出了 3 种假设：1)布谷鸟每次仅产一个蛋，而且放入哪个鸟巢是随机的；2)较好的鸟巢将会被保留到下一代；3)布谷鸟的鸟蛋有一定的概率会被宿主发现，发现的概率为 $P_a\in(0,1)$。下面给出 CS 算法的基本步骤。

Step1　初始化各个鸟窝。随机初始化 n 个鸟窝，即 $X=(X_1,X_2,\cdots,X_n)$，每个鸟窝中蛋的个数为 m，即 $x_i=(x_1,x_2,\cdots,x_m)$。其中鸟窝表示问题的解，鸟蛋表示解的维数。

Step2　根据适应度函数计算初始化后的各个鸟窝的适应度。

Step3　每一个鸟窝通过莱维飞行的原则进行改进，计算改进后每一个鸟窝的适应度，并将其与之前鸟窝的适应度进行比较，保留适应度较好的鸟窝，改进的原则如下：

$$X_i^{(t+1)}=X_i^{(t)}+\alpha\oplus S \tag{23}$$

其中，$X_i^{(t+1)}$ 表示改进后的鸟窝；$X_i^{(t)}$ 表示改进前的鸟窝，α 可以控制步长，在大多数情况下，$\alpha=1$；S 表示莱维飞行的步长：

$$S=0.01\times\frac{u}{|v|^{1/\beta}}\times(g_{\text{best}}-X_i^{(t)}) \tag{24}$$

其中，$u\sim N(0,\sigma_u^2)$，$v\sim N(0,\sigma_v^2)$，g_{best} 表示全局最优的鸟窝。

σ_u 可由式(25)表示，其中 Γ 表示标准的伽马函数，本文中 $\beta=3/2$，$\sigma_v=1$。

$$\sigma_u=\left\{\frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\times 2^{(\beta-1)/2}}\right\}^{1/\beta} \tag{25}$$

Step4　按照被宿主发现的概率，舍弃部分鸟窝，并用偏好随机游动产生新的鸟窝，以代替那些被舍弃的鸟窝，改进公式如下：

$$X_i^{(t+1)}=X_i^{(t)}+rand\times(X_j^{(t)}-X_i^{(t)}) \tag{26}$$

其中，$rand$ 为 0 到 1 之间的随机数，$X_j^{(t)}$ 是 $X_i^{(t)}$ 附近的一个鸟窝。

Step5　计算新的鸟窝的适应度，并将其与之前鸟窝的适应度进行比较，保留适应度较好的鸟窝。

Step6　重复执行 Step3－Step5，直到满足控制条件为止。

3　本文方法

由式(18)可知，每迭代一次就要进行 7 次加(减)法运算和一次乘法运算，如果按照传统的三维 Otsu 方法去寻找最优阈值，需要计算每一种 (s,t,q) 情况下的距离测度函数的值，运算时间较长。其实在大多数情况下，最优阈值 s_0,t_0,q_0 要比灰度级 L 小很多，因此只需要求得比 s_0,t_0,q_0 稍大的一个范围内的距离测度就能求得最优阈值。

文献[10]虽然为最佳阈值的求解限定了一个范围，但是该文献中 s,t,q 的范围都由原图像中目标区域的平均灰度值来限定。当有噪声时，邻域均值和邻域中值所组成的图像目标区域的平均灰度值和原图像目标区域的平均灰度值有一定的偏差，因此这样做会导致结果不准确。并且该文献只限定了上限，没有限定下限，而下限的限定有利于进一步缩小搜索空间的范围。

本文采用一维 Otsu 来提取上限和下限，因为一维Otsu 算法简单，运算速度快。首先用一维 Otsu 分别计算出由灰度值、邻域均值和邻域中值所组成的图像的低灰度区域的平均灰度和高灰度区域的平均灰度。将 3 幅图像的低灰度区域的平均灰度作为下限，并用 $(g_{s\_low},g_{t\_low},g_{q\_low})$ 来表示；将高灰度区域的平均灰度作为上限，并用 $(g_{s\_up},g_{t\_up},g_{q\_up})$ 来表示。然后计算 $(g_{s\_up},g_{t\_up},g_{q\_up})$ 范围内的 $\omega_0(s,t,q)$，$\mu_i(s,t,q)$，$\mu_j(s,t,q)$ 和 $\mu_k(s,t,q)$。最后利用布谷鸟搜索算法在上限和下限之间进行寻优。具体的实现步骤如下。

Step1　首先使用一维 Otsu 计算出上限和下限。

Step2　计算 $(g_{s\_up},g_{t\_up},g_{q\_up})$ 范围内的 $\omega_0(s,t,q)$，$\mu_i(s,t,q)$，$\mu_j(s,t,q)$ 和 $\mu_k(s,t,q)$。

Step3　在 $(g_{s\_low},g_{t\_low},g_{q\_low})$ 和 $(g_{s\_up},g_{t\_up},g_{q\_up})$ 之间初始化鸟窝。本文鸟窝的个数为 80，鸟蛋的个数为 3，分别对应灰度值、邻域均值和邻域中值。同时根据三维 Otsu 的适应度函数计算初始化后各个鸟窝的适应度值。

Step4　通过莱维飞行原则来改进当前的鸟窝，保留较好的鸟窝，本文取 $\alpha=1$。然后按照被宿主发现的概率来改进当前的鸟窝，保留较好的鸟窝(改进的范围始终在下限和上限之间)。

Step5　重复 Step3 和 Step4，直到达到最大的迭代次数为止，本文的最大迭代次数为 1000。此时，全局最优的鸟窝即为最优分割阈值。

从算法的复杂度上看，不使用递推公式的传统三维 Otsu 算法的复杂度为 $O(L^6)$，而本文算法的复杂度为 $O(L^3)$。虽然景晓军等[7]给出了递推公式使得算法的复杂度变为 $O(L^3)$，但是并没有给出式(15)、式(16)和式(17)中 ψ_{stq} 的递推公式，这使得计算时间增加。例如，在对 $\mu_j(s,t,q)$ 的计算上，每计算一次就需要 $3+t$ 次加(减)法和 t 次乘法，随着 t 值的增大，计算量会大大增加；而本文算法每计算一次只需要 7 次加(减)法和 1 次乘法。另外，本文算法缩小了搜索的空间，并用布谷鸟搜索算法进行进一步优化，使得运算时间明显少于传统的三维 Otsu 算法的运算时间。

4　对分割结果的处理

传统的三维 Otsu 认为，与整幅图像的像素点相比，边缘上的像素点以及噪声点非常少，因此假设区域 2－区域 7 所有的点加起来的概率为 0，从而忽略了区域 2－区域 7 上的点。然而实际情况中，区域 2－区域 7 除了边缘上的像素点和噪声以外，还有一部分属于背景或目标区域的点。对于一些图像，区域 2－区域 7 中属于背景或目标区域的点比较多，这就导致了错分。实际上，即使区域 2－区域 7 都是边缘上的像素点和噪声，也不应该忽略这些点，因为忽略这些点将导致边缘的缺失和背景信息或者目标区域信息的丢失，因此这些点也应该被分到背景或者目标区域。

下面提出一种针对区域 2－区域 7 像素点的处理方法。上文已经通过一维 Otsu 求出了低灰度区域的平均灰度和高灰度区域的平均灰度，可以利用区域 2－区域 7 的像素点距离低灰度区域平均灰度和高灰度区域平均灰度的距离来判断这些点属于背景还是目标。现定义 d_{low} 为这些像素点距离低灰度区域平均灰度的距离，并用欧氏距离表示：

$$d_{\text{low}}^2=(f(x,y)-g_{s\_\text{low}})^2+(g(x,y)-g_{t\_\text{low}})^2+(h(x,y)-g_{q\_\text{low}})^2 \tag{27}$$

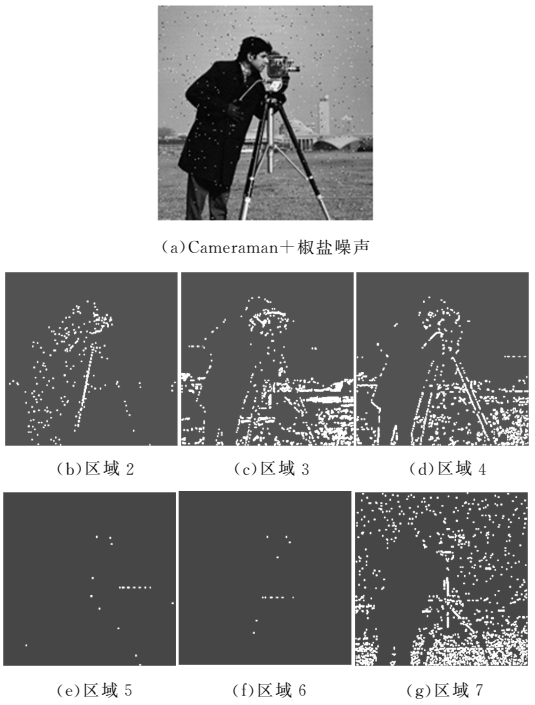
其中，$(f(x,y),g(x,y),h(x,y))\in B,B=2,3,4,5,6,7$；$f(x,y)$为图像$(x,y)$处的像素值；$g(x,y)$为$(x,y)$处的邻域均值；$h(x,y)$为$(x,y)$处的邻域中值。因为开根号的计算会占用大量的计算时间，所以这里用d_{low}^2来表示。定义d_{up}为这些像素点距离高灰度区域平均灰度的距离，其欧氏距离为：

$$d_{up}^2=(f(x,y)-g_{s\_up})^2+(g(x,y)-g_{t\_up})^2+(h(x,y)-g_{q\_up})^2 \tag{28}$$

其中，$(f(x,y),g(x,y),h(x,y))\in B,B=2,3,4,5,6,7$。

如果$d_{low}^2>d_{up}^2$，那么说明该像素点距离高灰度区域比较近，应将它分到目标区域。如果$d_{low}^2<d_{up}^2$，那么说明该像素点距离低灰度区域比较近，应将它分到背景区域。

但是如果原图中存在噪声，这些噪声点依旧按照上述距离公式进行分配就容易导致错分，因此对于噪声的分配，应该采用另外一种方式。由于邻域中值可以滤除椒盐噪声，而邻域均值对高斯噪声有很好的抑制作用，对于噪声点来说，它们的邻域中值和邻域均值应该很接近，而灰度值和邻域中值或邻域均值应该有明显的差异。因此基于上述分析，绝大多数噪声点应该存在于区域 2 和区域 7。通过对区域 2－区域 7 的像素点进行单独成像（如图 2 所示），验证了上述分析，即区域 2 中的点主要是背景区域的噪声点，而区域 7 中的点主要是目标区域的噪声点。



(a)Cameraman＋椒盐噪声

(b)区域 2 (c)区域 3 (d)区域 4

(e)区域 5 (f)区域 6 (g)区域 7

图 2 区域 2－区域 7 单独成像图

Fig. 2 Separate images in region 2 to 7

由于区域 2 和区域 7 中也存在一些不是噪声的像素点，因此要将噪声提取出来。下面给出一种区分点是否为噪声点的方法：在区域 2 或者区域 7 中，如果一个点是孤立点，它就有可能是噪声。因此若该点的邻域（本文用的是 3×3）中有m个的点同样属于该区域，则选择一个合适的阈值n，如果$m>n$，那么该点就不是噪声，反之就将其定义为噪声。由于噪声的随机性，3×3 邻域内出现 4 个或以上的噪声点的几率是很小的，因此当$n=3$时比较合理。

下面给出处理噪声的方法，由于区域 2 主要是背景区域的噪声，区域 7 主要是目标区域的噪声，因此对区域 2 和区域 7 进行单独处理。对于区域 2 中的噪声，如果它的邻域（本文用的是 3×3 邻域）中有像素点在区域 1 中，那么就将它归入目标区域，反之，归入背景区域。对于区域 7 中的噪声，如果它的邻域中有像素点在区域 0 中，那么就将它归入背景区域，反之，归入目标区域。这样处理的好处是被误分为噪声的像素点通过该处理方法也能被分给目标或背景。

具体的实现步骤如下。

Step1 判断该点属于哪个区域。

Step2 1）如果该点属于区域 0，则该点像素值置为 0。2）如果该点属于区域 1，则该点像素值置为 255。3）如果该点属于区域 2，则判断该点是否为噪声。如果是噪声，则用区域 2 噪声的处理方法判断其属于目标还是背景；如果不是噪声，则用距离公式判断其属于目标还是背景。4）如果该点属于区域 7，则判断该点是否为噪声。如果是噪声，则用区域 7 处理噪声的方法判断其属于目标还是背景；如果不是噪声，则用距离公式判断其属于目标还是背景。5）如果该点属于区域 3－区域 6，则用距离公式判断其属于目标还是背景。

Step3 重复 Step 1 和 Step 2，直到所有的点完成分割。

5 实验结果及分析

仿真实验是在 Intel（R） Core（TM） i5-3210，2. 5 GHz，8GB 内存的机器上进行的，编程环境为 Eclipse Neon. 3 Release (4. 6. 3)＋OPENCV3. 0。

实验 1 是对优化后的三维 Otsu 分割算法（不包含对分割结果的后续处理）进行仿真。本文采用的图片为 Cameraman 图和 Tungsten 图，尺寸大小分别为 256×256 和 252×298。

从表 1 的结果可以看出，本文算法要明显优于传统的三维 Otsu 算法和文献[8]的算法，运算时间分别缩短了 84％和 48％左右，节省了大量的时间，而且分割阈值是一样的。本文使用了文献[8]的递推公式，从表 1 也可以看出，该文献的递推公式要优于文献[7]中的递推公式。文献[10]使用了文献[7]中的递推公式，而且只限定了寻优上限，没有限定寻优下限；而本文使用了一维 Otsu 确定上下限，并用布谷鸟搜索算法进行寻优，可以看出，本文方法的运算时间比文献[10]算法的时间缩短了 27％左右。而文献[13]也考虑到了迭代空间对运算时间的影响，通过压缩三维直方图所在正方体的大小来缩小迭代空间，但是在运算时间上，本文算法还是稍优于文献[13]的算法。

表 1 各个算法的运行时间

Table 1 Running time of each algorithm

（单位：ms）

	Cameraman		Tungsten	
	最优分割阈值	程序运行时间	最优分割阈值	程序运行时间
文献[7]的算法	(112,110,88)	11289	(104,114,98)	12019
文献[8]的算法	(112,110,88)	2709	(104,114,98)	3576
文献[10]的算法	(112,110,88)	2093	(104,114,98)	2519
文献[13]的算法	(87,123,148)	1754	(98,68,93)	2149
本文算法	(112,110,88)	1408	(104,114,98)	1830

实验 2 是对分割结果后续处理的仿真，本文采用了 Cameraman 图和 Tungsten 图，尺寸分别为 256×256 和 252×

298。分别对两幅图加入椒盐噪声、高斯噪声和混合噪声，分割结果如图 3－图 10 所示。

(a)原图　(b)三维 Otsu　(c)文献[13]的算法　(d)本文方法

图 3　Cameraman 原图的实验结果

Fig. 3　Results of Cameraman

(a)椒盐噪声图　(b)三维 Otsu　(c)文献[13]的算法　(d)本文方法

图 4　Cameraman＋椒盐噪声的实验结果

Fig. 4　Results of Cameraman mixed with salt-and-pepper noise

(a)高斯噪声图　(b)三维 Otsu　(c)文献[13]的算法　(d)本文方法

图 5　Cameraman＋高斯噪声的实验结果

Fig. 5　Results of Cameraman mixed with Gaussian noise

(a)混合噪声图　(b)三维 Otsu　(c)文献[13]的算法　(d)本文方法

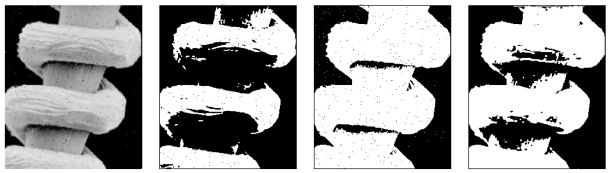
图 6　Cameraman 混合噪声的实验结果

Fig. 6　Results of Cameraman mixed with salt-and-pepper and Gaussian nosie

(a)原图　(b)三维 Otsu　(c)文献[13]的算法　(d)本文方法

图 7　Tungsten 原图的实验结果

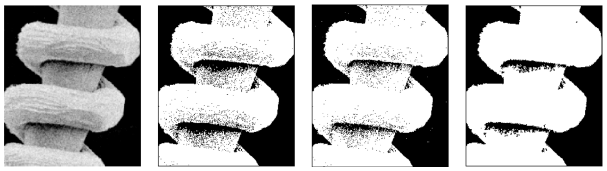
Fig. 7　Results of Tungsten



(a)椒盐噪声图　(b)三维 Otsu　(c)文献[13]的算法　(d)本文方法

图 8　Tungsten＋椒盐噪声的实验结果

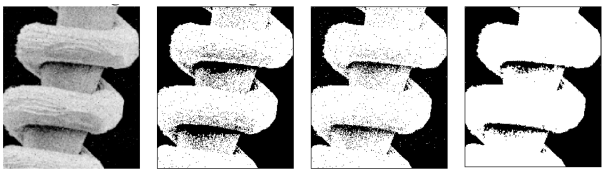
Fig. 8　Results of Tungsten mixed with salt-and-pepper nosie



(a)高斯噪声图　(b)三维 Otsu　(c)文献[13]的算法　(d)本文方法

图 9　Tungsten＋高斯噪声的实验结果

Fig. 9　Results of Tungsten mixed with Gaussian nosie



(a)混合噪声图　(b)三维 Otsu　(c)文献[13]的算法　(d)本文方法

图 10　Tungsten 混合噪声的实验结果

Fig. 10　Results of Tungsten mixed with salt-and-oepper and Gaussian nosie

由于文献[8]和文献[10]的分割阈值与传统的三维 Otsu 算法是一样的，且都没有做任何处理，因此文献[8]和文献[10]的分割结果与传统的三维 Otsu 算法相同，现把这 3 种算法放在一起分析。从实验 2 的结果可以看出，本文方法的分割效果要优于这 3 种算法。从 Cameraman 图的分割结果上来看，本文算法和这 3 种算法都能够将人和相机很好地分割出来，但是这 3 种算法将地面上很多本属于背景的点分给了目标，而本文算法对地面的分割要明显优于这 3 种算法；在加入了噪声的图像的分割结果上，本文算法依然能够保持很好的分割效果。从 Tungsten 图的分割结果中也能看出这 3 种算法的错分问题，这是因为这 3 种算法可以很好地将背景和目标分离出来，但是由于忽略了区域 2－区域 7，导致分割结果存在错分。例如，加入椒盐噪声的 Tungsten 图，用传统的三维 Otsu 分割错分会比较严重，如果考虑到区域 2－区域 7，分割结果将稍好一些。文献[13]的算法也考虑到了错分的问题，可以从实验 2 的结果中看出，其要好于前面 3 种算法，但是相比于本文算法，文献[13]的算法错分的点更多，本文算法除了在加入椒盐噪声的 Tungsten 图的分割上要稍劣于文献[13]的算法外，其他分割结果都要优于文献[13]的算法。

综合实验 1 和实验 2，本文算法不仅在运算时间上明显优于传统的三维 Otsu 算法、文献[8]的算法以及文献[10]的算法，而且分割结果要比这 3 种算法好很多。而相比于文献[13]的算法，本文算法虽然在运算效率上提高得不太明显，但是在分割结果上，本文算法要优于文献[13]的算法。综上，本文算法进一步提高了运算效率，并且明显改善了分割的效果。

结束语　本文首先指出了传统的三维 Otsu 分割算法的计算量大，运算时间长等不足，并提出了一种能够有效缩短运算时间的算法。该算法首先通过一维 Otsu 算法计算出低灰度区域的平均灰度和高灰度区域的平均灰度，并以此为上下限来限定迭代空间和搜索空间，然后利用布谷鸟搜索算法进行寻优，显著减少了算法的运行时间。其次，本文指出传统的三维 Otsu 分割算法因忽略区域 2－区域 7 而导致分割结果存在错分的不足，提出了一种能够较好地对分割结果进行后续处理的方法。该方法对区域 2－区域 7 的噪声点和非噪声点分开进行处理，利用欧氏距离来判断非噪声点属于目标还是背景，实验结果证明该方法的分割结果要优于传统的三维 Otsu 分割算法。虽然本文算法可以很好地抑制噪声，但是对于有些加入椒盐噪声的图像还是存在错分的问题，因此下一步将继续优化分割结果。

参 考 文 献

[1]　WU Y Q, MENG T L, WU S H. Research progress of image

thresholding methods in recent 20 years(1994－2014)[J]. Journal of Data Acquisition and Processing, 2015, 30(1): 1-23. (in Chinese)
吴一全,孟天亮,吴诗婳. 图像阈值分割方法研究进展 20 年(1994－2014)[J]. 数据采集与处理,2015,30(1):1-23.

[2] ZHANG P F, LU S F, LI J Q. Multi-component segmentation of X-ray computed tomography (CT) image using multi-Otsu thresholding algorithm and scanning electron microscopy[J]. Energy Exploration & Exploitation, 2017, 35(3): 281-294.

[3] YIN P Y, WU T H. Multi-objective and multi-level image thresholding based on dominance and diversity criteria[J]. Applied Soft Computing, 2017, 54: 62-73.

[4] OTSU N. A threshold selection method from gray-level histogram[J]. IEEE Transactions on Systems, 1979, 9(1): 62-66.

[5] JYOTIKA P, GAURAV G. Image Segmentation Using Genetic Algorithm OTSU[C]// 5th International Conference on Soft Computing for Problem Solving(SocProS). 2016: 473-480.

[6] LIU J Z, LI W Q. The Automatic Thresholding of Gray-Level Pictures Via Two-Dimensional OTSU Method[J]. Acta Automatica Sinica, 1993, 91(1): 101-105. (in Chinese)
刘健庄,栗文青. 灰度图象的二维 Otsu 自动阈值分割法[J]. 自动化学报,1993,19(1):101-105.

[7] JING X J, LI J F, LIU Y L. Image Segmentation Based on 3-D Maximum Between-Cluster Variance[J]. Acta Electronica Sinica, 2003, 31(9): 1281-1285. (in Chinese)
景晓军,李剑峰,刘郁林. 一种基于三维最大类间方差的图像分割算法[J]. 电子学报,2003,31(9):1281-1285.

[8] FAN J L, ZHAO F, ZHANG X F. Recursive Algorithm for Three-Dimensional Otsu's Thresholding Segmentation Method [J]. Acta Electronica Sinica, 2007, 35(7): 1398-1402. (in Chinese)
范九伦,赵凤,张雪峰. 三维 Otsu 阈值分割方法的递推算法[J]. 电子学报,2007,35(7):1398-1402.

[9] ZHAO F, FAN J L. Three-dimensional Otsu's method with non local spatial gray information[J]. Computer Engineering and Applications, 2013, 49(3): 30-33. (in Chinese)
赵凤,范九伦. 融合非局部空间灰度信息的三维 Otsu 法[J]. 计算机工程与应用,2013,49(3):30-33.

[10] ZENG Y Z, WANG R M. A method for three-dimension OTSU image segmentation based on adaptive particle swarm optimization[J]. Electronic Design Engineering, 2011, 19(13): 173-175. (in Chinese)
曾业战,王润民. 基于自适应粒子群优化的三维 OTSU 图像分割算法[J]. 电子设计工程,2011,19(13):173-175.

[11] SHILPA S, SHYAM L. Multilevel thresholding based on Chaotic Darwinian Particle Swarm Optimization for segmentation of satellite images[J]. Applied Soft Computing, 2017(55): 503-522.

[12] HE L F, HUANG S W. Modified firefly algorithm based multilevel thresholding for color image segmentation[J]. Neurocomputing, 2017, 240: 152-174.

[13] FAN J W, WANG Q P, LUO H, et al. Fast Iterative Algorithm for Segmentation Based on an Improved Three-Dimensional Otsu[C]// 2015 National Microwave and Millimeter Wave Conference. 2015: 5. (in Chinese)
范加武,王青平,罗慧,等. 基于改进的三维 Otsu 分割快速迭代算法[C]// 2015 年全国微波毫米波会议. 2015: 5.

[14] PARE S, KUMAR A, BAJAJ V. A multilevel color image segmentation technique based on cuckoo search algorithm and energy curve[J]. Applied Soft Computing, 2016, 47(C): 76-102.

[15] SUDARSHAN N, YANG X S, PRATIM S P. Color Image Segmentation By Cuckoo Search[J]. Intelligent Automation and Soft Computing, 2015, 21(4): 673-685.

[16] HE X S, LI N, YANG X S, et al. Multi-objective Cuckoo Search Algorithm[J]. Journal of System Simulation, 2015, 27(4): 731-737. (in Chinese)
贺兴时,李娜,杨新社,等. 多目标布谷鸟搜索算法[J]. 系统仿真学报,2015,27(4):731-737.

[17] YANG X S, DEB S. Cuckoo Search via Levy Flights[C]// Proc. of World Congress on Nature & Biologically Inspired Computing, India. USA: IEEE Publications, 2009: 210-214.

[18] LIU X N, MA M. Application of Cuckoo Algorithm in Multithreshold Image Segmeutation[J]. Computer Engineering, 2013, 39(7): 274-278. (in Chinese)
柳新妮,马苗. 布谷鸟搜索算法在多阈值图像分割中的应用[J]. 计算机工程,2013,39(7):274-278.

(上接第 217 页)

[23] HAWKINS D M. The Problem of Overfitting[J]. Cheminform, 2004, 35(19): 1-12.

[24] HINTON G E, SRIVASTAVA N, KRIZHEVSKY A, et al. Improving neural networks by preventing co-adaptation of feature detectors[J]. Computer Science, 2012, 3(4): 212-223.

[25] MAAS A L, DALY R E, PHAM P T, et al. Learning word vectors for sentiment analysis[C]// Meeting of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2011: 142-150.

[26] NAKOV P, RITTER A, Rosenthal S, et al. SemEval-2016 Task 4: Sentiment Analysis in Twitter[C]// International Workshop on Semantic Evaluation. 2016: 1-18.

[27] ANOOP V S, PREM S C. Generating and visualizing topic hierarchies from microblogs: An iterative latent dirichlet allocation approach[C]// 2015 International Conference on Advances in Computing Communications and Informatics (ICACCI). 2015: 824-828.

[28] LE Q V, MIKOLOV T. Distributed Representations of Sentences and Documents[J]. ICML, 2014, 4(2): 1188-1196.

[29] DUCHI J, HAZAN E, SINGER Y. Adaptive subgradient methods for online learning and stochastic optimization[J]. Journal of Machine Learning Research, 2011, 12(7): 257-269.

[30] ZEILER M D. Adadelta: an adaptive learning rate method[J]. arXiv preprint arXiv: 1212.5701, 2012.

[31] KINGMA D, BA J. Adam: amethod for stochastic optimization [J]. arXiv preprint arXiv: 1412.6980, 2014.