2 维 Otsu 自适应阈值的快速算法

郝颖明 朱 枫
(中国科学院沈阳自动化研究所,沈阳 110016)

摘 要 Otsu 自适应阈值法作为图像阈值分割的经典算法,在图像处理领域得到了广泛的应用,在其基础上发展起来的 2 维 Otsu 阈值法却因为计算时间长而制约了其应用。针对 2 维 Otsu 自适应阈值方法计算复杂度高的缺点,通过改变 2 维直方图上的区域划分,将 2 维阈值转换为 1 维阈值,从而提高了 2 维自适应阈值算法的计算速度。实验结果表明,该算法的计算时间远远小于原始 2 维 Otsu 算法,分割效果和原始算法基本一致。

关键词 2 维 Otsu 阈值 最大类间方差 阈值分割
中图法分类号: TP391 **文献标识码**: A **文章编号**: 1006-8961(2005)04-0484-05

Fast Algorithm for Two-dimensional Otsu Adaptive Threshold Algorithm

HAO Ying-ming, ZHU Feng
(*Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang* 110016)

Abstract As a classical image segmentation method, Otsu adaptive threshold algorithm has applied widely in image processing. The application of the two-dimensional Otsu threshold algorithm based on the Otsu threshold algorithm has been restricted for the long-paying computation. This paper gives a fast algorithm for two-dimensional Otsu adaptive threshold algorithm that overcomes the disadvantage of high computational complexity. The fast algorithm changes the two-dimensional threshold to one-dimensional threshold by using new area partition method, and enhances the computational speed of the two-dimensional Otsu algorithm. The experimental result has demonstrated that the computational time of the fast method is far less than that of the source two-dimensional one.

Keywords two-dimensional Otsu threshold, maximal variance between classes, threshold segmentation

1 引 言

阈值分割作为图像分割的典型算法,广泛应用于图像处理和计算机视觉领域。如何选取阈值也成为图像处理的一个热点,并提出了多种阈值选取方法[1,2]。但在实际使用中,常用的方法只有:直方图双峰法、最大熵法、Otsu 法[3]、矩量保持法[4]、梯度统计法[5],以及这些方法在 2 维上的推广化方法。这些方法中,Otsu 法以其分割效果好、适用范围广而得到了广泛的应用。

Otsu 法是由日本人大津首先提出的,也称大津阈值法或最大类间方差法。该方法以图像的 1 维直方图为依据,以目标和背景的类间方差最大为阈值选取准则,在很多情况下都能取得很好的阈值。但在实际应用中,由于噪声等干扰因素的存在,灰度直方图不一定存在明显的波峰和波谷。因此,仅仅利用 1 维直方图来确定阈值往往会造成错误分割。利用原始图像和其邻域平滑图像联合直方图,将 1 维 Otsu 法推广到 2 维而得到的 2 维 Otsu 自适应阈值分割方法[6],使分割效果得到改善。但 2 维直方图的引入,大大增加了计算复杂性,即便在 P4 2.4C

基金项目:国家"863"计划项目(2002AA401001-4A)
收稿日期:2003-11-19;**改回日期**:2004-09-30

第一作者简介:郝颖明(1966 ~),女,副研究员。1990 年于中国科学院沈阳自动化研究所获模式识别与智能控制专业工学硕士学位。主要研究领域为图像处理、计算机视觉、3 维检测。E-mail: ymhao@sia.ac.cn

CPU 上,处理一幅 512 × 512 图像的时间也要几十秒。这就在很大程度上限制了该算法的应用范围。文献[4]提出了一种改进算法,通过改变 2 维直方图上的区域划分,使计算时间提高到原算法的 17.6%,但仍旧不能满足图像处理实际应用的需要。本文在此基础上,通过将 2 维阈值转换成 1 维阈值,大大提高了算法的计算速度。

2　2 维 Otsu 阈值分割方法

设一幅图像 $f(x,y)$ 的灰度级为 $L(0,1,\cdots,L-1)$,其邻域平滑图像 $g(x,y)$(以 3 × 3 邻域均值作为该像素灰度值)的灰度级也为 L,对于图像中的任何一个像素,就有了一个二元组:像素灰度值 i 和邻域平均灰度值 j。设像素灰度值为 i 且邻域平均灰度值为 j 的像素点数为 f_{ij},图像总像素数为 M,则 2 维联合概率密度为

$$p_{ij} = \frac{f_{ij}}{M} \tag{1}$$

且, $\sum_{i=0}^{L-1}\sum_{j=0}^{L-1} f_{ij} = M$, $\sum_{i=0}^{L-1}\sum_{j=0}^{L-1} p_{ij} = 1$

任意给定一个阈值 (s,t),就可以将图像分割成图 1 所示的 4 个区域:A,B,C,D。其中,对象线上的两个区域 B 和 C 分别对应于目标和背景。而远离对象线的区域 A 和 D 则对应边缘和噪声。设目标和背景分别为 C_1,C_0,其出现的概率分别为

$$\begin{cases} \omega_0 = \sum_{i=0}^{s-1}\sum_{j=0}^{t-1} p_{ij} \\ \omega_1 = \sum_{i=s}^{L-1}\sum_{j=t}^{L-1} p_{ij} \end{cases} \tag{2}$$

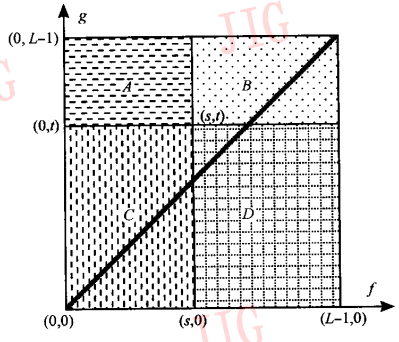


图 1　2 维阈值的原始分区

Fig. 1　The area partition of the 2D Otsu threshold

两类对应的均值矢量为

$$\begin{cases} \boldsymbol{\mu}_0 = (\mu_{0i},\mu_{0j})^{\mathrm{T}} = \left(\sum_{i=0}^{s-1}\sum_{j=0}^{t-1} ip_{ij}, \sum_{i=0}^{s-1}\sum_{j=0}^{t-1} jp_{ij}\right)^{\mathrm{T}} \\ \boldsymbol{\mu}_1 = (\mu_{1i},\mu_{1j})^{\mathrm{T}} = \left(\sum_{i=s}^{L-1}\sum_{j=t}^{L-1} ip_{ij}, \sum_{i=s}^{L-1}\sum_{j=t}^{L-1} jp_{ij}\right)^{\mathrm{T}} \end{cases} \tag{3}$$

2 维直方图的均值矢量为

$$\boldsymbol{\mu}_t = (\mu_{ti},\mu_{tj})^{\mathrm{T}} = \left(\sum_{i=0}^{L-1}\sum_{j=0}^{L-1} ip_{ij}, \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} jp_{ij}\right)^{\mathrm{T}} \tag{4}$$

在大多数情况下,远离对角线的概率可忽略不计,即假设图 1 中 A、D 两部分的概率为零。此时,可以证明:

$$\begin{cases} \omega_0 + \omega_1 \approx 1 \\ \boldsymbol{\mu}_t \approx \omega_0\boldsymbol{\mu}_0 + \omega_1\boldsymbol{\mu}_1 \end{cases} \tag{5}$$

则,类间离散矩阵为

$$\boldsymbol{S}_{\mathrm{b}} = \sum_{k=0}^{1} \omega_k[(\boldsymbol{\mu}_k - \boldsymbol{\mu}_t)(\boldsymbol{\mu}_k - \boldsymbol{\mu}_t)^{\mathrm{T}}] \tag{6}$$

离散矩阵的迹为

$$r_{\mathrm{trace}}(\boldsymbol{S}_{\mathrm{b}}) = \omega_0[(\mu_{0i} - \mu_{ti})^2 + (\mu_{0j} - \mu_{tj})^2] + \omega_1[(\mu_{1i} - \mu_{ti})^2 + (\mu_{1j} - \mu_{tj})^2] \tag{7}$$

最佳阈值为 $r_{\mathrm{trace}}(\boldsymbol{S}_{\mathrm{b}})$ 取最大值时的 (s,t)。

3　快速 Otsu 算法原理

图像的 2 维直方图分布具有一定的规律,即概率 p_{ij} 不为零的点都分布在沿对角线方向的一个窄带:$f = g - N$ 和 $f = g + N$ 内,远离对角线的部分概率都为零。其中,f 为原始图像灰度值,g 为邻域平均图像灰度值,N 为该窄带的宽度。这可以从图 2 的图像 2 维直方图分布中看出。

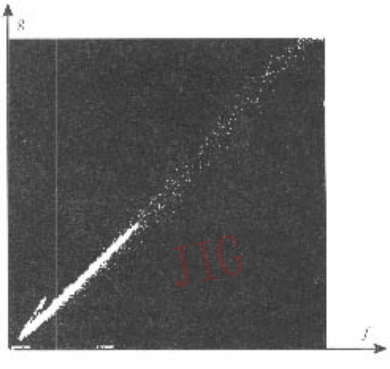


图 2　2 维直方图分布

(白色表示 $p_{ij} > 0$,黑色表示 $p_{ij} = 0$)

Fig. 2　The distribution of the 2D histogram

原始 Otsu 算法在取阈值时假设图 1 中 A、D 两部分的概率为零，在分割时，则根据目标的明暗，将 B、C 两部分分别取为目标和背景。由于概率为零的假设仅在远离对角线的部分成立，在靠近对角线的部分不再成立，如果在取阈值时将这些点全部忽略，势必会影响分割结果，而将所有的点都计算在内，计算量又很大。因此，有必要重新审视计算时的分块策略。

将 2 维直方图按图 3 的方式分块，仅将图中阴影部分的点参与运算，而将其余部分忽略不计，当 N 取足够大时，即可以将所有概率不为零的点包括进来。

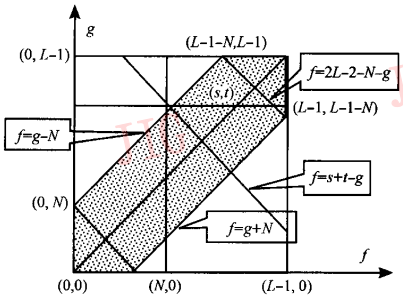


图 3 2 维直方图的新分区策略

Fig. 3 The new area partition of the 2D histogram

设分割阈值为 (s,t)，不以 A,B,C,D 进行分割，而以通过点 (s,t) 且与对角线垂直的直线来进行分割。该直线左面的点对应于 C_0，右边点对应于 C_1。

由于该直线的方程为

$$f = s + t - g \tag{8}$$

则当 $f \geqslant s+t-g$ 时，属于 C_1，当 $f < s+t-g$ 时，属于 C_0。由于分类准则只与 $s+t$ 有关，可以将 $s+t$ 整体做为一个阈值，从而将 2 维阈值转换成 1 维阈值。

在 2 维 Otsu 算法中，需要对每个 s,t 的组合计算一个离散度矩阵的迹，然后取离散度矩阵的迹最大时的 s,t 为分割阈值。在原始算法中，需要 s 和 t 的双重循环，而每次计算离散矩阵的迹，需要对 $st+(L-s)\times(L-t)$ 个点做累加运算，总累加次数为

$$A = \sum_{s=0}^{L-1}\sum_{t=0}^{L-1}[st + (L-s)(L-t)] \tag{9}$$

当 $L=256$ 时，$A=2\,147\,416\,416$。

对于本文的快速算法，由于判决标准取决于 $s+t$，而非单独的 s,t，因此仅需要 $2L$ 次循环，每次循环仅需 $N^2+2N(L-N)=N(2L-N)$ 个点做累加运算。设 $N=kL$，则总累加次数为

$$A' = 2(2k-k^2)L^3 \tag{10}$$

当 $L=256$，且 $N=40$ 时，可以将所有的非零概率点都计算在内，此时 $A'=966\,560=0.45\%A$。当 $N=20$ 时，$A'=5\,038\,080=0.23\%A$。可见其计算时间可以大大提高。

当然，在实际运算中，图像平滑和概率密度的计算对两种方法来说是相同的，其计算时间取决于图像的大小。改进算法只是在求最佳阈值的部分节省了时间，因此总体时间的比例要大于上述比例。

4 快速 Otsu 算法的实现

上述快速算法在具体实现时，需要根据 (s,t) 所处位置的不同分成不同的情况来分别进行处理。

(1) $s+t<N$

此时，分割直线在直线 $f=N-g$ 的左面，如图 4 所示。

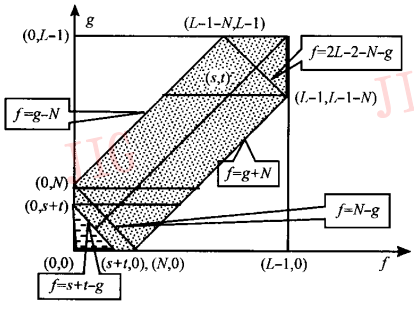


图 4 $s+t<N$ 的情况

Fig. 4 The case of $s+t<N$

仅有直线 $f=s+t-g$ 左面的部分属于 C_0。而 C_1 则包含了以下 4 段：

(a) $g<s+t$ 时，$s+t-g\leqslant f\leqslant g+N$

(b) $s+t\leqslant g<N$ 时，$0\leqslant f\leqslant g+N$

(c) $N\leqslant g<L-1-N$ 时，$g-N\leqslant f\leqslant g+N$

(d) $g\geqslant L-N-1$ 时，$g-N\leqslant f\leqslant L-1$

(2) $s+t\geqslant 2L-2-N$

此时，分割直线在直线 $f=2L-2-N-g$ 的右边，仅有直线 $g=s+t-f$ 右边的部分属于 C_1，而 C_0 则包含了以下 4 段：

(a) $g<N$ 时，$0\leqslant f\leqslant g+N$

(b) $N\leqslant g<L-1-N$ 时，$g-N\leqslant f\leqslant g+N$

(c) $L-N-1\leqslant g<s+t-L-1$ 时，

$g-N\leqslant f\leqslant L-1$

(d) $s+t-L-1\leqslant g$ 时，$g-N\leqslant f\leqslant s+t-g$

(3) $N \leqslant g \leqslant 2L-2-N$

此时 C_0 分段时,根据直线 $g=s+t-f$ 与直线 $g=f+N$ 交点的 g 坐标 $(s+t-N)/2$ 与 N 的关系,又可分为两种情况:

① $(s+t-N)/2<N$,即 $s+t<3N$

此时,C_0 包括以下 3 段:

(a) $g<(s+t-N)/2$ 时,$0 \leqslant f \leqslant g+N$

(b) $(s+t-N)/2 \leqslant g<N$ 时,$0 \leqslant f<s+t-g$

(c) $N \leqslant g \leqslant (s+t+N)/2$ 时,$g-N \leqslant f \leqslant g+N$

② $s+t \geqslant 3N$

此时,C_0 包括以下 3 段:

(a) $g<N$ 时,$0 \leqslant f \leqslant g+N$

(b) $N \leqslant g \leqslant (s+t-N)/2$ 时,$g-N \leqslant f \leqslant g+N$

(c) $(s+t-N)/2 \leqslant g<(s+t+N)/2$ 时,

$g-N \leqslant f<s+t-g$

C_1 分段时,根据直线 $g=s+t-f$ 与直线 $g=f-N$ 交点的 g 坐标 $(s+t+N)/2$ 与 $L-1-N$ 的关系,又可分为两种情况:

① $(s+t+N)/2<L-N-1$,即 $s+t<2L-2-3N$

此时,C_1 包括以下 3 段:

(a) $(s+t-N)/2 \leqslant g<(s+t+N)/2$ 时,

$s+t-g \leqslant f \leqslant g+N$

(b) $(s+t+N)/2 \leqslant g<L-1-N$ 时,

$g-N \leqslant f<g+N$

(c) $L-1-N \leqslant g<L$ 时,$g-N \leqslant f \leqslant L-1$

② $(s+t+N)/2 \geqslant L-N-1$,即 $s+t \geqslant 2L-2-3N$

此时,C_1 包括以下 3 段:

(a) $(s+t-N)/2 \leqslant g<L-1-N$ 时,

$s+t-g \leqslant f \leqslant g+N$

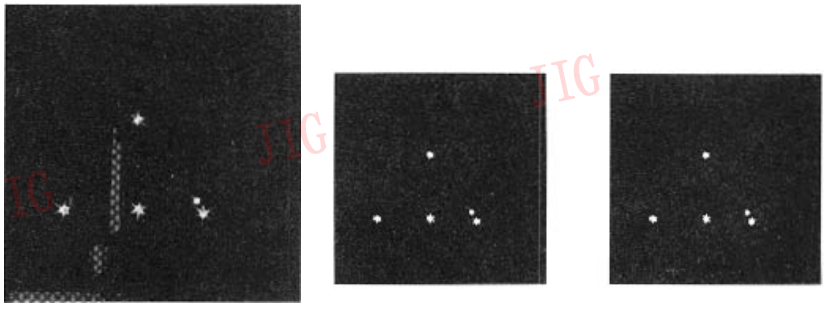
(b) $L-1-N \leqslant g<(s+t+N)/2$ 时,

$s+t-g \leqslant f \leqslant L-1$

(c) $(s+t+N)/2 \leqslant g<L-1$ 时,$g-N \leqslant f \leqslant L-1$

5　实验结果

图 5 给出了对一组目标灯的分割结果。



(a) 原始图像　　(b) 原始 2 维 Otsu 算法　　(c) 快速算法 ($N=20$)

图 5　图像分割结果

Fig. 5　The result of image segmentation

由此可见,在分割效果上,快速算法与原始算法几乎一样。在 P4 2.4C CPU,256M 内存的条件下,处理图 5 中 512 × 512 的图像。原始算法阈值为 (93,5),处理时间为 25.75s。快速算法在 $N=20$ 时,阈值为 178,处理时间为 0.188s,当 $N=40$ 时,阈值为 177,处理时间为 0.25s,当 $N=60$ 时,阈值为 177,处理时间为 0.328s。由此可见,当 $N \geqslant 40$ 时,阈值已不再改变。因此,可以取 $N=40$ 的阈值为最佳阈值,此时,处理时间为原始算法的 0.97%。当 $N=20$ 时,阈值已经很接近最佳阈值,此时的处理时间仅为原始算法的 0.73%。

从该算法的原理可知,算法的计算时间仅仅取决于图像的大小、图像的灰度分辨率和计算机的处理速度,而与具体的图像无关。实验也证明了这一点,在相同的机器上,处理同样大小的不同图像,计算时间均相同。而同样的图像,运行在不同的机器上,不仅绝对的计算时间不同,改进算法与原始算法所用时间的比例也不同。图 5 所示图像,运行在赛扬 1.7 的 CPU 和 128M 内存的机器上,原始算法的处理时间为 141.94s。快速算法在 $N=20$ 时,处理时间为 0.58s,为原始算法的 0.41%。当 $N=40$ 时,处理时间为 0.8s,为原始算法的 0.62%。

6 结 论

针对 2 维 Otsu 算法运算时间长,应用面较窄的缺点,提出了一种快速算法。该算法通过改变 2 维直方图上阈值分割的分块方式,巧妙地将 2 维阈值转成 1 维阈值,大大提高了处理速度。使得处理一幅 512×512 图像的速度小于 1s,扩大了 2 维阈值算法的应用范围。如果图像平滑可以用硬件实现,则还可以进一步提高处理速度。

参考文献(References)

1 Zhang Y J. A survey on evaluation methods for image segmentation [J]. Pattern Recognition, 1996, **29**(8): 1335 ~ 1346.

2 Han S Q, Wang L. The review of threshold methods in image segmentation[J]. System Engineering and Electronics, 2002, **4**(6): 91 ~ 102. [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, **4**(6):91 ~ 102.]

3 Otsu N. A threshold selection method from gray-level histograms[J]. IEEE Transactions on System Man and Cybernetic, 1979, **9**(1): 62 ~ 66.

4 Wang Y M, Yang W, Dong J P. Image's intensity moment and threshold [J]. Journal of Shanghai Teachers University (Natural Sciences), 2001, **30**(1): 48 ~ 51. [王耀明,严炜,董建萍. 图像的亮度矩和阈值分割[J]. 上海师范大学学报,2001, **30**(1): 48 ~ 51.]

5 Li L Y, Chen W N. A robust and completely deterministic method for gray-level picture thresholding[J]. Pattern Recognition and Artificial Intelligence, 1993, **6**(3): 235 ~ 241. [李立源,陈维楠. 一种强鲁棒的完全确定型的快速阈值化方法[J]. 模式识别与人工智能, 1993, **6**(3): 235 ~ 241.]

6 Liu J Z, Li W Q. The automatic thresholding of gray-level picture via 2D Otsu method [J]. Acta Automatica Sinica, 1993, **19**(1): 101 ~ 105. [刘建庄,栗文清. 灰度图像的二维 Otsu 自动阈值分割法[J]. 自动化学报, 1993, **19**(1): 101 ~ 105.]

7 Liang G M, Liu D H, Li B, *et al*. Improvement of a two-dimension Otsu adaptive thresholding segment algorithm [J]. Techniques of Automation and Applications, 2002, **21**(5): 43 ~ 47. [梁光明,刘东华,李波等. 二维 Otsu 自适应阈值分割算法的改进[J]. 自动化技术与应用,2002, **21**(5): 43 ~ 47.]