Otsu's method

From Wikipedia, the free encyclopedia

In computer vision and image processing, **Otsu's method**, named after Nobuyuki Otsu (大津展之 *Ōtsu Nobuyuki*), is used to automatically perform clustering-based image thresholding,[1] or, the reduction of a graylevel image to a binary image. The algorithm assumes that the image contains two classes of pixels following bi-modal histogram (foreground pixels and background pixels), it then calculates the optimum threshold separating the two classes so that their combined spread (intra-class variance) is minimal, or equivalently (because the sum of pairwise squared distances is constant), so that their inter-class variance is maximal.[2] Consequently, Otsu's method is roughly a one-dimensional, discrete analog of Fisher's Discriminant Analysis. Otsu's method is also directly related to the Jenks optimization method.

An example image thresholded using Otsu's algorithm

The extension of the original method to multi-level thresholding is referred to as the multi Otsu method.[3]

Original image

Contents

Method

In Otsu's method we exhaustively search for the threshold that minimizes the intra-class variance (the variance within the class), defined as a weighted sum of variances of the two classes:

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

Weights ω_0 and ω_1 are the probabilities of the two classes separated by a threshold t ,and σ_0^2 and σ_1^2 are variances of these two classes.

The class probability $\omega_{0,1}(t)$ is computed from the L bins of the histogram:

$$\omega_0(t) = \sum_{i=0}^{t-1} p(i)$$

$$\omega_1(t) = \sum_{i=t}^{L-1} p(i)$$

Otsu shows that minimizing the intra-class variance is the same as maximizing inter-class variance:[2]

$$\begin{aligned}\sigma_b^2(t) &= \sigma^2 - \sigma_w^2(t) = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 \\ &= \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2\end{aligned}$$

which is expressed in terms of class probabilities ω and class means μ.

while the class mean $\mu_{0,1,T}(t)$ is:

$$\mu_0(t) = \sum_{i=0}^{t-1} i\frac{p(i)}{\omega_0}$$

$$\mu_1(t) = \sum_{i=t}^{L-1} i\frac{p(i)}{\omega_1}$$

$$\mu_T = \sum_{i=0}^{L-1} ip(i)$$

The following relations can be easily verified:

$$\begin{aligned}\omega_0\mu_0 + \omega_1\mu_1 &= \mu_T \\ \omega_0 + \omega_1 &= 1\end{aligned}$$

The class probabilities and class means can be computed iteratively. This idea yields an effective algorithm.

Algorithm

1. Compute histogram and probabilities of each intensity level
2. Set up initial $\omega_i(0)$ and $\mu_i(0)$
3. Step through all possible thresholds $t = 1, \ldots$ maximum intensity
    1. Update ω_i and μ_i
    2. Compute $\sigma_b^2(t)$
4. Desired threshold corresponds to the maximum $\sigma_b^2(t)$

MATLAB implementation

histogramCounts is a 256-element histogram of a grayscale image different gray-levels (typical for 8-bit images). **level** is the threshold for the image (double).

```
function level = otsu(histogramCounts)
total = sum(histogramCounts); % '''total''' is the number of pixels in the given image.
%% OTSU automatic thresholding method
sumB = 0;
wB = 0;
maximum = 0.0;
sum1 = dot( (0:255), histogramCounts);
for ii=1:256
    wB = wB + histogramCounts(ii);
    wF = total - wB;
    if (wB == 0 || wF == 0)
        continue;
    end
    sumB = sumB +  (ii-1) * histogramCounts(ii);
    mF = (sum1 - sumB) / wF;
    between = wB * wF * ((sumB / wB) - mF) * ((sumB / wB) - mF);
    if ( between >= maximum )
        level = ii;
        maximum = between;
    end
end
end
```

Matlab has built-in functions `graythresh()` and `multithresh()` in the Image Processing Toolbox which are implemented with Otsu's method and Multi Otsu's method, respectively.

Limitations

Otsu's method exhibits the relatively good performance if the histogram can be assumed to have bimodal distribution and assumed to possess a deep and sharp valley between two peaks. But if the object area is small compared with the background area, the histogram no longer exhibits bimodality.[4] And if the variances of the object and the background intensities are large compared to the mean difference, or the image is severely corrupted by additive noise, the sharp valley of the gray level histogram is degraded. Then the possibly incorrect threshold determined by Otsu's method results in the segmentation error. (Here we define the object size to be the ratio of the object area to the entire image area and the mean difference to be the difference of the average intensities of the object and the background)

From the experimental results, the performance of global thresholding techniques including Otsu's method is shown to be limited by the small object size, the small mean difference, the large variances of the object and the background intensities, the large amount of noise added, and so on.[5]

Improvements

There are many improvements focusing on different limitations for Otsu's method.[6] One famous and effective way is known as **two-dimensional Otsu's method**. In this approach, the gray-level value of each pixel as well as the average value of its immediate neighborhood is studied so that the binarization results are greatly improved, especially for those images corrupted by noise.[7]

At each pixel, the average gray-level value of the neighborhood is calculated. Let the gray level of a given picture be divided into L values and the average gray level is also divided into the same L values. Then a pair is formed: the pixel gray level and the average of the neighborhood. Each pair belongs to a 2-dimensional bin. The total number of bins is obviously $L \times L$. The total number of occurrence(frequency), f_{ij}, of a pair(i, j) divided by the total number of pixels in the image N, defines the joint probability mass function in 2-dimensional histogram:

$$P_{ij} = \frac{f_{ij}}{N}, \qquad \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} P_{ij} = 1$$

And the 2-dimensional Otsu's method will be developed based on the 2-dimensional histogram as follows.

The probabilities of two classes can be denoted as:

$$\omega_0 = \sum_{i=0}^{s-1}\sum_{j=0}^{t-1} P_{ij}$$
$$\omega_1 = \sum_{i=s}^{L-1}\sum_{j=t}^{L-1} P_{ij}$$

The intensity means value vectors of two classes and total mean vector can be expressed as follows:

$$\mu_0 = [\mu_{0i}, \mu_{0j}]^T = \left[\sum_{i=0}^{s-1}\sum_{j=0}^{t-1} i\frac{P_{ij}}{\omega_0}, \sum_{i=0}^{s-1}\sum_{j=0}^{t-1} j\frac{P_{ij}}{\omega_0}\right]^T$$

$$\mu_1 = [\mu_{1i}, \mu_{1j}]^T = \left[\sum_{i=s}^{L-1}\sum_{j=t}^{L-1} i\frac{P_{ij}}{\omega_1}, \sum_{i=s}^{L-1}\sum_{j=t}^{L-1} j\frac{P_{ij}}{\omega_1}\right]^T$$

$$\mu_T = [\mu_{Ti}, \mu_{Tj}]^T = \left[\sum_{i=0}^{L-1}\sum_{j=0}^{L-1} iP_{ij}, \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} jP_{ij}\right]^T$$

In most cases, the probability off-diagonal will be negligible so it's easy to verify:

$$\omega_0 + \omega_1 \cong 1$$
$$\omega_0\mu_0 + \omega_1\mu_1 \cong \mu_T$$

The inter-class discrete matrix is defined as

$$S_b = \sum_{k=0}^{1} \omega_k[(\mu_k - \mu_T)(\mu_k - \mu_T)^T]$$

The trace of discrete matrix can be expressed as

$$\begin{aligned}
&\mathrm{tr}(S_b) \\
&= \omega_0[(\mu_{0i} - \mu_{Ti})^2 + (\mu_{0j} - \mu_{Tj})^2] + \omega_1[(\mu_{1i} - \mu_{Ti})^2 + (\mu_{1j} - \mu_{Tj})^2] \\
&= \frac{(\mu_{Ti}\omega_0 - \mu_i)^2 + (\mu_{Tj}\omega_0 - \mu_j)^2}{\omega_0(1-\omega_0)}
\end{aligned}$$

where

$$\mu_i = \sum_{i=0}^{s}\sum_{j=0}^{t} iP_{ij}$$
$$\mu_j = \sum_{i=0}^{s}\sum_{j=0}^{t} jP_{ij}$$

Similar to one-dimensional Otsu's method, the optimal threshold (s, t) is obtained by maximizing $\mathrm{tr}(S_b)$.

Algorithm

The s and t is obtained iteratively which is similar with one-dimensional Otsu's method. The values of s and t are changed till we obtain the maximum of $\mathrm{tr}(S_b)$, that is

```
max,s,t = 0;
for ss: 0 to L-1 do
    for tt: 0 to L-1 do
        evaluate tr(S_b);
        if tr(S_b) > max
            max = tr(S,b);
            s = ss;
            t = tt;
        end if
    end for
end for
return s,t;
```

Notice that for evaluating $\mathbf{tr}(S_b)$, we can use a fast recursive dynamic programming algorithm to improve time performance.[8] However, even with the dynamic programming approach, 2d Otsu's method still has large time complexity. Therefore, many researches have been done to reduce the computation cost.[9]

If summed area tables are used to build the 3 tables, sum over P_i_j, sum over i * P_i_j, and sum over j * P_i_j, then the runtime complexity is the maximum of (O(N_pixels), O(N_bins*N_bins)). Note that if only coarse resolution is needed in terms of threshold, N_bins can be reduced.

Matlab implementation

function inputs and output:

hists is a 256×256 2D-histogram of grayscale value and neighborhood average grayscale value pair.

total is the number of pairs in the given image.it is determined by the number of the bins of 2D-histogram at each direction.

threshold is the threshold obtained.

```
function threshold = 2D_otsu(hists, total)
maximum = 0.0;
threshold = 0;
helperVec = 0:255;
mu_t0 = sum(sum(repmat(helperVec',1,256).*hists));
mu_t1 = sum(sum(repmat(helperVec,256,1).*hists));
p_0 = zeros(256);
mu_i = p_0;
mu_j = p_0;
for ii = 1:256
    for jj = 1:256
        if jj == 1
            if ii == 1
                p_0(1,1) = hists(1,1);
            else
                p_0(ii,1) = p_0(ii-1,1) + hists(ii,1);
                mu_i(ii,1) = mu_i(ii-1,1)+(ii-1)*hists(ii,1);
                mu_j(ii,1) = mu_j(ii-1,1);
            end
        else
            p_0(ii,jj) = p_0(ii,jj-1)+p_0(ii-1,jj)-p_0(ii-1,jj-1)+hists(ii,jj);
            mu_i(ii,jj) = mu_i(ii,jj-1)+mu_i(ii-1,jj)-mu_i(ii-1,jj-1)+(ii-1)*hists(ii,jj);
            mu_j(ii,jj) = mu_j(ii,jj-1)+mu_j(ii-1,jj)-mu_j(ii-1,jj-1)+(jj-1)*hists(ii,jj);
        end

        if (p_0(ii,jj) == 0)
            continue;
        end
        if (p_0(ii,jj) == total)
            break;
        end
        tr = ((mu_i(ii,jj)-p_0(ii,jj)*mu_t0)^2 + (mu_j(ii,jj)-p_0(ii,jj)*mu_t0)^2)/(p_0(ii,jj)*(1-p_0(ii,jj)));

        if ( tr >= maximum )
            threshold = ii;
            maximum = tr;
        end
    end
end
end
```

References

1. M. Sezgin & B. Sankur (2004). "Survey over image thresholding techniques and quantitative performance evaluation". *Journal of Electronic Imaging*. **13** (1): 146–165. doi:10.1117/1.1631315 (https://doi.org/10.1117%2F1.1631315).

2. Nobuyuki Otsu (1979). "A threshold selection method from gray-level histograms". *IEEE Trans. Sys., Man., Cyber*. **9** (1): 62–66. doi:10.1109/TSMC.1979.4310076 (https://doi.org/10.1109%2FTSMC.1979.4310076).
3. Ping-Sung Liao and Tse-Sheng Chen and Pau-Choo Chung (2001). "A Fast Algorithm for Multilevel Thresholding". *J. Inf. Sci. Eng*. **17** (5): 713–727.
4. Kittler, Josef & Illingworth, John (1985). "On threshold selection using clustering criteria". *Systems, Man and Cybernetics, IEEE Transactions on*. SMC-15 (5): 652–655. doi:10.1109/tsmc.1985.6313443 (https://doi.org/10.1109%2Ftsmc.1985.6313443).
5. Lee, Sang Uk and Chung, Seok Yoon and Park, Rae Hong (1990). "A comparative performance study of several global thresholding techniques for segmentation". *Computer Vision, Graphics, and Image Processing*. **52** (2): 171–190. doi:10.1016/0734-189x(90)90053-x (https://doi.org/10.1016%2F0734-189x%2890%2990053-x).
6. Vala, HJ & Baxi, Astha (2013). "A review on Otsu image segmentation algorithm". *International Journal of Advanced Research in Computer Engineering \& Technology (IJARCET)*. **2** (2): 387.
7. Jianzhuang, Liu and Wenqing, Li and Yupeng, Tian (1991). "Automatic thresholding of gray-level pictures using two-dimension Otsu method". *Circuits and Systems, 1991. Conference Proceedings, China., 1991 International Conference on*: 325–327.
8. Zhang, Jun & Hu, Jinglu (2008). "Image segmentation based on 2D Otsu method with histogram analysis". *Computer Science and Software Engineering, 2008 International Conference on*. **6**: 105–108.
9. Zhu, Ningbo and Wang, Gang and Yang, Gaobo and Dai, Weiming (2009). "A fast 2d otsu thresholding algorithm based on improved histogram". *Pattern Recognition, 2009. CCPR 2009. Chinese Conference on*: 1–5.

External links

- Implementation of Otsu's thresholding method (https://github.com/cbhushan/script-collection/blob/master/GIMP-plugins/otsu-threshold.scm) as GIMP-plugin using Script-Fu (a Scheme-based language).
- Lecture notes on thresholding (http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MORSE/threshold.pdf) – covers the Otsu method.
- A plugin for ImageJ (http://rsb.info.nih.gov/ij/plugins/otsu-thresholding.html) using Otsu's method to do the threshold.
- A full explanation of Otsu's method (http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html) with a working example and Java implementation.
- Implementation of Otsu's method (http://www.itk.org/Doxygen/html/classitk_1_1OtsuThresholdImageFilter.html) in ITK
- Otsu Thresholding in C# (http://www.codeproject.com/KB/graphics/OtsuSharp.aspx) A straightforward C# implementation with explanation.
- Otsu's method using MATLAB (http://www.mathworks.com/discovery/image-thresholding.html)