Метод Оцу

Материал из Википедии — свободной энциклопедии

Метод Оцу (англ. *Otsu's method*) — это алгоритм вычисления порога бинаризации для полутонового изображения, используемый в области компьютерного распознавания образов и обработки изображений.

Алгоритм позволяет разделить пиксели двух классов («полезные» и «фоновые»), рассчитывая такой порог, чтобы внутриклассовая дисперсия была минимальной[1]. Метод Оцу также имеет улучшенную версию для поддержки нескольких уровней изображения[2], который получил название *мульти-Оцу метод*.

В различных русскоязычных источниках можно встретить различные способы написания фамилии автора, так, например, можно встретить *Метод Отса* и *Метод Отсу*.

Содержание

Метод

Метод Оцу ищет порог, уменьшающий дисперсию внутри класса, которая определяется как взвешенная сумма дисперсий двух классов:

$$\sigma_w^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t),$$

где веса ω_i — это вероятности двух классов, разделенных порогом t,
σ^2_i — дисперсия этих классов.

Оцу показал, что минимизация дисперсии *внутри* класса равносильна максимизации дисперсии *между* классами:[1]

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t) = \omega_1(t)\omega_2(t)\left[\mu_1(t) - \mu_2(t)\right]^2$$

которая выражается в терминах вероятности ω_i и среднего арифметического класса μ_i, которое, в свою очередь, может обновляться итеративно. Эта идея привела к эффективному алгоритму.

Алгоритм

1. Вычислить гистограмму и вероятность для каждого уровня интенсивности.
2. Вычислить начальные значения для $\omega_i(0)$ и $\mu_i(0)$.
3. Для каждого значения порога от t = 1 .. до максимальной интенсивности:
    1. Обновляем ω_i и μ_i
    2. Вычисляем $\sigma^2_b(t)$.
    3. Если $\sigma_b(t)$ больше, чем имеющееся, то запоминаем σ_b и значение порога t.
4. Искомый порог соответствует максимуму $\sigma^2_b(t)$.

Программная реализация

JavaScript

В данной функции аргумент total является общим числом пикселей изображения, а аргумент histogram — гистограмма 8-битового полутонового изображения, представленная в виде одномерного массива, в котором номер элемента кодирует номер полутона, а значение поля — количество пикселей с этим полутоном.

```
function otsu(histogram, total) {
    var sum = 0;
    for (var i = 1; i < 256; ++i)
        sum += i * histogram[i];
    var sumB = 0;
    var wB = 0;
    var wF = 0;
    var mB;
    var mF;
    var max = 0;
    var between;
    var threshold = 0;
    for (var i = 0; i < 256; ++i) {
        wB += histogram[i];
        if (wB == 0)
            continue;
        wF = total - wB;
        if (wF == 0)
            break;
        sumB += i * histogram[i];
        mB = sumB / wB;
        mF = (sum - sumB) / wF;
        between = wB * wF * Math.pow(mB - mF, 2);
        if (between > max) {
            max = between;
            threshold = i;
        }
    }
    return threshold;
}
```

Реализация на языке C

```
#include <stdlib.h> /* Для функций malloc и free*/

/* Функция возвращает порог бинаризации для полутонового изображения image с общим числом пикселей si
int otsuThreshold(IMAGE *image, int size)
{
   int min=image[0], max=image[0];
   int i, temp, temp1;
   int *hist;
   int histSize;
```

```
    int alpha, beta, threshold=0;
    double sigma, maxSigma=-1;
    double w1,a;

/**** Построение гистограммы ****/
/* Узнаем наибольший и наименьший полутон */
    for(i=1;i<size;i++)
    {
       temp=image[i];
       if(temp<min)   min = temp;
       if(temp>max)   max = temp;
    }

    histSize=max-min+1;
    if((hist=(int*) malloc(sizeof(int)*histSize))==NULL) return -1;

    for(i=0;i<histSize;i++)
       hist[i]=0;

/* Считаем сколько каких полутонов */
    for(i=0;i<size;i++)
       hist[image[i]-min]++;

/**** Гистограмма построена ****/

    temp=temp1=0;
    alpha=beta=0;
/* Для расчета математического ожидания первого класса */
    for(i=0;i<=(max-min);i++)
    {
       temp += i*hist[i];
       temp1 += hist[i];
    }

/* Основной цикл поиска порога
Пробегаемся по всем полутонам для поиска такого, при котором внутриклассовая дисперсия минимальна */
    for(i=0;i<(max-min);i++)
    {
       alpha+= i*hist[i];
       beta+=hist[i];

       w1 = (double)beta / temp1;
       a = (double)alpha / beta - (double)(temp - alpha) / (temp1 - beta);
       sigma=w1*(1-w1)*a*a;

       if(sigma>maxSigma)
       {
          maxSigma=sigma;
          threshold=i;
       }
    }
    free(hist);
    return threshold + min;
}
```

Примечания

1. N. Otsu (1979). «A threshold selection method from gray-level histograms». *IEEE Trans. Sys., Man., Cyber.* **9**: 62-66.
2. ↑ Ping-Sung Liao and Tse-Sheng Chen and Pau-Choo Chung (2001). «A Fast Algorithm for Multilevel Thresholding». *J. Inf. Sci. Eng.* **17**: 713-727.

Ссылки

- Lecture notes on thresholding (http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MORSE/threshold.pdf) — covers the Otsu method.

Источник — «https://ru.wikipedia.org/w/index.php?title=Метод_Оцу&oldid=61533382»
