PUCPR- Pontifícia Universidade Católica Do Paraná
PPGIA- Programa de Pós-Graduação Em Informática Aplicada

PROF. DR. JACQUES FACON

LIMIARIZAÇÃO POR ENTROPIA DE JOHANNSEN

Resumo: Este artigo descreve não só a teoria, mas também as ferramentas utilizadas para a implementação do algoritmo de Limiarização por Entropia de Johannsen.

Palavras Chave: Limiarização, Entropia, Segmentação Global, algoritmo de Johannsen

1. Introdução:

A limiarização tem como principal objetivo, dividir uma imagem em níveis de cinza em duas partes, transformando-a numa imagem binária, de maneira a minimizar a interdependência entre elas.
D Existem vários algoritmos para se processar a limiarização. Dentre eles, temos, Otsu, Kittler, Entropia, Johannsen e etc, sendo que o **algoritmo por Entropia de Johannsen** que será descrito abaixo.

2. Método por Entropia de Johannsen

O Método por Entropia de Johannsen baseia-se, na divisão dos níveis de cinza da imagem em duas partes, a fim de minimizar a interdependência entre elas. O Método de Johannsen baseia-se na Entropia. A entropia é uma medida de conteúdo de informação. A função entropia de um símbolo x é dada por: $E(x) = -x\log(x)$

Extrapolando-se o conceito acima, uma imagem, que é uma fonte de níveis de cinza, pode ser também vista como uma fonte de símbolos. O algoritmo por Entropia de Johannsen baseia-se na divisão do nível de cinza numa parte preta e numa parte branca. A entropia associada, para uma imagem com 256 níveis de cinza, com os pixels pretos tendo sido limiarizada utilizando-se o limiar t, é dada por Sb(t):

$$Sb(t) = \log(\sum_{i=0}^{t} p_i) + \frac{1}{\sum_{i=0}^{t} p_i}\left[E(p_t) + E(\sum_{i=0}^{t-1} p_i)\right]$$

De forma análoga, a entropia de pixels brancos é dada por Sw(t):

$$Sw(t) = \log(\sum_{i=t}^{255} p_i) + \frac{1}{\sum_{i=t}^{255} p_i}\left[E(p_t) + E(\sum_{i=t+1}^{255} p_i)\right]$$

Segundo o algoritmo de Johannsen, o valor do limiar ótimo é o valor de t que minimiza a soma

Sb(t)+Sw(t). Deve-se observar que os valores calculados para Sb(t) e Sw(t) devem ser desprezados para valores de t quando pt=0.

3. Conclusão:

Após desenvolvido o algoritmo contatou-se que surgiram dúvidas quanto ao uso do Log ou Log10 para o cálculo de Sb(t) e Sw(t), mas verificando os valores viu-se que os resultados são praticamente os mesmos, com pouca alteração.

4. Referências

Johannsen G. and Bille J.,"*A Threshold Selection Method using Information Measures*", Proceedings, 6th Int. Conf. Pattern Recognition, Munich, Germany, pp.140-143, 1982.

IMPLEMENTAÇÃO:

```
// Algoritmo de Limiarizacao por Entropia de Johannsen
BOOL CLimiar::LimiarEntropiaJohannsen()
{
    int             t,g;
    double          sum1,sum2,minimo;
    double          sb[256],sw[256],tl[256],p[256];

    BYTE            Limiar;

    if ( !(VerifyConsistentIn() && VerifyConsistentOut()) )
        return FALSE;

    CopyImageInOut();

    ClockStart();
    Histograma();

    // Calculo das probabilides a priori
    for(g=0;g<256;g++)
    {
        p [g] = (double)m_Histo[g]/m_TotalPixels;
        sb[g] = 0;
        sw[g] = 0;
        tl[g] = 0;
    }

    // Calculo
    for(t=0;t<256;t++)
    {
        sum1 = 0;
        sum2 = 0;

        for(g=0;g<=t  ;g++)
            sum1 += p[g];

        for(g=0;g<=t-1;g++)
            sum2 += p[g];

        if ( (p[t] != 0) && (sum1 != 0) && (sum2 != 0) )
        {
            sb[t]    =            (     log10((double)sum1))
                     - (1.0/sum1)*((p[t]*log10((double)p[t]))
                                  +(sum2*log10((double)sum2)));
        }
    }

    for(t=0;t<256;t++)
    {
        sum1 = 0;
        sum2 = 0;

        for(g=t  ;g<256;g++)
            sum1 += p[g];

        for(g=t+1;g<256;g++)
            sum2 += p[g];
```

```
        if ( (p[t] != 0) && (sum1 != 0) && (sum2 != 0) )
        {
            sw[t]   =           (     log10((double)sum1))
                    - (1.0/sum1)*((p[t]*log10((double)p[t]))
                            +(sum2*log10((double)sum2)));
        }
    }

    // Calculo do Limiar de Johannsen
    for(g=0;g<256;g++)
    {
        if ( (sb[g] != 0) && (sw[g] != 0) )
            tl[g] = sb[g] + sw[g];
    }

    t=0;
    while(!tl[t]) {t++;};
    minimo = tl[t];
    Limiar = (BYTE)t;

    for(g=t+1;g<256;g++)
    {
        if ( (tl[g] != 0) && (tl[g] < minimo) )
        {
            minimo = tl[g];
            Limiar = (BYTE)g;
        }
    }

    SetLimiar((BYTE)Limiar);

    AplicarLimiar();

    ClockFinish("Limiarizacao por Entropia de Johannsen");

    return TRUE;
}
```