LIMIARIZAÇÃO POR MATRIZ DE CO-OCORRÊNCIA

Resumo: O método da matriz de co-ocorrência, considerando imagens com tons de cinza, possui uma importante posição na análise de limiarizações. Várias medidas tomadas com base nessa matriz, através da análise de vários valores possíveis, são propostas para selecionar o melhor método de limiarização. Foram analisados vários métodos e escolhido o melhor para ser implementado, que ocupa menos tempo da CPU e menos espaço de memória. Este método é melhor usado para aplicações locais de limiarização.

Palavras-chave: Segmentação de Imagens; Limiarização de Tons de Cinza; Matriz de Co-ocorrência.

1. Introdução

Existem atualmente muitos métodos de limiarização para imagens com tons de cinza. Muitos deles usam o histograma da imagem para encontrar um ponto de classificação dos pixels. Por outro lado, as estatísticas dos níveis de cinza de uma imagem, espalhadas por toda sua extensão, são usadas para montar a matriz de co-ocorrência, que indica a distribuição dos níveis de cinza em uma imagem. Isto possibilita a identificação de pontos homogêneos na imagem, separando-os do fundo dela. Baseado nisso, vários cálculos são feitos para se escolher o melhor método de limiarização a ser usado em cada caso. O uso de uma matriz esparsa, porém, não é um método muito eficiente devido à sua lentidão. Iremos demonstrar, portanto, um método prático e rápido de análise por matriz de co-ocorrência.

A matriz de co-ocorrência contém informações a respeito da posição dos pixels que tem um mesmo valor de nível de cinza. Assim, podemos varrer a imagem nas direções vertical, horizontal ou diagonal e verificar quantos pixels, que diferem de um valor n estão separados por uma distância *d*. Mas para cada direção, se analisarmos a imagem inteira, teremos muita informação, o que diminuirá a eficiência da análise. Portanto podemos juntar estas informações em elementos menores, que são chamados de descritores.

2. Procedimento

Sendo G(x,y) uma imagem de L níveis de cinza, de tamanho MxN, sua matriz de co-ocorrência será:

$$C = \{C_{m,n}\} = C_0 + C_{\pi/2} + C_{\pi} + C_{3\pi/2},$$

onde C_0 é a estatística de pixels adjacentes na direção 0 (direita), $C_{\pi/2}$ é a estatística de pixels adjacentes na direção $\pi/2$ (acima), C_{π} é a estatística de pixels adjacentes na direção π, (esquerda) e $C_{3\pi/2}$ é a estatística de pixels adjacentes na direção $3\pi/2$ (abaixo). Como a matriz é simétrica, temos que $c_{m,n} = c_{n,m}$ e $c_{m,n} \geq 0$.

Sejam os pixels *P1* e *P2* (em níveis de cinza) separados horizontalmente por um número de pixels *d* no interior da imagem *G*. Considere-se $C_0[m,n]$ como sendo a matriz que armazena a quantidade de pixels com níveis de cinza *P1* e *P2* separados pela distância pré-definida *d* à direita. Os índices m e n de $C_0[m,n]$ são os níveis de cinza correspondentes aos pixels *P1* e *P2* respectivamente.

O cálculo utilizado para uma distância *d* horizontal à direita (direção 0) pode ser o seguinte:

```
for (m = 0; m < M; ++m)
{
    for (n = 0; n < N - d; ++n)
    {
        p1 = G[n][m];          // Obtem o nível de cinza do pixel P1
        p2 = G[n+d][m];        // Obtem o nível de cinza do pixel P2
        C0[p1][p2] + = 1;      // Incrementa elemento de C0
    }
}
```

Ao final da execução deste cálculo, a parcela da matrix de co-ocorrência C_0 *[m,n]* estará completa contento o número total de pixels em nível de cinza *P1* e *P2* separados distância *d* na direção 0.

Sabe-se que, o número de pixels representado por cada elemento de C_0 *[m,n]* dividido pelo número total de pixels analisados (MxN) nos dá a probabilidade de um pixel *P2* com nível de cinza *m* estar a uma distância horizontal à direita de um pixel *P1* com nível de cinza *n.*

Para calcular $C_{\pi/2}$, C_{π} e $C_{3\pi/2}$ o algoritmo deve sofrer pequenas alterações nos limites superiores dos loops e no cálculo de *P2*. Por exemplo, para $C_{3\pi/2}$, calcula-se *P2* como sendo G[n][m+d].

Após calculado em todas as 4 direções, soma-se todas as matrizes para se obter a matrix de co-ocorrência C = $\{C_{m,n}\}$.

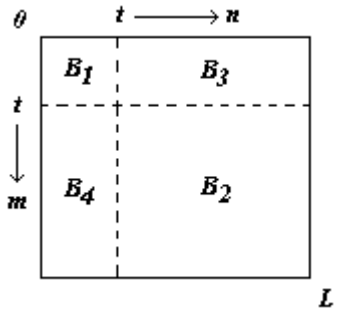
Sendo *t* o limiar para a binarização da matriz, *t* divide a matriz em quatro blocos distintos: $B_1(t)$, $B_2(t)$, $B_3(t)$, $B_4(t)$.

$$B_1(t) = \sum_{m=0}^{t-1}\sum_{n=0}^{t-1} w_{m,n}.c_{m,n}$$

$$B_2(t) = \sum_{m=t}^{L-1}\sum_{n=t}^{L-1} w_{m,n}.c_{m,n}$$

$$B_3(t) = \sum_{m=0}^{t-1}\sum_{n=t}^{L-1} w_{m,n}.c_{m,n}$$

$$B_4(t) = \sum_{m=t}^{L-1}\sum_{n=0}^{t-1} w_{m,n}.c_{m,n}$$



onde, $w_{m,n}$ é o coeficiente que indica o peso para cada entrada componente da matrix co-ocorrência $C_{m,n}$.

Alguns métodos de limiarização avaliam a matriz segmentada em *t* com medidas da matriz de co-ocorrência e escolhem um ponto *t** que otimiza a limiarização da imagem.

Duas medidas e seus objetivos são:

Medida Business, para o qual

$$t^* = \underset{0\le t\le L-1}{\text{mínimo}}\{B_3(t) + B_4(t)\},$$
$$\text{onde } w_{m,n} = 1, \text{ para } 0 \le m, n \le L-1$$

Medida da Probabilidade Condicional, para o qual

$$t^* = \underset{0\le t\le L-1}{\text{mínimo}}\left\{\frac{B_3(t)}{B_1(t)+B_3(t)} + \frac{B_4(t)}{B_2(t)+B_4(t)}\right\},$$
$$\text{onde } w_{m,n} = 1, \text{ para } 0 \le m, n \le L-1$$

Para a medida business, calcula-se t* obtendo-se o valor de t para o qual a soma entre $B_3(t)$ e $B_4(t)$ seja

mínima.

Para a medida da probabilidade condicional, calcula-se t* obtendo-se o valor mínimo para qual $\frac{B_3(t)}{B_1(t)+B_3(t)}+\frac{B_4(t)}{B_2(t)+B_4(t)}$ seja mínimo.

Ambas as medidas utilizam o coeficiente de peso somente com o valor 1. Outras medidas mais complexas exigem a utilizazação de coeficiente de peso ($w_{m,n}$) diferentes de 1.

Este método convencional pode ainda ser otimizado com relação aos requisitos de uso da memória, esforço computacional (processamento) e de tempo. Para tanto, pode-se dividir as diagonais de $\{c_{m,n}\}$ (assumindo que $\{w_{m,n}\}$ será simétrica) e trabalhar somente com a triangular superior ou a inferior da matriz $\{C_{m,n}\}$.

A partir deste princípio, pode-se implementar um método que aumente consideravelmente a eficiência da obtenção dos valores ideais de limiarização (t*).

Uma análise da performance de ambos os métodos (convencional e eficiente) nos dá a seguinte tabela:

	Pesagens	**Multiplicação Float**	**Adição Float**	**Adição Inteira**
Convencional	L^2	L^2+2L	L^2+4L	$2MN+L^2$
Eficiente	2MN	2L	6MN+4L	0

3. Conclusões

Utilizando o método de limiarização por matriz de co-ocorrência, pudemos observar alguns fatores interessantes quanto a este algoritmo. Ele é bem eficiente se a imagem tiver um vale bem definido, pois se a imagem possuir vários vales (verificado através de seu histograma) o vale escolhido para sua limiarização será o de menor amplitude, e em alguns casos a imagem limiarizada pode não ficar muito clara em relação à imagem original.

Obtivemos também alguns problemas quanto ao significado do termo "minimize" de determinados métodos em função de t. Inicialmente, este valor era sempre muito baixo, pois quando se está no início da análise da imagem, o valor de B3 e B4 (partes da matriz de co-ocorrência) são sempre muito baixos, e este valor baixo é escolhido como valor de limiar. Para resolver este problema fizemos algumas mudanças quando ao significado de minimize, para então escolhermos um valor razoável de limiar. Esta mudança consiste em, se o valor encontrado for igual a zero, ignore-o. Mas mesmo assim ainda podem haver problemas de limiar insatisfatório, e por isso este método só funciona se a imagem tiver dois vales bem definidos.

4. Referências

B. Chanda and D.D. Majunder, "A note on the use of the graylevel co-ocorrence matrix in threshold selection", Signal Processing, Vol 15, No.2, pp 149-167, Setembro 1988

Wen Nung Lie, "Na efficient threshold-evaluation algorithm for image segmentation based on spatial grayevel co-ocorrences", Signal Processing, Vol 33, pp 121-126, 1993

```
// Algoritmo de Limiarizacao por Matriz de Co-Ocorrência
BOOL CLimiar::LimiarMatrizCoOcorrencia()
{
    const BYTE      Distancia  = GetDistancia_MatrizCoOcorrencia(); // Default: 1
    const BYTE      W          = 1; // Coeficiente de peso (default = 1 para os critérios Business e Prob. Condicional)

    unsigned long   B1, B2, B3, B4;
    unsigned long   c[256][256];
    unsigned long   m, n;           // Variaveis usadas para navegar na matriz de pixels
    long            i, j;           // Variaveis usadas para navegar nas matrizes coocorrência

    unsigned long   AuxBusiness;
    double          AuxProbCond;

    BYTE            *bpBits;
    DWORD           dwWidth;
    BYTE            Limiar, Pixel1, Pixel2;
    BYTE            LimiarBusiness;     // Medida Business
    BYTE            LimiarProbCond;     // Medida da Prob Condicional

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

    CopyImageInOut();

    ClockStart();

    bpBits      = piRW_ImgIn ->GetImageBits();
    dwWidth  = piRW_ImgIn ->GetImageWidthBytes();

    // Inicialização das matrizes Coocorrência
    for (i = 0; i < 256; i++)
        for (j = 0; j < 256; j++)
            c[i][j] = 0;

    //  Direções:
    //      0 - Horizontal à direita
    //      1 - Vertical acima
    //      2 - Horizontal à esquerda
    //      3 - Vertical abaixo

    for (m = 0; m < m_Height ; ++m)
    {
        for (n = 0; n < m_Width; ++n)
        {
            Pixel1 = bpBits[(n       )+((m       )*dwWidth)];

            // Para o caso de ser direcao 0
            if(n < m_Width - Distancia)
            {
                Pixel2 = bpBits[(n+Distancia)+((m       )*dwWidth)];
                c[Pixel1][Pixel2]++;
            }

            // Para o caso de ser direcao 2
```

```
            if(n >= Distancia)
            {
                Pixel2 = bpBits[(n-Distancia)+((m          )*dwWidth)];
                c[Pixel1][Pixel2]++;
            }

            // Para o caso de ser direcao 3
            if(m >= Distancia)
            {
                Pixel2 = bpBits[(n         )+((m-Distancia)*dwWidth)];
                c[Pixel1][Pixel2]++;
            }

        }
    }

    AuxBusiness       = 0;
    AuxProbCond     = 0;
    for (int t = 0; t < 256; ++t)
    {
        // Cálculo de B1
        B1 = 0;
        for (i = 0; i < t ; ++i)
        {
            for (j = 0; j < t ; ++j)
            {
                B1 += W * c[i][j];
            }
        }

        // Cálculo de B2
        B2 = 0;
        for (i = t; i < 256 ; ++i)
        {
            for (j = t; j < 256 ; ++j)
            {
                B2 += W * c[i][j];
            }
        }

        // Cálculo de B3
        B3 = 0;
        for (i = 0; i < t ; ++i)
        {
            for (j = t; j < 256 ; ++j)
            {
                B3 += W * c[i][j];
            }
        }

        // Cálculo de B4
        B4 = 0;
        for (i = t; i < 256 ; ++i)
        {
            for (j = 0; j < t ; ++j)
            {
```

```
                B4 += W * c[i][j];
            }
        }

        // Calculando pela Medida Business
        if ( ((B3 + B4) != 0) )
        {
            if ( (AuxBusiness > (B3 + B4)) || (AuxBusiness == 0) )
//          if ( (AuxBusiness < (B3 + B4)) )
            {
                AuxBusiness       = B3 + B4;
                LimiarBusiness    = t;
            }
        }

        // Calculando pela Medida da Probabilidade Condicional
        if( ((B1 + B3) != 0) && ((B2 + B4) != 0) )
        {
            if ( (AuxProbCond > (((double)B3 / (B1 + B3)) + ((double)B4 / (B2 + B4)))) || (AuxProbCond == 0) )
//          if ( (AuxProbCond < (((double)B3 / (B1 + B3)) + ((double)B4 / (B2 + B4)))) )
            {
                AuxProbCond          = (((double)B3 / (B1 + B3)) + ((double)B4 / (B2 + B4)));
                LimiarProbCond = t;
            }
        }
    }

    if (GetOpcao_MatrizCoOcorrencia() == 0)
        Limiar = LimiarBusiness;

    if (GetOpcao_MatrizCoOcorrencia() == 1)
        Limiar = LimiarProbCond;

    SetLimiar(Limiar);

    AplicarLimiar();

    ClockFinish("Limiarizacao por Matriz de Co-Ocorrência");

    return TRUE;
}
```