IMPLEMENTAÇÃO DE ALGORITMOS DE VISÃO COMPUTACIONAL NUMA REDE DE "TRANSPUTERS"

Vania Vieira Estrêla*

Osamu Saotome**

RESUMO.

Este artigo descreve uma implementação paralela de uma técnica de visão por computador conhecida como transformada de Hough, destinada à detecção de formas em imagens digitalizadas. Os algoritmos desta classe estão sendo desenvolvidos em máquinas baseadas em "transputers".

ABSTRACT.

This paper describes a parallel implementation of the Hough transform. It is a technique for detecting shapes in digitized images. This class of algorithms is being developed in machines based on transputers.

---

* Aluna de mestrado do IEEA/ITA; Engenheira Eletrônica pela UFRJ; áreas de interesse: visão computacional, processamento de imagens, arquiteturas de computadores e computação gráfica.

** D.Sc. pelo Tokyo Institute of Technology; Professor Adjunto do IEEA/ITA; áreas de interesse: processamento de sinais digitais, processamento de imagens e arquiteturas de computadores.

CTA - ITA - IEEA
12225 - São José dos Campos - SP
Tel.: (0123)412211/Ramais 134 e 330.

1 - INTRODUÇÃO.

Este trabalho se concentra numa técnica de visão por computador conhecida como Transformada de Hough, que é vista no item 2. As máquinas alvo, que são usadas para efeito de avaliação de desempenho dos algoritmos em estudo, baseiam-se em "transputers" e as características que motivaram tal escolha são mostradas no item 3. No item 4 são analisados os aspectos referentes às implementações paralelas propostas.

2. A TRANSFORMADA DE HOUGH.

2.1. Considerações Iniciais.

O estudo de visão computacional requer a identificação das características presentes na imagem e a representação simbólica das mesmas. Os principais problemas envolvidos são a compreensão do processo de visão e a sua posterior implementação num computador. Uma das questões mais importantes relacionadas com a visão por computador é o processamento em tempo real. Sistemas capazes de operar em tempo real devem reconhecer padrões com suficiente rapidez para controlar processos industriais. Computadores de propósito geral são, em se tratando da resolução de problemas complexos, lentos e isto se deve ao esforço computacional necessário para o reconhecimento.

A maior parte da informação pertinente a um objeto reside no seu contorno. Inicialmente, a imagem a ser processada deve ser codificada de modo a conter apenas informações sobre os

contornos dos objetos nela presentes. A imagem codificada pode conter somente os módulos das mudanças locais de intensidade ou o módulo e a direção do vetor gradiente em cada ponto, indicando, neste último caso, a severidade e a orientação para uma mudança local de nível de cinza.

2.2. Detecção de Linhas Retas.

Hough desenvolveu um método que substitui o problema da busca de pontos colineares por um outro, matematicamente equivalente, de busca de linhas concorrentes [1]. Especificamente, os procedimentos considerados destinam-se à detecção de conjuntos colineares de pixels pertencentes a uma borda, através do mapeamento dos mesmos num espaço de parâmetros (espaço de Hough). Um conjunto de pixels colineares dá origem a um pico no espaço de Hough. Os pixels de borda, que serão processados pela transformada, são obtidos aplicando-se um operador de bordas à imagem, como por exemplo o de Sobel. Este operador calcula as aproximações digitais para a magnitude e a direção do gradiente dos níveis de cinza de cada pixel da imagem. Seja o pixel P, cujos vizinhos possuem os níveis de cinza dados a seguir:

A	B	C
D	P	E
F	G	H

Em primeiro lugar, os valores das diferenças de Sobel devem ser calculados [2], a partir das equações a seguir:

$$\Delta x \equiv \frac{1}{4} [(C + 2E + H) - (A + 2D + F)] \quad (2.1)$$

$$\Delta y \equiv \frac{1}{4} [(A + 2B + C) - (F + 2G + H)] \quad (2.2)$$

A magnitude (m) e a direção (θ) são definidas como:

$$m \equiv \sqrt{\Delta^2 x + \Delta^2 y} \quad (2.3) \qquad \theta \equiv tg^{-1} (\Delta y / \Delta x) \quad (2.4)$$

P será considerado um pixel de borda se o valor de m em P exceder um limiar pré-estabelecido.

Uma linha reta pode ser caracterizada pela sua inclinação α e comprimento ρ da perpendicular à mesma e que passa pela origem (figura 2.1). Se o ponto (x,y) pertence à imagem, tem-se que:

$$\rho = x.\text{sen}\alpha - y.\cos\alpha \quad (2.5)$$

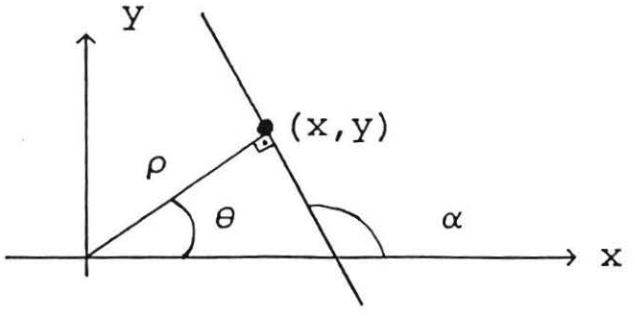

Figura 2.1 - Representação paramétrica de uma linha reta.

Um pixel pertencente a esta borda e com coordenadas (x,y) deve ter o gradiente orientado segundo uma direção θ, perpendicular a α, ou seja, $\alpha = \theta \pm \pi/2$. Reescrevendo-se a equação de ρ em termos de θ tem-se:

$$\rho = x.\cos\theta + y.\mathrm{sen}\theta \qquad (2.6)$$

O processo para o cálculo da transformada de Hough, no caso de linhas retas, pode ser resumido conforme segue:

a) Para cada pixel (x,y), calcula-se m = m(x,y) e $\theta = \theta(x,y)$; se m(x,y) $\geq$ T, onde T é um dado limiar, então o pixel (x,y) pertence ao contorno;

b) Toma-se o intervalo $[\theta(x,y) - \theta_o, \theta(x,y) + \theta_o]$ para cada pixel (x,y) pertencente a um contorno, onde θ_o é uma constante dada e calcula-se $\rho = x\cos\theta + y\mathrm{sen}\theta$; a correspondente posição (ρ,θ) da matriz acumuladora é incrementada; logo, após a varredura de toda a imagem, cada posição da mesma conterá o número de vezes que um par (ρ,θ) foi obtido como resultado do processamento dos pixels de um contorno.

2.3. Detecção de Curvas Analíticas.

Serão consideradas curvas analíticas aquelas que podem ser expressas sob a forma $\mathbf{f(x,a)} = 0$, onde **x** é um ponto da imagem e **a** é um vetor de parâmetros. Seja o problema da detecção de bordas circulares numa determinada imagem. A equação de um

círculo em coordenadas cartesianas é dada por:

$$(x - a)^2 + (y - b)^2 = r^2 \quad (2.7)$$

Seja uma imagem codificada, onde apenas os módulos das mudanças locais de intensidade são conhecidos. Caso um pixel pertença a um círculo, o lugar geométrico dos parâmetros que o descrevem é um cone circular (figura 2.2), que pode ser obtido fixando-se x e y e variando-se a, b e r. Se um conjunto de pixels pertencente a um contorno reside sobre um círculo, cujos parâmetros são a_o, b_o e r_o, os lugares geométricos dos parâmetros de cada par (x,y) se interceptarão num mesmo ponto do espaço paramétrico. Isto equivale à intersecção dos correspondentes cones circulares num ponto (a_o,b_o,r_o), dentro do referido espaço.

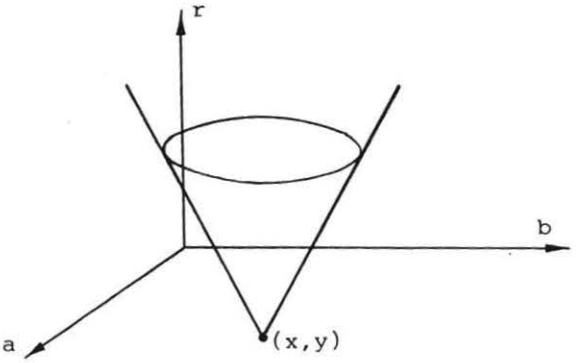

Figura 2.2 - Lugar geométrico dos parâmetros de um ponto.

2.3.1 - Efeito da Informação sobre as Direções dos Contornos.

Caso alguma informação a respeito da direção do contorno seja associada ao mesmo, o lugar geométrico dos parâmetros será reduzido a uma linha (figura 2.3) [1]. Formalmente, o círculo

requer três parâmetros, mas se a equação que o descreve for utilizada em conjunto com a sua derivada, o cálculo será simplificado, de modo que:

$$\frac{df(x,a)}{dx} = 0 \qquad (2.8)$$

Esta operação introduz um termo em dy/dx, dado por:

$$\frac{dy}{dx} = tg\ [\phi(x) - \pi/2] \qquad (2.9)$$

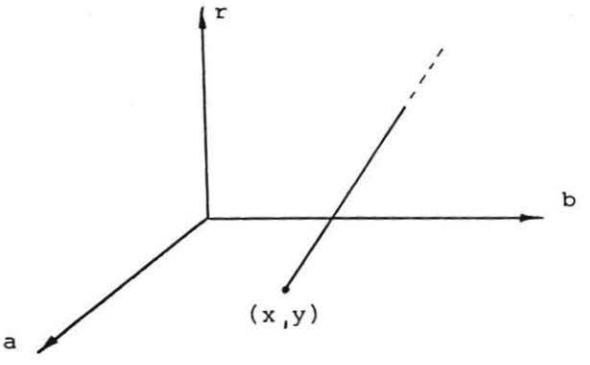

Figura 2.3 - Lugar geométrico dos parâmetros de um ponto com informação sobre a direção do contorno.

Onde $\phi(x)$ é a direção do gradiente.

2.4. Análise do Esforço Computacional.

Usando-se apenas a equação $f(x,a) = 0$, o número de cálculos necessários crescerá exponencialmente com o número de parâmetros menos 1. Seja m o número de parâmetros, os quais podem assumir M valores. Logo, a quantidade de cálculos é proporcional a M^{m-1},

já que a equação da curva é utilizada para determinar o último parâmetro. O uso do vetor gradiente resulta numa diminuição do nûmero de cálculos necessários. O esforço computacional é proporcional a M^{m-2}, para $m \geq 2$.

O método pode ser estendido a curvas e formas arbitrárias, bastando escolher um conjunto de parâmetros conveniente. Por exemplo, a altura, a largura e as coordenadas do centro de um retângulo podem ser usadas para caracterizá-lo. Quanto mais sofisticadas forem as formas a serem detectadas, maior será o número de parâmetros a serem levados em conta. O processamento desta transformada em tempo real requer uma grande quantidade de memória e esforço computacional considerável.

3. IMPLEMENTAÇÃO DA TRANSFORMADA DE HOUGH NUMA REDE DE "TRANSPUTERS".

O uso de uma arquitetura convencional consome muito tempo de processamento e memória. Uma alternativa eficiente consiste na utilização de paralelismo. Embora a maior parte das máquinas paralelas existentes para processamento de imagens opere no modo SIMD, o modo MIMD permite a implementação de programas mais complexos e melhor estruturados [2,3].

O "transputer" tem a oferecer várias vantagens em relação aos processadores convencionais, tais como: maior velocidade, redução do número de componentes presentes no sistema e facilidade de programação [4,5]. O desempenho pode ser aumentado

colocando-se um "transputer" (ou uma rede deles) dedicado a uma função particular, dentro de um dado programa. Este é, em geral, o melhor método para assegurar que a resposta em tempo real será satisfatória.

O paralelismo apresentado por estes componentes é melhor aproveitado quando é alocada uma certa quantidade de memória a cada um deles e as mensagens são trocadas via "links" seriais, o que caracteriza uma arquitetura MIMD com acoplamento fraco. Uma vez que um programa é definido como sendo um conjunto de processos, o mesmo pode ser mapeado numa máquina MIMD de várias formas: minimizando-se o custo, otimizando-se o desempenho, melhorando-se a resposta a determinados eventos,etc.

4. ASPECTOS RELATIVOS À PARALELIZAÇÃO DO SOFTWARE.

4.1. Alternativas.

Os algoritmos de processamento de imagem podem ser paralelizados de duas maneiras: a nível de imagem e a nível de tarefa [6]. Como o método em questão envolve um número pequeno de cálculos (poucas tarefas) e uma grande quantidade de pixels, a segunda alternativa é a mais conveniente para a implementação da transformada de Hough. Os espaços de imagem e de Hough devem ser distribuídos pelos nós de modo a extrair o máximo de paralelismo possível. Existem três possibilidades, a saber:

a) **Imagem completa/ partição do espaço de Hough.**

Cada processador acessa uma cópia local da imagem completa

e uma partição local do espaço de Hough (figura 4.1). A quantidade de memória necessária é de moderada a alta; há um desbalanceamento de carga (alguns nós estarão sobrecarregados e outros ociosos).

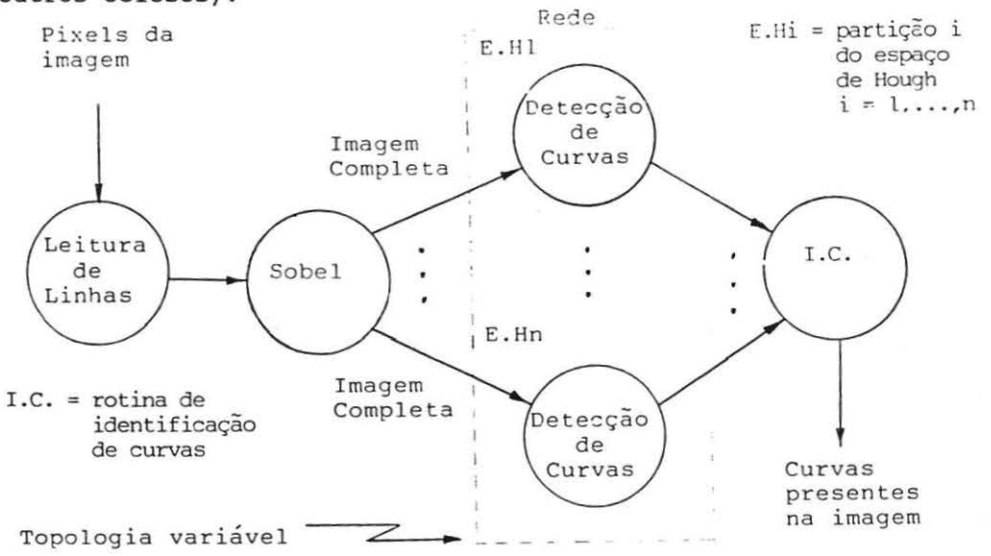

Figura 4.1 - Organização dos processos para a situação imagem completa/partição do espaço de Hough.

b) **Partição da imagem/ espaço de Hough completo.**

Cada processador acessa uma partição local da imagem e incrementa células na sua matriz acumuladora (espaço de Hough local). Uma vez que o espaço de Hough pode ser uma estrutura de dados muito grande, é necessária bastante memória local, especialmente para formas parametrizadas. Os resultados gerados são parciais (figura 4.2). O cálculo completo pode ser obtido de duas formas: os espaços de Hough são difundidos ao longo da rede de "transputers" e acumulados em cada nó ou as partições da imagem são trocadas até que cada nó possua uma cópia completa da

matriz acumuladora. A primeira opção despende muito tempo, uma vez que os espaços de Hough serão transmitidos serialmente e os mesmos são, em realidade, matrizes muito grandes. Já a segunda é menos onerosa, sob o ponto de vista de entrada e saída.

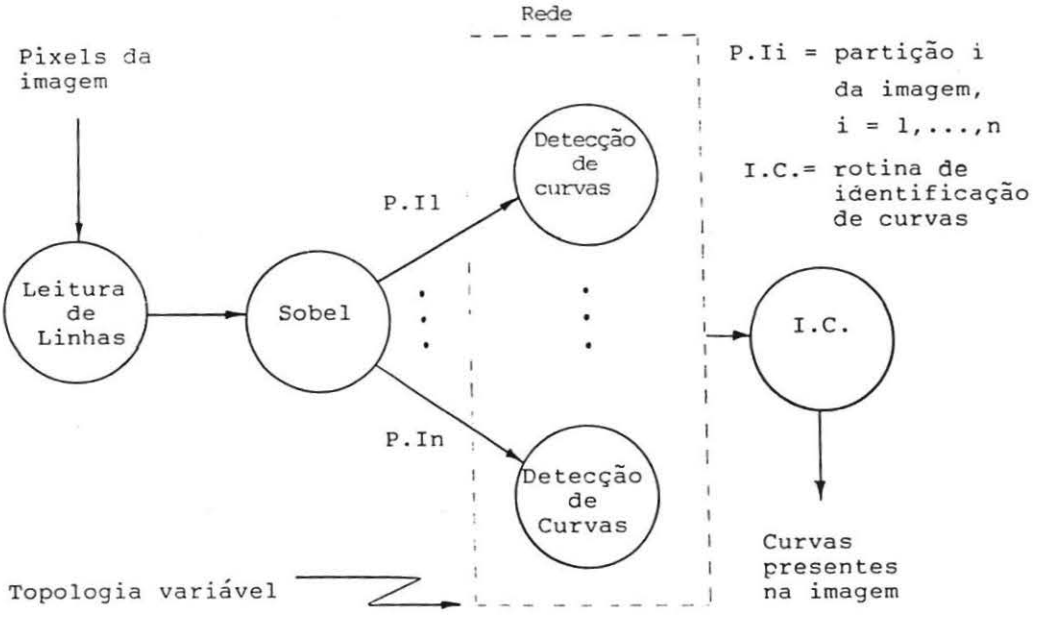

Figura 4.2 - Organização dos processos para a situação partição da imagem/espaço de Hough completo.

3) Partição da imagem/ partição do espaço de Hough.

Cada processador varre a sua partição da imagem e atualiza a sua partição do espaço de Hough (figura 4.3). Estes resultados são parciais e o cálculo total pode ser obtido difundindo-se as partições da imagem ou do espaço de Hough. A primeira alternativa é preferível e pode ser melhorada se for efetuado um pré-processamento, visando a obtenção dos pixels de borda, que serão trocados entre os nós. A carga é desbalanceada, mas pode ser equilibrada se pixels pré-processados forem eqüitativamente

distribuídos pelos nós.

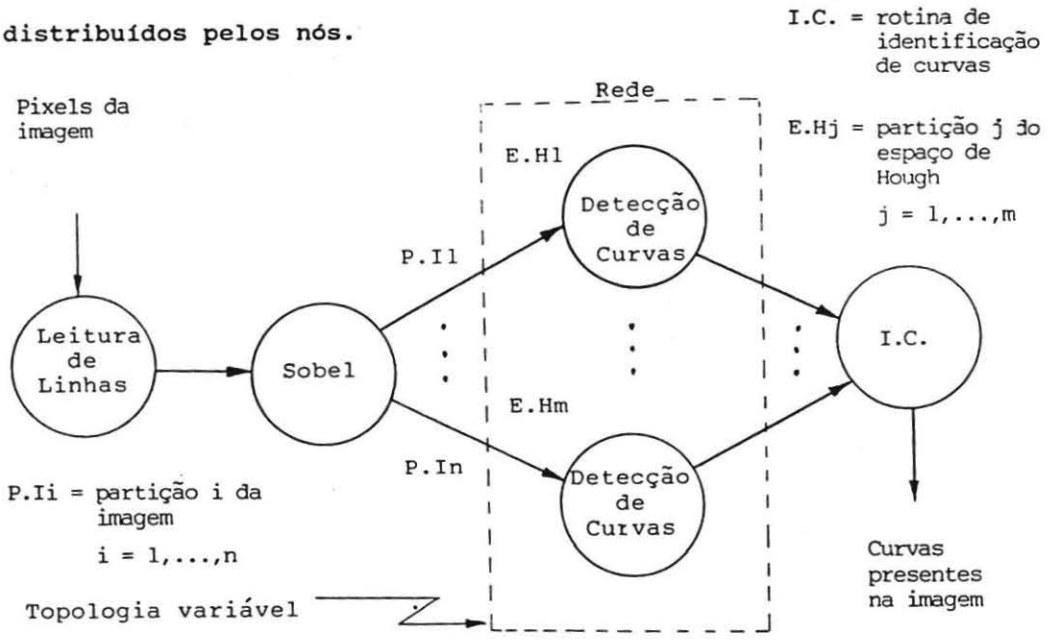

Figura 4.3 - Organização dos processos para a situação partição da imagem/ partição do espaço de Hough.

Das três possibilidades citadas anteriormente, a primeira e a terceira são as mais viáveis, devido às quantidades de memória local envolvidas.

4.2. Descrição do Programa.

Dois programas, escritos nas linguagens "C" e OCCAM2, estão sendo desenvolvidos, no Laboratório de Computação Paralela da COPPE/Sistemas/UFRJ. Este possui placas com um e quatro "transputers", além da máquina paralela NCP I, atualmente funcionando com oito nós. Todos os equipamentos são baseados no

componente T800. Os programas têm como objetivo uma avaliação preliminar das possibilidades existentes para a implementação da transformada de Hough mencionadas no item 4.1. Também serão estudadas diversas topologias para a rede de "transputers". O processamento numérico é realizado pelo programa escrito em OCCAM2, o qual será carregado na rede de transputers. Já a interface gráfica que será executada no computador hospedeiro (IBM-PC) foi programada em "C". A comunicação entre os dois é feita através de um programa servidor.

5. CONCLUSÕES.

O programa escrito em "C" e as rotinas do programa desenvolvido em OCCAM2 que não requerem comunicação estão concluídos. As rotinas que efetuam a comunicação estão sendo depuradas.

O maior problema apresentado pela transformada de Hough é a determinação dos máximos da matriz acumuladora. Há muitas fontes de erro que podem afetar o vetor de parâmetros a, de modo que vários pontos na vizinhança do ponto ideal sejam incrementados. Pode-se resolver este problema por meio de um modelamento formal dos erros na etapa de incrementação, especificando-se um conjunto de pontos próximos, ao invés de apenas um ponto. Uma outra solução é tentar compensar os valores contidos na matriz acumuladora com uma função, após concluída a etapa de incrementação. Estes métodos são equivalentes, supondo-se erros isotrópicos.

6. AGRADECIMENTOS.

Este trabalho está sendo desenvolvido no Núcleo de Computação Paralela da COPPE/Sistemas e no Laboratório de Instrumentação Nuclear da COPPE/Nuclear, ambos pertencentes à UFRJ. Agradecemos à CAPES pelo apoio financeiro.

7. BIBLIOGRAFIA.

[1] Ballard, D.H. - "Generalizing the Hough Transform to Detect Arbitrary Shapes", Pattern Recognition, Vol. 13, N° 2, 1981, pp. 111-122.

[2] Estrêla, V.V. & Saotome, O. - "Arquitetura Paralela para Visão por Computador", Anais do 4° Congresso Nacional de Automação Industrial, Julho, 1990, pp. 289-295.

[3] Maresca, M. & Lavin, M.A. - "Parallel Architectures for Vision", Proceedings of the IEEE/Special Issue on Computer Graphics, Vol. 76, N° 8, August, 1988, pp. 970-981.

[4] Stein, R. M. - "T800 and Counting", BYTE, November, 1988, pp. 287-296.

[5] Burns, Alan - "Programming in OCCAM 2", Addison-Wesley Publishing Company, Great Britain, 1988, pp. 104-105.

[6] Bowyer, K. & Lake, C. - "Computing the Hough Transform on

MIMD Parallel Architectures (extended abstract)", University of South Florida, 1989, pp. 1-11.