

# Sistema de Detecção de Objetos Circulares *através da Transformada de Hough Randômica*

## 1. *Introdução*

---

Esta monografia tem por objetivo apresentar toda a fundamentação teórica para a implementação de um sistema de localização de objetos circulares em uma imagem através da Transformada de Hough Randômica, como exigência da disciplina de Projeto Final.

No capítulo 2 é apresentado o conceito teórico e o formalismo matemático da Transformada de Hough clássica para detecção de círculos. O capítulo 3 traz a fundamentação teórica, bem como o algoritmo da Transformada de Hough Randômica (RHT) adaptado para a detecção de objetos circulares. No capítulo 4 é apresentado todo o formalismo matemático para a detecção de círculos utilizando a RHT.

É importante ressaltar que o objetivo principal deste projeto é implementar um sistema capaz de detectar círculos em imagens reais e artificiais, utilizando-se a Transformada de Hough Randômica (RHT). O estudo referente à detecção de elipses, será apresentado em anexo, apenas a nível teórico.

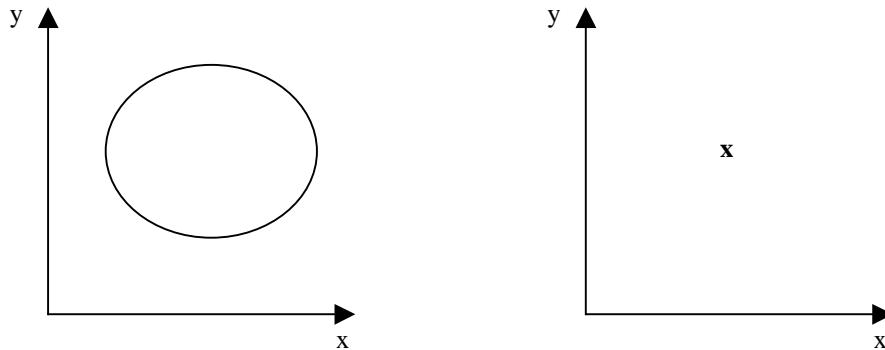
O capítulo 5 apresenta os testes realizados com diferentes tipos de imagens, bem como os resultados, utilizando a implementação feita. O capítulo 6 traz as conclusões do projeto. Finalizando, são apresentadas as Referências Bibliográficas. Em Anexo, é apresentada uma introdução à teoria da Transformada de Hough Clássica para detecção de Elipses.

## 2. A Transformada de Hough Clássica

---

Um problema freqüentemente encontrado na extração de primitivas em uma imagem é a detecção de curvas analíticas do tipo segmentos de reta, círculos, elipses, etc. A Transformada de Hough é um método matemático utilizado para detecção de bordas, aplicável quando se possui informações precisas acerca da forma da curva a ser detectada.

O princípio básico da Transformada de Hough consiste em obter, através das transformações do gradiente e da limiarização, pontos de uma imagem. A idéia é aplicar na imagem uma transformação tal que todos os pontos pertencentes a uma mesma curva sejam mapeados num único ponto de um novo espaço de parametrização da curva procurada. A Figura 1 ilustra esta técnica:



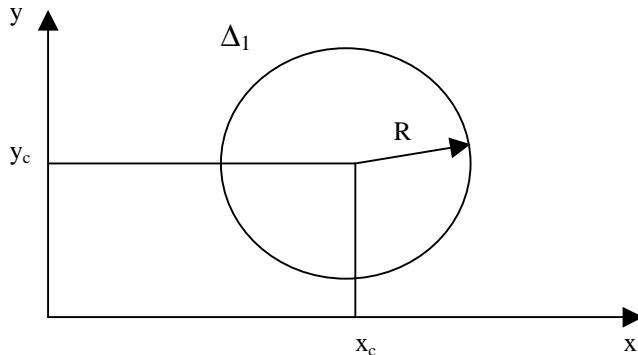
**Figura 1** – Uma curva qualquer parametrizada no espaço  $x - y$ , sofre a Transformada de Hough e passa a ser mapeada em um único ponto.

A Transformada de Hough tem como vantagem o fato de que pode ser aplicada ao tratamento de qualquer tipo de curva e, além disso, apresenta muita eficiência em imagens fortemente ruidosas.

No entanto, tem como desvantagem o fato de que se uma curva é mais complexa, ou seja, com um número maior de parâmetros, o esforço computacional exigido é bem maior.

## 2.1. A Transformada de Hough Clássica para Detecção de Círculos

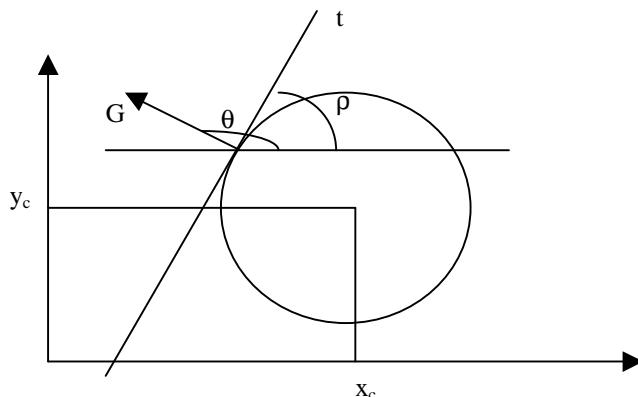
Considere o círculo  $\Delta_1$  a seguir, de raio  $R$  e centro de coordenadas  $(x_c, y_c)$ .



**Figura 2** – Um círculo  $\Delta_1$  perfeito de raio  $R$  e centro de coordenadas  $(x_c, y_c)$ .

Traçando uma tangente  $t$  ao círculo  $\Delta_1$  e um vetor Gradiente  $G$  perpendicular à  $t$ ,

temos:



**Figura 3** – São traçados ao círculo  $\Delta_1$  uma tangente  $t$  e um vetor Gradiente  $G$ , perpendicular à  $t$ .

Onde  $\theta$  é o ângulo formado entre o vetor Gradiente  $G$  e o eixo horizontal  $x$  e  $\rho$  é o ângulo formado entre a tangente  $t$  e o eixo horizontal  $x$ .

Da geometria elementar temos que a equação paramétrica do círculo é:

$$(x - x_c)^2 + (y - y_c)^2 = R^2 \quad (1)$$

Aplicando a derivada de primeira ordem, relativa ao Gradiente, em (1), chegamos a:

$$2(x - x_c) + 2(y - y_c) \frac{dy}{dx} = 0 \quad (2)$$

onde temos que:

$$\frac{dy}{dx} = \tan \rho \quad \text{e}$$

$$\theta = \pi / 2 + \rho, \text{ ou seja, } \rho = \theta - \pi / 2$$

Substituindo-se, temos que:

$$\frac{dy}{dx} = \tan (\theta - \pi / 2)$$

$$\frac{dy}{dx} = -1 / \tan \theta \quad (3)$$

Substituindo (3) em (2), obtemos:

$$2(x - x_c) + 2(y - y_c) \cdot -\frac{1}{\tan \theta} = 0$$

Simplificando:

$$(x - x_c) - \frac{(y - y_c)}{\tan \theta} = 0$$

Logo:

$$(x - x_c) = \frac{(y - y_c)}{\tan \theta} \longrightarrow (y - y_c) = (x - x_c) \tan \theta \quad (4)$$

Substituindo a equação (4) em (1), temos:

$$(x - x_c)^2 + (y - y_c)^2 \tan^2 \theta = R^2$$

Isolando  $(x - x_c)^2$ , chegamos a:

$$(x - x_c)^2 \cdot [1 + \tan^2 \theta] = R^2 \quad (5)$$

Como  $\tan^2 \theta = \sin^2 \theta / \cos^2 \theta$ , podemos substituir em (5), ficando com:

$$\begin{aligned} (x - x_c)^2 \left[ 1 + \frac{\sin^2 \theta}{\cos^2 \theta} \right] &= R^2 \\ (x - x_c)^2 \left[ \frac{\cos^2 \theta}{\cos^2 \theta} \cdot 1 + \frac{\sin^2 \theta}{\cos^2 \theta} \right] &= R^2 \\ (x - x_c)^2 \left[ \frac{\cos^2 \theta + \sin^2 \theta}{\cos^2 \theta} \right] &= R^2 \end{aligned}$$

Como  $\cos^2\theta + \sin^2\theta = 1$ , ficamos com:

$$(x - x_c)^2 \left[ \frac{1}{\cos^2 q} \right] = R^2$$

$$(x - x_c)^2 = R^2 \cdot \cos^2 q$$

Simplificando:

$$(x - x_c) = R \cdot \cos q \quad (6)$$

Substituindo (6) em (4), encontramos:

$$(y - y_c) = R \cdot \cos q \cdot \tan q$$

Como  $\tan \theta = \sin \theta / \cos \theta$ , concluímos que:

$$(y - y_c) = R \cdot \sin q \quad (7)$$

Onde  $x$  e  $y$  definem as coordenadas do ponto no novo espaço de parametrização do círculo,  $x_c$  e  $y_c$  definem as coordenadas do centro e  $R$  é o raio do mesmo.

### **3. A Transformada de Hough Randômica (RHT)**

---

#### **3.1. As Bases do Algoritmo RHT**

Conforme visto na seção 2, sobre a HT Clássica, o princípio da Transformada de Hough consiste em obter, a partir dos parâmetros da curva a ser detectada, um novo espaço de parametrização, ou seja, acumular, em um único ponto, todos os pontos de borda da curva.

A Transformada de Hough Randômica (RHT) tem como princípio básico obter, a partir dos pontos de borda de uma imagem binária, um pequeno subconjunto de pontos selecionados randomicamente, a partir dos quais se fará o mapeamento para o novo espaço de parametrização da curva procurada.

O método RHT está baseado no fato de que o ponto único no novo espaço de parametrização pode ser determinado a partir de um pequeno subconjunto de pontos da borda da imagem original. O tamanho deste subconjunto depende exclusivamente da complexidade das curvas a serem detectadas. Por exemplo, no caso de detecção de retas, o ponto de acumulação no novo espaço de parametrização pode ser obtido a partir de dois pontos de borda da imagem binária original. Tais pares de pontos ( $d_i, d_j$ ) são escolhidos randomicamente. Em seguida, o ponto ( $d_i, d_j$ ) é usado no cálculo da equação de reta, e a célula A ( $d_i, d_j$ ) é armazenada em uma tabela de acumulação. A RHT é executada para se descobrir os valores na tabela de acumulação que são superiores ou iguais a um limiar  $t$ , considerado mínimo.

Para o caso dos círculos, que tem uma parametrização bem mais complexa que as retas, dois pontos são insuficientes para o cálculo da RHT. Sendo assim, três ou quatro pontos serão selecionados randomicamente a partir da borda da imagem binária original. Maiores detalhes são apresentados nas seções seguintes.

A tabela de acumulação é uma estrutura utilizada para armazenar as coordenadas dos pontos obtidos após o cálculo da Transformada de Hough. Para o caso de detecção de círculos, a tabela contém três colunas, conforme ilustra a Tabela 1:

**Tabela 1 – Exemplo de Tabela de Acumulação**

<b>x<sub>c</sub></b>	<b>y<sub>c</sub></b>	<b>Counter</b>

Onde a coluna **x<sub>c</sub>** indica a coordenada x do centro, a coluna **y<sub>c</sub>** a coordenada y do centro e a coluna ‘**Counter**’ indica o número de pontos acumulados sobre o centro indicado nas duas colunas anteriores.

Em relação à Transformada de Hough Clássica para círculos, a RHT tem como principal diferença o fato de que apenas um pequeno subconjunto de pontos da borda da imagem é necessário para se fazer o mapeamento do novo espaço de parametrização da curva procurada, enquanto que na HT Clássica todos os pontos da borda são utilizados. A segunda grande diferença é que a RHT não utiliza o cálculo da reta tangente às curvas, bem como o cálculo do vetor Gradiente.

Essas diferenças acarretam em um tempo de processamento bem menor da RHT em relação à HT Clássica, otimizando, desta maneira a demanda de memória computacional, visto que o formalismo matemático a ser utilizado é mais simples.

### 3.2. O Kernel do Algoritmo RHT para Detecção de Círculos

A geometria elementar nos mostra que a equação paramétrica do círculo é:

$$(x - x_c)^2 + (y - y_c)^2 = R^2$$

Sendo assim, a partir de uma imagem binária obtida como input, o algoritmo RHT para detecção de círculos descreve-se da seguinte maneira:

1. Forme o conjunto D com todos os pontos de borda da imagem de input.

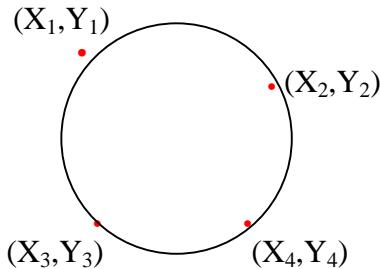
2. Selecione randomicamente 3 ou 4 pontos ( $d_i, d_j$ ) do conjunto D.
3. Se os pontos não satisfazem uma distância mínima, vá ao Passo 2; caso contrário, continue do passo 4.
4. Calcule o ponto do novo espaço de parametrização ( $x, y$ ) usando a equação paramétrica do círculo utilizando os pontos ( $d_i, d_j$ ).
5. Armazene a célula A ( $x_c, y_c$ ) na tabela de acumulação.
6. Se a célula A ( $x_c, y_c$ ) é superior ou igual ao limiar  $t$ , os parâmetros x e y descrevem os parâmetros da curva detectada. Caso contrário, continue do passo 2.
7. Calcule a distância do ponto ( $x, y$ ) até o centro ( $x_c, y_c$ ) para obter o raio do círculo.

Para definir a distância limite no passo 3, os pontos  $d_i$  e  $d_j$  não devem estar muito próximos um do outro, o que chamaremos de critério de distância de ponto. Este critério deve ser utilizado basicamente para evitar perda de precisão na localização do centro do círculo, conforme a técnica apresentada na seção 4 a seguir.

## 4. Detecção de Círculos com a Transformada de Hough Randômica

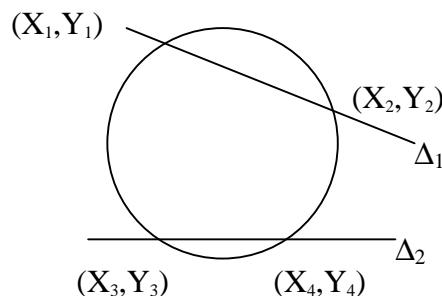
---

Cada círculo, em uma imagem binarizada, é parametrizado por dois valores ( $x_c, y_c$ ) que representam as coordenadas do seu centro. Assuma inicialmente que três ou quatro pixels pretos de coordenadas ( $x, y$ ) estão sobre a borda de um círculo, a uma distância mínima. A Figura 5 ilustra um exemplo (os pixels são representados em cor vermelha apenas para ilustração):



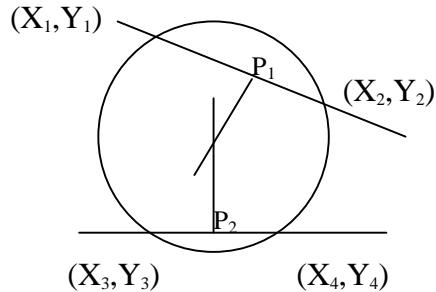
**Figura 4** – Exemplo de círculo com pontos detectados sobre suas bordas.

Para detecção do centro do círculo, inicialmente devemos traçar duas retas:  $\Delta_1$  entre os pontos  $(X_1, Y_1)$  e  $(X_2, Y_2)$  e  $\Delta_2$  entre os pontos  $(X_3, Y_3)$  e  $(X_4, Y_4)$ , conforme ilustra a figura 5:



**Figura 5** – Duas retas são traçadas:  $\Delta_1$  entre os pontos  $(X_1, Y_1)$  e  $(X_2, Y_2)$  e  $\Delta_2$  entre os pontos  $(X_3, Y_3)$  e  $(X_4, Y_4)$ .

Encontrando os pontos médios  $\mathbf{P}_1$  entre  $(X_1, Y_1)$ ,  $(X_2, Y_2)$  e  $\mathbf{P}_2$  entre  $(X_3, Y_3)$ ,  $(X_4, Y_4)$  e traçando duas retas perpendiculares a partir destes, temos que a intersecção das perpendiculares dá-se sobre o centro do círculo, conforme nos mostra a figura 6:



**Figura 6** – A intersecção das perpendiculares dá-se sobre o centro do círculo.

A descrição matemática da técnica apresentada anteriormente é mostrada a seguir. Usando a Forma da Reta Perpendicular para os pontos  $(X_1, Y_1)$  e  $(X_2, Y_2)$ , temos a seguinte equação:

$$\left( \sqrt{(x - x_1)^2 + (y - y_1)^2} \right)^2 = \left( \sqrt{(x - x_2)^2 + (y - y_2)^2} \right)^2 \quad (8)$$

Eliminando a raiz quadrada em (8), ficamos com:

$$(x - x_1)^2 + (y - y_1)^2 = (x - x_2)^2 + (y - y_2)^2 \quad (9)$$

Aplicando a fórmula algébrica em (9), temos:

$$x^2 - 2.x.x_1 + x_1^2 + y^2 - 2.y.y_1 + y_1^2 = x^2 - 2.x.x_2 + x_2^2 + y^2 - 2.y.y_2 + y_2^2 \quad (10)$$

Isolando as coordenadas X e Y em (10), ficamos com:

$$x [2.x_2 - 2.x_1] + y [2.y_2 - 2.y_1] + x_1^2 + y_1^2 - y_2^2 - x_2^2 = 0 \quad (11)$$

Para simplificação da equação (11), temos:

$$A_1 = 2.x_2 - 2.x_1$$

$$B_1 = 2.y_2 - 2.y_1$$

$$C_1 = x_1^2 + y_1^2 - y_2^2 - x_2^2$$

Obtendo, por fim, a primeira equação da reta perpendicular ao ponto médio  $\mathbf{P}_1$ :

$$A_1 \cdot x + B_1 \cdot y + C_1 = 0 \quad (12)$$

Refazendo os mesmos cálculos para os pontos  $(X_3, Y_3)$  e  $(X_4, Y_4)$ , temos:

$$\left( \sqrt{(x - x_3)^2 + (y - y_3)^2} \right)^2 = \left( \sqrt{(x - x_4)^2 + (y - y_4)^2} \right)^2 \quad (13)$$

Eliminando a raiz quadrada em (13), ficamos com:

$$(x - x_3)^2 + (y - y_3)^2 = (x - x_4)^2 + (y - y_4)^2 \quad (14)$$

Aplicando a fórmula algébrica em (14), temos:

$$x^2 - 2 \cdot x \cdot x_3 + x_3^2 + y^2 - 2 \cdot y \cdot y_3 + y_3^2 = x^2 - 2 \cdot x \cdot x_4 + x_4^2 + y^2 - 2 \cdot y \cdot y_4 + y_4^2 \quad (15)$$

Isolando as coordenadas x e y em (15), ficamos com:

$$x [2 \cdot x_4 - 2 \cdot x_3] + y [2 \cdot y_4 - 2 \cdot y_3] + x_3^2 + y_3^2 - y_4^2 - x_4^2 = 0 \quad (16)$$

Para simplificação da equação (16), temos:

$$A_1 = 2 \cdot x_4 - 2 \cdot x_3$$

$$B_1 = 2 \cdot y_4 - 2 \cdot y_3$$

$$C_1 = x_3^2 + y_3^2 - y_4^2 - x_4^2$$

Obtendo, assim, a segunda equação da reta perpendicular ao ponto médio  $\mathbf{P}_2$ :

$$A_2 \cdot x + B_2 \cdot y + C_2 = 0 \quad (17)$$

Tomando as duas equações das retas perpendiculares obtidas (12) e (17), temos o seguinte sistema de equações:

$$\begin{cases} A_1 \cdot x_c + B_1 \cdot y_c + C_1 = 0 & (18) \\ A_2 \cdot x_c + B_2 \cdot y_c + C_2 = 0 & (19) \end{cases}$$

Onde temos que:

$$A_1 \cdot x_c + B_1 \cdot y_c = -C_1 \quad (20)$$

$$A_2 \cdot x_c + B_2 \cdot y_c = -C_2 \quad (21)$$

Somando-se as equações (20) e (21), ficamos com:

$$(A_1 + A_2) \cdot x_c + (B_1 + B_2) \cdot y_c = -C_1 - C_2 \quad (22)$$

Isolando em  $x_c$  (22), temos:

$$(A_1 + A_2) \cdot x_c = -C_1 - C_2 - (B_1 + B_2) \cdot y_c$$

$$x_c = \frac{-C_1 - C_2 - (B_1 + B_2) \cdot y_c}{(A_1 + A_2)} \quad (23)$$

Substituindo (23) em (20), ficamos com:

$$\frac{A_1 \cdot (-C_1 - C_2 - (B_1 + B_2) \cdot y_c)}{(A_1 + A_2)} + B_1 \cdot y_c = -C_1 \quad (24)$$

Efetuando a soma em (24), temos:

$$\frac{A_1 \cdot (-C_1 - C_2 - (B_1 + B_2) \cdot y_c) + B_1 \cdot y_c \cdot (A_1 + A_2)}{(A_1 + A_2)} = -C_1 \quad (25)$$

Passando  $(A_1 + A_2)$  multiplicando  $-C_1$ , passamos a ter:

$$A_1 \cdot (-C_1 - C_2 - (B_1 + B_2) \cdot y_c) + B_1 \cdot y_c \cdot (A_1 + A_2) = -C_1 \cdot (A_1 + A_2) \quad (26)$$

Multiplicando os termos em (26), temos:

$$-C_1 \cdot A_1 - C_2 \cdot A_1 - A_1 \cdot y_c \cdot (B_1 + B_2) + B_1 \cdot y_c \cdot (A_1 + A_2) = -C_1 \cdot (A_1 + A_2) \quad (27)$$

Isolando  $y_c$  no lado esquerdo da equação e multiplicando o lado esquerdo em (27), ficamos com:

$$y_c \cdot [B_1 \cdot (A_1 + A_2) - A_1 \cdot (B_1 + B_2)] = A_1 \cdot C_1 + A_1 \cdot C_2 - A_1 \cdot C_1 - A_2 \cdot C_1$$

Onde finalmente chegamos ao valor final de  $y_c$ :

$$y_c = \frac{A_1 \cdot C_2 - A_2 \cdot C_1}{B_1 \cdot (A_1 + A_2) - A_1 \cdot (B_1 + B_2)} \quad (28)$$

Substituindo (28) em (23), obtemos a equação:

$$x_c = \frac{-C_1 - C_2 - (B_1 + B_2) \cdot \frac{A_1 \cdot C_2 - A_2 \cdot C_1}{B_1 \cdot (A_1 + A_2) - A_1 \cdot (B_1 + B_2)}}{(A_1 + A_2)} \quad (29)$$

Modificando a equação (29), ficamos com:

$$x_c = \frac{(-C_1 - C_2) - \frac{(B_1 + B_2)(A_1 C_2 - A_2 C_1)}{B_1(A_1 + A_2) - A_1(B_1 + B_2)}}{(A_1 + A_2)} \quad (30)$$

Continuando em (30), obtemos:

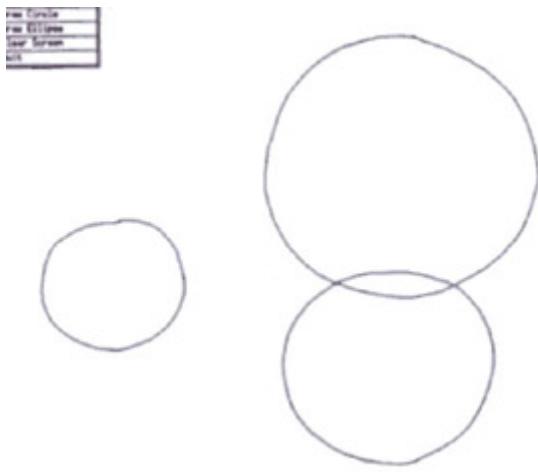
$$x_c = \frac{\frac{\{(-C_1 - C_2)[B_1(A_1 + A_2) - A_1(B_1 + B_2)]\} - [(B_1 + B_2)(A_1 C_2 - A_2 C_1)]}{[B_1(A_1 + A_2) - A_1(B_1 + B_2)]}}{(A_1 + A_2)} \quad (31)$$

Calculando a divisão pelo denominador em (31), obtemos, por fim, a equação de  $x_c$ :

$$x_c = \frac{\frac{\{(-C_1 - C_2)[B_1(A_1 + A_2) - A_1(B_1 + B_2)]\} - [(B_1 + B_2)(A_1 C_2 - A_2 C_1)]}{[B_1(A_1 + A_2) - A_1(B_1 + B_2)]}}{(A_1 + A_2)} \quad (32)$$

Onde  $x_c$  e  $y_c$  definem as coordenadas do centro do círculo. O formalismo matemático apresentado acima foi testado com sucesso para o caso de círculos perfeitos.

Para o caso de círculos imperfeitos, ou seja, com pequenas deformações, após a aplicação da técnica acima, haverá um acúmulo de pontos como prováveis centros de círculo, conforme nos ilustram as figuras 7 e 8 a seguir.



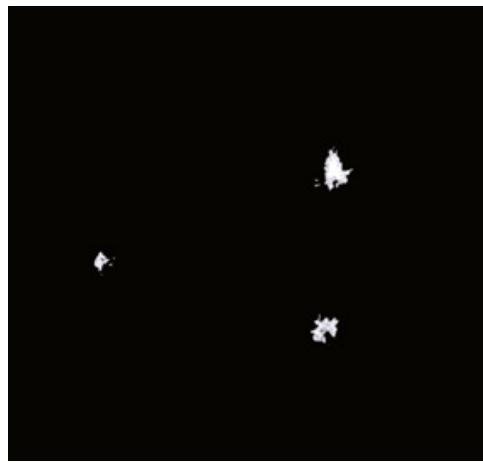
**Figura 7** – Imagem típica com círculos imperfeitos.



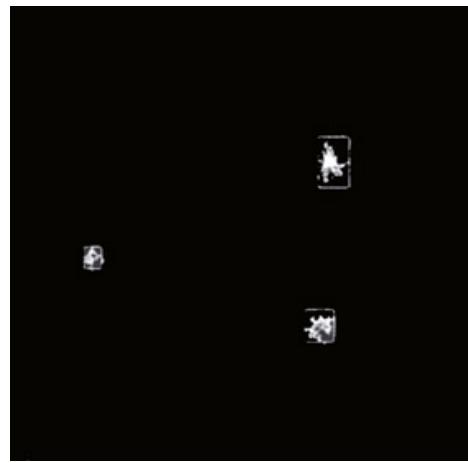
**Figura 8** – Transformada de Hough Randômica da imagem. Há um acúmulo de pontos, considerados prováveis centros de círculos.

A imagem 8, obtida após o cálculo da Transformada de Hough Randômica, apresenta alguns pontos de ruído, que podem ser eliminados através da limiarização. Primeiro, um valor médio é encontrado para homogeneizar a tabela de acumulação. Em seguida, os maiores valores da tabela são usados para se ajustar um valor limiar  $t$ . Todos os valores que estão localizados abaixo desse limiar (referentes aos pontos de ruído) são zerados, eliminando os pontos ruidosos (Figura 9).

Em seguida, cada uma das regiões não-zeradas são identificadas e uma “bounding box” é colocada ao redor de cada uma (Figura 10). Cada bounding box é assumida para conter um provável centro. Retomando os valores originais não-limiarizados, nós consideramos o centro da área dentro de cada bounding box e tomamos este como o centro do círculo.



**Figura 9** – Um valor médio é encontrado e a tabela de acumulação é limiarizada.



**Figura 10** – Uma “bounding box” é colocada ao redor de cada região não-zerada.

Encontrados os centros dos círculos na imagem, precisamos encontrar o raio. Para isto, basta obter, a partir da tabela de acumulação, todos os pontos acumulados que tiverem um valor maior ou igual ao limiar  $t$ . Considerando que todos os pontos que satisfazem esta condição fazem parte da borda da curva, basta calcular a distância entre esses pontos e o centro do círculo, que equivale ao raio do mesmo.

Para a detecção final de um objeto circular, cada ponto da borda de coordenadas  $(x, y)$  será armazenado em uma estrutura de dados com seu respectivo centro, de coordenadas  $(x_c, y_c)$ .

## 5. Implementação

---

### 5.1. Detecção de Bordas

A detecção de bordas, no sistema implementada com o nome de Morfologia, foi dividida em duas etapas:

- ✓ Erosão;
- ✓ Binarização;

O primeiro passo consiste em erodir a imagem original, ou seja, detectar os pontos de borda da imagem através da aplicação do elemento estruturante  $3 \times 3$  em formato de cruz, com o ponto central exatamente no meio da estrutura (representado com um 'X' vermelho na Figura 11). O elemento cruz é utilizado pelo fato de objetos circulares não apresentarem uma curvatura perfeita na tela do computador, conforme ilustra a Figura 11:

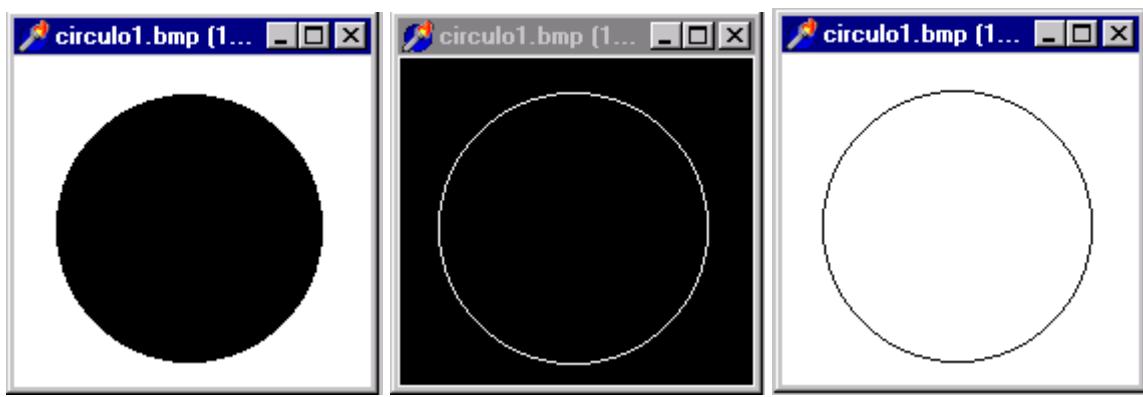


**Figura 11** – À esquerda está representado o elemento estruturante em cruz e à direita, a representação de um pedaço da borda binária de um objeto circular, visto no monitor de um computador. O elemento cruz é o que melhor se encaixa na borda.

Na erosão, todos os pontos da imagem serão varridos pelo elemento estruturante cruz, quando os pixels são “marcados”, ou seja, a imagem é invertida (pixels branco tornam-se pretos e vice-versa). A imagem erodida é subtraída da imagem original, resultando em uma imagem com a borda ressaltada (imagem central da Figura 12). A partir desta imagem, aplique-se o processo de binarização que consiste em obter-se todos os pixels diferentes de preto da imagem (em níveis de cinza), calculando-se o valor da cor média presente na imagem. Em

seguida, solicita-se ao usuário uma porcentagem abaixo da média desejada que será usada para encontrar a borda da imagem, da seguinte maneira:

1. Compara-se o valor de cada pixel com o valor da nova média, calculada após o valor da porcentagem fornecida;
2. Para valores maiores que a média, os pixels são desenhados, na imagem resultado, na cor preta. Caso contrário, são desenhados em branco (imagem à direita da Figura 12).



**Figura 12** – À esquerda é representada a imagem original. No centro, a imagem obtida após a subtração da imagem erodida a partir da imagem original e à direita a imagem resultante do processo de binarização, já com as bordas detectadas.

## 5.2. A Transformada de Hough Randômica (RHT)

O formalismo matemático apresentado na seção 4 foi implementado oferecendo-se ao usuário a opção de utilizar três ou quatro pontos para o cálculo da RHT. O primeiro ponto ( $X_1, Y_1$ ) é obtido randomicamente, a partir da imagem com as bordas detectadas. Os pontos seguintes são obtidos a partir deste e com base nos critérios de distâncias mínima e máxima, conforme descrito no algoritmo da seção 3.2. Os valores das distâncias, próximos de um ideal, são sugeridos ao usuário.

Em seguida, é feita a acumulação dos 3 ou 4 pontos de borda usados no cálculo da RHT, bem como os centros relevantes encontrados. Os valores serão armazenados em uma estrutura de dados, a partir da qual será criada a Tabela de Acumulação (visível ao usuário). A

Tabela é composta de três colunas: Xc, Yc e Contador, onde **Xc** e **Yc** representam as coordenadas cartesianas do provável centro do objeto circular e o **Contador** mostra o número de pixels encontrados referentes ao centro (Xc,Yc) correspondente. Para objetos circulares irregulares, ou seja, que não são círculos perfeitos (o que acontece em imagens do mundo real), haverá mais de um provável centro, que serão apresentados em mais linhas da Tabela de Acumulação.

A Tabela de Acumulação deverá sofrer um ajuste, visando acumular os pontos considerados relevantes para localização do possível centro do objeto circular. Este ajuste é feito usando uma “bounding box” de dimensão 5 x 5, onde o ponto central desta estrutura é o provável centro do objeto circular. O ponto central é o ponto de maior acumulação (valor do campo ‘Contador’) da Tabela de Acumulação. Os pontos localizados dentro deste limite da janela 5 x 5 serão ajustados para o centro desta.

Após o ajuste da Tabela de Acumulação, o usuário deverá fornecer um valor para um limiar que será utilizado para obterem-se os centros verdadeiramente relevantes dos prováveis objetos circulares, ou seja, valores abaixo desse limiar são desconsiderados (seriam prováveis ruídos presentes na imagem original). O valor do limiar deve ser fornecido com base nos valores do campo ‘Contador’ da Tabela de Acumulação.

Por último, são apresentados os resultados. Com base na Tabela de Acumulação limiarizada, o sistema obtém, a partir de uma comparação com a estrutura de dados criada inicialmente, os prováveis centros dos objetos circulares da imagem. A partir do resultado desta comparação, os pontos de borda são desenhados em uma imagem resultante e apresentados ao usuário.

## 6. Testes e Resultados

---

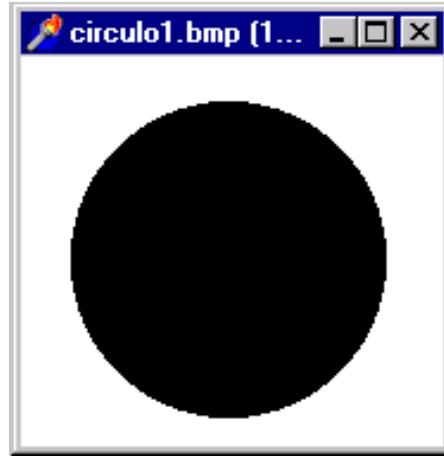
Os testes realizados com o software Trans\_Hough 2.0 foram feitos utilizando-se três tipos de imagens: 1) Imagem Artificial Simples; 2) Imagem Artificial Complexa e 3) Imagem Real Complexa. Todas as imagens foram testadas em dois computadores diferentes:

- ✓ Pentium II, 200 MHz e com 64 MB de memória RAM (será chamado de P II).
- ✓ Pentium III, 450 MHz e com 64 MB de memória RAM (será chamado de P III).

Os resultados apresentados a seguir foram obtidos a partir dos seguintes parâmetros: porcentagem de pixels de borda analisados, tempos de processamento de cada imagem em segundos, utilizando-se ambas as possibilidades: três e quatro pontos, o número de objetos circulares encontrados e distâncias mínima e máxima utilizadas, de acordo com o formalismo matemático apresentado no capítulo 4.

### 6.1. *Imagen Artificial Simples*

A primeira imagem testada foi a de um único círculo (Figura 13). Os tempos de processamento são extremamente baixos, no entanto, apresentam uma variação muito grande para as porcentagens mínima e máxima, conforme as Tabelas 2 e 3. As Figuras 14 e 15 mostram visualmente os resultados obtidos para todas as porcentagens utilizadas, respectivamente para três e quatro pontos. Nas tabelas 2 e 3 são apresentados, respectivamente, os resultados dos processamentos utilizando os computadores P II e P III.



**Figura 13** – Imagem artificial simples, contendo apenas um círculo

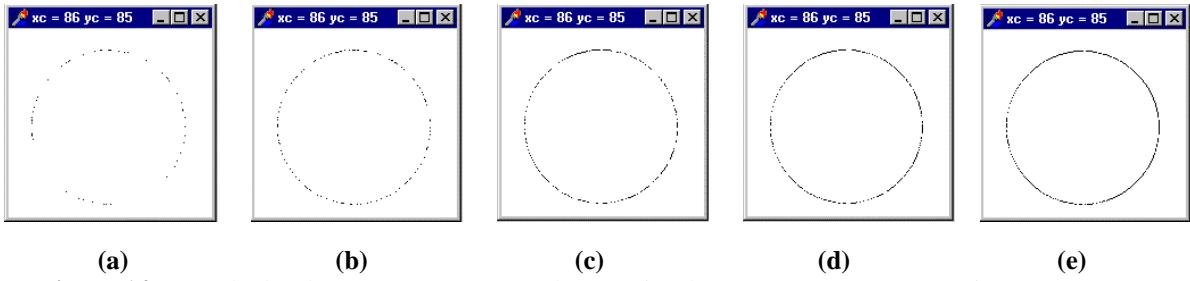
<i>Imagen: Circulo1.bmp</i>		<i>Distâncias: 20 – 89</i>			
Porcentagem de Pixels	3 Pontos		4 Pontos		Encontrados
	Tempo (s)	Encontrados	Tempo (s)	Encontrados	
20 %	0,72	1 círculo	0,86	1 círculo	
40 %	0,78	1 círculo	0,91	1 círculo	
60 %	0,94	1 círculo	0,94	1 círculo	
80 %	1,00	1 círculo	1,00	1 círculo	
100 %	1,41	1 círculo	1,41	1 círculo	

**Tabela 2** – Resultados do teste com Imagem Artificial Simples, usando o computador P II

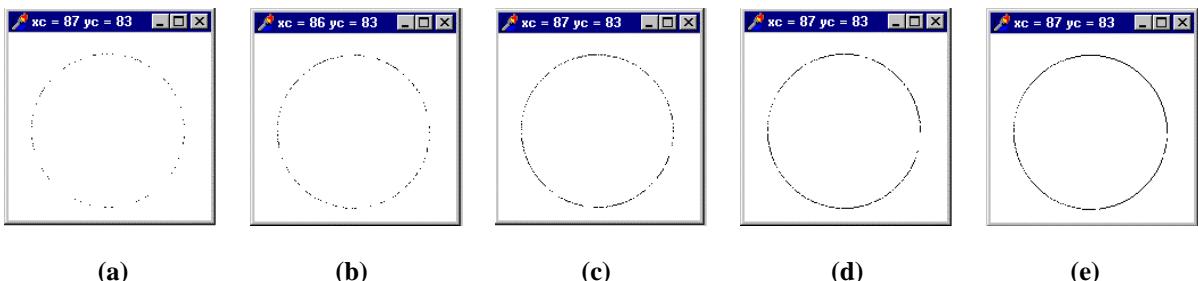
<i>Imagen: Circulo1.bmp</i>		<i>Distâncias: 20 – 89</i>			
Porcentagem de Pixels	3 Pontos		4 Pontos		Encontrados
	Tempo (s)	Encontrados	Tempo (s)	Encontrados	
20 %	0,23	1 círculo	0,23	1 círculo	
40 %	0,33	1 círculo	0,33	1 círculo	
60 %	0,43	1 círculo	0,43	1 círculo	
80 %	0,53	1 círculo	0,53	1 círculo	
100 %	0,63	1 círculo	0,63	1 círculo	

**Tabela 3** – Resultados do teste com Imagem Artificial Simples, usando o computador P III

Por se tratar de uma imagem de baixíssima complexidade e sem a presença de ruído, os tempos de processamento são muito baixos e considerados excelentes para um processador como o do Pentium III. Utilizando-se ambas as técnicas para o processamento (três e quatro pontos), os resultados obtidos são considerados muito bons utilizando 60 ou 80 % dos pixels apresentando por volta de 25 % de variação máxima no tempo de processamento (computador PIII), o que justifica a implementação do modelo da RHT Randômica.



**Figura 14** – Resultados do processamento usando a técnica de três pontos, para as seguintes porcentagens de pixels detectados: (a) 20 %; (b) 40 %; (c) 60 %; (d) 80 %; (e) 100 %.



**Figura 15** – Resultados do processamento usando a técnica de quatro pontos, para as seguintes porcentagens de pixels detectados: (a) 20 %; (b) 40 %; (c) 60 %; (d) 80 %; (e) 100 %.

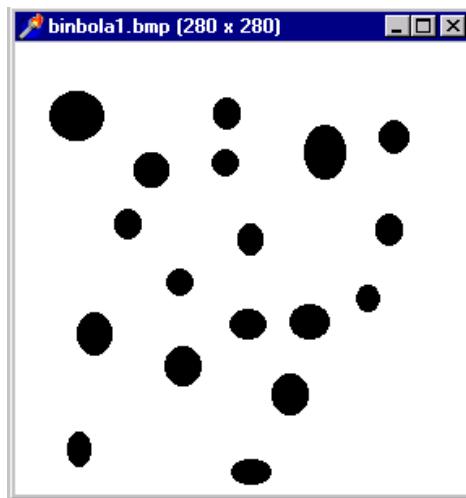
Visualmente, o resultado para 100 % é melhor, mas apresentou um acréscimo de 41 % no tempo de processamento, o que não aconselha sua utilização. Para este caso específico de imagem, as distâncias utilizadas foram as sugeridas pelo próprio software (calculadas internamente, quando executada uma imagem), não sendo necessários testes mais específicos para se descobrir distâncias melhores.

## 6.2. Imagem Artificial Complexa

A segunda imagem testada apresenta vários objetos circulares distribuídos aleatoriamente na imagem (Figura 16) e com diferentes tamanhos. Deve-se tomar um cuidado especial com este tipo de imagem no que se refere às distâncias mínima e principalmente a máxima, devido à pequena distância dos objetos. Uma distância máxima obtida de forma errônea pode obter pixels de objetos diferentes, acarretando em um sério problema de falsas detecções.

As tabelas 4 e 5 mostram, respectivamente, os resultados dos processamentos utilizando os computadores P II e P III. As figuras 17 e 18 mostram visualmente os resultados

obtidos para todas as porcentagens utilizadas, respectivamente para as técnicas de três e quatro pontos. No entanto, devido aos vários objetos circulares encontrados, serão mostradas apenas uma das imagens resultantes para cada porcentagem utilizada.



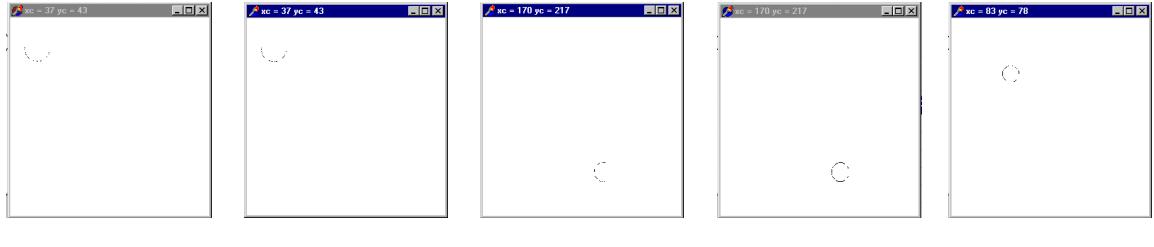
**Figura 16** – Imagem artificial complexa, contendo vários objetos circulares de diferentes tamanhos

<b>Imagen: Binbola.bmp</b>			<b>Distâncias: 10 – 20</b>	
Porcentagem de Pixels	3 Pontos		4 Pontos	
	Tempo (s)	Encontrados	Tempo (s)	Encontrados
20 %	3,72	4 círculos	4,73	4 círculos
40 %	5,89	7 círculos	6,91	5 círculos
60 %	7,40	8 círculos	9,22	8 círculos
80 %	9,53	11 círculos	12,13	10 círculos
100 %	12,50	11 círculos	15,96	13 círculos

**Tabela 4** – Resultados do teste com Imagem Artificial Complexa, usando o computador P II

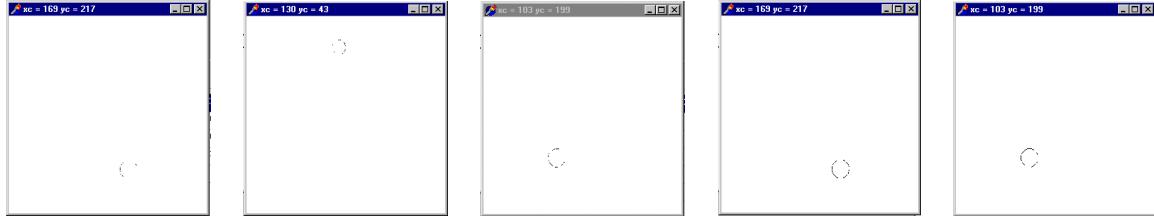
<b>Imagen: Binbola.bmp</b>			<b>Distâncias: 10 – 20</b>	
Porcentagem de Pixels	3 Pontos		4 Pontos	
	Tempo (s)	Encontrados	Tempo (s)	Encontrados
20 %	1,14	3 círculos	1,50	4 círculos
40 %	1,85	4 círculos	2,19	7 círculos
60 %	2,59	9 círculos	3,31	9 círculos
80 %	3,37	10 círculos	4,43	10 círculos
100 %	4,75	12 círculos	5,76	13 círculos

**Tabela 5** – Resultados do teste com Imagem Artificial Complexa, usando o computador P III



(a) (b) (c) (d) (e)

**Figura 17** – Alguns dos resultados do processamento usando a técnica de três pontos, para as seguintes porcentagens de pixels detectados: (a) 20 %; (b) 40 %; (c) 60 %; (d) 80 %; (e) 100 %.



(a) (b) (c) (d) (e)

**Figura 18** – Alguns dos resultados do processamento usando a técnica de quatro pontos, para as seguintes porcentagens de pixels detectados: (a) 20 %; (b) 40 %; (c) 60 %; (d) 80 %; (e) 100 %.

Considerada a complexidade da imagem, os resultados obtidos são considerados bons, principalmente no que se refere ao tempo de processamento, considerado baixo. O resultado visual para 80 % é considerado satisfatório e mais vantajoso ainda se usada a técnica de três pontos, que apresenta uma redução de 25 % no tempo de processamento, justificando, assim, a implementação do modelo da RHT Randômica.

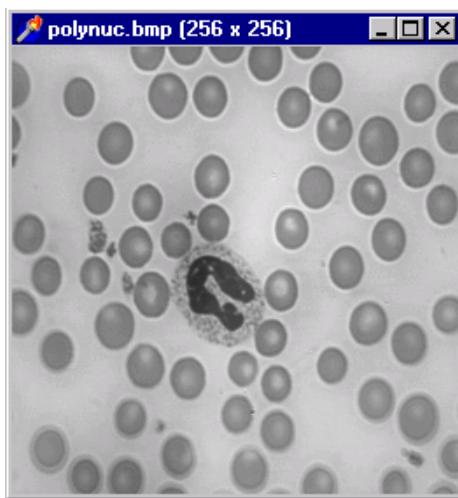
O resultado visual para 100 % é razoavelmente melhor, mas não justifica a utilização devido ao acréscimo de 25 % que apresenta no tempo de processamento. É importante ressaltar que alguns dos objetos presentes na imagem não são encontrados pelo fato de serem elipses, o que está fora do escopo deste projeto.

### 6.3. Imagem Real Complexa

A terceira imagem testada foi uma imagem em níveis de cinza, que mostra um conjunto de células verdadeiras (Figura 19). Os tempos de processamento são extremamente altos, devido à grande complexidade da imagem. As tabelas 4 e 5 mostram, respectivamente, os resultados dos processamentos utilizando os computadores P II e P III. As figuras 19 e 20 mostram visualmente os resultados obtidos para todas as porcentagens utilizadas,

respectivamente para as técnicas de três e quatro pontos. No entanto, devido aos vários objetos circulares encontrados, serão mostradas apenas uma das imagens resultantes para cada porcentagem utilizada.

Neste tipo de imagem também deve-se tomar um cuidado especial na obtenção das distâncias mínima e máxima, já que distâncias obtidas diferentes da realidade podem acarretar em um sério problema de falsas detecções.



**Figura 15** – Imagem Real Complexa em níveis de cinza, contendo um conjunto de células verdadeiras

<b>Imagen: Polynuc.bmp</b>			<b>Distâncias: 10 – 20</b>	
Porcentagem de Pixels	3 Pontos		4 Pontos	
	Tempo (s)	Encontrados	Tempo (s)	Encontrados
20 %	142,79	1 círculo	188,88	2 círculos
40 %	361,47	5 círculos	423,31	4 círculos
60 %	567,56	7 círculos	765,93	7 círculos
80 %	927,33	7 círculos	1132,14	8 círculos
100 %	1446,45	6 círculos	1714,05	11 círculos

**Tabela 6** – Resultados do teste com Imagem Real Complexa, usando o computador P II

<b>Imagen: Polynuc.bmp</b>			<b>Distâncias: 10 – 20</b>	
Porcentagem de Pixels	3 Pontos		4 Pontos	
	Tempo (s)	Encontrados	Tempo (s)	Encontrados
20 %	60,29	1 círculo	120,20	3 círculos
40 %	60,23	1 círculo	300,10	4 círculos
60 %	360,29	3 círculos	480,22	6 círculos
80 %	540,46	8 círculos	720,10	6 círculos
100 %	840,35	10 círculos	1020,27	8 círculos

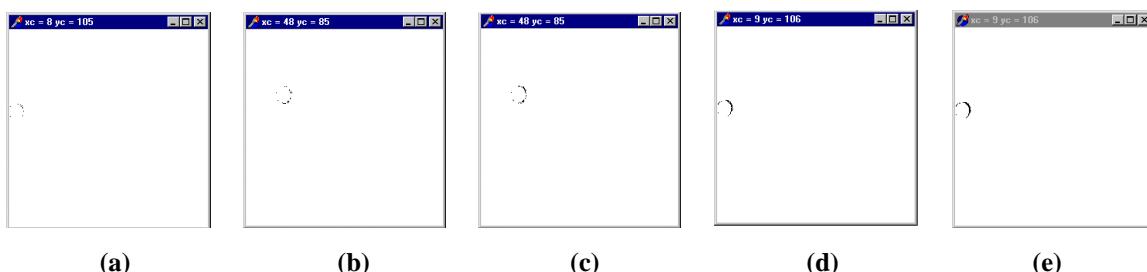
**Tabela 7** – Resultados do teste com Imagem Real Complexa, usando o computador P III

Devido à enorme quantidade de pontos de borda presentes na imagem e usados para o cálculo da RHT, os tempos de processamento deste tipo de imagem são extremamente grandes e o resultado obtido é razoável. No entanto, devido à grande presença de ruído na imagem, alguns dos objetos encontrados não são verdadeiramente objetos circulares e sim ruídos da imagem.

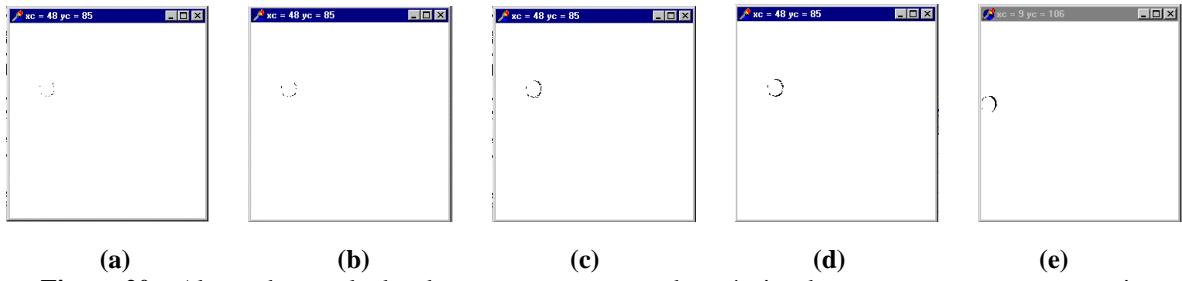
Considerando-se ambas as técnicas (três e quatro pontos) e também os dois computadores utilizados, os tempos de processamento apresentam uma grande variação, de acordo com a porcentagem escolhida: vão por volta de 1m (20% dos pontos, no computador PIII) até 28m30s, aproximadamente (100% dos pontos, no computador PII).

Especificamente neste tipo de imagem, sugere-se que os valores das distâncias máxima e mínima sejam alterados, efetuando-se outros testes. Os valores usados aqui, foram utilizados considerando-se sempre o número de objetos presentes na imagem e também o tamanho médio dos mesmos, para evitar que pixels de diferentes objetos sejam usados no mesmo cálculo, acarretando em falsas detecções.

Visualmente, os resultados obtidos para 60 e 80 % são considerados bons, com um tempo razoável, considerando-se a complexidade da imagem, o que também justifica a implementação da RHT Randômica.



**Figura 19** – Alguns dos resultados do processamento usando a técnica de três pontos, para as seguintes porcentagens de pixels detectados: (a) 20 %; (b) 40 %; (c) 60 %; (d) 80 %; (e) 100 %.



**Figura 20** – Alguns dos resultados do processamento usando a técnica de quatro pontos, para as seguintes porcentagens de pixels detectados: (a) 20 %; (b) 40 %; (c) 60 %; (d) 80 %; (e) 100 %.

## 7. Conclusão

---

Dentre as várias versões probabilísticas existentes da Transformada de Hough, procuramos mostrar nesta monografia que o modelo randômico, que utiliza a idéia de seleção aleatória de pontos, ou seja, faz uso de um pequeno subconjunto de dados da imagem original, é o método mais eficiente para solucionar o problema de detecção de objetos circulares com um bom tempo de processamento e utilização de memória, quando comparado ao modelo clássico da Transformada de Hough.

## Referências Bibliográficas

---

- [1] FACON, Jacques. **Processamento e Análise de Imagens.** Apostila, ano 2000.
- [2] KÄLVIÄINEN, Heikki; HIRVONEN, Petri; XU, Lei & OJA, Erkki. **Probabilistic and non-probabilistic Hough transforms: overview and comparisons.** Image and Vision Computing, Vol. 13 – No. 4, Maio 1995, pp. 1 – 14.
- [3] KÄLVIÄINEN, Heikki & HIRVONEN, Petri. **Connective Randomized Hough Transform.** Proc. 9th Scandinavian Conference on Image Analysis, Uppsala, Sweden, June 1995, pp. 1029 – 1036.
- [4] KULTANEN, P.; XU, L. & OJA, E. **Randomized Hough Transform (RHT),** Proc. 10th Int. Conf. Patt. Recogn., Atlantic City, NJ, June 1990, pp. 631 – 635.
- [5] McLAUGHLIN, Robert A. **Technical Report – Randomized Hough Transform: Improved Ellipse Detection with Comparison.** 1997, pp. 1 – 12.  
[http://ciips.ee.uwa.edu.au/Papers/Technical\\_Reports/1997/01/Index.html](http://ciips.ee.uwa.edu.au/Papers/Technical_Reports/1997/01/Index.html)
- [6] McLAUGHLIN, Robert A. & ALDER, Michael D. **The Hough Transform versus the UpWrite.** University of Western Australia, Australia. 1995, pp. 1 – 13.  
[http://ciips.ee.uwa.edu.au/Papers/Technical\\_Reports/1995/03/Index.html](http://ciips.ee.uwa.edu.au/Papers/Technical_Reports/1995/03/Index.html)
- [7] XU, L.; OJA, E. & KULTANEN, P. **A new curve detection method: Randomized Hough Transform (RHT).** Patt. Recogn. Lett., Vol. 11, No 5, 1990, pp. 331 – 338.

---

## FONTES EM DELPHI da TRANSFORMADA de HOUGH para Reta e Circulo

---

```

unit UfrmHough;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, ExtCtrls, StdCtrls, Grids;

type
  { Necessário para manipular a lista (inicial) de pixels }
  PInicial = ^TInicial;
  TInicial = record
    Prev : PInicial;
    x,y : integer;
    cont : Integer;
    checked : Boolean;
    Next: PInicial;
  end;

  { Descritor da Lista Inicial de Pixels }
  PDescriptor = ^TDescriptor;
  TDescriptor = record
    Prev : PInicial;
    Next : PInicial;
  end;

TFrmHough = class(TForm)
  MainMenu1: TMainMenu;
  Arquivol: TMenuItem;
  Abrir: TMenuItem;
  Salvar: TMenuItem;
  SalvarComo: TMenuItem;
  Imprimir: TMenuItem;
  ConfigurarImpressora: TMenuItem;
  OpenFile: TOpenDialog;
  SaveFile: TSaveDialog;
  PrintFile: TPrintDialog;
  PrinterSetup: TPrinterSetupDialog;
  Morfologial: TMenuItem;
  Sair: TMenuItem;
  Edit1: TEdit;
  Edit2: TEdit;
  CRHT1: TMenuItem;
  Retas: TMenuItem;
  Edit3: TEdit;
  StringGrid1: TStringGrid;
  StringGrid2: TStringGrid;
  StringGrid3: TStringGrid;
  StringGrid4: TStringGrid;
  Acumulaoaeb1: TMenuItem;
  Desenhal: TMenuItem;
  PnlAcum: TPanel;
  StringGrid5: TStringGrid;
  AcumulaoInicial1: TMenuItem;
  StringGrid6: TStringGrid;
  StringGrid7: TStringGrid;
  DetecaodeBordasporerosaoQuadrada1: TMenuItem;

```

```

Transformadaparacirculos: TMenuItem;
Acumulacao: TMenuItem;
StringGrid8: TStringGrid;
Ajuda2: TMenuItem;
TpicosdeAjuda2: TMenuItem;
Sobre2: TMenuItem;
Limiari1: TMenuItem;
visualizaol: TMenuItem;
Edit4: TEdit;
StringGrid9: TStringGrid;
StringGrid10: TStringGrid;
StringGrid11: TStringGrid;
PorcentagemdePixelanalisisados1: TMenuItem;
CalculoUsandoTresPontos1: TMenuItem;
CalculoUsandoQuadroPontos1: TMenuItem;
procedure AbrirClick(Sender: TObject);
procedure SalvarClick(Sender: TObject);
procedure SalvarComoClick(Sender: TObject);
procedure ImprimirClick(Sender: TObject);
procedure ConfigurarImpressoraClick(Sender: TObject);
procedure ErosaoCinza (var dBitmap:TBitmap; li,ls:integer);
procedure BinarizaImagem(var dBitmap : TBitmap; li,ls : Integer);
procedure AlocaImagemOriginal(var dBitmap: TBitmap; li,ls : Integer);
procedure SairClick(Sender: TObject);
procedure RetasClick(Sender: TObject);
procedure ProcuraNodo(var aBitmap : TBitmap);
procedure CriaJanela(var aBitmap : TBitmap);
procedure Acumulaoaeb1Click(Sender: TObject);
procedure DesenhalClick(Sender: TObject);
procedure FormActivate(Sender: TObject);
procedure AcumulaoInicial1Click(Sender: TObject);
procedure DetecaodeBordasporerosaoQuadrada1Click(Sender: TObject);
procedure TpicosdeAjuda2Click(Sender: TObject);
procedure Sobre2Click(Sender: TObject);
procedure ProcuraNodo1(var aBitmap : TBitmap;li,ls : integer);
procedure ProcuraNodo2(var aBitmap : TBitmap;li,ls : integer);
procedure AcumulacaoClick(Sender: TObject);
procedure Limiar1Click(Sender: TObject);
procedure AlocaImagemOriginal1(var dBitmap: TBitmap; li,ls : Integer);
procedure AlocaImagemOriginal2(var dBitmap: TBitmap; li,ls : Integer);
procedure CalculatransformadadehoughparacirculosUsandoTresPontos(var
aBitmap : TBitmap; li,ls : integer);
procedure calculatransformadadehoughparacirculosUsandoQuadroPontos(var
aBitmap : TBitmap; li,ls :integer);
procedure visualizaolClick(Sender: TObject);
procedure ajustar(var xcl, ycl:integer);
procedure CalculoUsandoTresPontos1Click(Sender: TObject);
procedure CalculoUsandoQuadroPontos1Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
  bEE: byte;
end;

var
  FrmHough: TfrmHough;
  desc : PDescriptor;
  desc1 : PDescriptor;
  p : PInicial;
  contchecked : Double;

```

```

r : Integer;
x,y,x1,y1,x2,y2,x3,y3,x4,y4 :integer;
newcont,percent,maxdistancia,distanciaminima,xc,yc : Integer;
a,b,a1,b1,a2,b2,a3,b3,a4,b4,a5,b5,a6,b6,a7,b7,newpercent : Double;
abitmap : Tbitmap;
implementation

uses UfrmImagen, topajupas, ftSobre, ufrmvisual ;
{$R *.DFM}

procedure TFrmHough.TpicosdeAjuda2Click(Sender: TObject);
begin
TopAju.Show;
end;

procedure TFrmHough.Sobre2Click(Sender: TObject);
begin
fSobre1.Show;
end;

procedure TFrmHough.AbrirClick(Sender: TObject);
var
sFileExt: String[4];
i,j : Integer;
bBitMap: TBitMap;
begin
for i := 0 to StringGrid1.ColCount do
  for j := 0 to StringGrid1.RowCount do
    begin
      StringGrid1.Cells[i,j] :='';
    end;
  for i := 0 to StringGrid2.ColCount do
    for j := 0 to StringGrid2.RowCount do
      begin
        StringGrid2.Cells[i,j] :='';
      end;
    for i := 0 to StringGrid3.ColCount do
      for j := 0 to StringGrid3.RowCount do
        begin
          StringGrid3.Cells[i,j] :='';
        end;
      for i := 0 to StringGrid4.ColCount do
        for j := 0 to StringGrid4.RowCount do
          begin
            StringGrid4.Cells[i,j] :='';
          end;
        for i := 0 to StringGrid5.ColCount do
          for j := 0 to StringGrid5.RowCount do
            begin
              StringGrid5.Cells[i,j] :='';
            end;
          for i := 0 to StringGrid6.ColCount do
            for j := 0 to StringGrid6.RowCount do
              begin
                StringGrid6.Cells[i,j] :='';
              end;
            for i := 0 to StringGrid7.ColCount do
              for j := 0 to StringGrid7.RowCount do
                begin
                  StringGrid7.Cells[i,j] :='';
                end;

```

```

for i := 0 to StringGrid8.ColCount do
  for j := 0 to StringGrid8.RowCount do
    begin
      StringGrid8.Cells[i,j] :='';
    end;
for i := 0 to StringGrid9.ColCount do
  for j := 0 to StringGrid9.RowCount do
    begin
      StringGrid9.Cells[i,j] :='';
    end;
for i := 0 to StringGrid10.ColCount do
  for j := 0 to StringGrid10.RowCount do
    begin
      StringGrid10.Cells[i,j] :='';
    end;
for i := 0 to StringGrid11.ColCount do
  for j := 0 to StringGrid11.RowCount do
    begin
      StringGrid11.Cells[i,j] :='';
    end;
StringGrid2.ColCount := 0;
StringGrid2.RowCount := 0;
StringGrid1.ColCount := 6;
StringGrid1.RowCount := 1;
StringGrid3.ColCount := 0;
StringGrid3.RowCount := 0;
StringGrid4.ColCount := 0;
StringGrid4.RowCount := 0;
StringGrid5.ColCount := 0;
StringGrid5.RowCount := 0;
StringGrid6.ColCount := 0;
StringGrid6.RowCount := 0;
StringGrid7.ColCount := 0;
StringGrid7.RowCount := 0;
StringGrid8.ColCount := 0;
StringGrid8.RowCount := 0;
StringGrid9.ColCount := 0;
StringGrid9.RowCount := 0;
StringGrid10.ColCount := 0;
StringGrid10.RowCount := 0;
StringGrid11.ColCount := 0;
StringGrid11.RowCount := 0;

if OpenFile.Execute then
begin
  Screen.Cursor := crHourglass; { cursor de espera }
  sFileExt := Uppercase (ExtractFileExt (OpenFile.FileName));
  if (sFileExt = '.BMP') then
  begin
    FrmImagem := TFrmImagem.Create (Self);
    FrmImagem.iBitmap := TBitmap.Create;
    FrmImagem.Open(OpenFile.FileName);
    FrmImagem.Visible := True;
    FrmImagem.SetFocus;
  end;

  Screen.Cursor := crDefault;
end;
end;

```

```

procedure TfrmHough.SalvarClick(Sender: TObject);
begin
  FrmImagem.iBitMap.SaveToFile (OpenFile.FileName);
end;

procedure TfrmHough.SalvarComoClick(Sender: TObject);
begin
  if SaveFile.Execute then
  begin
    FrmImagem.iBitMap.SaveToFile (SaveFile.FileName);
    FrmImagem.Caption := LowerCase(ExtractFileName (SaveFile.FileName))
      + Format(' (%d x %d)',[FrmImagem.iBitMap.Height,
    FrmImagem.iBitMap.Width]);
  end;
end;

procedure TfrmHough.ImprimirClick(Sender: TObject);
begin
  if PrintFile.Execute then
  begin
    { Add code to print current file }
  end;
end;

procedure TfrmHough.ConfigurarImpressoraClick(Sender: TObject);
begin
  PrinterSetup.Execute;
end;

procedure TfrmHough.ErosaoCinza (var dBitMap:TBitMap; li,ls:integer);
var
  i,j,It:integer;
  aBitMap: TBitMap;
  iMin,temp: LongInt;
  NewString : String;
begin
  aBitMap := TBitMap.Create;
  aBitMap.Assign (dBitMap); { copia a imagem original }

  for i := 0 to ls do {vertical}
  begin
    for j := 0 to li do {horizontal}
    begin
      iMin := dBitMap.Canvas.Pixels[i,j]; { ponto central }
      temp:=dBitMap.Canvas.Pixels[i,j-1];
      if (temp < iMin) then
        iMin := temp; {erosao usando o
elemento cruz}
      temp:=dBitMap.Canvas.Pixels[i,j+1];
      if (temp < iMin)then
        iMin := temp;
      temp:=dBitMap.Canvas.Pixels[i-1,j];
      if (temp < iMin)then
        iMin := temp;
      temp:=dBitMap.Canvas.Pixels[i+1,j];
      if (temp < iMin)then
        iMin := temp;
      aBitMap.Canvas.Pixels[i,j] := $02000000 + iMin; {ponto
central }
    end;
  end;
end;

```

```

        end;
    end;
    dBitMap.Assign (aBitMap); { atualiza imagem original }
    { copia imagem dilatada para parametro por referencia }

end;

procedure TfrmHough.BinarizaImagem(var dBitMap : TBitMap; li,ls : Integer);
var
  i,j,total : longint;
  grad : double;
  media,newmedia : Double;
  aBitMap : TBitMap;
  NewString : String;
  ClickedOK: Boolean;
begin
  aBitMap := TBitMap.Create;
  aBitMap.Assign (dBitMap); { copia a imagem original }
  media := 0.0;

  total := 0;

  for i := 0 to ls do {vertical}
  begin
    for j := 0 to li do {horizontal}
    begin
      grad := aBitMap.Canvas.Pixels[i,j]; {coloca a cor do pixel na
variavel grad}
      if grad <> 0 then {verifica se grad é diferente de preto}
      begin
        media := media + grad; {soma o grad na variavel media}
        total := total + 1;
      end;
    end;
  end;

  media := media / total;           {calcula a media das cores dos pixels}

  NewString := '';
  ClickedOK := InputQuery('Porcentagem abaixo da média', 'Detecção de
borda', NewString);
  if ClickedOK then
  begin
    newmedia := media -((StrToInt(NewString))/100)* media;// + newmedia;
    end      {calcula a nova media usando a percentagem usada pelo usuário}
    else
      exit;

  for i := 1 to ls-1 do {vertical}
  begin
    for j := 1 to li-1 do {horizontal}
    begin
      if aBitMap.Canvas.Pixels[i,j] > newmedia then {verifica se a cor do
pixel é maior que a nova media}
      begin
        aBitMap.Canvas.Pixels[i,j] := $00000000 ; {imprime o pixel preto}
      end
      else
      begin
        aBitMap.Canvas.Pixels[i,j] := $00FFFFFF ; {imprime o pixel branco}
      end;
    end;
  end;
end;

```

```

    end;
end;
dBitMap.Assign (aBitMap); { copia a imagem original }
end;

procedure TFrmHough.DetectaodeBordasporerosaoQuadrada1Click(Sender: TObject);
var
  li,ls,i,j : Integer;
  aBitMap, bBitMap, cBitmap: TBitMap;
  aux :LongInt;
  NewString : String;
  var ok :integer;
begin
  Screen.Cursor := crHourglass; { cursor de espera }

  FrmImagem := TFrmImagem.Create(self); { cria uma nova imagem}
  FrmImagem.iBitMap := TBitMap.Create; { aloca a imagem}
  FrmImagem.Open(OpenFile.FileName); { abre a imagem}
  li := (FrmImagem.iBitMap.Height)-1; { Altura da imagem }
  ls := (FrmImagem.iBitMap.Width)-1; { Largura da imagem }
  aBitMap := TBitMap.Create; { aloca as imagens auxiliares}
  bBitMap := TBitMap.Create;
  cBitmap := TBitMap.Create;
  aBitMap.Assign (FrmImagem.iBitMap); { imagem original }
  cBitmap.Assign (FrmImagem.iBitMap); { imagem original }
  ErosaoCinza (FrmImagem.iBitMap, li,ls); { erosão }
  bBitMap.Assign (FrmImagem.iBitMap); { imagem erudida }
  FrmImagem.FImage.Picture.Assign (FrmImagem.iBitMap); { atualiza formulário }
  for i := 1 to ls-1 do
    begin
      for j := 1 to li-1 do {realiza a subtração da imagem original da
      imagem erudida}
      begin
        aux := aBitMap.Canvas.Pixels[i,j] -
      bBitMap.Canvas.Pixels[i,j];
        cBitmap.Canvas.Pixels[i,j] := $02000000 + aux;
      end;
    end;
  repeat
  begin
    FrmImagem.FImage.Picture.Assign (cBitmap);
    FrmImagem.iBitMap.assign(cBitmap); {atualiza o formulário}
    BinarizaImagem(frmImagem.ibitmap,li,ls); {binariza a imagem}
    bBitMap.Assign (FrmImagem.iBitMap); { imagem binarizada }
    FrmImagem.FImage.Picture.Assign (bBitMap); { atualiza formulário }
    FrmImagem.iBitMap.Assign (bBitMap);
    ok:=MessageDlg('A detecção das Bordas teve um Bom resultado?', mtConfirmation, [mbYes,mbNo], 0);
    end;
  until(ok=6);{laço de repetição}
  aBitMap.Free;
  bBitMap.Free;
  cBitmap.Free; { libera memória ocupada pelas imagens auxiliares}
  Screen.Cursor := crDefault; { cursor normal }
end;

procedure TFrmHough.SairClick(Sender: TObject);
begin

```

```

    FrmHough.Close;
end;

procedure TFrmHough.RetasClick(Sender: TObject);
var
  aBitMap: TBitMap;
  li,ls,i,j,aux : Integer;
  NewString : String;
  ClickedOK: Boolean;
begin
  newcont := 0;
  percent:=0;
  newpercent:=0;
  for i := 0 to StringGrid1.ColCount do
    for j := 0 to StringGrid1.RowCount do
      begin
        StringGrid1.Cells[i,j] :='';
      end;
  for i := 0 to StringGrid2.ColCount do
    for j := 0 to StringGrid2.RowCount do
      begin
        StringGrid2.Cells[i,j] :='';
      end;
  for i := 0 to StringGrid3.ColCount do
    for j := 0 to StringGrid3.RowCount do
      begin
        StringGrid3.Cells[i,j] :='';
      end;
  for i := 0 to StringGrid4.ColCount do
    for j := 0 to StringGrid4.RowCount do
      begin
        StringGrid4.Cells[i,j] :='';
      end;
  for i := 0 to StringGrid5.ColCount do
    for j := 0 to StringGrid5.RowCount do
      begin
        StringGrid5.Cells[i,j] :='';
      end;
  for i := 0 to StringGrid6.ColCount do
    for j := 0 to StringGrid6.RowCount do
      begin
        StringGrid6.Cells[i,j] :='';
      end;
  for i := 0 to StringGrid7.ColCount do
    for j := 0 to StringGrid7.RowCount do
      begin
        StringGrid7.Cells[i,j] :='';
      end;
  for i := 0 to StringGrid8.ColCount do
    for j := 0 to StringGrid8.RowCount do
      begin
        StringGrid8.Cells[i,j] :='';
      end;
  for i := 0 to StringGrid9.ColCount do
    for j := 0 to StringGrid9.RowCount do
      begin
        StringGrid9.Cells[i,j] :='';
      end;
  for i := 0 to StringGrid10.ColCount do
    for j := 0 to StringGrid10.RowCount do
      begin

```

```

        StringGrid10.Cells[i,j] := '';
end;
for i := 0 to StringGrid11.ColCount do
for j := 0 to StringGrid11.RowCount do
begin
        StringGrid11.Cells[i,j] := '';
end;

StringGrid2.ColCount := 0;
StringGrid2.RowCount := 0;
StringGrid1.ColCount := 6;
StringGrid1.RowCount := 1;
StringGrid3.ColCount := 0;
StringGrid3.RowCount := 0;
StringGrid4.ColCount := 0;
StringGrid4.RowCount := 0;
StringGrid5.ColCount := 0;
StringGrid5.RowCount := 0;
StringGrid6.ColCount := 0;
StringGrid6.RowCount := 0;
StringGrid7.ColCount := 0;
StringGrid7.RowCount := 0;
StringGrid8.ColCount := 0;
StringGrid8.RowCount := 0;
StringGrid9.ColCount := 0;
StringGrid9.RowCount := 0;
StringGrid10.ColCount := 0;
StringGrid10.RowCount := 0;
StringGrid11.ColCount := 0;
StringGrid11.RowCount := 0;

NewString := '';
ClickedOK := InputQuery('Porcentagem', 'Detecção', NewString);
if ClickedOK then
begin
        percent := StrToInt(NewString);
end
else
        exit;
newpercent := (100 - percent)/100;

Screen.Cursor := crHourglass; { cursor de espera }
li := (FrmImagem.iBitMap.Height)-1; { Altura da imagem }
ls := (FrmImagem.iBitMap.Width)-1; { Largura da imagem }

aBitMap := TBitMap.Create;
aBitMap.Assign (FrmImagem.iBitMap); { imagem original }
aBitMap.PixelFormat := pf1bit;

AlocaImagemOriginal(FrmImagem.iBitMap, li,ls); { aloca imagem original }

aBitMap.Free;
Screen.Cursor := crDefault; { cursor normal }

StringGrid1.ColCount := 6;
StringGrid1.RowCount := 1;
StringGrid1.Cells[0,0] := 'a';
StringGrid1.Cells[1,0] := 'b';
StringGrid1.Cells[2,0] := 'x1';
StringGrid1.Cells[3,0] := 'y1';

```

```

StringGrid1.Cells[4,0] :='x2';
StringGrid1.Cells[5,0] :='y2';

StringGrid2.ColCount := 7;
StringGrid2.Cells[0,0] :='a';
StringGrid2.Cells[1,0] :='b';
StringGrid2.Cells[2,0] :='x1';
StringGrid2.Cells[3,0] :='y1';
StringGrid2.Cells[4,0] :='x2';
StringGrid2.Cells[5,0] :='y2';
StringGrid2.Cells[6,0] :='Contador';

StringGrid5.ColCount := 7;
StringGrid5.Cells[0,0] :='a';
StringGrid5.Cells[1,0] :='b';
StringGrid5.Cells[2,0] :='x1';
StringGrid5.Cells[3,0] :='y1';
StringGrid5.Cells[4,0] :='x2';
StringGrid5.Cells[5,0] :='y2';
StringGrid5.Cells[6,0] :='Contador';

StringGrid7.ColCount := 3;
StringGrid7.Cells[0,0] :='a';
StringGrid7.Cells[1,0] :='b';
StringGrid7.Cells[2,0] := 'Contador';

end;

procedure TfrmHough.AlocaImagemOriginal(var dBitmap: TBitmap; li,ls : Integer);
var
  aBitmap : TBitmap;
  x,y,i,j,cont,contador: Integer;
  aux : LongInt;
  cont1 : Double;
begin
  aBitmap := TBitmap.Create;
  aBitmap.Assign(dBitmap);
  aBitmap.PixelFormat := pflbit;
  contador := 0;

  New(desc); // Aloca descriptor
  desc^.Prev := nil;
  desc^.Next := nil;

  for x := 0 to ls do {horizontal}
  begin
    for y := 0 to li do {vertical}
    begin
      if aBitmap.Canvas.Pixels[x,y] = 0 then
      begin
        if (desc^.Prev = nil) and (desc^.Next = nil) then {primeiro nodo}
        begin
          New(p); // Aloca nodo
          desc^.Next := p;
          desc^.Prev := p;
          p^.Prev := nil;
          p^.Next := nil;
          p^.x := x;
          p^.y := y;
          p^.checked := False;
        end;
      end;
    end;
  end;
end;

```

```

    p^.cont := contador ; // Necessário para escolha randômica
end
else
begin
  New(p); // Aloca nodo
  p^.Prev := desc^.Next;
  p^.Next := nil;
  (desc^.Next)^.Next := p;
  desc^.Next := p;
  p^.x := x;
  p^.y := y;
  p^.checked := False;
  p^.cont := contador + 1; // Necessário para escolha randômica
  contador := contador + 1;
end;
end;
end;

contchecked := contador;

Edit1.Text := IntToStr(contador);
li := (FrmImagen.iBitMap.Height)-1; { Altura da imagem }
ls := (FrmImagen.iBitMap.Width)-1;

// Deve- se incluir um comando de checagem para ver se o nodo já foi
escolhido anteriormente
// Deve- se iniciar o descritor em sua configuração inicial
cont := 0;

cont1 := contchecked;

while ((contchecked - (cont1 * newpercent)) > 0) do
begin
  cont := cont + 1;

  // Atualiza descritor (configuração inicial)
  while (desc^.Prev)^.Prev <> nil do
    desc^.Prev := (desc^.Prev)^.Prev;

  // Randomize;
  r := Random(contador);
  Edit2.Text := IntToStr(r);
  ProcuraNodo(aBitMap);
end;
Dispose(desc);
aBitMap.Assign(dBitMap);
showmessage('Calculo Terminado');
end;

procedure TFrmHough.ProcuraNodo(var aBitMap : TBitMap);
begin
  while (desc^.Prev)^.Next <> nil do
  begin
    if (desc^.Prev)^.cont = r then
    begin
      if (desc^.Prev)^.checked = True then
        break
      else
        begin
          (desc^.Prev)^.checked := True;

```

```

        contchecked := contchecked - 1;
        CriaJanela(aBitMap);
        break;
    end;
end
else
    desc^.Prev := (desc^.Prev)^.Next;
end;
end;

procedure TfrmHough.CriaJanela(var aBitMap : TBitMap);
var
i,j,p,q : Integer;
a1,a2,b1,b2 : Double;
begin
// Este é o nodo central
x := (desc^.Prev)^.x;
y := (desc^.Prev)^.y;
Edit1.Text := floatToStr(x);
Edit2.Text := floatToStr(y);

x1 := x;
y1 := y;

// Atualiza descriptor (configuração inicial)
while (desc^.Prev)^.Prev <> nil do
    desc^.Prev := (desc^.Prev)^.Prev;

// Aqui começa a janela 3x3
// 1º pixel
// (decrementa x e y)
x := x - 1;
y := y - 1;
if aBitMap.Canvas.Pixels[x,y] = 0 then
begin
    while (desc^.Prev)^.Next <> nil do
    begin
        if ((desc^.Prev)^.x = x) and ((desc^.Prev)^.y = y) then
        begin
            x2 := (desc^.Prev)^.x;
            y2 := (desc^.Prev)^.y;
            Edit1.Text := IntToStr(x);
            Edit2.Text := IntToStr(y);
            if(x1 <> x2) then //Verifica se não há divisão por zero
            begin
                a := (y2 - y1) / (x2 - x1);    //Calcula a do ponto 2 em
                //relação ao ponto central
                b := y1 - x1 * ((y2 - y1) / (x2 - x1));
                //Calcula b do ponto 2 em relação ao ponto central da janela
                StringGrid1.RowCount := StringGrid1.RowCount + 1;
                StringGrid1.Cells[0,1] := FloatToStr(a);
                StringGrid1.Cells[1,1] := FloatToStr(b);
                StringGrid1.Cells[2,1] := FloatToStr(x1);
                StringGrid1.Cells[3,1] := FloatToStr(y1);
                StringGrid1.Cells[4,1] := FloatToStr(x2);
                StringGrid1.Cells[5,1] := FloatToStr(y2);

                StringGrid2.RowCount := StringGrid1.RowCount + 1;
                //Incrementa a tabela StringGrid2 em uma linha
            end;
        end;
    end;
end;

```

```

        StringGrid2.Cells[0, StringGrid2.RowCount - 1] :=
FloatToStr(a);
        StringGrid2.Cells[1, StringGrid2.RowCount - 1] :=
FloatToStr(b);
        StringGrid2.Cells[2, StringGrid2.RowCount - 1] :=
FloatToStr(x1);
        StringGrid2.Cells[3, StringGrid2.RowCount - 1] :=
FloatToStr(y1);
        StringGrid2.Cells[4, StringGrid2.RowCount - 1] :=
FloatToStr(x2);
        StringGrid2.Cells[5, StringGrid2.RowCount - 1] :=
FloatToStr(y2);
        //Insere os valores de a, b, x1, y1, x2, y2 na tabela

        if((StringGrid2.Cells[6, StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[6, StringGrid2.RowCount - 1]= ' '))then
            begin //Se na posição do contador não houver nenhum valor, é
colocado o valor 1 no mesmo
                StringGrid2.Cells[6, StringGrid2.RowCount - 1] :='1';
            end;
        p := StringGrid2.RowCount - 2;
        q := StringGrid2.RowCount - 1;
        newcont := 0;
        //Faz a comparação do contador da última linha da tabela com
as demais e troca
        //os valores para as mesmas linhas
        while(p >= 2) do
        begin
            if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
                begin
                    a1 := StrToInt(StringGrid2.Cells[0,p]);
                    b1 := StrToInt(StringGrid2.Cells[1,p]);
                    a2 :=
StrToInt(StringGrid2.Cells[0, StringGrid2.RowCount - 1]);
                    b2 :=
StrToInt(StringGrid2.Cells[1, StringGrid2.RowCount - 1]);
                    if((a1 = a2) and (b1 = b2)) then
                        begin
                            newcont := (StrToInt(StringGrid2.Cells[6,p]))+
(StrToInt(StringGrid2.Cells[6, StringGrid2.RowCount - 1]));
                            StringGrid2.Cells[6,p] := IntToStr(newcont);
                            StringGrid2.Cells[6, StringGrid2.RowCount - 1]
:= IntToStr(newcont);
                            break;
                        end;
                    end;
                    p := p - 1;
                end;
            break;
        end;

        if(x1 = x2) then //Caso seja uma divisão por zero
begin
        a := 1000; //a recebe um valor muito grande
        b := x2; //b recebe o valor de x2
        StringGrid1.RowCount := StringGrid1.RowCount + 1;
        StringGrid1.Cells[0,1] := FloatToStr(a);
        StringGrid1.Cells[1,1] := FloatToStr(b);
        StringGrid1.Cells[2,1] := FloatToStr(x1);

```

```

StringGrid1.Cells[3,1] := FloatToStr(y1);
StringGrid1.Cells[4,1] := FloatToStr(x2);
StringGrid1.Cells[5,1] := FloatToStr(y2);

StringGrid2.RowCount := StringGrid1.RowCount + 1;
//Incrementa a tabela StringGrid2 em uma linha
StringGrid2.Cells[0,StringGrid2.RowCount - 1] :=
FloatToStr(a);
StringGrid2.Cells[1,StringGrid2.RowCount - 1] :=
FloatToStr(b);
StringGrid2.Cells[2,StringGrid2.RowCount - 1] :=
FloatToStr(x1);
StringGrid2.Cells[3,StringGrid2.RowCount - 1] :=
FloatToStr(y1);
StringGrid2.Cells[4,StringGrid2.RowCount - 1] :=
FloatToStr(x2);
StringGrid2.Cells[5,StringGrid2.RowCount - 1] :=
FloatToStr(y2);
//Insere os valores de a, b, x1, y1, x2, y2 na tabela
if((StringGrid2.Cells[6,StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[6,StringGrid2.RowCount - 1]= ' '))then
begin //Se na posição do contador não houver nenhum valor, é
colocado o valor 1 no mesmo
StringGrid2.Cells[6,StringGrid2.RowCount - 1] :='1';
end;
//Faz a comparação do contador da última linha da tabela com
as demais e troca
//os valores para as mesmas linhas
p := StringGrid2.RowCount - 2;
q := StringGrid2.RowCount - 1;
newcont := 0;
while(p >= 2) do
begin
if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
begin
a1 := StrToFloat(StringGrid2.Cells[0,p]);
b1 := StrToFloat(StringGrid2.Cells[1,p]);
a2 :=
StrToFloat(StringGrid2.Cells[0,StringGrid2.RowCount - 1]);
b2 :=
StrToFloat(StringGrid2.Cells[1,StringGrid2.RowCount - 1]);
if((a1 = a2) and (b1 = b2)) then
begin
newcont := (StrToInt(StringGrid2.Cells[6,p])) +
(StrToInt(StringGrid2.Cells[6,StringGrid2.RowCount - 1]));
StringGrid2.Cells[6,p] := IntToStr(newcont);
StringGrid2.Cells[6,StringGrid2.RowCount - 1] :=
IntToStr(newcont);
break;
end;
p := p - 1;
end;
break;
end;

end
else
begin

```

```

        desc^.Prev := (desc^.Prev)^.Next;
    end;
end;
end;

// 2º pixel
// incrementa x de uma posição
x := x + 1;
if aBitMap.Canvas.Pixels[x,y] = 0 then
begin
    while (desc^.Prev)^.Next <> nil do
begin
    if ((desc^.Prev)^.x = x) and ((desc^.Prev)^.y = y) then
begin
    x2 := (desc^.Prev)^.x;
    y2 := (desc^.Prev)^.y;
    Edit1.Text := IntToStr(x);
    Edit2.Text := IntToStr(y);
    if(x1 <> x2) then
begin
        a1 := (y2 - y1) / (x2 - x1);
        b1 := y1 - x1 * ((y2 - y1) / (x2 - x1));
        StringGrid1.RowCount := StringGrid1.RowCount + 1;
        StringGrid1.Cells[0,2] := FloatToStr(a1);
        StringGrid1.Cells[1,2] := FloatToStr(b1);
        StringGrid1.Cells[2,2] := FloatToStr(x1);
        StringGrid1.Cells[3,2] := FloatToStr(y1);
        StringGrid1.Cells[4,2] := FloatToStr(x2);
        StringGrid1.Cells[5,2] := FloatToStr(y2);

        StringGrid2.RowCount := StringGrid1.RowCount + 1;
        StringGrid2.Cells[0,StringGrid2.RowCount - 1] :=
FloatToStr(a1);
        StringGrid2.Cells[1,StringGrid2.RowCount - 1] :=
FloatToStr(b1);
        StringGrid2.Cells[2,StringGrid2.RowCount - 1] :=
FloatToStr(x1);
        StringGrid2.Cells[3,StringGrid2.RowCount - 1] :=
FloatToStr(y1);
        StringGrid2.Cells[4,StringGrid2.RowCount - 1] :=
FloatToStr(x2);
        StringGrid2.Cells[5,StringGrid2.RowCount - 1] :=
FloatToStr(y2);

        if((StringGrid2.Cells[6,StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[6,StringGrid2.RowCount - 1]= ' '))then
begin
            StringGrid2.Cells[6,StringGrid2.RowCount - 1] :='1';
        end;
        p := StringGrid2.RowCount - 2;
        q := StringGrid2.RowCount - 1;
        newcont := 0;
        while(p >= 2) do
begin
            if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
begin
                a1 := StrToFloat(StringGrid2.Cells[0,p]);
                b1 := StrToFloat(StringGrid2.Cells[1,p]);
                a2 := StrToFloat(StringGrid2.Cells[0,q]);

```

```

        b2 := StrToFloat(StringGrid2.Cells[1,q]);
        if((a1 = a2) and (b1 = b2)) then
        begin
            newcont := StrToInt(StringGrid2.Cells[6,p])+
(StrToInt(StringGrid2.Cells[6,StringGrid2.RowCount - 1]));
            StringGrid2.Cells[6,p] := IntToStr(newcont);
            StringGrid2.Cells[6,StringGrid2.RowCount - 1]
:= IntToStr(newcont);
            break;
        end;
        p := p - 1;
    end;
    break;
end;

if(x1 = x2) then
begin
    a1 := 1000;
    b1 := x2;
    StringGrid1.RowCount := StringGrid1.RowCount + 1;
    StringGrid1.Cells[0,1] := FloatToStr(a1);
    StringGrid1.Cells[1,1] := FloatToStr(b1);
    StringGrid1.Cells[2,1] := FloatToStr(x1);
    StringGrid1.Cells[3,1] := FloatToStr(y1);
    StringGrid1.Cells[4,1] := FloatToStr(x2);
    StringGrid1.Cells[5,1] := FloatToStr(y2);

    StringGrid2.RowCount := StringGrid1.RowCount + 1;
    StringGrid2.Cells[0,StringGrid2.RowCount - 1] :=
FloatToStr(a1);
    StringGrid2.Cells[1,StringGrid2.RowCount - 1] :=
FloatToStr(b1);
    StringGrid2.Cells[2,StringGrid2.RowCount - 1] :=
FloatToStr(x1);
    StringGrid2.Cells[3,StringGrid2.RowCount - 1] :=
FloatToStr(y1);
    StringGrid2.Cells[4,StringGrid2.RowCount - 1] :=
FloatToStr(x2);
    StringGrid2.Cells[5,StringGrid2.RowCount - 1] :=
FloatToStr(y2);

    if((StringGrid2.Cells[6,StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[6,StringGrid2.RowCount - 1]= ' '))then
    begin
        StringGrid2.Cells[6,StringGrid2.RowCount - 1] :='1';
    end;
    p := StringGrid2.RowCount - 2;
    q := StringGrid2.RowCount - 1;
    newcont := 0;
    while(p >= 2) do
    begin
        if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
        begin
            a1 := StrToFloat(StringGrid2.Cells[0,p]);
            b1 := StrToFloat(StringGrid2.Cells[1,p]);
            a2 :=
StrToFloat(StringGrid2.Cells[0,StringGrid2.RowCount - 1]);
            StringGrid2.Cells[6,p] := IntToStr(newcont);
            StringGrid2.Cells[6,StringGrid2.RowCount - 1]
:= IntToStr(newcont);
            break;
        end;
        p := p - 1;
    end;
    break;
end;

```

```

        b2 :=
StrToFloat(StringGrid2.Cells[1, StringGrid2.RowCount - 1]);
        if((a1 = a2) and (b1 = b2)) then
begin
        newcont := (StrToInt(StringGrid2.Cells[6,p]))
+ (StrToInt(StringGrid2.Cells[6, StringGrid2.RowCount - 1]));
        StringGrid2.Cells[6,p] := IntToStr(newcont);
        StringGrid2.Cells[6, StringGrid2.RowCount - 1]
:= IntToStr(newcont);
        break;
end;
        p := p - 1;
end;
        break;
end;
else
begin
        desc^.Prev := (desc^.Prev)^.Next;
end;
end;
end;

// 3º pixel
// incrementa x de uma posição
x := x + 1;
if aBitMap.Canvas.Pixels[x,y] = 0 then
begin
while (desc^.Prev)^.Next <> nil do
begin
if ((desc^.Prev)^.x = x) and ((desc^.Prev)^.y = y) then
begin
x2 := (desc^.Prev)^.x;
y2 := (desc^.Prev)^.y;
Edit1.Text := IntToStr(x);
Edit2.Text := IntToStr(y);
if(x1 <> x2) then
begin
a2 := (y2 - y1) / (x2 - x1);
b2 := y1 - x1 * ((y2 - y1) / (x2 - x1));
StringGrid1.RowCount := StringGrid1.RowCount + 1;
StringGrid1.Cells[0,3] := FloatToStr(a2);
StringGrid1.Cells[1,3] := FloatToStr(b2);
StringGrid1.Cells[2,3] := FloatToStr(x1);
StringGrid1.Cells[3,3] := FloatToStr(y1);
StringGrid1.Cells[4,3] := FloatToStr(x2);
StringGrid1.Cells[5,3] := FloatToStr(y2);

StringGrid2.RowCount := StringGrid1.RowCount + 1;
StringGrid2.Cells[0, StringGrid2.RowCount - 1] :=
FloatToStr(a2);
StringGrid2.Cells[1, StringGrid2.RowCount - 1] :=
FloatToStr(b2);
StringGrid2.Cells[2, StringGrid2.RowCount - 1] :=
FloatToStr(x1);
StringGrid2.Cells[3, StringGrid2.RowCount - 1] :=
FloatToStr(y1);
StringGrid2.Cells[4, StringGrid2.RowCount - 1] :=
FloatToStr(x2);

```

```

        StringGrid2.Cells[5, StringGrid2.RowCount - 1] :=
FloatToStr(y2);
        if((StringGrid2.Cells[6, StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[6, StringGrid2.RowCount - 1]= ' '))then
begin
        StringGrid2.Cells[6, StringGrid2.RowCount - 1] :='1';
end;
p := StringGrid2.RowCount - 2;
q := StringGrid2.RowCount - 1;
newcont := 0;
while(p >= 2) do
begin
if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
begin
a1 := StrToFloat(StringGrid2.Cells[0,p]);
b1 := StrToFloat(StringGrid2.Cells[1,p]);
a2 := StrToFloat(StringGrid2.Cells[0,q]);
b2 := StrToFloat(StringGrid2.Cells[1,q]);
if((a1 = a2) and (b1 = b2)) then
begin
newcont := StrToInt(StringGrid2.Cells[6,p])+
(StrToInt(StringGrid2.Cells[6, StringGrid2.RowCount - 1]));
StringGrid2.Cells[6,p] := IntToStr(newcont);
StringGrid2.Cells[6, StringGrid2.RowCount - 1]
:= IntToStr(newcont);
break;
end;
p := p - 1;
end;
break;
end;

if(x1 = x2) then
begin
a2 := 1000;
b2 := x2;
StringGrid1.RowCount := StringGrid1.RowCount + 1;
StringGrid1.Cells[0,1] := FloatToStr(a2);
StringGrid1.Cells[1,1] := FloatToStr(b2);
StringGrid1.Cells[2,1] := FloatToStr(x1);
StringGrid1.Cells[3,1] := FloatToStr(y1);
StringGrid1.Cells[4,1] := FloatToStr(x2);
StringGrid1.Cells[5,1] := FloatToStr(y2);

StringGrid2.RowCount := StringGrid1.RowCount + 1;
StringGrid2.Cells[0, StringGrid2.RowCount - 1] :=
FloatToStr(a2);
StringGrid2.Cells[1, StringGrid2.RowCount - 1] :=
FloatToStr(b2);
StringGrid2.Cells[2, StringGrid2.RowCount - 1] :=
FloatToStr(x1);
StringGrid2.Cells[3, StringGrid2.RowCount - 1] :=
FloatToStr(y1);
StringGrid2.Cells[4, StringGrid2.RowCount - 1] :=
FloatToStr(x2);
StringGrid2.Cells[5, StringGrid2.RowCount - 1] :=
FloatToStr(y2);

```

```

        if((StringGrid2.Cells[6,StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[6,StringGrid2.RowCount - 1]= ' '))then
begin
        StringGrid2.Cells[6,StringGrid2.RowCount - 1] :='1';
end;
p := StringGrid2.RowCount - 2;
q := StringGrid2.RowCount - 1;
newcont := 0;
while(p >= 2) do
begin
        if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
begin
        a1 := StrToInt(StringGrid2.Cells[0,p]);
        b1 := StrToInt(StringGrid2.Cells[1,p]);
        a2 :=
StrToInt(StringGrid2.Cells[0,StringGrid2.RowCount - 1]);
        b2 :=
StrToInt(StringGrid2.Cells[1,StringGrid2.RowCount - 1]);
        if((a1 = a2) and (b1 = b2)) then
begin
        newcont := (StrToInt(StringGrid2.Cells[6,p]))+
(StrToInt(StringGrid2.Cells[6,StringGrid2.RowCount - 1]));
        StringGrid2.Cells[6,p] := IntToStr(newcont);
        StringGrid2.Cells[6,StringGrid2.RowCount - 1]
:= IntToStr(newcont);
        break;
end;
        p := p - 1;
end;
break;
end;
end
else
begin
        desc^.Prev := (desc^.Prev)^.Next;
end;
end;
end;

// 4º pixel (decrementa x de duas posições e incrementa y de uma posição)
x := x - 2;
y := y + 1;
if aBitMap.Canvas.Pixels[x,y] = 0 then
begin
while (desc^.Prev)^.Next <> nil do
begin
        if ((desc^.Prev)^.x = x) and ((desc^.Prev)^.y = y) then
begin
        x2 := (desc^.Prev)^.x;
        y2 := (desc^.Prev)^.y;
        Edit1.Text := IntToStr(x);
        Edit2.Text := IntToStr(y);
        if(xl <> x2) then
begin
        a3 := (y2 - y1) / (x2 - x1);
        b3 := y1 - x1 * ((y2 - y1) / (x2 - x1));
        StringGrid1.RowCount := StringGrid1.RowCount + 1;
        StringGrid1.Cells[0,4] := FloatToStr(a3);

```

```

StringGrid1.Cells[1,4] := FloatToStr(b3);
StringGrid1.Cells[2,4] := FloatToStr(x1);
StringGrid1.Cells[3,4] := FloatToStr(y1);
StringGrid1.Cells[4,4] := FloatToStr(x2);
StringGrid1.Cells[5,4] := FloatToStr(y2);

StringGrid2.RowCount := StringGrid1.RowCount + 1;
StringGrid2.Cells[0,StringGrid2.RowCount - 1] :=
FloatToStr(a3);
StringGrid2.Cells[1,StringGrid2.RowCount - 1] :=
FloatToStr(b3);
StringGrid2.Cells[2,StringGrid2.RowCount - 1] :=
FloatToStr(x1);
StringGrid2.Cells[3,StringGrid2.RowCount - 1] :=
FloatToStr(y1);
StringGrid2.Cells[4,StringGrid2.RowCount - 1] :=
FloatToStr(x2);
StringGrid2.Cells[5,StringGrid2.RowCount - 1] :=
FloatToStr(y2);
if((StringGrid2.Cells[6,StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[6,StringGrid2.RowCount - 1]= ' '))then
begin
    StringGrid2.Cells[6,StringGrid2.RowCount - 1] :='1';
end;
p := StringGrid2.RowCount - 2;
q := StringGrid2.RowCount - 1;
newcont := 0;
while(p >= 2) do
begin
    if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
        begin
            a1 := StrToFloat(StringGrid2.Cells[0,p]);
            b1 := StrToFloat(StringGrid2.Cells[1,p]);
            a2 := StrToFloat(StringGrid2.Cells[0,q]);
            b2 := StrToFloat(StringGrid2.Cells[1,q]);
            if((a1 = a2) and (b1 = b2)) then
                begin
                    newcont := StrToInt(StringGrid2.Cells[6,p])+
(StrToInt(StringGrid2.Cells[6,StringGrid2.RowCount - 1]));
                    StringGrid2.Cells[6,p] := IntToStr(newcont);
                    StringGrid2.Cells[6,StringGrid2.RowCount - 1]
:= IntToStr(newcont);
                    break;
                end;
            end;
            p := p - 1;
        end;
    break;
end;

if(x1 = x2) then
begin
    a3 := 1000;
    b3 := x2;
    StringGrid1.RowCount := StringGrid1.RowCount + 1;
    StringGrid1.Cells[0,1] := FloatToStr(a3);
    StringGrid1.Cells[1,1] := FloatToStr(b3);

```

```

StringGrid1.Cells[2,1] := FloatToStr(x1);
StringGrid1.Cells[3,1] := FloatToStr(y1);
StringGrid1.Cells[4,1] := FloatToStr(x2);
StringGrid1.Cells[5,1] := FloatToStr(y2);

StringGrid2.RowCount := StringGrid1.RowCount + 1;
StringGrid2.Cells[0,StringGrid2.RowCount - 1] :=
FloatToStr(a3);
StringGrid2.Cells[1,StringGrid2.RowCount - 1] :=
FloatToStr(b3);
StringGrid2.Cells[2,StringGrid2.RowCount - 1] :=
FloatToStr(x1);
StringGrid2.Cells[3,StringGrid2.RowCount - 1] :=
FloatToStr(y1);
StringGrid2.Cells[4,StringGrid2.RowCount - 1] :=
FloatToStr(x2);
StringGrid2.Cells[5,StringGrid2.RowCount - 1] :=
FloatToStr(y2);

if((StringGrid2.Cells[6,StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[6,StringGrid2.RowCount - 1]= ' '))then
begin
    StringGrid2.Cells[6,StringGrid2.RowCount - 1] :='1';
end;
p := StringGrid2.RowCount - 2;
q := StringGrid2.RowCount - 1;
newcont := 0;
while(p >= 2) do
begin
    if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
        begin
            a1 := StrToFloat(StringGrid2.Cells[0,p]);
            b1 := StrToFloat(StringGrid2.Cells[1,p]);
            a2 :=
StrToFloat(StringGrid2.Cells[0,StringGrid2.RowCount - 1]);
            b2 :=
StrToFloat(StringGrid2.Cells[1,StringGrid2.RowCount - 1]);
            if((a1 = a2) and (b1 = b2)) then
                begin
                    newcont := (StrToInt(StringGrid2.Cells[6,p]))+
(StrToInt(StringGrid2.Cells[6,StringGrid2.RowCount - 1]));
                    StringGrid2.Cells[6,p] := IntToStr(newcont);
                    StringGrid2.Cells[6,StringGrid2.RowCount - 1]
:= IntToStr(newcont);
                    break;
                end;
            p := p - 1;
        end;
    break;
end;
end
else
begin
    desc^.Prev := (desc^.Prev)^.Next;
end;
end;
end;

```

```

// 5º pixel (incrementa x de duas posições)
x := x + 2;
if aBitMap.Canvas.Pixels[x,y] = 0 then
begin
  while (desc^.Prev)^.Next <> nil do
  begin
    if ((desc^.Prev)^.x = x) and ((desc^.Prev)^.y = y) then
    begin
      x2 := (desc^.Prev)^.x;
      y2 := (desc^.Prev)^.y;
      Edit1.Text := IntToStr(x);
      Edit2.Text := IntToStr(y);
      if(x1 <> x2) then
      begin
        a4 := (y2 - y1) / (x2 - x1);
        b4 := y1 - x1 * ((y2 - y1) / (x2 - x1));
        StringGrid1.RowCount := StringGrid1.RowCount + 1;
        StringGrid1.Cells[0,5] := FloatToStr(a4);
        StringGrid1.Cells[1,5] := FloatToStr(b4);
        StringGrid1.Cells[2,5] := FloatToStr(x1);
        StringGrid1.Cells[3,5] := FloatToStr(y1);
        StringGrid1.Cells[4,5] := FloatToStr(x2);
        StringGrid1.Cells[5,5] := FloatToStr(y2);

        StringGrid2.RowCount := StringGrid1.RowCount + 1;
        StringGrid2.Cells[0,StringGrid2.RowCount - 1] :=
FloatToStr(a4);
        StringGrid2.Cells[1,StringGrid2.RowCount - 1] :=
FloatToStr(b4);
        StringGrid2.Cells[2,StringGrid2.RowCount - 1] :=
FloatToStr(x1);
        StringGrid2.Cells[3,StringGrid2.RowCount - 1] :=
FloatToStr(y1);
        StringGrid2.Cells[4,StringGrid2.RowCount - 1] :=
FloatToStr(x2);
        StringGrid2.Cells[5,StringGrid2.RowCount - 1] :=
FloatToStr(y2);
        if((StringGrid2.Cells[6,StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[6,StringGrid2.RowCount - 1]= ' '))then
        begin
          StringGrid2.Cells[6,StringGrid2.RowCount - 1] := '1';
        end;
        p := StringGrid2.RowCount - 2;
        q := StringGrid2.RowCount - 1;
        newcont := 0;
        while(p >= 2) do
        begin
          if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
          begin
            a1 := StrToFloat(StringGrid2.Cells[0,p]);
            b1 := StrToFloat(StringGrid2.Cells[1,p]);
            a2 := StrToFloat(StringGrid2.Cells[0,q]);
            b2 := StrToFloat(StringGrid2.Cells[1,q]);
            if((a1 = a2) and (b1 = b2)) then
            begin
              newcont := StrToInt(StringGrid2.Cells[6,p])+
(StrToInt(StringGrid2.Cells[6,StringGrid2.RowCount - 1]));
              StringGrid2.Cells[6,p] := IntToStr(newcont);
            end;
          end;
        end;
      end;
    end;
  end;
end;

```

```

StringGrid2.Cells[6, StringGrid2.RowCount - 1]
:= IntToStr(newcont);
begin
  if (x1 = x2) then
    begin
      a4 := 1000;
      b4 := x2;
      StringGrid1.RowCount := StringGrid1.RowCount + 1;
      StringGrid1.Cells[0,1] := FloatToStr(a4);
      StringGrid1.Cells[1,1] := FloatToStr(b4);
      StringGrid1.Cells[2,1] := FloatToStr(x1);
      StringGrid1.Cells[3,1] := FloatToStr(y1);
      StringGrid1.Cells[4,1] := FloatToStr(x2);
      StringGrid1.Cells[5,1] := FloatToStr(y2);

      StringGrid2.RowCount := StringGrid1.RowCount + 1;
      StringGrid2.Cells[0, StringGrid2.RowCount - 1] :=
      FloatToStr(a4);
      StringGrid2.Cells[1, StringGrid2.RowCount - 1] :=
      FloatToStr(b4);
      StringGrid2.Cells[2, StringGrid2.RowCount - 1] :=
      FloatToStr(x1);
      StringGrid2.Cells[3, StringGrid2.RowCount - 1] :=
     FloatToStr(y1);
      StringGrid2.Cells[4, StringGrid2.RowCount - 1] :=
      FloatToStr(x2);
      StringGrid2.Cells[5, StringGrid2.RowCount - 1] :=
      FloatToStr(y2);

      if ((StringGrid2.Cells[6, StringGrid2.RowCount -
      1]='')or(StringGrid2.Cells[6, StringGrid2.RowCount - 1]= ' '))then
        begin
          StringGrid2.Cells[6, StringGrid2.RowCount - 1] := '1';
        end;
      p := StringGrid2.RowCount - 2;
      q := StringGrid2.RowCount - 1;
      newcont := 0;
      while(p >= 2) do
        begin
          if((StringGrid2.Cells[0,p] <>'') and
          (StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
          (StringGrid2.Cells[1,q] <>'')) then
            begin
              a1 := StrToFloat(StringGrid2.Cells[0,p]);
              b1 := StrToFloat(StringGrid2.Cells[1,p]);
              a2 :=
              StrToFloat(StringGrid2.Cells[0, StringGrid2.RowCount - 1]);
              b2 :=
              StrToFloat(StringGrid2.Cells[1, StringGrid2.RowCount - 1]);
              if((a1 = a2) and (b1 = b2)) then
                begin
                  newcont := (StrToInt(StringGrid2.Cells[6,p])) +
                  (StrToInt(StringGrid2.Cells[6, StringGrid2.RowCount - 1]));
                  StringGrid2.Cells[6,p] := IntToStr(newcont);
                end;
            end;
        end;
      end;
    end;
  end;
end;

```

```

StringGrid2.Cells[6, StringGrid2.RowCount - 1]
:= IntToStr(newcont);
begin
  if newcont <> '' then
    begin
      if (StringGrid2.Cells[6, StringGrid2.RowCount - 1] = '1') or
         (StringGrid2.Cells[6, StringGrid2.RowCount - 1] = ' ') then
        begin
          StringGrid2.Cells[6, StringGrid2.RowCount - 1] := '0';
          break;
        end;
      end;
      p := p + 1;
    end;
  end;
end;

// 6º pixel
// decrementa x de duas posições e incrementa y de uma posição
x := x - 2;
y := y + 1;
if aBitMap.Canvas.Pixels[x,y] = 0 then
begin
  while (desc^.Prev)^.Next <> nil do
  begin
    if ((desc^.Prev)^.x = x) and ((desc^.Prev)^.y = y) then
    begin
      x2 := (desc^.Prev)^.x;
      y2 := (desc^.Prev)^.y;
      Edit1.Text := IntToStr(x);
      Edit2.Text := IntToStr(y);
      if(x1 <> x2) then
      begin
        a5 := (y2 - y1) / (x2 - x1);
        b5 := y1 - x1 * ((y2 - y1) / (x2 - x1));
        StringGrid1.RowCount := StringGrid1.RowCount + 1;
        StringGrid1.Cells[0,6] := FloatToStr(a5);
        StringGrid1.Cells[1,6] := FloatToStr(b5);
        StringGrid1.Cells[2,6] := FloatToStr(x1);
        StringGrid1.Cells[3,6] := FloatToStr(y1);
        StringGrid1.Cells[4,6] := FloatToStr(x2);
        StringGrid1.Cells[5,6] := FloatToStr(y2);

        StringGrid2.RowCount := StringGrid1.RowCount + 1;
        StringGrid2.Cells[0, StringGrid2.RowCount - 1] :=
FloatToStr(a5);
        StringGrid2.Cells[1, StringGrid2.RowCount - 1] :=
FloatToStr(b5);
        StringGrid2.Cells[2, StringGrid2.RowCount - 1] :=
FloatToStr(x1);
        StringGrid2.Cells[3, StringGrid2.RowCount - 1] :=
FloatToStr(y1);
        StringGrid2.Cells[4, StringGrid2.RowCount - 1] :=
FloatToStr(x2);
        StringGrid2.Cells[5, StringGrid2.RowCount - 1] :=
FloatToStr(y2);
        if((StringGrid2.Cells[6, StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[6, StringGrid2.RowCount - 1]=' '))then
          begin
            StringGrid2.Cells[6, StringGrid2.RowCount - 1] := '1';
          end;
      end;
    end;
  end;
end;

```

```

p := StringGrid2.RowCount - 2;
q := StringGrid2.RowCount - 1;
newcont := 0;
while(p >= 2) do
begin
  if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
    begin
      a1 := StrToInt(StringGrid2.Cells[0,p]);
      b1 := StrToInt(StringGrid2.Cells[1,p]);
      a2 := StrToInt(StringGrid2.Cells[0,q]);
      b2 := StrToInt(StringGrid2.Cells[1,q]);
      if((a1 = a2) and (b1 = b2)) then
        begin
          newcont := StrToInt(StringGrid2.Cells[6,p]) +
(StrToInt(StringGrid2.Cells[6,StringGrid2.RowCount - 1]));
          StringGrid2.Cells[6,p] := IntToStr(newcont);
          StringGrid2.Cells[6,StringGrid2.RowCount - 1]
:= IntToStr(newcont);
          break;
        end;
      end;
      p := p - 1;
    end;
  break;
end;

if(x1 = x2) then
begin
  a5 := 1000;
  b5 := x2;
  StringGrid1.RowCount := StringGrid1.RowCount + 1;
  StringGrid1.Cells[0,1] := FloatToStr(a5);
  StringGrid1.Cells[1,1] := FloatToStr(b5);
  StringGrid1.Cells[2,1] := FloatToStr(x1);
  StringGrid1.Cells[3,1] := FloatToStr(y1);
  StringGrid1.Cells[4,1] := FloatToStr(x2);
  StringGrid1.Cells[5,1] := FloatToStr(y2);

  StringGrid2.RowCount := StringGrid1.RowCount + 1;
  StringGrid2.Cells[0,StringGrid2.RowCount - 1] :=
FloatToStr(a5);
  StringGrid2.Cells[1,StringGrid2.RowCount - 1] :=
FloatToStr(b5);
  StringGrid2.Cells[2,StringGrid2.RowCount - 1] :=
FloatToStr(x1);
  StringGrid2.Cells[3,StringGrid2.RowCount - 1] :=
FloatToStr(y1);
  StringGrid2.Cells[4,StringGrid2.RowCount - 1] :=
FloatToStr(x2);
  StringGrid2.Cells[5,StringGrid2.RowCount - 1] :=
FloatToStr(y2);

  if((StringGrid2.Cells[6,StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[6,StringGrid2.RowCount - 1]= ' '))then
begin
  StringGrid2.Cells[6,StringGrid2.RowCount - 1] :='1';
end;
p := StringGrid2.RowCount - 2;
q := StringGrid2.RowCount - 1;

```

```

    newcont := 0;
    while(p >= 2) do
    begin
        if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
            begin
                a1 := StrToFloat(StringGrid2.Cells[0,p]);
                b1 := StrToFloat(StringGrid2.Cells[1,p]);
                a2 :=
StrToFloat(StringGrid2.Cells[0, StringGrid2.RowCount - 1]);
                b2 :=
StrToFloat(StringGrid2.Cells[1, StringGrid2.RowCount - 1]);
                if((a1 = a2) and (b1 = b2)) then
                    begin
                        newcont := (StrToInt(StringGrid2.Cells[6,p]))+
(StrToInt(StringGrid2.Cells[6, StringGrid2.RowCount - 1]));
                        StringGrid2.Cells[6,p] := IntToStr(newcont);
                        StringGrid2.Cells[6, StringGrid2.RowCount - 1]
:= IntToStr(newcont);
                        break;
                    end;
                end;
                p := p - 1;
            end;
            break;
        end;
    end
else
begin
    desc^.Prev := (desc^.Prev)^.Next;
end;
end;
end;

// 7º pixel
// incrementa x de uma posição
x := x + 1;
if aBitMap.Canvas.Pixels[x,y] = 0 then
begin
    while (desc^.Prev)^.Next <> nil do
begin
    if ((desc^.Prev)^.x = x) and ((desc^.Prev)^.y = y) then
begin
        x2 := (desc^.Prev)^.x;
        y2 := (desc^.Prev)^.y;
        Edit1.Text := IntToStr(x);
        Edit2.Text := IntToStr(y);
        if(x1 <> x2) then
begin
            a6 := (y2 - y1) / (x2 - x1);
            b6 := y1 - x1 * ((y2 - y1) / (x2 - x1));
            StringGrid1.RowCount := StringGrid1.RowCount + 1;
            StringGrid1.Cells[0,7] := FloatToStr(a6);
            StringGrid1.Cells[1,7] := FloatToStr(b6);
            StringGrid1.Cells[2,7] := FloatToStr(x1);
            StringGrid1.Cells[3,7] := FloatToStr(y1);
            StringGrid1.Cells[4,7] := FloatToStr(x2);
            StringGrid1.Cells[5,7] := FloatToStr(y2);

            StringGrid2.RowCount := StringGrid1.RowCount + 1;

```

```

        StringGrid2.Cells[0, StringGrid2.RowCount - 1] :=
FloatToStr(a6);
        StringGrid2.Cells[1, StringGrid2.RowCount - 1] :=
FloatToStr(b6);
        StringGrid2.Cells[2, StringGrid2.RowCount - 1] :=
FloatToStr(x1);
        StringGrid2.Cells[3, StringGrid2.RowCount - 1] :=
FloatToStr(y1);
        StringGrid2.Cells[4, StringGrid2.RowCount - 1] :=
FloatToStr(x2);
        StringGrid2.Cells[5, StringGrid2.RowCount - 1] :=
FloatToStr(y2);
        if((StringGrid2.Cells[6, StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[6, StringGrid2.RowCount - 1]= ' '))then
begin
        StringGrid2.Cells[6, StringGrid2.RowCount - 1] :='1';
end;
p := StringGrid2.RowCount - 2;
q := StringGrid2.RowCount - 1;
newcont := 0;
while(p >= 2) do
begin
        if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
begin
        a1 := StrToFloat(StringGrid2.Cells[0,p]);
        b1 := StrToFloat(StringGrid2.Cells[1,p]);
        a2 := StrToFloat(StringGrid2.Cells[0,q]);
        b2 := StrToFloat(StringGrid2.Cells[1,q]);
        if((a1 = a2) and (b1 = b2)) then
begin
        newcont := StrToInt(StringGrid2.Cells[6,p])+
(StrToInt(StringGrid2.Cells[6, StringGrid2.RowCount - 1]));
        StringGrid2.Cells[6,p] := IntToStr(newcont);
        StringGrid2.Cells[6, StringGrid2.RowCount - 1]
:= IntToStr(newcont);
        break;
end;
end;
p := p - 1;
end;
break;
end;

if(x1 = x2) then
begin
        a6 := 1000;
        b6 := x2;
        StringGrid1.RowCount := StringGrid1.RowCount + 1;
        StringGrid1.Cells[0,1] := FloatToStr(a6);
        StringGrid1.Cells[1,1] := FloatToStr(b6);
        StringGrid1.Cells[2,1] := FloatToStr(x1);
        StringGrid1.Cells[3,1] := FloatToStr(y1);
        StringGrid1.Cells[4,1] := FloatToStr(x2);
        StringGrid1.Cells[5,1] := FloatToStr(y2);

        StringGrid2.RowCount := StringGrid1.RowCount + 1;
        StringGrid2.Cells[0, StringGrid2.RowCount - 1] :=
FloatToStr(a6);

```

```

        StringGrid2.Cells[1, StringGrid2.RowCount - 1] :=
FloatToStr(b6);
        StringGrid2.Cells[2, StringGrid2.RowCount - 1] :=
FloatToStr(x1);
        StringGrid2.Cells[3, StringGrid2.RowCount - 1] :=
FloatToStr(y1);
        StringGrid2.Cells[4, StringGrid2.RowCount - 1] :=
FloatToStr(x2);
        StringGrid2.Cells[5, StringGrid2.RowCount - 1] :=
FloatToStr(y2);

        if((StringGrid2.Cells[6, StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[6, StringGrid2.RowCount - 1]= ' '))then
begin
        StringGrid2.Cells[6, StringGrid2.RowCount - 1] := '1';
end;
p := StringGrid2.RowCount - 2;
q := StringGrid2.RowCount - 1;
newcont := 0;
while(p >= 2) do
begin
        if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
begin
        a1 := StrToFloat(StringGrid2.Cells[0,p]);
        b1 := StrToFloat(StringGrid2.Cells[1,p]);
        a2 :=
StrToFloat(StringGrid2.Cells[0, StringGrid2.RowCount - 1]);
        b2 :=
StrToFloat(StringGrid2.Cells[1, StringGrid2.RowCount - 1]);
        if((a1 = a2) and (b1 = b2)) then
begin
        newcont := (StrToInt(StringGrid2.Cells[6,p]))+
(StrToInt(StringGrid2.Cells[6, StringGrid2.RowCount - 1]));
        StringGrid2.Cells[6,p] := IntToStr(newcont);
        StringGrid2.Cells[6, StringGrid2.RowCount - 1] :=
IntToStr(newcont);
        break;
end;
        end;
        p := p - 1;
end;
break;
end;
end;
else
begin
        desc^.Prev := (desc^.Prev)^.Next;
end;
end;
end;

// 8° pixel
// incrementa x de uma posição
x := x + 1;
if aBitmap.Canvas.Pixels[x,y] = 0 then
begin
while (desc^.Prev)^.Next <> nil do
begin

```

```

if ((desc^.Prev)^.x = x) and ((desc^.Prev)^.y = y) then
begin
  x2 := (desc^.Prev)^.x;
  y2 := (desc^.Prev)^.y;
  Edit1.Text := IntToStr(x);
  Edit2.Text := IntToStr(y);
  if(x1 <> x2) then
  begin
    a7 := (y2 - y1) / (x2 - x1);
    b7 := y1 - x1 * ((y2 - y1) / (x2 - x1));
    StringGrid1.RowCount := StringGrid1.RowCount + 1;
    StringGrid1.Cells[0,8] := FloatToStr(a7);
    StringGrid1.Cells[1,8] := FloatToStr(b7);
    StringGrid1.Cells[2,8] := FloatToStr(x1);
    StringGrid1.Cells[3,8] := FloatToStr(y1);
    StringGrid1.Cells[4,8] := FloatToStr(x2);
    StringGrid1.Cells[5,8] := FloatToStr(y2);

    StringGrid2.RowCount := StringGrid1.RowCount + 1;
    StringGrid2.Cells[0,StringGrid2.RowCount - 1] :=
FloatToStr(a7);
    StringGrid2.Cells[1,StringGrid2.RowCount - 1] :=
FloatToStr(b7);
    StringGrid2.Cells[2,StringGrid2.RowCount - 1] :=
FloatToStr(x1);
    StringGrid2.Cells[3,StringGrid2.RowCount - 1] :=
FloatToStr(y1);
    StringGrid2.Cells[4,StringGrid2.RowCount - 1] :=
FloatToStr(x2);
    StringGrid2.Cells[5,StringGrid2.RowCount - 1] :=
FloatToStr(y2);
    if((StringGrid2.Cells[6,StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[6,StringGrid2.RowCount - 1]= ' '))then
      begin
        StringGrid2.Cells[6,StringGrid2.RowCount - 1] :='1';
      end;
    p := StringGrid2.RowCount - 2;
    q := StringGrid2.RowCount - 1;
    newcont := 0;
    while(p >= 2) do
    begin
      if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
        begin
          a1 := StrToFloat(StringGrid2.Cells[0,p]);
          b1 := StrToFloat(StringGrid2.Cells[1,p]);
          a2 := StrToFloat(StringGrid2.Cells[0,q]);
          b2 := StrToFloat(StringGrid2.Cells[1,q]);
          if((a1 = a2) and (b1 = b2)) then
            begin
              newcont := StrToInt(StringGrid2.Cells[6,p])+
(StrToInt(StringGrid2.Cells[6,StringGrid2.RowCount - 1]));
              StringGrid2.Cells[6,p] := IntToStr(newcont);
              StringGrid2.Cells[6,StringGrid2.RowCount - 1]
:= IntToStr(newcont);
              break;
            end;
          p := p - 1;
        end;
    end;
  end;
end;

```

```

        break;
end;

if(x1 = x2) then
begin
  a7 := 1000;
  b7 := x2;
  StringGrid1.RowCount := StringGrid1.RowCount + 1;
  StringGrid1.Cells[0,1] := FloatToStr(a7);
  StringGrid1.Cells[1,1] := FloatToStr(b7);
  StringGrid1.Cells[2,1] := FloatToStr(x1);
  StringGrid1.Cells[3,1] := FloatToStr(y1);
  StringGrid1.Cells[4,1] := FloatToStr(x2);
  StringGrid1.Cells[5,1] := FloatToStr(y2);

  StringGrid2.RowCount := StringGrid1.RowCount + 1;
  StringGrid2.Cells[0,StringGrid2.RowCount - 1] :=
FloatToStr(a7);
  StringGrid2.Cells[1,StringGrid2.RowCount - 1] :=
FloatToStr(b7);
  StringGrid2.Cells[2,StringGrid2.RowCount - 1] :=
FloatToStr(x1);
  StringGrid2.Cells[3,StringGrid2.RowCount - 1] :=
FloatToStr(y1);
  StringGrid2.Cells[4,StringGrid2.RowCount - 1] :=
FloatToStr(x2);
  StringGrid2.Cells[5,StringGrid2.RowCount - 1] :=
FloatToStr(y2);

  if((StringGrid2.Cells[6,StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[6,StringGrid2.RowCount - 1]= ' '))then
begin
  StringGrid2.Cells[6,StringGrid2.RowCount - 1] :='1';
end;
p := StringGrid2.RowCount - 2;
q := StringGrid2.RowCount - 1;
newcont := 0;
while(p >= 2) do
begin
  if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
  begin
    a1 := StrToFloat(StringGrid2.Cells[0,p]);
    b1 := StrToFloat(StringGrid2.Cells[1,p]);
    a2 :=
StrToFloat(StringGrid2.Cells[0,StringGrid2.RowCount - 1]);
    b2 :=
StrToFloat(StringGrid2.Cells[1,StringGrid2.RowCount - 1]);
    if((a1 = a2) and (b1 = b2)) then
    begin
      newcont := (StrToInt(StringGrid2.Cells[6,p])) +
(StrToInt(StringGrid2.Cells[6,StringGrid2.RowCount - 1]));
      StringGrid2.Cells[6,p] := IntToStr(newcont);
      StringGrid2.Cells[6,StringGrid2.RowCount - 1]
:= IntToStr(newcont);
      break;
    end;
    p := p - 1;
  end;
end;

```

```

        break;
    end;
end
else
begin
    desc^.Prev := (desc^.Prev)^.Next;
end;
end;
end;
p := 0;
q := 0;
newcont := 0;

// Atualiza descritor (configuração inicial)
while (desc^.Prev)^.Prev <> nil do
    desc^.Prev := (desc^.Prev)^.Prev;

end;

procedure TfrmHough.AcumulaoInicial1Click(Sender: TObject);
var
    i,j,c,d,e,n,temp,ccont1,ccont2,ccont3,ccont4,igualdade : Integer;
begin
    c := 1;
    StringGrid5.ColCount := 7;
    igualdade := 0;
    StringGrid5.RowCount := 2;
    //Se o valor da acumulação da reta for maior, insere a mesma na tabela
    StringGrid5.Cells[0,1] := StringGrid2.Cells[0,2];
    StringGrid5.Cells[1,1] := StringGrid2.Cells[1,2];
    StringGrid5.Cells[2,1] := StringGrid2.Cells[2,2];
    StringGrid5.Cells[3,1] := StringGrid2.Cells[3,2];
    StringGrid5.Cells[4,1] := StringGrid2.Cells[4,2];
    StringGrid5.Cells[5,1] := StringGrid2.Cells[5,2];
    StringGrid5.Cells[6,1] := StringGrid2.Cells[6,2];

    for i := 3 to (StringGrid2.RowCount - 1) do
begin
    for j := 0 to (StringGrid5.RowCount - 1) do
begin
    if(StringGrid5.Cells[0,j] = StringGrid2.Cells[0,i]) then
begin
    if(StringGrid5.Cells[1,j] = StringGrid2.Cells[1,i]) then
begin
        c := 0;
        ccont1 := StrToInt(StringGrid5.Cells[6,j]);
        ccont2 := StrToInt(StringGrid2.Cells[6,i]);
        if(ccont1 < ccont2) then
begin
            StringGrid5.Cells[6,j] := StringGrid2.Cells[6,i];
        end;

        break;
    end;
end;
end;
if(StrToInt(StringGrid2.Cells[6,i]) > 2) then
begin
    if(c = 1) then
begin
    StringGrid5.RowCount := StringGrid5.RowCount + 1;

```

```

        StringGrid5.Cells[0, StringGrid5.RowCount - 1] :=
StringGrid2.Cells[0,i];
        StringGrid5.Cells[1, StringGrid5.RowCount - 1] :=
StringGrid2.Cells[1,i];
        StringGrid5.Cells[2, StringGrid5.RowCount - 1] :=
StringGrid2.Cells[2,i];
        StringGrid5.Cells[3, StringGrid5.RowCount - 1] :=
StringGrid2.Cells[3,i];
        StringGrid5.Cells[4, StringGrid5.RowCount - 1] :=
StringGrid2.Cells[4,i];
        StringGrid5.Cells[5, StringGrid5.RowCount - 1] :=
StringGrid2.Cells[5,i];
        StringGrid5.Cells[6, StringGrid5.RowCount - 1] :=
StringGrid2.Cells[6,i];
      end;
      c := 1;
    end;
  end;
StringGrid6.ColCount := StringGrid5.RowCount;
StringGrid6.RowCount := 1;
for i := 0 to (StringGrid6.ColCount-1) do
begin
  StringGrid6.Cells[i,0] := '0';
end;
for i := 1 to (StringGrid5.RowCount - 1) do
begin
  StringGrid6.Cells[i-1,0] := StringGrid5.Cells[6,i];
end;
for i := 0 to (StringGrid6.colCount - 2) do
begin
  for n:=i+1 to StringGrid6.colCount -1 do
  begin

if(strtoint(StringGrid6.Cells[n,0])>=(strtoint(StringGrid6.Cells[i,0])))then
n
begin
  temp:=strtoint(StringGrid6.Cells[i,0]);
  StringGrid6.Cells[i,0]:=StringGrid6.Cells[n,0];
  StringGrid6.Cells[n,0]:=inttostr(temp);
end;
end;
StringGrid7.RowCount := StringGrid5.RowCount;
StringGrid7.ColCount := 3;
ccont4 := 1;
for i := 0 to (StringGrid6.ColCount - 1) do
begin

  for j := 1 to (StringGrid5.RowCount - 1) do
  begin
    if(StringGrid5.Cells[6,j] = StringGrid6.Cells[i,0]) then
    begin
      if(ccont4 = 1) then
      begin
        StringGrid7.Cells[0,ccont4] :=
StringGrid5.Cells[0,j];
        StringGrid7.Cells[1,ccont4] :=
StringGrid5.Cells[1,j];
        StringGrid7.Cells[2,ccont4] :=
StringGrid5.Cells[6,j];
      end;
    end;
  end;

```

```

        end;
        if(ccont4 > 1) then
        begin
            if(StringGrid7.Cells[0,ccont4] =
StringGrid7.Cells[0,ccont4 - 1]) then
                begin
                    igualdade := igualdade + 1;
                end;
                if(StringGrid7.Cells[1,ccont4] =
StringGrid7.Cells[1,ccont4 - 1]) then
                    begin
                        igualdade := igualdade + 1;
                    end;
                    if(igualdade < 2) then
                    begin
                        StringGrid7.Cells[0,ccont4] :=
StringGrid5.Cells[0,j];
                        StringGrid7.Cells[1,ccont4] :=
StringGrid5.Cells[1,j];
                        StringGrid7.Cells[2,ccont4] :=
StringGrid5.Cells[6,j];
                        igualdade := 0;
                    end;
                    if(igualdade = 2) then
                    begin
                        igualdade := 0;
                    end;
                end;
            end;
        end;
    end;
    ccont4 := ccont4 + 1;
end;
for i := 0 to (StringGrid5.ColCount - 1) do
begin
    for j := 0 to (StringGrid5.RowCount - 1) do
    begin
        StringGrid5.Cells[i,j] := '';
    end;
end;
for i := 0 to (StringGrid6.ColCount - 1) do
begin
    for j := 0 to (StringGrid6.RowCount - 1) do
    begin
        StringGrid6.Cells[i,j] := '';
    end;
end;
StringGrid5.ColCount := 0;
StringGrid5.RowCount := 0;
StringGrid6.ColCount := 0;
StringGrid6.RowCount := 0;

StringGrid11.ColCount := 3;
StringGrid11.RowCount := 3;
StringGrid11.Cells[0,0] := 'a';
StringGrid11.Cells[1,0] := 'b';
StringGrid11.Cells[2,0] := 'Contador';
ccont3 := 0;

```

```

for i := 1 to (StringGrid7.RowCount - 1) do
begin
  ccont4 := 0;
  if(StringGrid11.RowCount = 3) then
  begin
    StringGrid11.Cells[0,1] := StringGrid7.Cells[0,1];
    StringGrid11.Cells[1,1] := StringGrid7.Cells[1,1];
    StringGrid11.Cells[2,1] := StringGrid7.Cells[2,1];
    ccont3 := 1;
  end;
  if(ccont3 = 1) then
  begin

    for j := 1 to (StringGrid11.RowCount - 1) do
    begin
      if((StringGrid11.Cells[0,j] =
StringGrid7.Cells[0,i])and(StringGrid11.Cells[1,j] =
StringGrid7.Cells[1,i])) then
      begin
        ccont4 := 1;
        //ShowMessage('teste ' + IntToStr(ccont4));
        break;
      end;

      end;
      if(ccont4 = 0) then
      begin
        StringGrid11.RowCount := StringGrid11.RowCount + 1;
        StringGrid11.Cells[0,(StringGrid11.RowCount - 2)] :=
StringGrid7.Cells[0,i];
        StringGrid11.Cells[1,(StringGrid11.RowCount - 2)] :=
StringGrid7.Cells[1,i];
        StringGrid11.Cells[2,(StringGrid11.RowCount - 2)] :=
StringGrid7.Cells[2,i];

        //ShowMessage('testeA ' + IntToStr(ccont4));

      end;
      //ccont4 := 0;
    end;
  end;
end;

procedure TFrmHough.Acumulaoaeb1Click(Sender: TObject);
var
  i,j,c,d,e,nret : Integer;
  NewString : String;
  ClickedOK : Boolean;
begin
  NewString := '';
  ClickedOK := InputQuery('Acumulação', 'Limiar (Menor que '+
stringgrid11.cells[2,1] + ')' , NewString);
  if ClickedOK then
  begin
    e := StrToInt(NewString);
  end
  else
  exit;
  //Recebe a porcentagem da acumulação

```

```

c := 1;
nret := 0;
StringGrid4.ColCount := 2;//Cria duas colunas na tabela StringGrid4
//Cria uma linha na tabela StringGrid4
StringGrid4.RowCount := 1;
//Se o valor da acumulação da reta for maior, insere a mesma na tabela
i := 2;
while(i < (StringGrid11.RowCount - 2)) do
begin
  if(StrToInt(StringGrid11.Cells[2,i]) > e) then
  begin
    StringGrid4.Cells[0,0] := StringGrid11.Cells[0,i];
    StringGrid4.Cells[1,0] := StringGrid11.Cells[1,i];
    i := StringGrid11.RowCount + 2;
    nret := 1;
  end;
  i := i + 1;
end;
for i := 1 to (StringGrid11.RowCount - 2) do
begin
  for j := 0 to (StringGrid4.RowCount - 1) do
  begin
    if(StringGrid4.Cells[0,j] = StringGrid11.Cells[0,i]) then
    begin
      if(StringGrid4.Cells[1,j] = StringGrid11.Cells[1,i]) then
      begin
        c := 0;
        break;
      end;
    end;
    if(StrToInt(StringGrid11.Cells[2,i]) > e) then
    begin
      if(c = 1) then
      begin
        StringGrid4.RowCount := StringGrid4.RowCount + 1;
        StringGrid4.Cells[0,StringGrid4.RowCount - 1] :=
StringGrid11.Cells[0,i];
        StringGrid4.Cells[1,StringGrid4.RowCount - 1] :=
StringGrid11.Cells[1,i];

        nret := 1;
      end;
      c := 1;
    end;
  end;
  //Exibe o número de retas detectadas
  if(nret > 0 ) then
  begin
    ShowMessage('Foram detectadas ' + IntToStr(StringGrid4.RowCount) + ' '
retas!');
  end
  else
  begin
    ShowMessage('Não foi detectada nenhuma reta com este limiar! ');
  end;
end;
procedure TFrmHough.DesenhalClick(Sender: TObject);

```

```

var
  i,j,v,s,li,ls : Integer;
  aBitMap,bBitMap: TBitMap;
begin
  for i := 0 to (StringGrid4.RowCount - 1) do
begin
  visual := Tvisual.Create(self);
  visual.nBitMap := TBitMap.Create;
  visual.Open(OpenFile.FileName);
  visual.visible:=true;
  li := (visual.nBitMap.Height)-1; { Altura da imagem }
  ls := (visual.nBitMap.Width)-1;
  for v := 0 to li do
begin
  for s := 0 to ls do
begin
  visual.nBitMap.Canvas.Pixels[s,v] := $00FFFFFF;
end;
end;

  for j := 2 to (StringGrid2.RowCount - 1) do
begin
if(StringGrid4.Cells[0,i] = StringGrid2.Cells[0,j]) then
begin
  if(StringGrid4.Cells[1,i] = StringGrid2.Cells[1,j]) then
begin
  visual.nBitMap.Canvas.Pixels[StrToInt(StringGrid2.Cells[2,j]),StrToInt(StringGrid2.Cells[3,j])] := $00000000;

  visual.nBitMap.Canvas.Pixels[StrToInt(StringGrid2.Cells[4,j]),StrToInt(StringGrid2.Cells[5,j])] := $00000000;
      end;
    end;
  end;
  visual.caption:='a = '+ StringGrid4.Cells[0,i] + ' b = ' +
StringGrid4.Cells[1,i];
  visual.FImage.Picture.Assign (visual.nBitMap);
  end;
end;

procedure TFrmHough.FormActivate(Sender: TObject);
begin
  PnlAcum.Caption := 'Tabela de Acumulação';
end;

//-----
-- 
// aqui comeca a parte do código para a transformada de hough para circulos
//-----
-- 

//calculo usando quatro pontos

procedure TFrmHough.CalculoUsandoQuadroPontos1Click(Sender: TObject);
var
  aBitMap: TBitMap;
  max,n,li,ls,i,j,aux : Integer;
  NewString : String;
  NewString1 : String;

```

```

ClickedOK: Boolean;
begin
    {igual para tres pontos}
    for i := 0 to StringGrid1.ColCount do
        for j := 0 to StringGrid1.RowCount do
        begin
            StringGrid1.Cells[i,j] :='';
        end;
    for i := 0 to StringGrid2.ColCount do
        for j := 0 to StringGrid2.RowCount do
        begin
            StringGrid2.Cells[i,j] :='';
        end;
    for i := 0 to StringGrid3.ColCount do
        for j := 0 to StringGrid3.RowCount do
        begin
            StringGrid3.Cells[i,j] :='';
        end;
    for i := 0 to StringGrid4.ColCount do
        for j := 0 to StringGrid4.RowCount do
        begin
            StringGrid4.Cells[i,j] :='';
        end;
    for i := 0 to StringGrid5.ColCount do
        for j := 0 to StringGrid5.RowCount do
        begin
            StringGrid5.Cells[i,j] :='';
        end;
    for i := 0 to StringGrid6.ColCount do
        for j := 0 to StringGrid6.RowCount do
        begin
            StringGrid6.Cells[i,j] :='';
        end;
    for i := 0 to StringGrid7.ColCount do
        for j := 0 to StringGrid7.RowCount do
        begin
            StringGrid7.Cells[i,j] :='';
        end;
    for i := 0 to StringGrid8.ColCount do
        for j := 0 to StringGrid8.RowCount do
        begin
            StringGrid8.Cells[i,j] :='';
        end;
    for i := 0 to StringGrid9.ColCount do
        for j := 0 to StringGrid9.RowCount do
        begin
            StringGrid9.Cells[i,j] :='';
        end;
    for i := 0 to StringGrid10.ColCount do
        for j := 0 to StringGrid10.RowCount do
        begin
            StringGrid10.Cells[i,j] :='';
        end;
    for i := 0 to StringGrid11.ColCount do
        for j := 0 to StringGrid11.RowCount do
        begin
            StringGrid11.Cells[i,j] :='';
        end;

StringGrid2.ColCount := 0;
StringGrid2.RowCount := 0;
StringGrid1.ColCount := 6;

```

```

StringGrid1.RowCount := 1;
StringGrid3.ColCount := 0;
StringGrid3.RowCount := 0;
StringGrid4.ColCount := 0;
StringGrid4.RowCount := 0;
StringGrid5.ColCount := 0;
StringGrid5.RowCount := 0;
StringGrid6.ColCount := 0;
StringGrid6.RowCount := 0;
StringGrid7.ColCount := 0;
StringGrid7.RowCount := 0;
StringGrid8.ColCount := 0;
StringGrid8.RowCount := 0;
StringGrid9.ColCount := 0;
StringGrid9.RowCount := 0;
StringGrid10.ColCount := 0;
StringGrid10.RowCount := 0;
StringGrid11.ColCount := 0;
StringGrid11.RowCount := 0;

newcont := 0;
distanciaminima:=0;
maxdistancia:=0;
newpercent:=0;
percent:=0;
NewString := '';
ClickedOK := InputQuery('Porcentagem', 'Detecção', NewString);
if ClickedOK then
begin
    percent := StrToInt(NewString);
end
else
    exit;
newpercent := (100 - percent)/100;

NewString1 := '20';
ClickedOK := InputQuery('Distancia Mínima da Borda do Círculo', 'Entre
com o Valor Sugerido(maior ou igual a 20 pixels)', NewString1);
if ClickedOK then
begin
    distanciaminima := StrToInt(NewString1);
end
else
    exit;
distanciaminima :=distanciaminima * distanciaminima;
Screen.Cursor := crHourglass; { cursor de espera }
li := (FrmImagen.iBitmap.Height); { Altura da imagem }
ls := (FrmImagen.iBitmap.Width); { Largura da imagem }
max:=ls div 2;
NewString1 := inttostr(max);
ClickedOK := InputQuery('Distancia Máxima da Borda do Círculo', 'Entre
com o Valor Sugerido (menor ou igual a ' + inttostr(max) + ' pixels)',
NewString1);
if ClickedOK then
begin
    maxdistancia := StrToInt(NewString1);
end
else
    exit;
maxdistancia:=maxdistancia * maxdistancia;
aBitmap := TBitmap.Create;
aBitmap.Assign (FrmImagen.iBitmap); { imagem original }

```

```

aBitMap.PixelFormat := pflbit;
AlocaImagemOriginal1(aBitMap, li,ls); { aloca imagem original }
aBitMap.Free;
Screen.Cursor := crDefault; { cursor normal }

StringGrid1.ColCount := 11;
StringGrid1.RowCount := 1;
StringGrid1.Cells[0,0] := 'xc';
StringGrid1.Cells[1,0] := 'yc';
StringGrid1.Cells[2,0] := 'x1';
StringGrid1.Cells[3,0] := 'y1';
StringGrid1.Cells[4,0] := 'x2';
StringGrid1.Cells[5,0] := 'y2';
StringGrid1.Cells[6,0] := 'x3';
StringGrid1.Cells[7,0] := 'y3';
StringGrid1.Cells[8,0] := 'x4';
StringGrid1.Cells[9,0] := 'y4';
StringGrid1.Cells[10,0] := 'Raio';

StringGrid2.ColCount := 12;
StringGrid2.Cells[0,0] := 'xc';
StringGrid2.Cells[1,0] := 'yc';
StringGrid2.Cells[2,0] := 'x1';
StringGrid2.Cells[3,0] := 'y1';
StringGrid2.Cells[4,0] := 'x2';
StringGrid2.Cells[5,0] := 'y2';
StringGrid2.Cells[6,0] := 'x3';
StringGrid2.Cells[7,0] := 'y3';
StringGrid2.Cells[8,0] := 'x4';
StringGrid2.Cells[9,0] := 'y4';
StringGrid2.Cells[10,0] := 'Raio';
StringGrid2.Cells[11,0] := 'Contador';
StringGrid5.ColCount := 12;
StringGrid5.Cells[0,0] := 'xc';
StringGrid5.Cells[1,0] := 'yc';
StringGrid5.Cells[2,0] := 'x1';
StringGrid5.Cells[3,0] := 'y1';
StringGrid5.Cells[4,0] := 'x2';
StringGrid5.Cells[5,0] := 'y2';
StringGrid5.Cells[6,0] := 'x3';
StringGrid5.Cells[7,0] := 'y3';
StringGrid5.Cells[8,0] := 'x4';
StringGrid5.Cells[9,0] := 'y4';
StringGrid5.Cells[10,0] := 'Raio';
StringGrid5.Cells[11,0] := 'Contador';

StringGrid7.ColCount := 3;
StringGrid7.Cells[0,0] := 'Xc';
StringGrid7.Cells[1,0] := 'Yc';
StringGrid7.Cells[2,0] := 'Contador';

end;

procedure TfrmHough.AlocaImagemOriginal1(var dBitMap: TBitMap; li,ls : Integer);
var
  aBitMap : TBitMap;
  x,y,i,j,cont,contador: Integer;
  aux : LongInt;
  cont1 : Double;

```

```

begin
  aBitMap := TBitmap.Create;
  aBitMap.Assign(dBitMap);
  aBitMap.PixelFormat := pfb1bit;
  contador := 0;
  contchecked := 0;
  New(desc1); // Aloca descritor
  desc1^.Prev := nil;
  desc1^.Next := nil;

  for x := 0 to ls do {horizontal}
begin
  for y := 0 to li do {vertical}
begin
  if aBitMap.Canvas.Pixels[x,y] = 0 then
begin
  if (desc1^.Prev = nil) and (desc1^.Next = nil) then {primeiro nodo}
begin
  New(p); // Aloca nodo
  desc1^.Next := p;
  desc1^.Prev := p;
  p^.Prev := nil;
  p^.Next := nil;
  p^.x := x;
  p^.y := y;
  p^.checked := False;
  p^.cont := contador ; // Necessário para escolha randômica
end
else
begin
  New(p); // Aloca nodo
  p^.Prev := desc1^.Next;
  p^.Next := nil;
  (desc1^.Next)^.Next := p;
  desc1^.Next := p;
  p^.x := x;
  p^.y := y;
  p^.checked := False;
  p^.cont := contador + 1; // Necessário para escolha randômica
  contador := contador + 1;
end;
end;
end;
end;
stringgrid9.colcount:=2;
stringgrid9.rowcount:=2;
stringgrid9.cells[0,0]:='X';
stringgrid9.cells[1,0]:='Y';
for x := 0 to ls do {horizontal}
begin
  for y := 0 to li do {vertical}
begin
  if aBitMap.Canvas.Pixels[x,y] = 0 then
begin
  stringgrid9.cells[0,stringgrid9.rowcount-1]:= inttostr(x);
  stringgrid9.cells[1,stringgrid9.rowcount-1]:= inttostr(y);
  stringgrid9.rowcount:=stringgrid9.rowcount+1;
end;
end;
end;

```

```

contchecked := contador;

Edit1.Text := IntToStr(contador);
li := (FrmImagenet.iBitMap.Height)-1; { Altura da imagem }
ls := (FrmImagenet.iBitMap.Width)-1;

// Deve- se incluir um comando de checagem para ver se o nodo já foi
escolhido anteriormente
// Deve- se iniciar o descritor em sua configuração inicial
cont := 0;
cont1 := 0;
cont1 := contchecked;
r:=0;
while ((contchecked - (cont1 * newpercent)) > 0) do
begin
  cont := cont + 1;

  // Atualiza descritor (configuração inicial)
  while (desc1^.Prev)^.Prev <> nil do
    desc1^.Prev := (desc1^.Prev)^.Prev;

  //Randomize;
  r := Random(contador);
  Edit2.Text := IntToStr(r);
  ProcuraNodo1(aBitMap,li,ls);
end;
Dispose(desc1);
aBitMap.Assign(dBitMap);
aBitMap.Free;
showmessage('Calculo Terminado');
end;

procedure TFrmHough.ProcuraNodo1(var aBitMap : TBitMap; li,ls : integer);
begin
  while (desc1^.Prev)^.Next <> nil do
begin
  if (desc1^.Prev)^.cont = r then
  begin
    if (desc1^.Prev)^.checked = True then
      break
    else
      begin
        (desc1^.Prev)^.checked := True;
        contchecked := contchecked - 1;

calculatransformadadehoughparacirculosUsandoQuadroPontos(aBitMap,li,ls);
        break;
      end;
  end
  else
    desc1^.Prev := (desc1^.Prev)^.Next;
end;
end;

procedure
TFrmHough.calculatransformadadehoughparacirculosUsandoQuadroPontos(var
aBitMap : TBitMap; li,ls :integer);
var

```

```

i,ny,j,p,q,a1,a2,b1,b2,c,c1,c2,xc1,xc2,ycl,yc2: integer;
raio, aux, aux1, aux2, aux3 : double;
begin

  x := (desc1^.Prev)^.x;
  y := (desc1^.Prev)^.y;

  Edit1.Text := IntToStr(x);
  Edit2.Text := IntToStr(y);

  x1 := x;
  y1 := y;
  // Atualiza descritor (configuração inicial)
  while (desc1^.Prev)^(.Prev) <> nil do
    desc1^.Prev := (desc1^.Prev)^.Prev;
  //captura valor de x2,y2
  for ny := 1 to stringgrid9.RowCount-2 do {vertical}
  begin
    x:=strtoint(stringgrid9.Cells[0,ny]);
    y:=strtoint(stringgrid9.Cells[1,ny]);
    aux:=(((x-x1)*(x-x1))+((y-y1)*(y-y1)));
    if((aux>=distanciaminima)and(aux<=maxdistancia))then
    begin
      x2 := x;
      y2 := y;
      break;
    end;
  end;

  //captura valor de x3,y3
  for ny := 1 to stringgrid9.RowCount-2 do {vertical}
  begin
    x:=strtoint(stringgrid9.Cells[0,ny]);
    y:=strtoint(stringgrid9.Cells[1,ny]);
    aux:=(((x-x2)*(x-x2))+((y-y2)*(y-y2)));
    if((aux>=distanciaminima)and(aux<=maxdistancia))then
    begin
      x3 := x;
      y3 := y;
      break;
    end;
  end;

  //captura valor de x4,y4
  for ny := 1 to stringgrid9.RowCount-2 do {vertical}
  begin
    x:=strtoint(stringgrid9.Cells[0,ny]);
    y:=strtoint(stringgrid9.Cells[1,ny]);
    aux:=(((x-x3)*(x-x3))+((y-y3)*(y-y3)));
    if((aux>=distanciaminima)and(aux<=maxdistancia))then
    begin
      x4 := x;
      y4 := y;
      break;
    end;
  end;
  a1:=2*(x2-x1);
  b1:=2*(y2-y1);
  c1:=(x1*x1)+(y1*y1)-(x2*x2)-(y2*y2);
  a2:=2*(x4-x3);
  b2:=2*(y4-y3);

```

```

c2:=(x3*x3)+(y3*y3)-(x4*x4)-(y4*y4);
aux:=((b1*(a1+a2))-(a1*(b1+b2)));
aux1:=((a1*c2)-(a2*c1));
if(aux=0)then
begin
  yc:=0;
end
else
aux2:=aux1 / aux;
aux1:=(-c1-c2-(b1+b2)*aux2);
aux:=a1+a2;
if(aux=0)then
begin
  xc:=0;
end
else
aux3:=aux1 / aux;

xc:=round(int(aux3));
yc:=round(int(aux2));

raio:=(((x1-xc)*(x1-xc))+((y1-yc)*(y1-yc)));
if(raio>=0)then
raio:=sqrt(raio);

StringGrid1.RowCount := StringGrid1.RowCount + 1;
StringGrid1.Cells[0,1] := intToStr(xc);
StringGrid1.Cells[1,1] := intToStr(yc);
StringGrid1.Cells[2,1] := intToStr(x1);
StringGrid1.Cells[3,1] := intToStr(y1);
StringGrid1.Cells[4,1] := intToStr(x2);
StringGrid1.Cells[5,1] := intToStr(y2);
StringGrid1.Cells[6,1] := intToStr(x3);
StringGrid1.Cells[7,1] := intToStr(y3);
StringGrid1.Cells[8,1] := intToStr(x4);
StringGrid1.Cells[9,1] := intToStr(y4);
StringGrid1.Cells[10,1] := floatToStr(raio);
StringGrid2.RowCount := StringGrid1.RowCount + 1;
//Incrementa a tabela StringGrid2 em uma linha
StringGrid2.Cells[0,StringGrid2.RowCount - 1] :=
intToStr(xc);
StringGrid2.Cells[1,StringGrid2.RowCount - 1] :=
intToStr(yc);
StringGrid2.Cells[2,StringGrid2.RowCount - 1] :=
intToStr(x1);
StringGrid2.Cells[3,StringGrid2.RowCount - 1] :=
intToStr(y1);
StringGrid2.Cells[4,StringGrid2.RowCount - 1] :=
intToStr(x2);
StringGrid2.Cells[5,StringGrid2.RowCount - 1] :=
intToStr(y2);
StringGrid2.Cells[6,StringGrid2.RowCount - 1] :=
intToStr(x3);
StringGrid2.Cells[7,StringGrid2.RowCount - 1] :=
intToStr(y3);
StringGrid2.Cells[8,StringGrid2.RowCount - 1] :=
intToStr(x4);
StringGrid2.Cells[9,StringGrid2.RowCount - 1] :=
intToStr(y4);
StringGrid2.Cells[10,StringGrid2.RowCount - 1] :=
floatToStr(raio);

```

```

//Insere os valores de xc, yc, x1, y1, x2, y2, x3, y3, x4,
y4 na tabela

    if((StringGrid2.Cells[11,StringGrid2.RowCount -
1]='')or(StringGrid2.Cells[11,StringGrid2.RowCount - 1] = ' '))then
        begin //Se na posição do contador não houver nenhum valor, é
colocado o valor 1 no mesmo
            StringGrid2.Cells[11,StringGrid2.RowCount - 1] :='1';
        end;
    p := StringGrid2.RowCount - 2;
    q := StringGrid2.RowCount - 1;
    newcont := 0;
    //Faz a comparação do contador da última linha da tabela com
as demais e troca
    //os valores para as mesmas linhas
    while(p >= 2) do
        begin
            if((StringGrid2.Cells[0,p] <>'') and
(StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
(StringGrid2.Cells[1,q] <>'')) then
                begin
                    xc1 := StrToInt(StringGrid2.Cells[0,p]);
                    yc1 := StrToInt(StringGrid2.Cells[1,p]);
                    xc2 :=
StrToInt(StringGrid2.Cells[0,StringGrid2.RowCount - 1]);
                    yc2 :=
StrToInt(StringGrid2.Cells[1,StringGrid2.RowCount - 1]);
                    if((xc1 = xc2) and (yc1 = yc2)) then
                        begin
                            newcont :=
(StrToInt(StringGrid2.Cells[11,p])) +
(StrToInt(StringGrid2.Cells[11,StringGrid2.RowCount -1 ]));
                            StringGrid2.Cells[11,p] := IntToStr(newcont);
                            StringGrid2.Cells[11,StringGrid2.RowCount -
1] := IntToStr(newcont);
                            break;
                        end;
                    end;
                    p := p - 1;
                end;
        end;

    p := 0;
    q := 0;
    newcont := 0;
end;

//Para o calculo usando Tres pontos
procedure TFrmHough.CalculoUsandoTresPontos1Click(Sender: TObject);
var
  aBitMap: TBitMap;
  max,n,li,ls,i,j,aux : Integer;
  NewString : String;
  NewString1 : String;
  ClickedOK: Boolean;
begin
  for i := 0 to StringGrid1.ColCount do
    for j := 0 to StringGrid1.RowCount do {limpa as estruturas de
dados}
      begin
        StringGrid1.Cells[i,j] :='';
      end;

```

```

for i := 0 to StringGrid2.ColCount do
  for j := 0 to StringGrid2.RowCount do
    begin
      StringGrid2.Cells[i,j] :='';
    end;
for i := 0 to StringGrid3.ColCount do
  for j := 0 to StringGrid3.RowCount do
    begin
      StringGrid3.Cells[i,j] :='';
    end;
for i := 0 to StringGrid4.ColCount do
  for j := 0 to StringGrid4.RowCount do
    begin
      StringGrid4.Cells[i,j] :='';
    end;
for i := 0 to StringGrid5.ColCount do
  for j := 0 to StringGrid5.RowCount do
    begin
      StringGrid5.Cells[i,j] :='';
    end;
for i := 0 to StringGrid6.ColCount do
  for j := 0 to StringGrid6.RowCount do
    begin
      StringGrid6.Cells[i,j] :='';
    end;
for i := 0 to StringGrid7.ColCount do
  for j := 0 to StringGrid7.RowCount do
    begin
      StringGrid7.Cells[i,j] :='';
    end;
for i := 0 to StringGrid8.ColCount do
  for j := 0 to StringGrid8.RowCount do
    begin
      StringGrid8.Cells[i,j] :='';
    end;
for i := 0 to StringGrid9.ColCount do
  for j := 0 to StringGrid9.RowCount do
    begin
      StringGrid9.Cells[i,j] :='';
    end;
for i := 0 to StringGrid10.ColCount do
  for j := 0 to StringGrid10.RowCount do
    begin
      StringGrid10.Cells[i,j] :='';
    end;
for i := 0 to StringGrid11.ColCount do
  for j := 0 to StringGrid11.RowCount do
    begin
      StringGrid11.Cells[i,j] :='';
    end;

StringGrid2.ColCount := 0;
StringGrid2.RowCount := 0;
StringGrid1.ColCount := 6;
StringGrid1.RowCount := 1;
StringGrid3.ColCount := 0;
StringGrid3.RowCount := 0;
StringGrid4.ColCount := 0;
StringGrid4.RowCount := 0;
StringGrid5.ColCount := 0;
StringGrid5.RowCount := 0;

```

```

StringGrid6.ColCount := 0;
StringGrid6.RowCount := 0;
StringGrid7.ColCount := 0;
StringGrid7.RowCount := 0;
StringGrid8.ColCount := 0;
StringGrid8.RowCount := 0;
StringGrid9.ColCount := 0;
StringGrid9.RowCount := 0;
StringGrid10.ColCount := 0;
StringGrid10.RowCount := 0;
StringGrid11.ColCount := 0;
StringGrid11.RowCount := 0;
newcont := 0;
distanciaminima:=0;
maxdistancia:=0;
newpercent:=0;
percent:=0;
NewString := '';
ClickedOK := InputQuery('Porcentagem', 'Detecção', NewString);
if ClickedOK then
begin
    percent := StrToInt(NewString);
end
else
    exit;
newpercent := (100 - percent)/100;

NewString1 := '20';
ClickedOK := InputQuery('Distância Mínima da Borda do Círculo', 'Entre
com o Valor Sugirido (maior ou igual a 20 pixels)', NewString1);
if ClickedOK then
begin
    distanciaminima := StrToInt(NewString1);
end
else
    exit;
distanciaminima :=distanciaminima * distanciaminima; {calcula a distancia
minima}
Screen.Cursor := crHourglass; { cursor de espera }
li := (FrmImagem.iBitMap.Height); { Altura da imagem }
ls := (FrmImagem.iBitMap.Width); { Largura da imagem }
max:=ls div 2;
NewString1 := inttostr(max);
ClickedOK := InputQuery('Distância Máxima da Borda do Círculo', 'Entre
com o Valor Sugirido (menor ou igual a ' + inttostr(max) + ' pixels)',
NewString1);
if ClickedOK then
begin
    maxdistancia := StrToInt(NewString1);
end
else
    exit;
maxdistancia:=maxdistancia * maxdistancia; {calcula a distancia
maxima}
aBitMap := TBitMap.Create;
aBitMap.Assign (FrmImagem.iBitMap); { imagem original }
aBitMap.PixelFormat := pf1bit;
AlocaImagemOriginal2(aBitMap, li,ls); { aloca imagem original }
aBitMap.Free; {libera memoria
Screen.Cursor := crDefault; { cursor normal }

```

```

StringGrid1.ColCount := 11;
StringGrid1.RowCount := 1;
StringGrid1.Cells[0,0] := 'xc';
StringGrid1.Cells[1,0] := 'yc';
StringGrid1.Cells[2,0] := 'x1';
StringGrid1.Cells[3,0] := 'y1';
StringGrid1.Cells[4,0] := 'x2';
StringGrid1.Cells[5,0] := 'y2';
StringGrid1.Cells[6,0] := 'x3';
StringGrid1.Cells[7,0] := 'y3';
StringGrid1.Cells[8,0] := 'x4';
StringGrid1.Cells[9,0] := 'y4';
StringGrid1.Cells[10,0] := 'Raio';

StringGrid2.ColCount := 12;
StringGrid2.Cells[0,0] := 'xc';
StringGrid2.Cells[1,0] := 'yc';
StringGrid2.Cells[2,0] := 'x1';
StringGrid2.Cells[3,0] := 'y1';
StringGrid2.Cells[4,0] := 'x2';
StringGrid2.Cells[5,0] := 'y2';
StringGrid2.Cells[6,0] := 'x3';
StringGrid2.Cells[7,0] := 'y3';
StringGrid2.Cells[8,0] := 'x4';
StringGrid2.Cells[9,0] := 'y4';
StringGrid2.Cells[10,0] := 'Raio';
StringGrid2.Cells[11,0] := 'Contador';
StringGrid5.ColCount := 12;
StringGrid5.Cells[0,0] := 'xc';
StringGrid5.Cells[1,0] := 'yc';
StringGrid5.Cells[2,0] := 'x1';
StringGrid5.Cells[3,0] := 'y1';
StringGrid5.Cells[4,0] := 'x2';
StringGrid5.Cells[5,0] := 'y2';
StringGrid5.Cells[6,0] := 'x3';
StringGrid5.Cells[7,0] := 'y3';
StringGrid5.Cells[8,0] := 'x4';
StringGrid5.Cells[9,0] := 'y4';
StringGrid5.Cells[10,0] := 'Raio';
StringGrid5.Cells[11,0] := 'Contador';

StringGrid7.ColCount := 3;
StringGrid7.Cells[0,0] := 'Xc';
StringGrid7.Cells[1,0] := 'Yc';
StringGrid7.Cells[2,0] := 'Contador';

end;

procedure TFrmHough.AlocaImagemOriginal2(var dBitmap: TBitmap; li,ls : Integer);
var
  aBitmap : TBitmap;
  x,y,i,j,cont,contador: Integer;
  aux : LongInt;
  cont1 : Double;
begin
  aBitmap := TBitmap.Create;
  aBitmap.Assign(dBitmap);
  aBitmap.PixelFormat := pfb1bit;
  contador := 0;
  contchecked := 0;

```

```

New(desc1); // Aloca descritor
desc1^.Prev := nil;
desc1^.Next := nil;

for x := 0 to ls do {horizontal}
begin
  for y := 0 to li do {vertical}
  begin
    if aBitMap.Canvas.Pixels[x,y] = 0 then
    begin
      if (desc1^.Prev = nil) and (desc1^.Next = nil) then {primeiro nodo}
      begin
        New(p); // Aloca nodo
        desc1^.Next := p;
        desc1^.Prev := p;
        p^.Prev := nil;
        p^.Next := nil;
        p^.x := x; //aloca o valor de x
        p^.y := y; //aloca o valor de y
        p^.checked := False; //controle de pontos checados
        p^.cont := contador ; // Necessário para escolha randômica
      end
      else
      begin
        New(p); // Aloca nodo
        p^.Prev := desc1^.Next;
        p^.Next := nil;
        (desc1^.Next)^.Next := p;
        desc1^.Next := p;
        p^.x := x; //aloca o valor de x
        p^.y := y; //aloca o valor de y
        p^.checked := False; //controle de pontos checados
        p^.cont := contador + 1; // Necessário para escolha randômica
        contador := contador + 1; //numeros de pontos da borda da
        imagem
      end;
    end;
  end;
end;
stringgrid9.colcount:=2;
stringgrid9.rowcount:=2;
stringgrid9.cells[0,0]:='X';
stringgrid9.cells[1,0]:='Y';
for x := 0 to ls do {horizontal}
begin
  for y := 0 to li do {vertical}
  begin
    if aBitMap.Canvas.Pixels[x,y] = 0 then
    begin
      auxiliar} {extrutura
      stringgrid9.cells[0,stringgrid9.rowcount-1]:= inttostr(x);
      stringgrid9.cells[1,stringgrid9.rowcount-1]:= inttostr(y);
      stringgrid9.rowcount:=stringgrid9.rowcount+1;
    end;
  end;
end;

contchecked := contador;

Edit1.Text := IntToStr(contador);
li := (FrmImagem.iBitMap.Height)-1; { Altura da imagem }

```

```

ls := (FrmImagenet.iBitMap.Width)-1; { largura da imagem }

// Deve- se incluir um comando de checagem para ver se o nodo já foi
escolhido anteriormente
// Deve- se iniciar o descritor em sua configuração inicial
cont := 0;
cont1 := 0;
cont1 := contchecked;
r:=0;
while ((contchecked - (cont1 * newpercent)) > 0) do
begin
  cont := cont + 1;

  // Atualiza descritor (configuração inicial)
  while (desc1^.Prev)^.Prev <> nil do
    desc1^.Prev := (desc1^.Prev)^.Prev;

  // Randomize;
  r := Random(contador);
  Edit2.Text := IntToStr(r);
  ProcuraNodo2(aBitMap,li,ls); //procura proximo ponto
end;
Dispose(desc1);
aBitMap.Assign(dBitMap);
aBitMap.Free;
showmessage('Calculo Terminado');
end;

procedure TFrmHough.ProcuraNodo2(var aBitMap : TBitMap; li,ls : integer);
begin
  while (desc1^.Prev)^.Next <> nil do //verifica se proximo nodo é nulo
  begin
    if (desc1^.Prev)^.cont = r then //verifica se cont é igual ao valor
    randomico
    begin
      if (desc1^.Prev)^.checked = True then //verifica se o ponto ja foi
    chegado
        break
      else
        begin
          (desc1^.Prev)^.checked := True; //marca o ponto como chegado
          contchecked := contchecked - 1;

calculatransformadadehoughparacirculosUsandoTresPontos(aBitMap,li,ls); //cha
ma o calculo para tres pontos
          break;
        end;
    end
    else
      desc1^.Prev := (desc1^.Prev)^.Next;
  end;
end;

procedure
TFrmHough.calculatransformadadehoughparacirculosUsandoTresPontos(var
aBitMap : TBitMap; li,ls :integer);
var
i,ny,j,p,q,a1,a2,b1,b2,c,c1,c2,xcl,xc2,ycl,yc2 : integer;

```

```

raio, aux, aux1, aux2, aux3 : double;
begin

  x := (desc1^.Prev)^.x;
  y := (desc1^.Prev)^.y;

  Edit1.Text := IntToStr(x);
  Edit2.Text := IntToStr(y);
  //captura os pontos x1 e x2;
  x1 := x;
  y1 := y;
  // Atualiza descriptor (configuração inicial)
  while (desc1^.Prev)^(.Prev <> nil do
    desc1^.Prev := (desc1^.Prev)^.Prev;
  //captura valor de x2,y2

  for ny := 1 to stringgrid9.RowCount-2 do {vertical}
  begin
    x:=strtoint(stringgrid9.Cells[0,ny]);
    y:=strtoint(stringgrid9.Cells[1,ny]);
    aux:=(((x-x1)*(x-x1))+((y-y1)*(y-y1)));
    if((aux>=distanciaminima)and(aux<=maxdistancia))then
    begin
      x2 := x;
      y2 := y;
      break;
    end;
  end;

  //captura valor de x3,y3

  x3:=x2;
  y3:=y2;

  //captura valor de x4,y4
  for ny := 1 to stringgrid9.RowCount-2 do {vertical}
  begin
    x:=strtoint(stringgrid9.Cells[0,ny]);
    y:=strtoint(stringgrid9.Cells[1,ny]);
    aux:=(((x-x3)*(x-x3))+((y-y3)*(y-y3)));
    if((aux>=distanciaminima)and(aux<=maxdistancia))then
    begin
      x4 := x;
      y4 := y;
      break;
    end;
  end;                                //calculo da transformada de hough
  a1:=2*(x2-x1);
  b1:=2*(y2-y1);
  c1:=(x1*x1)+(y1*y1)-(x2*x2)-(y2*y2);
  a2:=2*(x4-x3);
  b2:=2*(y4-y3);
  c2:=(x3*x3)+(y3*y3)-(x4*x4)-(y4*y4);
  aux:=((b1*(a1+a2))-(a1*(b1+b2)));
  aux1:=((a1*c2)-(a2*c1));
  if(aux=0)then
  begin
    yc:=0;
  end
  else
  aux2:=aux1 / aux;      //aux2 = yc

```

```

aux1:=(-c1-c2-(b1+b2)*aux2);
aux:=a1+a2;
if(aux=0)then
begin
  xc:=0;
end
else
  aux3:=aux1 / aux;    //aux3 =  xc
  xc:=round(int(aux3));
  yc:=round(int(aux2));

raio:=(((x1-xc)*(x1-xc))+((y1-yc)*(y1-yc)));
if(raio>=0)then
  raio:=sqrt(raio);
  {aloca os pontos iniciais}
StringGrid1.RowCount := StringGrid1.RowCount + 1;
StringGrid1.Cells[0,1] := intToStr(xc);
StringGrid1.Cells[1,1] := intToStr(yc);
StringGrid1.Cells[2,1] := intToStr(x1);
StringGrid1.Cells[3,1] := intToStr(y1);
StringGrid1.Cells[4,1] := intToStr(x2);
StringGrid1.Cells[5,1] := intToStr(y2);
StringGrid1.Cells[6,1] := intToStr(x3);
StringGrid1.Cells[7,1] := intToStr(y3);
StringGrid1.Cells[8,1] := intToStr(x4);
StringGrid1.Cells[9,1] := intToStr(y4);
StringGrid1.Cells[10,1] := floatToStr(raio);
StringGrid2.RowCount := StringGrid1.RowCount + 1;
//Incrementa a tabela StringGrid2 em uma linha
StringGrid2.Cells[0,StringGrid2.RowCount - 1] :=
intToStr(xc);
StringGrid2.Cells[1,StringGrid2.RowCount - 1] :=
intToStr(yc);
StringGrid2.Cells[2,StringGrid2.RowCount - 1] :=
intToStr(x1);
StringGrid2.Cells[3,StringGrid2.RowCount - 1] :=
intToStr(y1);
StringGrid2.Cells[4,StringGrid2.RowCount - 1] :=
intToStr(x2);
StringGrid2.Cells[5,StringGrid2.RowCount - 1] :=
intToStr(y2);
StringGrid2.Cells[6,StringGrid2.RowCount - 1] :=
intToStr(x3);
StringGrid2.Cells[7,StringGrid2.RowCount - 1] :=
intToStr(y3);
StringGrid2.Cells[8,StringGrid2.RowCount - 1] :=
intToStr(x4);
StringGrid2.Cells[9,StringGrid2.RowCount - 1] :=
intToStr(y4);
StringGrid2.Cells[10,StringGrid2.RowCount - 1] :=
floatToStr(raio);
//Insere os valores de xc, yc, x1, y1, x2, y2 ,x3, y3, x4,
y4 na tabela

  if((StringGrid2.Cells[11,StringGrid2.RowCount -
1]=' ')or(StringGrid2.Cells[11,StringGrid2.RowCount - 1]= ' '))then
    begin //Se na posição do contador não houver nenhum valor, é
colocado o valor 1 no mesmo
      StringGrid2.Cells[11,StringGrid2.RowCount - 1]:='1';
    end;
  p := StringGrid2.RowCount - 2;

```

```

        q := StringGrid2.RowCount - 1;
        newcont := 0;
        //Faz a comparação do contador da última linha da tabela com
        as demais e troca
            //os valores para as mesmas linhas
            while(p >= 2) do
                begin
                    if((StringGrid2.Cells[0,p] <>'') and
                    (StringGrid2.Cells[1,p] <>'') and (StringGrid2.Cells[0,q] <>'') and
                    (StringGrid2.Cells[1,q] <>'')) then
                        begin
                            xc1 := StrToInt(StringGrid2.Cells[0,p]);
                            yc1 := StrToInt(StringGrid2.Cells[1,p]);
                            xc2 :=
                            StrToInt(StringGrid2.Cells[0,StringGrid2.RowCount - 1]);
                            yc2 :=
                            StrToInt(StringGrid2.Cells[1,StringGrid2.RowCount - 1]);
                            if((xc1 = xc2) and (yc1 = yc2)) then
                                begin
                                    newcont :=
                                    (StrToInt(StringGrid2.Cells[11,p])) +
                                    (StrToInt(StringGrid2.Cells[11,StringGrid2.RowCount - 1]));
                                    StringGrid2.Cells[11,p] := IntToStr(newcont);
                                    StringGrid2.Cells[11,StringGrid2.RowCount -
                                    1] := IntToStr(newcont);
                                    break;
                                end;
                            end;
                            p := p - 1;
                        end;
                end;

        p := 0;
        q := 0;
        newcont := 0;
    end;
}

```

```

procedure TFrmHough.AcumulacaoClick(Sender: TObject);
var
    i,j,n,s,m,c,d,e,cont,ccont1,ccont2,ccont3,ccont4,igualdade,temp,temp1:
    integer;
begin
    c := 1;
    StringGrid5.ColCount := 12;
    igualdade := 0;
    StringGrid5.RowCount := 2;
    //Se o valor da acumulação do circulo for maior, insere o mesmo na tabela
    StringGrid5.Cells[0,1] := StringGrid2.Cells[0,2];
    StringGrid5.Cells[1,1] := StringGrid2.Cells[1,2];
    StringGrid5.Cells[2,1] := StringGrid2.Cells[2,2];
    StringGrid5.Cells[3,1] := StringGrid2.Cells[3,2];
    StringGrid5.Cells[4,1] := StringGrid2.Cells[4,2];
    StringGrid5.Cells[5,1] := StringGrid2.Cells[5,2];
    StringGrid5.Cells[6,1] := StringGrid2.Cells[6,2];
    StringGrid5.Cells[7,1] := StringGrid2.Cells[7,2];
    StringGrid5.Cells[8,1] := StringGrid2.Cells[8,2];

```

```

StringGrid5.Cells[9,1] := StringGrid2.Cells[9,2];
StringGrid5.Cells[10,1] := StringGrid2.Cells[10,2];
StringGrid5.Cells[11,1] := StringGrid2.Cells[11,2];
for i := 3 to (StringGrid2.RowCount - 1) do
begin
  for j := 0 to (StringGrid5.RowCount - 1) do {verifica se tem um centro
igual com um contador maior que o anterior}
  begin
    if((StringGrid5.Cells[0,j] = StringGrid2.Cells[0,i])) then
    begin
      if((StringGrid5.Cells[1,j] = StringGrid2.Cells[1,i])) then
      begin
        c := 0;
        ccont1 := StrToInt(StringGrid5.Cells[11,j]);
        ccont2 := StrToInt(StringGrid2.Cells[11,i]);
        if(ccont1 <= ccont2) then
        begin
          StringGrid5.Cells[11,j] := StringGrid2.Cells[11,i];//se sim
substitui pelo maior
        end;
        break;
      end;
    end;
  end;
  if(StrToInt(StringGrid2.Cells[11,i]) > 2) then
begin
  if(c = 1) then
  begin
    StringGrid5.RowCount := StringGrid5.RowCount + 1;
    StringGrid5.Cells[0,StringGrid5.RowCount - 1] :=
StringGrid2.Cells[0,i];
    StringGrid5.Cells[1,StringGrid5.RowCount - 1] :=
StringGrid2.Cells[1,i];
    StringGrid5.Cells[2,StringGrid5.RowCount - 1] :=
StringGrid2.Cells[2,i];
    StringGrid5.Cells[3,StringGrid5.RowCount - 1] :=
StringGrid2.Cells[3,i];
    StringGrid5.Cells[4,StringGrid5.RowCount - 1] :=
StringGrid2.Cells[4,i];
    StringGrid5.Cells[5,StringGrid5.RowCount - 1] :=
StringGrid2.Cells[5,i];
    StringGrid5.Cells[6,StringGrid5.RowCount - 1] :=
StringGrid2.Cells[6,i];
    StringGrid5.Cells[7,StringGrid5.RowCount - 1] :=
StringGrid2.Cells[7,i];
    StringGrid5.Cells[8,StringGrid5.RowCount - 1] :=
StringGrid2.Cells[8,i];
    StringGrid5.Cells[9,StringGrid5.RowCount - 1] :=
StringGrid2.Cells[9,i];
    StringGrid5.Cells[10,StringGrid5.RowCount - 1] :=
StringGrid2.Cells[10,i];
    StringGrid5.Cells[11,StringGrid5.RowCount - 1] :=
StringGrid2.Cells[11,i];
  end;
  c := 1;
end;
StringGrid6.ColCount := StringGrid5.RowCount;
StringGrid6.RowCount := 1;
for i := 0 to (StringGrid6.ColCount - 1) do
begin

```

```

        StringGrid6.Cells[i,0] :='0';
end;
for i := 1 to (StringGrid5.RowCount - 1) do
begin
        StringGrid6.Cells[i-1,0] := StringGrid5.Cells[11,i];
end;
for i := 0 to (StringGrid6.colCount - 1) do
begin
        for n:=i+1 to StringGrid6.colCount-1 do //reordena os valores
acumulados do maior para o menor
begin
if(strtoint(StringGrid6.Cells[n,0])>=(strtoint(StringGrid6.Cells[i,0])))the
n
begin
temp:=strtoint(StringGrid6.Cells[i,0]);
StringGrid6.Cells[i,0]:=StringGrid6.Cells[n,0];
StringGrid6.Cells[n,0]:=inttostr(temp);
end;
end;
end;
StringGrid7.RowCount := StringGrid5.RowCount;
StringGrid7.ColCount := 3;
ccont4 := 1;
for i := 0 to (StringGrid6.ColCount - 1) do
begin

for j := 1 to (StringGrid5.RowCount - 1) do
begin
if(StringGrid5.Cells[11,j] = StringGrid6.Cells[i,0]) then
begin
if(ccont4 = 1) then
begin
StringGrid7.Cells[0,ccont4] :=
StringGrid5.Cells[0,j];
StringGrid7.Cells[1,ccont4] :=
StringGrid5.Cells[1,j];
StringGrid7.Cells[2,ccont4] :=
StringGrid5.Cells[11,j];
end;
if(ccont4 > 1) then
begin
if(StringGrid7.Cells[0,ccont4] =
StringGrid7.Cells[0,ccont4 - 1]) then
begin
igualdade := igualdade + 1;
end;
if(StringGrid7.Cells[1,ccont4] =
StringGrid7.Cells[1,ccont4 - 1]) then
begin
igualdade := igualdade + 1;
end;
if(igualdade < 2) then
begin
StringGrid7.Cells[0,ccont4] :=
StringGrid5.Cells[0,j];
StringGrid7.Cells[1,ccont4] :=
StringGrid5.Cells[1,j];
StringGrid7.Cells[2,ccont4] :=
StringGrid5.Cells[11,j];
igualdade := 0;
end;
end;
end;
end;
end;
end;
end;

```

```

        end;
        if(igualdade = 2) then
        begin

            igualdade := 0;
        end;

        end;
    end;

    end;
    ccont4 := ccont4 + 1;
end;
for i := 0 to (StringGrid5.ColCount - 1) do
begin
    for j := 0 to (StringGrid5.RowCount - 1) do
    begin
        StringGrid5.Cells[i,j] :='';
    end;
end;
for i := 0 to (StringGrid6.ColCount - 1) do
begin
    for j := 0 to (StringGrid6.RowCount - 1) do
    begin
        StringGrid6.Cells[i,j] :='';
    end;
end;
StringGrid5.ColCount := 0;
StringGrid5.RowCount := 0;
StringGrid6.ColCount := 0;
StringGrid6.RowCount := 0;

StringGrid8.ColCount := 4;
StringGrid8.RowCount := 2;
StringGrid8.Cells[0,0] :='xc';
StringGrid8.Cells[1,0] :='yc';
StringGrid8.Cells[2,0] :='Contador';
StringGrid8.Cells[3,0] :='Usados';
ccont3 := 0;

for i := 1 to (StringGrid7.RowCount - 1) do
begin
    ccont4 := 0;
    if(StringGrid8.RowCount = 2) then
    begin
        StringGrid8.Cells[0,1] := StringGrid7.Cells[0,1];
        StringGrid8.Cells[1,1] := StringGrid7.Cells[1,1];
        StringGrid8.Cells[2,1] := StringGrid7.Cells[2,1];
        StringGrid8.Cells[3,1] :='0';
        ccont3 := 1;
    end;
    if(ccont3 = 1) then
    begin

        for j := 1 to (StringGrid8.RowCount - 1) do
        begin
            if((StringGrid8.Cells[0,j] =
StringGrid7.Cells[0,i])and(StringGrid8.Cells[1,j] =
StringGrid7.Cells[1,i])) then
            begin
                ccont4 := 1;
            end;
        end;
    end;
end;

```

```

                //ShowMessage('teste ' + IntToStr(ccont4));
                break;
            end;

            end;
            if(ccont4 = 0) then
            begin
                StringGrid8.RowCount := StringGrid8.RowCount + 1;
                StringGrid8.Cells[0,(StringGrid8.RowCount - 1)] :=
StringGrid7.Cells[0,i];
                StringGrid8.Cells[1,(StringGrid8.RowCount - 1)] :=
StringGrid7.Cells[1,i];
                StringGrid8.Cells[2,(StringGrid8.RowCount - 1)] :=
StringGrid7.Cells[2,i];
                StringGrid8.Cells[3,(StringGrid8.RowCount - 1)]
:= '0';
                //ShowMessage('testeA ' + IntToStr(ccont4));

            end;
            //ccont4 := 0;
        end;
    end;

//calculo da janera 5x5 para ajustar o centro
    stringgrid10.RowCount:=2;
    stringgrid10.ColCount:=3;
    stringgrid10.cells[0,0]:='Xc';
    stringgrid10.cells[1,0]:='Yc';
    stringgrid10.cells[2,0]:='Contador';
    stringgrid10.cells[0,1]:=stringgrid8.cells[0,1];
    stringgrid10.cells[1,1]:=stringgrid8.cells[1,1];
    stringgrid10.cells[2,1]:=stringgrid8.cells[2,1];

    n:=1;
    s:=1;
    for n:=1 to stringgrid8.RowCount-2 do
    begin
        if((strToInt(stringgrid8.cells[0,n])<=0) or
(strToInt(stringgrid8.cells[1,n])<=0))then
        begin
            for m:=n to stringgrid8.RowCount-2 do
            begin
                stringgrid8.cells[0,m]:=stringgrid8.cells[0,m+1];
//retira valores negativos e menores de zero
                stringgrid8.cells[1,m]:=stringgrid8.cells[1,m+1];
                stringgrid8.cells[2,m]:=stringgrid8.cells[2,m+1];
                stringgrid8.cells[3,m]:=stringgrid8.cells[3,m+1];
            end;
            stringgrid8.RowCount:=stringgrid8.RowCount-1;
        end;
    end;
    for n:=1 to stringgrid8.RowCount-2 do
    begin

if((stringgrid8.cells[0,n]<>'')and(stringgrid8.cells[1,n]<>''))then
begin
    xc:=strToInt(stringgrid8.cells[0,n]);
    yc:=strToInt(stringgrid8.cells[1,n]);
    if((strToInt(stringgrid8.cells[3,n]))=0))then
        begin

```

```

s:=n-cont;
stringgrid10.cells[0,n]:=inttostr(xc);
stringgrid10.cells[1,n]:=inttostr(yc);
stringgrid10.cells[2,n]:=stringgrid8.cells[2,n];
for m:=n+1 to stringgrid8.RowCount-2 do
begin
  if(strtoint(stringgrid8.cells[3,n])=0) then
    begin
      //Verifica se o valor de
da estrutura estao dentro da janela
      if((strtoint(stringgrid8.cells[0,m])=xc) and
(strtoint(stringgrid8.cells[1,m])=yc+1))then
        begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
          stringgrid8.cells[3,m]:='1'; //se sim soma os contadores
e ajusta os centros que estao proximos ao pontos do centro
          temp:=xc;
          temp1:=yc+1;
          ajustar(temp,temp1); //funcao de justa do centros
        end;
        if((strtoint(stringgrid8.cells[0,m])=xc-1) and
(strtoint(stringgrid8.cells[1,m])=yc-1))then
          begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
          stringgrid8.cells[3,m]:='1';
          temp:= xc-1;
          temp1:=yc-1;
          ajustar(temp,temp1);
        end;
        if((strtoint(stringgrid8.cells[0,m])=xc-1) and
(strtoint(stringgrid8.cells[1,m])=yc))then
          begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
          stringgrid8.cells[3,m]:='1';
          temp:=xc-1;
          temp1:=yc;
          ajustar(temp,temp1);
        end;
        if((strtoint(stringgrid8.cells[0,m])=xc-1) and
(strtoint(stringgrid8.cells[1,m])=yc+1))then
          begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
          stringgrid8.cells[3,m]:='1';
          temp:= xc-1;
          temp1:=yc+1;
          ajustar(temp,temp1);
        end;
        if((strtoint(stringgrid8.cells[0,m])=xc) and
(strtoint(stringgrid8.cells[1,m])=yc-1))then
          begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
          stringgrid8.cells[3,m]:='1';

```

```

        temp:=xc;
        temp1:=yc-1;
        ajustar(temp,temp1);
    end;
    if((strtoint(stringgrid8.cells[0,m])=xc+1) and
(strtoint(stringgrid8.cells[1,m])=yc-1))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
        stringgrid8.cells[3,m]:='1';
        temp:=xc+1;
        temp1:=yc-1;
        ajustar(temp,temp1);
    end;
    if((strtoint(stringgrid8.cells[0,m])=xc+1) and
(strtoint(stringgrid8.cells[1,m])=yc))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
        stringgrid8.cells[3,m]:='1';
        temp:=xc+1;
        temp1:=yc;
        ajustar(temp,temp1);
    end;
    if((strtoint(stringgrid8.cells[0,m])=xc+1) and
(strtoint(stringgrid8.cells[1,m])=yc+1))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
        stringgrid8.cells[3,m]:='1';
        temp:=xc+1;
        temp1:=yc+1;
        ajustar(temp,temp1);
    end;
    if((strtoint(stringgrid8.cells[0,m])=xc-2) and
(strtoint(stringgrid8.cells[1,m])=yc-2))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
        stringgrid8.cells[3,m]:='1';
        temp:= xc-2;
        temp1:=yc-2;
        ajustar(temp,temp1);
    end;
    if((strtoint(stringgrid8.cells[0,m])=xc-1) and
(strtoint(stringgrid8.cells[1,m])=yc-2))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
        stringgrid8.cells[3,m]:='1';
        temp:= xc-1;
        temp1:=yc-2;
        ajustar(temp,temp1);
    end;
    if((strtoint(stringgrid8.cells[0,m])=xc) and
(strtoint(stringgrid8.cells[1,m])=yc-2))then
begin

```

```

begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
    stringgrid8.cells[3,m]:='1';
    temp:= xc;
    temp1:=yc-2;
    ajustar(temp,temp1);
    end;
    if((strtoint(stringgrid8.cells[0,m])=xc+1) and
(strtoint(stringgrid8.cells[1,m])=yc-2))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
    stringgrid8.cells[3,m]:='1';
    temp:= xc+1;
    temp1:=yc-2;
    ajustar(temp,temp1);
    end;
    if((strtoint(stringgrid8.cells[0,m])=xc+2) and
(strtoint(stringgrid8.cells[1,m])=yc-2))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
    stringgrid8.cells[3,m]:='1';
    temp:= xc+1;
    temp1:=yc-2;
    ajustar(temp,temp1);
    end;
    if((strtoint(stringgrid8.cells[0,m])=xc-2) and
(strtoint(stringgrid8.cells[1,m])=yc-1))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
    stringgrid8.cells[3,m]:='1';
    temp:= xc-2;
    temp1:=yc-1;
    ajustar(temp,temp1);
    end;
    if((strtoint(stringgrid8.cells[0,m])=xc-2) and
(strtoint(stringgrid8.cells[1,m])=yc))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
    stringgrid8.cells[3,m]:='1';
    temp:= xc-2;
    temp1:=yc;
    ajustar(temp,temp1);
    end;
    if((strtoint(stringgrid8.cells[0,m])=xc-2) and
(strtoint(stringgrid8.cells[1,m])=yc+1))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
    stringgrid8.cells[3,m]:='1';
    temp:= xc-2;

```

```

        temp1:=yc+1;
        ajustar(temp,temp1);
        end;
        if((strtoint(stringgrid8.cells[0,m])=xc-2) and
(strtoint(stringgrid8.cells[1,m])=yc+2))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
        stringgrid8.cells[3,m]:='1';
        temp:= xc-1;
        temp1:=yc-2;
        ajustar(temp,temp1);
        end;
        if((strtoint(stringgrid8.cells[0,m])=xc-1) and
(strtoint(stringgrid8.cells[1,m])=yc+2))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
        stringgrid8.cells[3,m]:='1';
        temp:= xc-1;
        temp1:=yc+2;
        ajustar(temp,temp1);
        end;
        if((strtoint(stringgrid8.cells[0,m])=xc) and
(strtoint(stringgrid8.cells[1,m])=yc+2))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
        stringgrid8.cells[3,m]:='1';
        temp:= xc;
        temp1:=yc+2;
        ajustar(temp,temp1);
        end;
        if((strtoint(stringgrid8.cells[0,m])=xc+1) and
(strtoint(stringgrid8.cells[1,m])=yc+2))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
        stringgrid8.cells[3,m]:='1';
        temp:= xc+1;
        temp1:=yc+2;
        ajustar(temp,temp1);
        end;
        if((strtoint(stringgrid8.cells[0,m])=xc+2) and
(strtoint(stringgrid8.cells[1,m])=yc+2))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
        stringgrid8.cells[3,m]:='1';
        temp:= xc+2;
        temp1:=yc+2;
        ajustar(temp,temp1);
        end;
        if((strtoint(stringgrid8.cells[0,m])=xc+2) and
(strtoint(stringgrid8.cells[1,m])=yc+1))then
begin

```

```

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
    stringgrid8.cells[3,m]:='1';
    temp:= xc+2;
    temp1:=yc+1;
    ajustar(temp,temp1);
    end;
    if((strtoint(stringgrid8.cells[0,m])=xc+2) and
(strtoint(stringgrid8.cells[1,m])=yc))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
    stringgrid8.cells[3,m]:='1';
    temp:= xc+2;
    temp1:=yc;
    ajustar(temp,temp1);
    end;
    if((strtoint(stringgrid8.cells[0,m])=xc+2) and
(strtoint(stringgrid8.cells[1,m])=yc-1))then
begin

stringgrid10.cells[2,n]:=inttostr(strtoint(stringgrid10.cells[2,n]) +
strtoint(stringgrid8.cells[2,m]));
    stringgrid8.cells[3,m]:='1';
    temp:= xc+2;
    temp1:=yc-1;
    ajustar(temp,temp1);
    end;
    end;

stringgrid10.RowCount:=stringgrid10.RowCount+1;
end;
end;
end; {tabela de acumulação atualizada}
stringgrid11.ColCount:=3;
stringgrid11.RowCount:=2;
stringgrid11.cells[0,0]:='Xc';
stringgrid11.cells[1,0]:='Yc';
stringgrid11.cells[2,0]:='Contador';
for n:=1 to stringgrid10.RowCount-1 do
begin
  for m:=n to stringgrid10.RowCount-1 do
begin
  if(stringgrid10.cells[0,m]<>'')then
begin
  stringgrid11.cells[0,n]:=stringgrid10.cells[0,m];
  stringgrid11.cells[1,n]:=stringgrid10.cells[1,m];
  stringgrid11.cells[2,n]:=stringgrid10.cells[2,m];
  stringgrid10.cells[0,m]:='';
  stringgrid10.cells[1,m]:='';
  stringgrid10.cells[2,m]:='';
  stringgrid11.RowCount:=stringgrid11.RowCount+1;
  break;
end;
end;
end;
//reordena a tabela de acumulacao
for i := 1 to (StringGrid11.RowCount-3 ) do

```

```

begin
  for n:=i+1 to StringGrid11.RowCount-2 do
  begin
    begin
      temp:=strtoint(StringGrid11.Cells[0,i]);
      StringGrid11.Cells[0,i]:=StringGrid11.Cells[0,n];
      StringGrid11.Cells[0,n]:=inttostr(temp);
      temp:=strtoint(StringGrid11.Cells[1,i]);
      StringGrid11.Cells[1,i]:=StringGrid11.Cells[1,n];
      StringGrid11.Cells[1,n]:=inttostr(temp);
      temp:=strtoint(StringGrid11.Cells[2,i]);
      StringGrid11.Cells[2,i]:=StringGrid11.Cells[2,n];
      StringGrid11.Cells[2,n]:=inttostr(temp);
    end;
  end;
end;

procedure TfrmHough.ajustar(var xc1, yc1:integer);
var i: integer;
begin
  for i:=2 to stringgrid2.RowCount-1 do
  begin
    if(strtoint(stringgrid2.Cells[0,i])= xc1)then
    begin
      {verifica se os pontos da tabela sao iguais aos pontos que estao dentro da
      janela}
      if((strtoint(stringgrid2.Cells[1,i])= yc1))then
      begin
        stringgrid2.Cells[0,i]:=inttostr(xc);           //se sim
        troca os pontos pelo ponto central da janela
        stringgrid2.Cells[1,i]:=inttostr(yc);
      end;
    end;
  end;
end;

procedure TfrmHough.Limiar1Click(Sender: TObject);
var
  i,j,c,d,e,ncirculo : Integer;
  NewString : String;
  ClickedOK : Boolean;
begin
  NewString := '';
  ClickedOK := InputQuery('Acumulação', 'Limiar (Menor que '+
  stringgrid11.Cells[2,1] + ')', NewString);
  if ClickedOK then
  begin
    e := StrToInt(NewString);
  end
  else
  exit;
  //Recebe a porcentagem da acumulação
  c := 1;
  ncirculo := 0;
  StringGrid4.ColCount := 2;//Cria duas colunas na tabela StringGrid4

```

```

//Cria uma linha na tabela StringGrid4
StringGrid4.RowCount := 1;
//Se o valor da acumulação do circulo for maior, insere a mesma na tabela
i := 1;
while(i < (StringGrid11.RowCount - 2)) do
begin
  if(StrToInt(StringGrid11.Cells[2,i]) > e) then
  begin
    StringGrid4.Cells[0,0] := StringGrid11.Cells[0,i];
    StringGrid4.Cells[1,0] := StringGrid11.Cells[1,i];
    i := StringGrid8.RowCount + 2;
    ncirculo := 1;
  end;
  i := i + 1;
end;
for i := 1 to (StringGrid11.RowCount - 2) do
begin
  for j := 1 to (StringGrid4.RowCount - 1) do
  begin
    if(StringGrid4.Cells[0,j] = StringGrid11.Cells[0,i]) then
    begin
      if(StringGrid4.Cells[1,j] = StringGrid11.Cells[1,i]) then
      begin
        c := 0;
        break;
      end;
    end;
    end;
    if(StrToInt(StringGrid11.Cells[2,i]) > e) then
    begin
      if(c = 1) then
      begin
        StringGrid4.RowCount := StringGrid4.RowCount + 1;
        StringGrid4.Cells[0,StringGrid4.RowCount - 1] :=
StringGrid11.Cells[0,i];
        StringGrid4.Cells[1,StringGrid4.RowCount - 1] :=
StringGrid11.Cells[1,i];
        ncirculo:= 1;
      end;
      c := 1;
    end;
  end;
  //Exibe o número de circulos detectados
  if(ncirculo > 0 ) then
  begin
    ShowMessage('Foram detectados ' + IntToStr(StringGrid4.RowCount-1) +
' circulos!');
  end
  else
  begin
    ShowMessage('Não foi detectado nenhum circulo com este limiar! ');
  end;
end;

procedure TfrmHough.visualiza01Click(Sender: TObject);
var
  i,j,v,s,li,ls,xcl,ycl,x1l,y1l : Integer;
  aBitMap,bBitMap: TBitMap;
begin

```

```

for i := 1 to (StringGrid4.RowCount - 1) do
begin
  visual := Tvisual.Create(self);
  visual.nBitMap := TBitMap.Create;
  visual.Open(OpenFile.FileName);
  visual.visible:=true;
  li := (visual.nBitMap.Height)-1; { Altura da imagem }
  ls := (visual.nBitMap.Width)-1;
  for v := 0 to li do
  begin
    for s := 0 to ls do
    begin
      visual.nBitMap.Canvas.Pixels[s,v] := $00FFFFFF;
    end;
  end;

  for j := 2 to (StringGrid2.RowCount - 1) do
  begin
    if(StringGrid4.Cells[0,i] = StringGrid2.Cells[0,j]) then
    begin
      if(StringGrid4.Cells[1,i] = StringGrid2.Cells[1,j]) then
{imprime na tela os pixels encontrados}
      begin
        visual.nBitMap.Canvas.Pixels[StrToInt(StringGrid2.Cells[2,j]),StrToInt(StringGrid2.Cells[3,j])] := $00000000;

        visual.nBitMap.Canvas.Pixels[StrToInt(StringGrid2.Cells[4,j]),StrToInt(StringGrid2.Cells[5,j])] := $00000000;

        visual.nBitMap.Canvas.Pixels[StrToInt(StringGrid2.Cells[6,j]),StrToInt(StringGrid2.Cells[7,j])] := $00000000;

        visual.nBitMap.Canvas.Pixels[StrToInt(StringGrid2.Cells[8,j]),StrToInt(StringGrid2.Cells[9,j])] := $00000000;
      end;
    end;
    visual.caption:='xc = '+ StringGrid4.Cells[0,i] + ' yc = ' +
StringGrid4.Cells[1,i];
    visual.FImage.Picture.Assign (visual.nBitMap);
  end;

end;
end.

```