Detecção de círculos em imagens através da tranformada de Hough

Leyza Elmeri Baldo Dorini[1], **Anderson de Rezende Rocha**[1]

[1]Unicamp – Disciplina de Introdução ao Processamento de Imagens Digitais

leyza@inf.unochapeco.rct-sc.br - anderson.rocha@ic.unicamp.br

Resumo. *Este trabalho trata da transformada de Hough para detecção de círculos em imagens. Foram realizadas duas implementações diferentes, sendo a primeira baseada na proposta de Duda e Hart (1972), e a segunda na proposta de Kimme, Ballard e Slansky (1975). O trabalho estrutura-se da seguinte forma: primeiro é apresentada uma breve fundamentação teórica, descrevendo também as duas propostas implementadas, depois são apresentados os resultados obtidos e por fim as conclusões.*

1. Introdução

Supondo que, para n pontos em uma imagem, deseja-se achar subconjuntos destes pontos que estejam alinhados em retas. Uma solução possível é a de primeiro achar todas as linhas determinadas por cada par de pontos, e depois buscar todos os subconjuntos de pontos que estejam próximos de determinadas linhas.
O problema com este procedimento é que o número de segmentos de retas digitais em uma imagem possui ordem superior a $\mathcal{O}(n^4)$ (mais detalhes podem ser obtidos em [3]).
Uma abordagem alternativa, denominada Tranformada de Hough foi proposta em 1962 por Paul Hough. Esta foi elaborada originalmente para detectar linhas retas e curvas, e este método pode ser usado se equações analíticas dos objetos de fronteira são conhecidos. Como vantagem deste método pode-se citar que não é muito sensível a ruídos ou dados imperfeitos. [1];[4].
Existem diferentes propostas para implementação da tranformada de Hough, dentre elas [2]:

- Transformada de Hough padrão: é a implementação utilizada neste trabalho. Será descrita com detalhes posteriormente;
- Tranformada de Hough probabilística: introduz a idéia de calcular a transformada de Hough de apenas uma parte dos pixels da imagem ($0\% \leq \alpha \leq 100\%$), que devem ser escolhidos de forma aleatória. Mais detalhes podem ser obtidos em [8] e [9]. Este último propõe a utilização de α entre 5% e 15%;
- Transformada de Hough hierárquica: onde a Tranformada de Hough é aplicada a uma estrutura piramidal da imagem. Exemplos de utilização podem ser encontrados em [10];[11];[12];[13];

- Transformada de Hough Aleatória: foi introduzida por [14] e em vez de mapear cada pixel da imagem em uma curva no espaço de parâmetros, mapeia uma n-tupla de pixels da imagem em um único ponto no espaço de parâmetros.
- Transformada de Hough Generalizada: onde a Transformada de Hough foi estendida para permitir a identificação de padrões de formas genéricas.

Estas e outras diferentes implementações poderiam ser amplamente discutidas e detalhadas. No entanto, como este não é o foco principal deste trabalho, optou-se por discutir apenas propostas que envolvem a Transformada de Hough padrão.
A seguir são apresentadas algumas propostas para detecção de linhas retas e círculos em imagens.

2. A transformada de Hough para detecção de linhas retas

A proposta de Hough para detectar linhas retas em uma imagem é descrita a seguir. Considerando um ponto (x_i, y_i) e a equação geral da reta na forma *slope-intercept*

$$y_i = ax_i + b$$

Neste caso, existirão infinitas linhas que passam pelo ponto (x_i, y_i), satisfazendo a equação $y_i = ax_i + b$ para diferentes valores de a e b.
No entanto, se esta equação for escrita como $b = -ax_i + y_i$ e considerar-se o plano ab (denomidado espaço de parâmetros), tem-se uma equação de uma única linha que passa pelo ponto (x_i, y_i). Um segundo ponto (x_j, y_j) também terá associada uma linha no espaço de parâmetros.
As retas no espaço de parâmetros associadas aos pontos (x_i, y_i) e (x_j, y_j) irão se interceptar em ab no ponto (a', b'), onde a' representa a inclinação e b' representa o ponto de intersecção com o eixo y da linha que contém (x_i, y_i) e (x_j, y_j) no plano xy [5].
Em resumo, pode-se dizer que os pontos que pertencem à mesma linha no plano xy irão se interceptar no mesmo ponto (a', b') no espaço de parâmetros. Em termos computacionais, a vantagem da tranformada de Hough vem da divisão do plano ab (espaço de parâmetros) em células acumuladoras (Figura 1) [5].

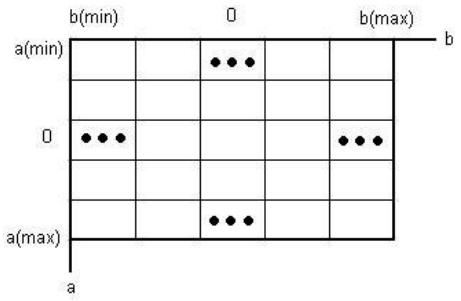
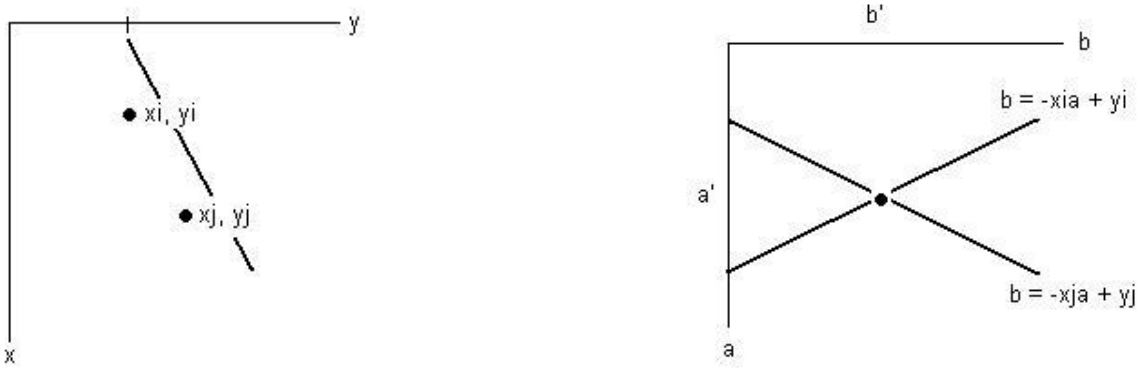


Figure 1: Espaço de parâmetros

Tais células são divididas através da quantização dos valores de a entre a_{min} e a_{max} e dos valores de b entre b_{min} e b_{max}. Inicialmente as células são zeradas. Para cada ponto (x_k, y_k) (pertencentes ao plano da imagem), varrem-se os valores de a entre a_{min} e a_{max} e acha-se o valor correspondente de b através da equação $b = -ax_k + y_k$ (se necessário, é feito um arredondamento do valor obtido para o mais próximo valor quantizado de b). Se a escolha de um certo a_p resulta em uma solução b_q, a célula A(p,q) no espaço de parâmetros é incrementada em um.
No final deste processo, ou seja, quando todos os pontos (x_i, y_i) foram percorridos, um valor V em uma célula A(i,j) no espaço de parâmetros indica que V pontos são colineares na reta $y = a_i x + b_j$. Esta idéia é ilustrada a seguir na Figura 2 [6].



(a) (b)

Figure 2: (a) plano xy (b) plano ab

Sendo assim, pode-se dizer de forma resumida que a Transformada de Hough consiste em mapear cada pixel da imagem para o espaço de parâmetros, que é organizado em forma de um acumulador n-dimensional, onde n corresponde ao número de parâmetros. Analisando um pixel da imagem, procura-se todas as linhas que passam por este ponto, e com base no mapeamento feito para o espaço de parâmetros, a posição do acumulador correspondente ao resutado do mapeamento é incrementado. Assim, pode-se utilizar um limiar que estabelecerá um valor mínimo para decidir se o valor acumulado corresponde a um objeto procurado [2].
O procedimento discutido anteriormente é linear em p (número de pontos da imagem), ou seja, subdividindo-se o eixo a em K incrementos fornece para cada ponto (x_k, y_k), K valores de b correspondendo aos K possíveis valores de a. Desta forma, considerando p pontos na imagem, o método envolverá pK computações [5].
Um problema na utilização da equação geral da reta na forma *slope-intercept* é que tanto a inclinação (*slope*) quanto o ponto de interceção (*intercept*) são ilimitados (quando a reta torna-se vertical, tendem ao infinito) [5];[6];[7].
Para superar esta limitação, Richard O. Duda e Peter E. Hart (7) sugeriram um parametrização alternativa, através da representação normal de uma linha:

$$x \cos\theta \ + \ y \sin\theta \ = \ \rho$$

Se θ for restringida ao intervalo $[0, \ \pi]$ então os parâmetros normais para uma linha são

únicos. Com esta restrição, cada linha no plano xy corresponderá a um único ponto no plano $\theta\rho$.
Supondo agora que tem-se um conjunto $(x_1, y_1), \ldots, (x_n, y_n)$ de n pontos e é desejado encontrar um conjunto de linhas retas que ajuste-os. O que é feito é transformar os pontos de n pontos (x_i, y_i) em curvas no plano $\theta\rho$, definido por:

$$\rho \ = \ x_i \cos\theta \ + \ y_i \sin\theta$$

Pode-se mostrar que as curvas correspondentes à pontos colineares na imagem tem um ponto comum de interseção. Este ponto no plano $\theta\rho$, definido (θ_0, ρ_0) define a linha passando pelos pontos colineares. Assim, o problema de detectar pontos colineares pode ser convertido ao problema de encontrar curvas concorrentes [7].
Uma propriedade dual da tranformada ponto-curva pode ser estabelecida. Suponha que se tenha um conjunto $(\theta_1, \rho_1), \ldots, (\theta_k, \rho_k)$ de pontos no plano $\theta\rho$, todos sobre a curva

$$\rho \ = \ x_0 \cos\theta \ + \ y_0 \sin\theta$$

Pode-se mostrar que todos estes pontos correspondem à linhas no plano xy passando pelo ponto (x_0, y_0). Pode-se sumarizar estas propriedades em quatro items [7]:

- propriedade 1: um ponto no plano da imagem corresponde à uma curva senoidal na plano dos parâmetros;
- propriedade 2: um ponto no plano dos parâmetros corresponde à uma linha reta no plano da imagem;
- propriedade 3: pontos sobre a mesma linha reta no plano da imagem correspondem à curvas sobre um ponto comum no plano dos parâmetros;
- propriedade 4: pontos sobre a mesma curva no plano dos parâmetros correspondem à curvas sobre o mesmo ponto no plano da imagem.

Para evitar um esforço computacional elevado, pode ser feita uma quantização do plano $\theta\rho$, assim como era feito para o plano ab (o intervalo sugerido por [7] é $0 \leq \theta < \pi, -R \leq \rho \leq R$, onde R é o tamanho da "retina").
O'Gorman e Clowes propuseram uma modificação da transformada de Hough, na qual o ângulo θ para entrada no espaço $\theta\rho$ é obtido a partir da direção do gradiente de um contorno [15].

3. A transformada de Hough para detecção de círculos

A seguir serão descritas as duas abordagens que foram implementadas no trabalho, sendo elas baseadas em [7] e [14].

3.1. Algoritmo sugerido por Duda e Hart

No algoritmo sugerido por Richard O. Duda e Peter E. Hart [7], a aplicação da Transformada de Hough para detectar círculos em uma imagem digitizada requer um array de

acumuladores de três dimensões. Este array é indexado por três parâmetros que especificam a localização e o tamanho de um círculo de centro $a = (a_1, a_2)$ e raio r.
Seja $p(x)$ uma imagem resultante de uma detecção de contorno e $x = (x_1, x_2)$ o par de coordenadas de um pixel. Para cada x há um conjunto $\mathcal{C}_x$ de círculos que passam por x. Seja $\mathcal{X}_p$ o conjunto de pontos $\{x \mid p(x) \neq 0\}$.
O algoritmo sugerido por [7] encontra o centro $a(x)$ e o raio $r(x)$ para cada membro de $\mathcal{C}_x$, tal que $x \in \mathcal{X}_p$. Para tal, utiliza-se a representação paramétrica

$$(x - A)^2 \ + \ (y - B)^2 \ = \ C^2$$

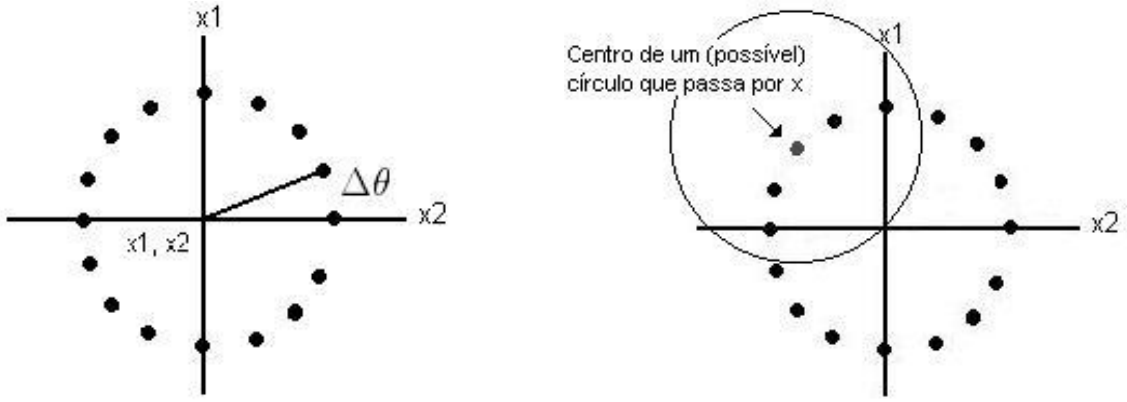
O conjunto resultante de pares centros-raios é denotado por $\{(a, r) \mid x \in \mathcal{X}_p\}$. Para cada membro de $\{(a, r) \mid x \in \mathcal{X}_p\}$ um acumulador em $(a(x), r(x))$ no espaço ab é incrementado em uma unidade.
Depois que todos os membros de $\mathcal{X}_p$ foram processados, o acumulador em $(a(x), r(x))$ irá conter o número de elementos de $\mathcal{X}_p$ que estão sobre o círculo de raio $r(x)$ com centro $a(x)$.

3.2. Algoritmo sugerido por Kimme, Ballard e Sklansky

Este algoritmo é na verdade uma extensão/melhoramento do algoritmo descrito anteriormente, e baseou-se na idéia do algoritmo de O'Gorman e Clowes, citado no final da seção anterior. Para implementar o algoritmo de uma maneira eficiente, os autores utilizaram:

1. um procedimento para gerar os membros digitizados de $\mathcal{C}_x$;
2. a direção do gradiente para eliminar porções de cada membro de $\mathcal{C}_x$.

Vale ressaltar que a notação sendo utilizada segue a da subseção anterior.
Para gerar cada membro de $\mathcal{C}_x$, é calculada a digitização de $(r \cos\theta e \sin\theta)$ para um r fixo, enquanto θ é cresce em incrementos de $\Delta\theta$, onde $\Delta\theta$ varia aproximadamente em $1/r$.
Na verdade, tal digitização gera possíveis centros de círculos que passam pelo ponto x. Uma ilustração é mostrada na Figura 3.



(a) (b)

Figure 3: (a) possíveis centros de círculos de passam por x (b) exemplo de um círculo

No entanto, em vez de guardar no espaço de parâmetros informações sobre todos estes possíveis círculos, pode-se utilizar a informação da direção do gradiente para selecionar apenas aquelas porções de $\mathcal{C}_x$ que provavelmente são centros de círculos.
Considerando as seguintes definições:

- $f(x)$ é o nível de cinza em x na imagem original;
- $g(x)$ é o gradiente de $f(x)$;
- $\phi(x)$ é a direção de $g(x)$;
- $g_1(x)$ e $g_2(x)$ são, respectivamente, os componentes horizontal e vertical de $g(x)$.

Assim, $g(x)$ é definida em termos de $g_1(x)$ e $g_2(x)$:

$$g_1(x) = \frac{1}{2s}\left[f(x\ +\ (s,0)) - f(x-(s,0))\right]$$

$$g_2(x) = \frac{1}{2s}\left[f(x\ +\ (0,s)) - f(x-(0,s))\right]$$

onde s é um inteiro positivo.
A direção do gradiente é dada então por:

$$\phi(x) = \tan^{-1}\left(\frac{g_2(x)}{g_1(x)}\right)$$

Um conceito importante é que se x_1 está sobre um círculo em $f(x)$, então o gradiente de $f(x)$ em $x = x_1$ irá apontar para o centro do círculo.
O procedimento adotado consiste então em selecionar apenas a porção de cada membro de $\mathcal{C}_\S$ tal que $g(x)$ aponta para $\{a, r \mid x \in \mathcal{X}\}$.
Desta forma, pode-se ver que o programa irá precisar como entrada a imagem em tons de cinza ($f(x)$) e também a imagem com detecção de contorno ($p(x)$). Em resumo, pode-se dizer que o procedimento consiste no seguinte:

1. se o pixel em $p(x)$ é diferente de zero (ou seja, é um pixel pertencente à borda), calcula-se o gradiente deste ponto em $f(x)$ (imagem em tons de cinza);
2. com base na direção do gradiente obtida, seleciona-se apenas a porção de $\mathcal{C}_x$ que aponta para esta direção.

A ilustração a seguir busca deixar estes conceitos mais claros.

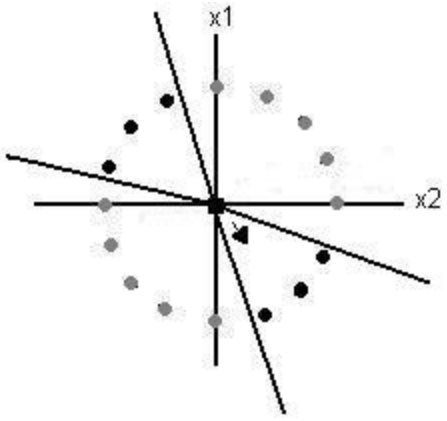


Figure 4: Porções do círculo consideradas

Sendo assim, a abordagem considera apenas os possíveis centros que estão dentro de uma porção da circunferência discretizada na direção do gradiente. Considera-se a direção oposta porque está se trabalhando com arco-tangente.

3.3. Comparação entre os dois algortimos

Obviamente o segundo algoritmo apresenta um desempenho superior ao primeiro, além de reduzir o nível de erro em relação à escolha dos possíveis centros de círculos.
No caso do algoritmo sem a utilização da direção do gradiente, para cada ponto da imagem ((x, y)) que pertencesse ao contorno, percorria-se todo o espaço de parâmetros. Desta forma, com base na equação

$$(x - a)^2 + (y - b)^2 = c^2$$

para cada ponto (x, y) varria-se o espaço de parâmetros encontrando-se os respectivos valores de (a, b) e r.
Se o problema for restringido para a procura de raios fixos (como é a proposta do algoritmo de Kimme, Ballard e Sklansky), para cada ponto (x, y), deve-se percorrer todo o espaço de parâmetros de forma a encontrar/incrementar os pontos (a, b) que satisfaçam à equação apresentada acima para o raio (c) pré-determinado, ou seja:

$$(x - a)^2 + (y - b)^2 = raio\_determinado^2$$

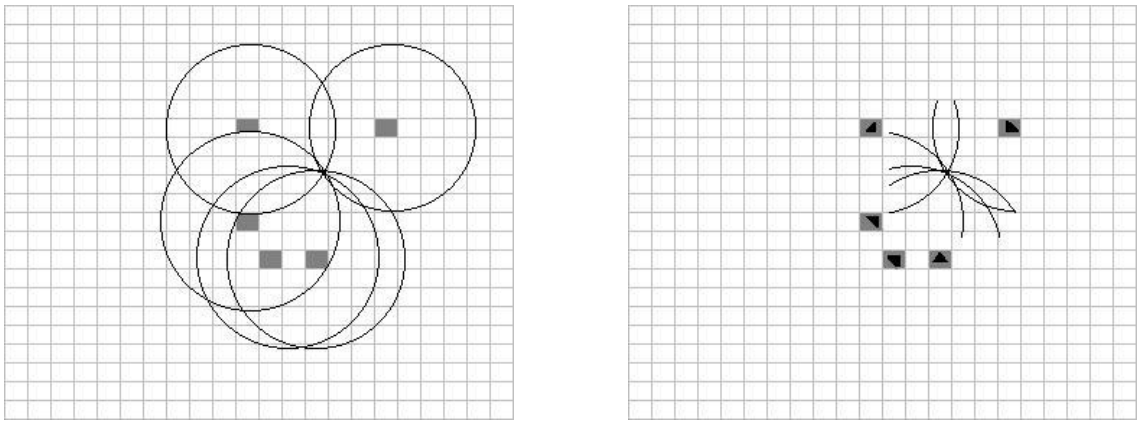
Já o segundo algoritmo, que utiliza a direção do gradiente, irá considerar apenas os pontos que estão dentro da porção do círculo que está na mesma direção que a que o gradiente está apontando. Para determinar tais pontos, utiliza-se a equação

$$(x, y) = r \cos t, r \sin t$$

onde

r = raio do círculo
$t \in [0, 2\pi]$

A Figura 4 a seguir ilustra as afirmações acima.



(a) (b)

Figure 5: (a) espaço de parâmetros sem informação do gradiente (b) espaço de parâmetros com informação do gradiente

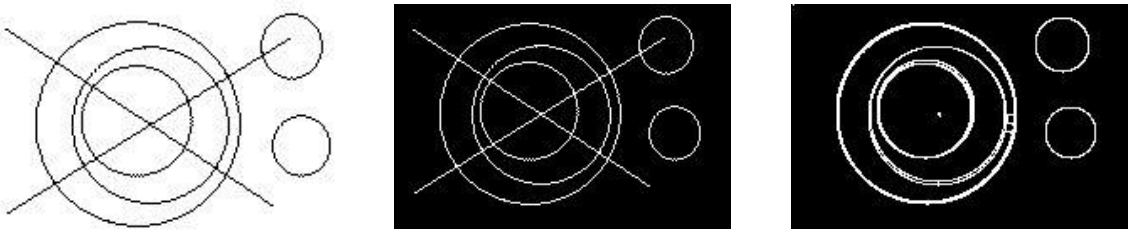
Além das abordagens citadas existem diversas outras para detecção de círculos em imagens. Dentre elas, pode-se citar a abordagem de Davies (1990), que propõe a divisão do cálculo da tranformada em primeiro localizar todos os centros dos círculos e depois encontrar o raio de cada círculo. Um método proposto por [16] baseia-se esta idéia, e utiliza "*intersecting chords*" para localizar os centros dos círculos.

4. Resultados obtidos

Será subdividida a apresentação dos resultados de acordo com os dois tipos de algoritmos implementados, tendo em vista que enquanto um localiza círculos de todos os raios o outro localiza apenas circulos para um raio pré-determinado.

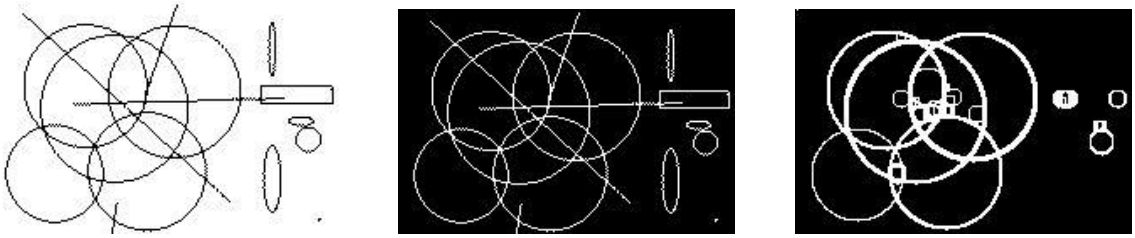
4.1. Resultados obtidos para o algoritmo proposto por Duda e Hart

A apresentação dos resultados irá apresentar a imagem original, a imagem com detecção de contorno e o resultado obtido.



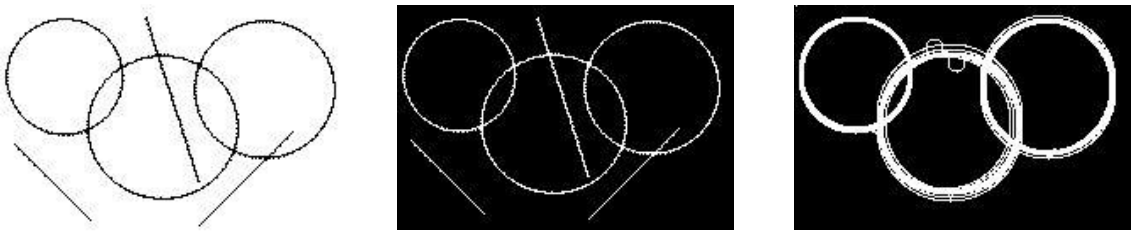
(a) (b) (c)

Figure 6: (a) imagem original (b) detecção de contorno (c) resultado final



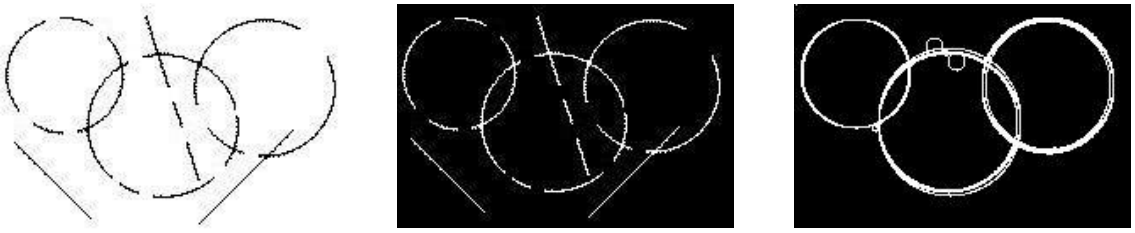
(a) (b) (c)

Figure 7: (a) imagem original (b) detecção de contorno (c) resultado final



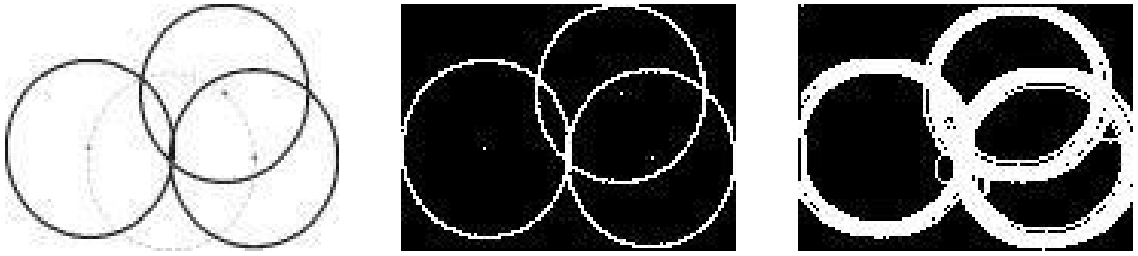
(a) (b) (c)

Figure 8: (a) imagem original (b) detecção de contorno (c) resultado final



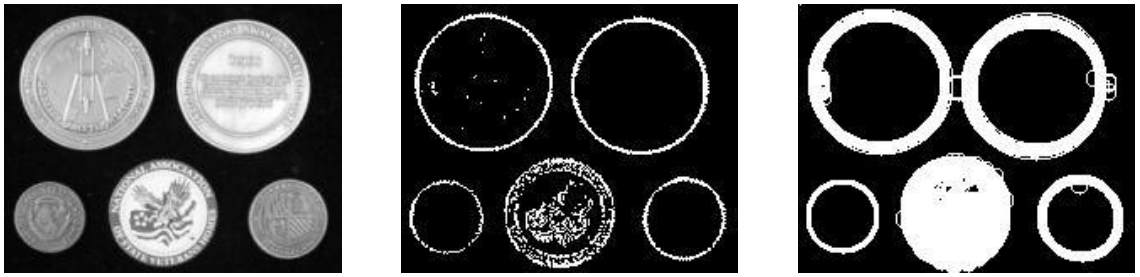
(a) (b) (c)

Figure 9: (a) imagem original (b) detecção de contorno (c) resultado final



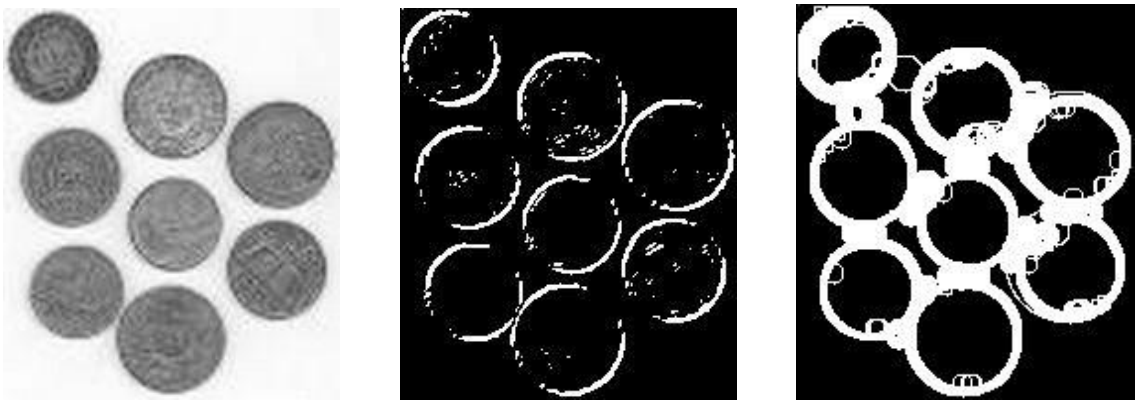
(a) (b) (c)

Figure 10: (a) imagem original (b) detecção de contorno (c) resultado final



(a) (b) (c)

Figure 11: (a) imagem original (b) detecção de contorno (c) resultado final



(a) (b) (c)

Figure 12: (a) imagem original (b) detecção de contorno (c) resultado final

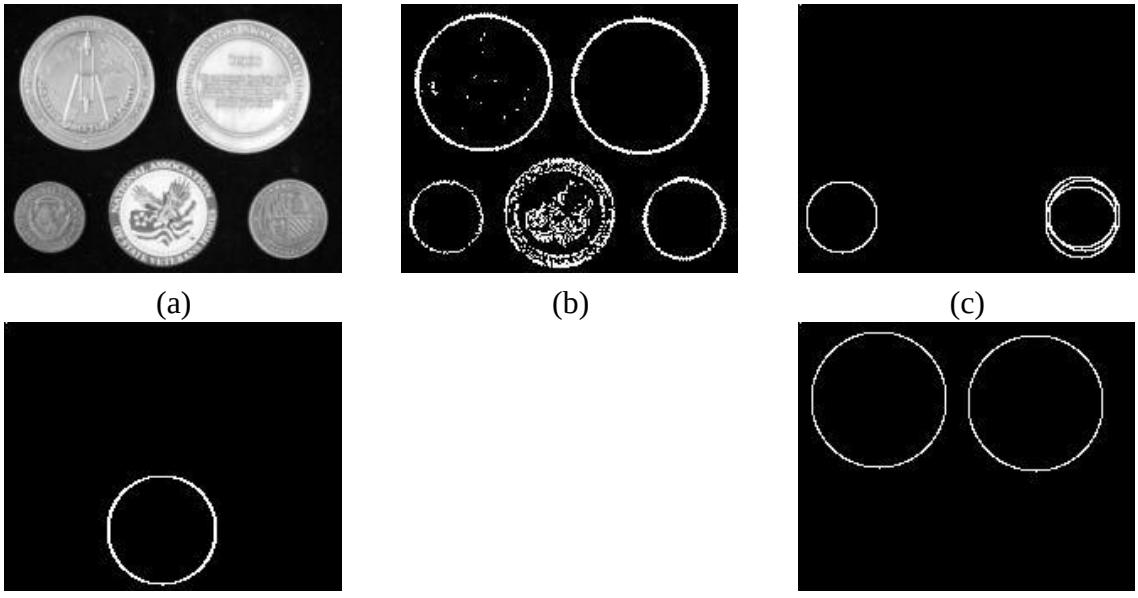
Pode-se dizer que esta abordagem funciona melhor para imagens sintéticas do que para imagens reais. Isto porque a detecção de contorno de imagens reais acaba gerando algumas ”sujeiras” (ou seja, pontos que não pertencem de fato ao contorno) que podem vir a ser interpretadas como círculos de raio pequeno na imagem.
No entanto, acredita-se com com uma calibragem mais exata dos parâmetros (valor do limiar, por exemplo) seria possível detectar menos ”círculos falsos”.

4.2. Resultados obtidos para o algoritmo proposto por Kimme, Ballard, Sklansky

A apresentação dos resultados irá apresentar a imagem original, a imagem com detecção de contorno e o resultado obtido.

(a) (b) (c)

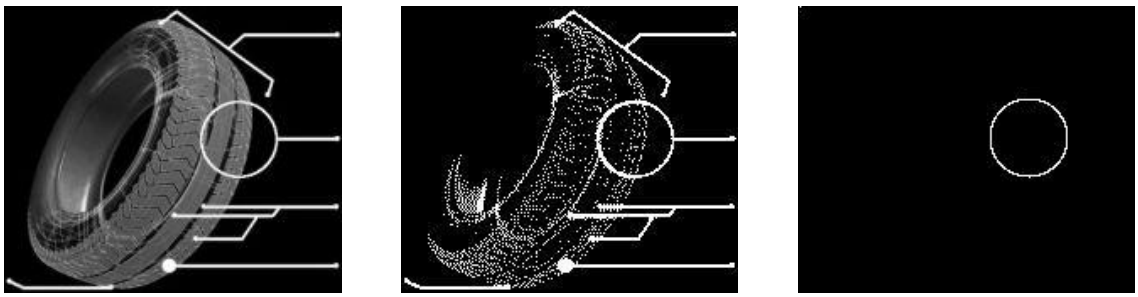
Figure 13: (a) imagem original (b) detecção de contorno (c) resultado para raio = 23



(a) (b) (c)

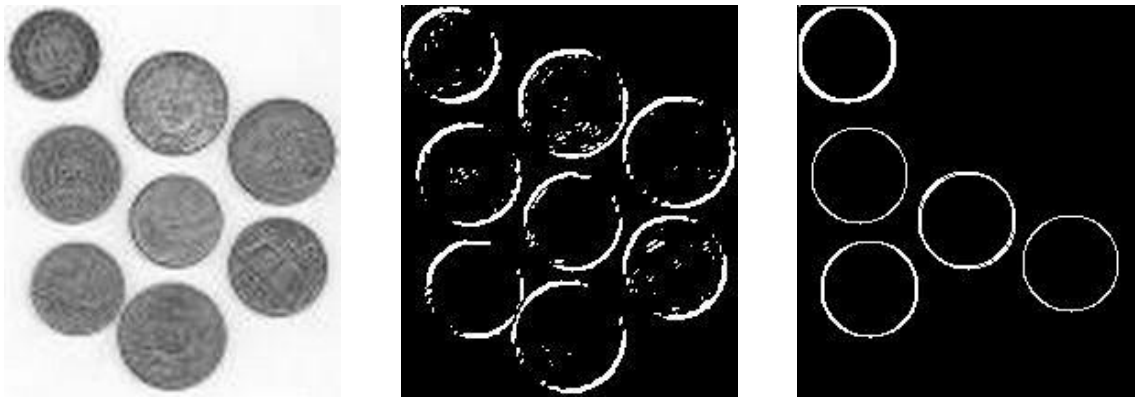
(d) (e)

Figure 14: (a) imagem original (b) detecção de contorno (c) resultado para raio = 21 (d) resultado para raio = 32 (e) resultado para raio = 40

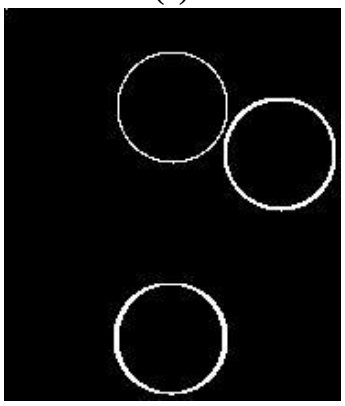


(a) (b) (c)

Figure 15: (a) imagem original (b) detecção de contorno (c) resultado para raio = 32



(a) (b) (c)



(d)

Figure 16: (a) imagem original (b) detecção de contorno (c) resultado para raio = 26 (d) resultado para raio = 30

Como pode-se observar, o algoritmo obteve um bom desempenho. Algumas conclusões são apresentadas a seguir.

5. Conclusões

Em relação ao primeiro algoritmo implementado, sugerido por Duda e Hart, pode-se dizer que para o caso específico da implementação este se apresentou melhor para imagens sintéticas. Acredita-se que isto se deve ao limiar escolhido, sendo que quando é feita a detecção de contorno em imagens reais pode-se ter resíduos (ou seja, pontos que não pertencem de fato ao contorno), que podem vir a ser encarados como pequenos círculos.
Além disso, as bordas podem não estar totalmente legíveis (embora o algoritmo seja robusto a ruídos, conforme pode-se obsevar na Figura 9). Desta forma, seria necessário um estudo mais aprofundado sobre a escolha do limiar ideal.
O outro algoritmo, sugerido por Kimme, Ballard, Sklansky, apresentou-se muito bom. No entanto, o problema em relação à escolha de um limiar ideal continua. Se é dado um limiar muito baixo, pode acontecer de falsos círculos serem detectados, e por outro lado, se o limiar é muito alto círculos podem não ser detectados.
Outra questão a ser observada é que, embora o limiar seja dependente do raio, um círculo com falhas nas bordas exige um limiar mais baixo para ser detectado que um com bordas inteiras.
Como uma forma de comparar os dois algoritmos desenvolvidos, foi feita uma adaptação no algoritmo sugerido por Duda e Hart de forma que este detectasse apenas círculos de um raio fixo (no caso do teste, raio igual a 40). Como parâmetros de comparação foram medidos o tempo de execução e a quantidade de elementos no espaço de parâmetros. Os resultados são apresentados a seguir:

- o algoritmo sugerido por Duda e Hart apresentou 521.410 elementos no espaço de parâmetros e levou 361 segundos para executar;
- o outro algoritmo armazenou 145.662 elementos no espaço de parâmetros e levou 1 segundo para executar.

As saídas são apresentadas abaixo na Figura 17.

(a) (b)

Figure 17: (a) Duda e Hart (b) Kimme, Ballard e Sklansky

É importante ressaltar que os critérios utilizados para definição do limiar foram diferentes.

6. Referências

1 JAMUNDA, T. Reconhecimento de formas: a tranformada de Hough. Disciplina de Seminário em Visão Computacional. UFSC - Universidade Federal de Santa Catarina, 2000.

2 HADAD, R. M. Identificação de padrões em imagens de satélites - formas circulares. Tese de doutorado. UFMG - Universidade Federal de Minas Gerais, Belo Horizonte, Minas Gerais. Setembro de 2000.

3 KOPLOWITZ, J.; LINDENBAUM, M.; BRUCKTEIN A.. The number of digital straight lines on a NxN grid. IEEE Trns. Theory, 36(1), Jan. 1990, 192-197.

4 SONKA, M. Image processing, Analysis And Machine Vision. Chapman & Hall, 1993.

5 GONZALEZ, Rafael C; WOODS, Richard E. Processamento de imagens digitais. São Paulo. Edgard Blucher: 2000.

6 ???

7 DUDA, R.; HART, P. Use of the Hough transform to detect lines and curves in pictures. Graphics and Image processing.

8 BERGEN. J. R.; SHVAYTSER H. A probabilistic algorithm for computing Hough Transforms, Journal os Algorithms, vol. 12, pp. 639-656, 1991

9 KIRYATI, N.; ELDAR, Y.; BRUCKSTEIN, M. A probabilistic Hough Transform, Pattern Recognition, vol. 24, n.4, pp. 306-316, 1991

10 PRINCEN, J.; ILLINGWORTH, J.; KITTLER, J. A hierarchical approach to line extraction, Proceedings CVPŔ89. IEEE, Jun 1989, vol. 1, pp. 92-97, IEEE Comp. Soc. Press, 1989

11 PRINCEN, J.; ILLINGWORTH, J.; KITTLER, J. A hierarchical approach to line extraction based on the Hough transform, Computer Vision, Graphics and Image Processing, vol. 52, pp. 57-77, 1990

12 YACOUB, S.; JOLION J. HIerarc hical line extraction. IEEE Proceedings - Vision, Image and Signal Processing, vol. 142, n. 1, pp. 7-14, Feb. 1995

13 JOLION J.; ROSENFELD A. An O(log n) pyramid Hough transform, Pattern Recognition Letters, vol 9, pp. 343-349, Jun 1989

14 KIMME, C.; BALLARD, D.; SKLANSKY J. Finding Circles by an Array of Accumulators. Communications of ACM, Vol. 18, pp. 120-122, 1975

15 PRATT, Wlliam K. Digital Image Processing. New York. John Wiley & Sons: 1991.