dx.doi.org/10.7437/NT2236-7640.2012.01.003
Notas Técnicas, v. 2, n. 1, 2012

Uso da Transformada de Hough para detecção de arcos

C.R. Bom* & Marcelo P. de Albuquerque†
Centro Brasileiro de Pesquisas Físicas, Rua Dr. Xavier Sigaud 150, Rio de Janeiro, RJ - 22290-180, Brazil

M. Makler‡
Centro Brasileiro de Pesquisas Físicas, Rua Dr. Xavier Sigaud 150, Rio de Janeiro, RJ - 22290-180, Brazil &
Laboratório Interinstitucional de e-Astronomia - LIneA,
Rua Gal. José Cristino 77, Rio de Janeiro, RJ - 20921-400, Brazil

Esta Nota Técnica tem como objetivo o estudo sistematico da Transformada de Hough implementada para detecção de arcos de circunferência. Neste estudo faremos uma análise da eficiência desse método com diferentes aberturas angulares e espessuras dos arcos.

1. INTRODUÇÃO

A Transformada de Hough (TH), é uma técnica de processamento de imagem digital desenvolvida para encontrar linhas retas que foi posteriormente generalizada para detectar formas arbitrárias que possam ser parametrizáveis [1, 2].

Esse método independe da orientação e da forma do objeto em análise e é uma ferramenta comumente utilizada na visão artificial de reconhecimento de formas [3].

No contexto desse trabalho aplicaremos a TH na detecção de arcos circulares ou objetos que possam ser aproximados como arcos de círculos. A detecção de objetos arqueados é um problema comum em vetorização de imagens [3] e na busca de objetos astronômicos, como arcos gravitacionais. Esses objetos, oriundos de casos extremos do fenômeno de lenteamento gravitacional [4, 5] formam figuras arqueadas nas imagens astronômicas e são objetos de interesse em estudos de Cosmologia e Astrofísica [6–13].

2. MÉTODO

2.1. Transformada de Hough

Essa técnica consiste no mapeamento, no caso da imagens digitais, de um pixel em um espaço cartesiano discreto (x,y), em um espaço abstrato dos n parâmetros a_i, $(a_1,a_2,a_3...,a_n)$, que definem o Espaço de Hough, ou espaço dos parâmetros.

Se um conjunto de pontos p_i que pertencem à imagem está associado a uma curva parametrizada descrita por um mesmo conjunto de parâmetros $(\alpha_1,\alpha_2,...,\alpha_n)$, cada um desses pontos será mapeado no espaço dos parâmetros nas mesmas coordenadas. A este conjunto de parâmetros será adicionado um valor, chamado de voto, para cada ponto p_i mapeado em $(\alpha_1,\alpha_2,...,\alpha_n)$. A(s) coordenada(s) no espaço de parâmetros com os maiores votos são os candidatos mais prováveis para os parâmetros da forma procurada. No caso de retas que satisfazem a equação $y = ax + b$, por exemplo, os pontos (x,y) serão mapeados no espaço dos parâmetros (a,b). Como no processamento imagens digitais tratamos de espaços discretos, o Espaço de Hough também é discreto.

2.2. Transformada de Hough para detecção de arcos de circunferência

Um arco de circunferência é descrito, essencialmente, pela mesma equação polar do círculo, sua única diferença é o domínio do ângulo polar θ. Por essa razão utilizaremos a T.H. implementada para a detecção de círculos.

Uma circunferência é descrita por três parâmetros: o raio r e a posição do centro $O = (x_c,y_c)$. E, portanto, iremos mapear cada ponto da imagem (x,y) em um espaço de parâmetros (x_c,y_c,r).

Explicitamente, podemos escrever a transformação de uma coordenada da imagem para o Espaço de Hough da seguinte maneira. Em uma imagem binarizada, para cada ponto ponto aceso, isto é de intensidade não nula, são calculadas todas as circunferências que passam pelo ponto $p_j = (x_j,y_j)$ em um dado intervalo de raio.

$$x_c = x_j - r_i * sin\theta, \tag{1}$$

$$y_c = y_j - r_i * sin\theta, \tag{2}$$

$$0 \leq \theta \leq 2\pi, \tag{3}$$

$$r_{min} \leq r_i \leq r_{max}. \tag{4}$$

Sendo r_{min} e r_{max} os limites superior e inferior do raio dos círculos que estamos procurando, respectivamente. Para o raio fixo r_i e para cada θ são determinados x_c e y_c e esse ponto receberá um voto na matriz do Espaço de Hough, ou matriz acumuladora:

$$HoughSpace[x_c,y_c] += 1. \tag{5}$$

Como estamos trabalhando em um domínio discreto, é preciso definir um incremento de θ, $d\theta$ tal que seja possível

*Electronic address: debom@cbpf.br
†Electronic address: marcelo@cbpf.br
‡Electronic address: martin@cbpf.br

gerar um círculo a partir de dado raio e centro sem falhas, isto é, com pixels conectados. A partir de testes preliminares, foi possível determinar que:

$$d\theta = \frac{1}{|r_i|} \tag{6}$$

satisfaz essa condição.

Esse resultado pode ser atribuído ao fato de que o tamanho do pixel é a unidade de comprimento e, como o raio é dado em pixels vale a relação para comprimento A de arco de circunferência (θ em radianos):

$$A = r * \theta. \tag{7}$$

Podemos concluir, assim, que a menor unidade de arco seria $A = 1$ o que define um ângulo mínimo $d\theta$ recuperando a condição estabelecida em 6.

2.3. Algoritmo

Devido a inúmeras facilidades para a manipulação de matrizes e imagens, o algoritmo da TH para círculos foi escrito na linguagem Python. Como a motivação imediata da aplicação do método é a detecção de arcos gravitacionais analisaremos imagens no fortmato FITS, um formato padrão para imagens astronômicas [14].

No problema da detecção desses objetos analisaremos uma imagem no formato FITS onde cada objeto já foi identificado e será processado individualmente, isto é, que foi previamente rotulada (ou "labelizada", no jargão de processamento de imagens).

Nesse trabalho o processamento de cada raio é feito de forma separada, evitando o uso de uma matriz 3D. Fixado um raio r_i a imagem é mapeada em uma matriz (x_c, y_c). Após analisar essa matriz a procura de arcos ela é apagada da memória e uma nova matriz, dessa vez para o raio $r_{i+1} = r_i + dr$, é criada e o procedimento repetido.

Este procedimento otimiza o uso de memória. Em testes preliminares, utilizando uma matriz 3D, uma imagem em níveis de cinza e de tamanho em disco 1000 x 1000 x 8 bits e um computador com 1GByte de memória RAM o algoritmo não podia ser executado por falta de memória.

Podemos verificar isso fazendo uma estimativa do uso de memória. Consideremos uma imagem 1000 x 1000 e um teste típico onde procuraremos circunferências em um intervalo de raios de 1% à 50% da largura da imagem (neste caso de 10 à 500 pixels com incremento de um pixel) se cada componente da matriz é um inteiro de 2 bytes e se a matriz (x_c, y_c) tem o mesmo tamanho da imagem, ou seja, o centro da circunferência procurada está localizado no interior da imagem, então o tamanho em memória da matriz (x_c, y_c, r) seria $2 \times 10^3 \times 10^3 \times 5 \times 10^2 = 10^9$, ou seja, da ordem de 1GBytes de memória RAM alocada somente para a matriz que representa o Espaço de Hough. Se os inteiros forem de 4 bytes necessitaremos de uma matriz de cerca de 2GBytes .

Por outro lado, se calcularmos a matriz dos parâmetros (x_c, y_c) para cada raio r, salvando o(s) ponto(s) de maior votação, esta matriz teria tamanho em memória $2 \times 10^3 \times 10^3 = 2 \times 10^6$ bytes, ou cerca de 2MB.

No algoritmo a seguir executaremos a TH para buscar o melhor valor para o raio do arco com o centro da curvatura contido no espaço definido por $n_i \times n_j$, sendo n_i e n_j as dimensões do eixo i e j, respectivamente, no qual desejamos procurar os arcos. Os parâmetros de entrada da função serão, portanto, n_i e n_j, a lista de pontos pertencentes ao objeto (i.e. pontos acesos da imagem) $p(x,y) = p[0][1]$ e o intervalo do raio r_i procurado, r_{min} e r_{max}. É possível também definir opcionalmente o valor do incremento do raio, dr. A seguir mostraremos a implementação do algoritmo em python.

```
from math import sin,cos,atan
from numpy import *

def Hough(p,r_min,r_max,n_i,n_j,dr=1):
   pi=4*atan(1)
   r=range(r_min,r_max,dr)
   r_votacao=numpy.zeros((len(r)+1))
   for k in range(0,len(r)):
      HoughSpace=numpy.zeros((n_i,n_j))
      dtheta=float(r[k])
      dtheta=1/dtheta
      theta=range(0,2*pi,dtheta)
      for i in range(0,len(p[0]):
         Controle=numpy.zeros((n_i,n_j))
         x=p[1][i]
         y=p[0][i]
         for t in range(0,len(theta):
            y_c=y-(raio*sin(theta[t]))
            x_c=x-(raio*cos(theta[t]))
            x_c=int(round(x_c,0))
            y_c=int(round(y_c,0))
            if (n_j>y_c) and (n_i>x_c)
             and (Controle[y_c,x_c]==0):
               Controle[yc,xc]+=1
               HoughSpace[yc,xc]+=1
      Maior_Votacao=HoughSpace.max()
      r_votacao[k]=Maior_Votacao
   indice_melhor_raio=numpy.where
        (r_votacao==r_votacao.max())
   Melhor_Raio=r[indice_melhor_raio[0]]
   return MelhorRaio
```

Na função acima foram utilizados os módulos *math* e *numpy* da Linguagem Python. A Figura 1 ilustra a matriz acumuladora de um dado raio r_i que obteve o pixel de cordenadas (x_c, y_c) de maior votação em todo intervalo de parâmetros testados. Neste exemplo ambos os arcos têm o mesmo raio. Note que, como esperado, conforme nos aproximamos do centro de curvatura a intensidade, que na imagem representa a votação, cresce rapidamente. De fato o centro da curvatura é o único ponto que recebe votação de todos os pontos que pertencem ao arco. Nas seções seguintes, discutiremos testes do algoritmo de busca do melhor raio, para arcos de circunferência gerados com diferentes aberturas angulares, raios e espessuras.

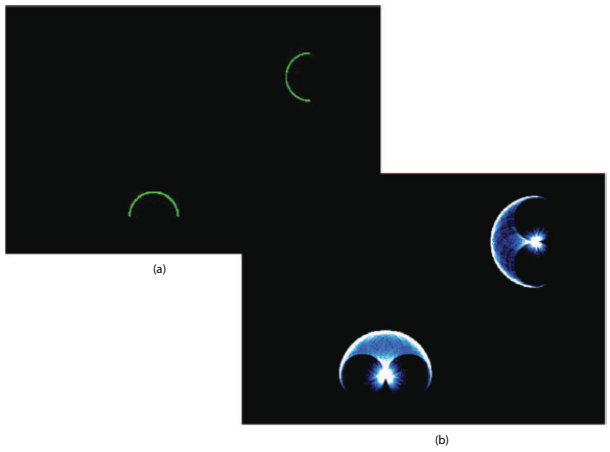

Figura 1: (a) Exemplo de imagem com dois arcos de circunferência. (b) Representação da Matriz Acumuladora para o melhor raio. Observe que na posição do centro de cada arco a intensidade é maior.

3. ANÁLISE

A função Hough() calcula a maior contagem dada a um único ponto da matriz acumuladora $HoughSpace(x_c, y_c)$ para cada raio individualmente. O ponto de maior contagem individual é o centro mais provável para uma dada circunferência de raio r_i. A maior contagem associada ao raio r_i nos fornece uma informação de quanto é mais provável uma circunferência com este raio na imagem. Esse comportamento esta ilustrado no gráfico da Figura 2, onde podemos observar um pico de contagem para o raio utilizado como entrada para desenhar o arco na imagem, neste caso $r_i = 60$ pixels.

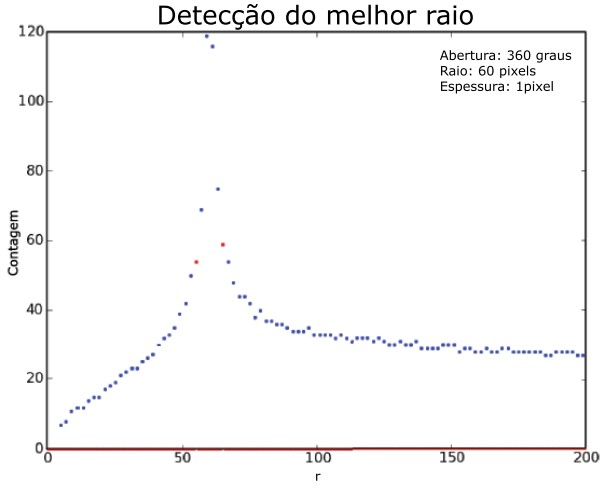

Figura 2: Gráfico da máxima contagem em um Espaço de Hough (x_c, y_c, r_i) pelo raio r_i em uma imagem com um arco de circunferência.

No gráfico da Figura 3 analisamos a dependência do pico de melhor raio em diferentes aberturas angulares. Podemos observar que a acurácia do método começa a cair significativamente com aberturas inferiores a 40 graus para as três configurações de raios testados.

Na Figura 4 podemos observar a variação da largura à

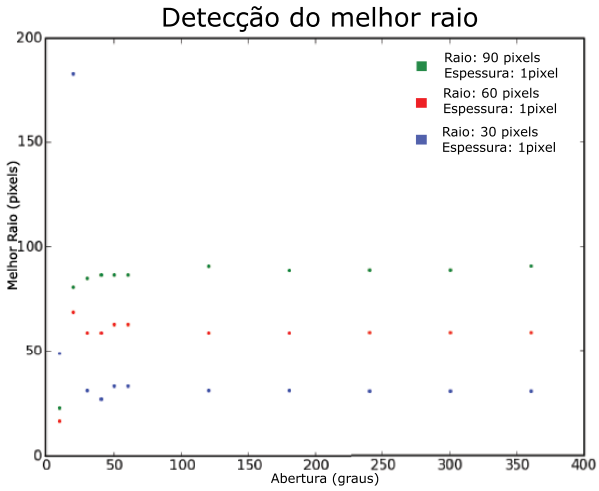

Figura 3: Análise do melhor raio encontrado em função da abertura angular para arcos de diferentes raios.

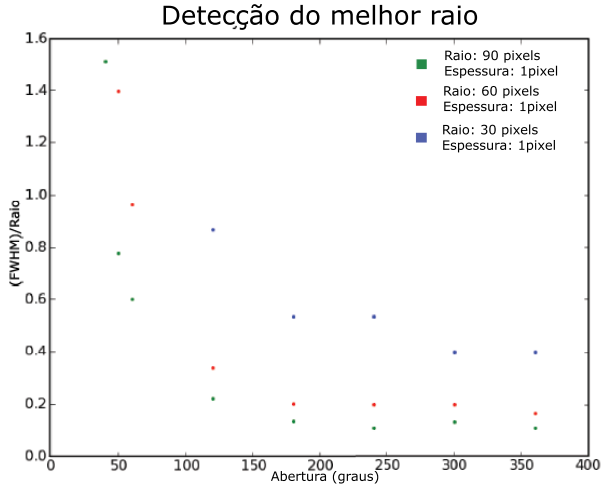

Figura 4: FWHM em função da abertura angular para arcos de diferentes raios.

meia altura (Full Width at Half Maximum,FWHM) das curvas de contagem $\times$ raio e notamos que os picos de contagem também se mostraram mais largos para pequenas aberturas angulares. Como um critério para detecção do arco podemos utilizar a intensidade e largura das medidas dos picos de contagem, fica claro, então que o método deve perder eficiência rapidamente para pequenas aberturas, também em acordo com as medidas apresentadas nos gráficos 3 e 5.

Consideramos também o efeito causado pelo aumento da espessura do arco. Um arco de círculo com espessura e tal que $e > 1$ pixel pode ser considerado como um conjunto de arcos próximos e concêntricos, mas, com raios diferentes. Por esta razão, esperamos que, com o aumento da espessura, os picos da curvas de contagem em função do raio sejam mais largos. O gráfico da Figura 6 ilustra esse efeito.

Na Figura 7 mostramos um exemplo do gráfico de contagem em função do raio para uma abertura menor que aquelas apresentadas no gráfico 6. Observamos que o efeito combinado da espessura e da menor abertura angular fazem

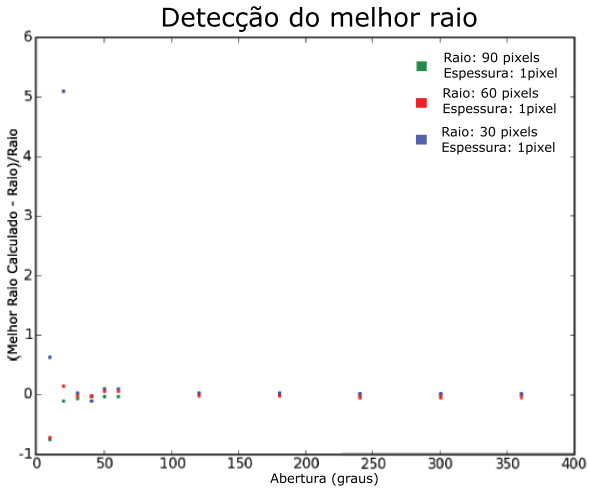

Figura 5: Análise da acurácia do método em função da abertura angular para arcos de diferentes raios.

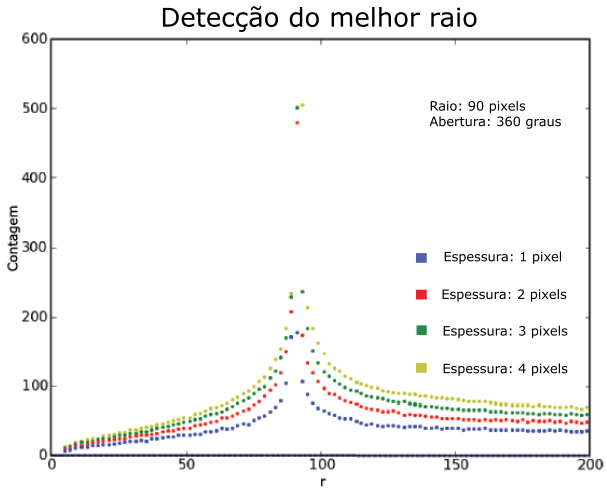

Figura 6: Contagem em função do raio para diferentes espessuras do arco usando uma abertura de 360 graus.

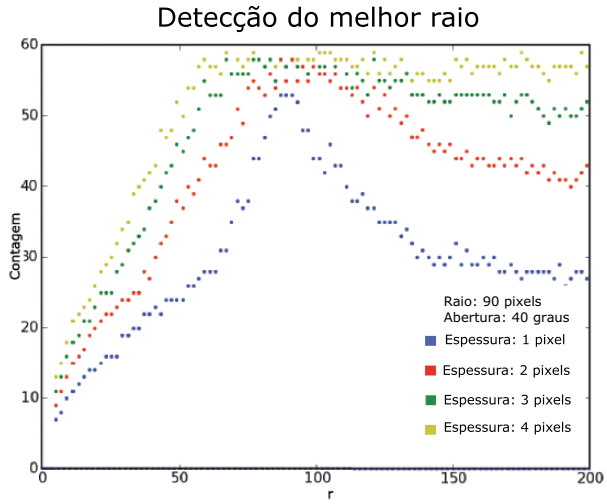

Figura 7: Gráfico de contagem em função do raio para diferentes espessuras abertura de 40 graus.

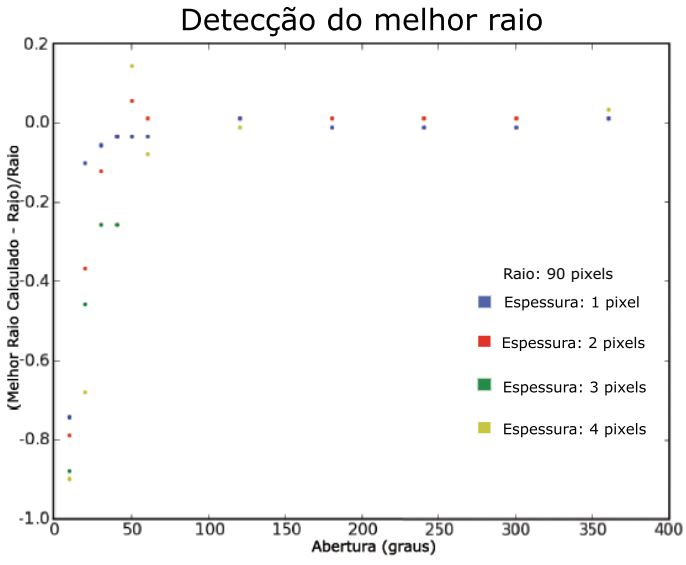

Figura 8: Análise da acurácia do método em função da abertura angular para arcos de diferentes espessuras.

com que seja muito difícil identificar um pico em aberturas com $e \geq 3pixels$.

O gráfico da Figura 8 mostra a perda da eficiência em função da abertura angular para diferentes espessuras. Para aberturas $\approx$ 60 graus, observamos uma discrepância significativa da ordem de 10%. Notamos que o algoritmo, para pequenas aberturas, esteja subestimando o raio. Isso se deve ao fato de que os arcos testados são construídos do maior raio para o menor.

Como a sensibilidade à espessura está ligada diretamente ao raio do arco, pois, com $r \gg e$, o conjunto de arcos concêntricos $r_1 \approx r_2 \approx r_3 \approx ...r_{e-r}$ não devem interferir significativamente na estimativa do raio. Realizamos um teste no qual mantivemos a razão r/e fixa. O resultado, mostrado no gráfico da Figura 9 mostra perfis quase idênticos.

4. CONCLUSÕES

O método da Transformada de Hough foi aplicado à identificação de segmentos de círculos em imagens digitais para diferentes raios, aberturas e espessuras e verificamos que esse método é eficiente para aberturas superiores a 40 graus, quando o erro percentual na medida do melhor raio é da ordem de 10% ou menor. A análise realizada nos arcos mostrou que a detecção é bastante sensível a relação r/e favorecendo a detecção de arcos mais finos. Apesar do FWHM dos picos de contagem de melhor raio ter se mostrado sensível a razão r/e , o erro percentual da medida do raio não muda significativamente com espessuras maiores nos intervalos testados, pois o erro percentual permaneceu menor que 5% para arcos de abertura $\geq$ 60 graus.

O método apresentado nesta Nota Técnica parece razoável para detecção de arcos de circunferência com aber-

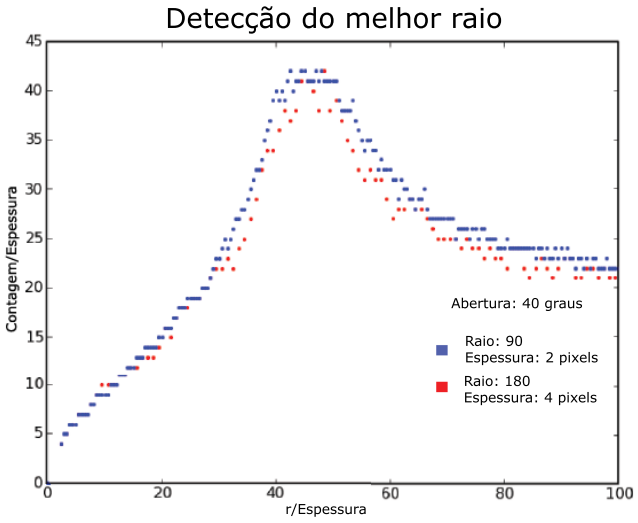

Figura 9: Gráfico da razão Contagem/Espessura em função da razão Raio/Espessura para perfis de mesma razão Raio/Espessura.

turas $>$ 40 graus. Entretanto o método traz uma enorme desvantagem, o tempo de processamento. Os testes duram em média cerca de uma hora por objeto testado, objetos com dezenas à centenas de pontos em um computador Celeron 2.4GHz com 256MByte de memória RAM em uma imagem com resolução de 1000 x 1000. Esse tempo poderia ser reduzido razoavelmente se houvesse uma estimativa inicial do raio a ser procurado, já que foi preciso fazer a transformada em um intervalo grande de raios (de 10% a 50% da largura da imagem).

5. AGRADECIMENTOS

Esse trabalho foi possivel graças ao apoio do Laboratório Interinstitucional de e-Astronomia (LIneA) operado em conjunto pelo o Centro Brasileiro de Pesquisas Físicas (CBPF), o Laboratório Nacional de Computação Científica (LNCC) e o Observatório Nacional (ON) e financiado pelo Ministério da Ciência e Tecnologia e Inovação (MCTI).

Esse trabalho foi parcialmente financiado por agencias de fomento Brasileiras CNPQ (projetos 312425/2006-6, 486138/2007-0 e 312876/2009-2), FINEP (projeto 01.06.0383.00), e FAPERJ (projeto E-26/171.206/2006).

[1] Duda, R.; Hart, P.; *Use of the Hough Transformation to Detect Lines and Curves in Pictures*, 1972, Comm.of ACM 15, 1. 11;

[2] Hough, P.V.C.; *Machine Analysis of Bubble Chamber Pictures*, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

[3] Macedo, M.M.G.; *Uso da Transformada de Hough na Vetorização de Moldes e Outras Aplicações*, 2005, Dissertação (Mestrado em Ciência da Computação) - UFF.

[4] Schneider, P.; Ehlers, J.; Falco, E.E.; *Gravitational Lenses*, 1992, Springer-Verlag, Berlin.

[5] Mollerach, S.; Roulet, E.; *Gravitational Lensing and Microlensing*, 2002, World Scientific.

[6] Rozo, E.; Nagai, D.; Keeton, C., Kravtsov, A. *The Impact of Baryonic Cooling on Giant Arc Abundances*, 2008, ApJ 687 22; astro-ph/0609621.

[7] Meneghetti,M.; Jain, B.; Bartelmann, M.; Dolag, K. *Constraints on Dark Energy Models from Galaxy Clusters with Multiple Arcs*, 2005, Mon.Not.Roy.Astron.Soc 362, 1301; astro-ph/0409030.

[8] Alard, C.; *Automated Detection of Gravitational Arcs* astro-ph/0606757.

[9] Seidel, G.; Bartelmann, M. *Arcfinder: An Algorithm for the Automatic Detection of Gravitational Arcs*, 2007, A&A 472, 341; astro-ph/0607547.

[10] Caminha, G.B.; *Cálculo da Abundância de Arcos Gravitacionais am Aglomerados de Galáxias*, 2009, Dissertação (Mestrado em Física) - CBPF .

[11] Ferreira, P. C.; *Simulações de arcos gravitacionais em imagens e estudo de sua raz ao axial*, 2010, Dissertação (Mestrado em Física) - UFRJ .

[12] Dumet M., H.; *Modelagens Semianalíticas para Arcos Gravitacionais: Seção de Choque e Método Perturbativo em Lentes Pseudoelípticas*, 2011, Tese (Doutorado em Física) - CBPF .

[13] Bom, C.R.; Makler M.; Albuquerque, Marcelo P. ; *Busca de um Método Automatizado para a Detecção de Arcos Gravitacionais*, em preparação.

[14] Pence, W.D.; Chiappetti, L.; Page, C.G.; Shaw, R.A.; Stobie, E. *Definition of the Flexible Image Transport System (FITS), version 3.0*, 2010, A&A 524, A42.