Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM
Disciplina: BCC 326 - Processamento de Imagem
Professor: Guillermo Camara Chavez

Lista 05 - PDI

Aluno: Felipe Augusto Vasconcelos e Silva
Matrícula: 16.2.4358

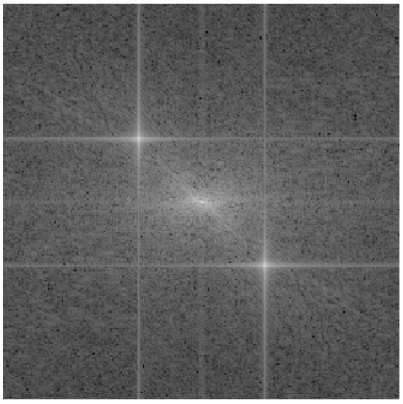
Questão 01) Dada a seguinte imagem, eliminar o ruído produzido pelo ruído periódico. Utilize a filtragem no domínio da frequência, primeiro calcule o espectro de Fourier (Figura b) e elimine a região ao redor dos “spikes” (assinalados com a seta vermelha). Os “spikes” estão localizados nas coordenadas (88,88) e (170,170). A Figura (c) mostra o processo depois de apagar esses valores. Para apagar os valores, basta atribuir zero para todos os elementos da região. Defina uma região ao redor de cada coordenada, teste diferentes tamanhos. Por exemplo, seja M a matriz de coeficientes de Fourier, para “apagar” os coeficientes, atribuir M(88-5:88+5, 88-5:88+5) = 0.

```
% 01 - Leitura da imagem a ser tratada
img = imread('lenna_periodico2.png');
imshow(img);
```

Resultado:

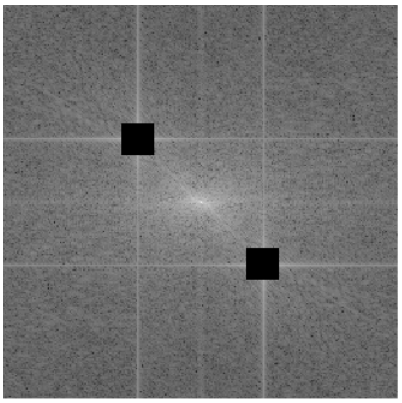
```
% 02 - Calculando a tranformada de Fourier.
fimg = fftshift(fft2(img));
imshow(log(abs(fimg)+1), []);
```

Resultado:



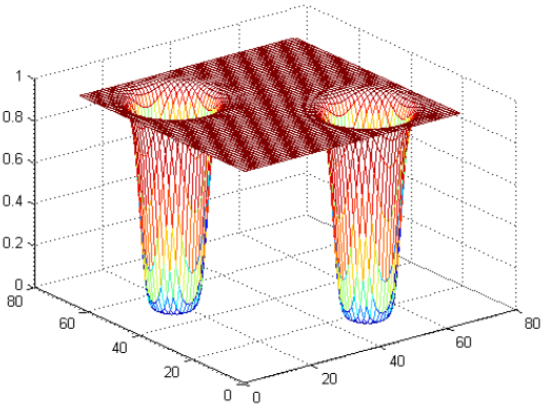
```
% 03 - Removendo os spikes
aux = 10;
fimg(88-aux:88+aux, 88-aux:88+aux) = 0;
fimg(169-aux:169+aux, 169-aux:169+aux) = 0;
imshow(log(abs(fimg)+1), []);
```

Resultado:



```
% 04 - Retornando no domínio espacial
nimg = ifft2(fimg);
imshow(abs(nimg), []);
```

Questão 02) - Filtro notch: São filtros capazes de rejeitar uma faixa bastante estreita de frequências. Sua utilização é recomendada quando o sinal a ser atenuado é bem definido. Pelo fato de atuarem em faixas reduzidas de frequências, filtros notch interferem pouco na qualidade do sinal. A figura a continuação mostra apenas um par de regiões sendo retirado.



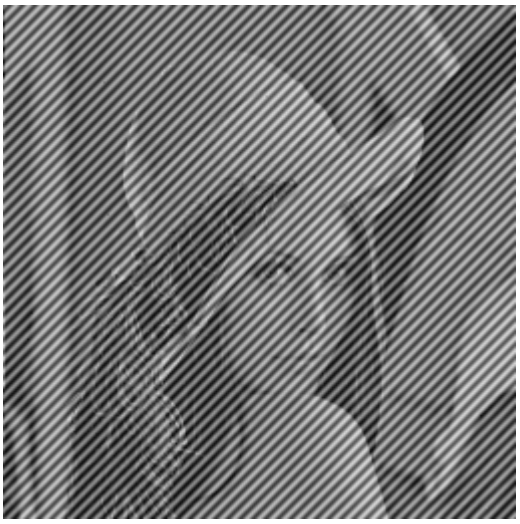
A área em torno da frequência de corte escolhida (D0) que pode ser retirada é definida na construção do filtro. Seja D0 a frequência de corte do filtro notch centrado em (u0, v0) e, por simetria (−u0, −v0):

$$D_1(u,v) = \sqrt{(u - M/2 - u_0)^2 + (v - N/2 - v_0)^2}$$

$$D_2(u,v) = \sqrt{(u - M/2 + u_0)^2 + (v - N/2 + v_0)^2}$$

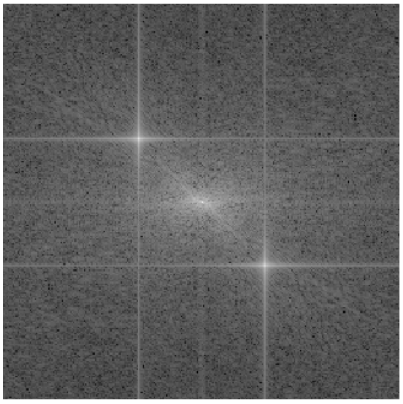
$$H(u,v) = \frac{1}{1 + \left(\frac{D_0^2}{D_1(u,v)D_2(u,v)}\right)^n}$$

Crie um filtro notch para remover a ruído periódico da seguinte imagem:



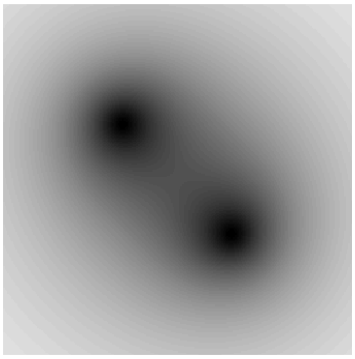
```
% 02 - Calculando a tranformada de Fourier.
fimg = fftshift(fft2(img));
imshow(log(abs(fimg)+1), []);
```

Resultado:

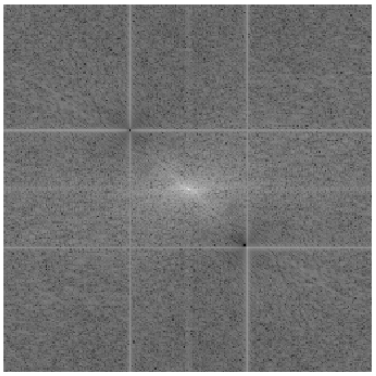


Construíndo a máscara

```
function mask = notch_filter(img, d0, n)
  [rows, cols] = size(img);
  nimg = zeros(rows, cols);

  uo = rows/2 - 88;
  vo = cols/2 - 88;

  for i = 1: rows
    for j = 1 : cols
      d1 = sqrt(((i - (rows/2) - uo)**2) + ((j - (cols/2) - vo)**2));
      d2 = sqrt(((i - (rows/2) + uo)**2) + ((j - (cols/2) + vo)**2));
      h = 1/power((1+(d0**2)/(d1*d2)),n);
      nimg(i,j) = h;
    endfor
  endfor

  mask = nimg;

% 03 - Calculando a máscara
mask = notch_filter(img, 50, 2);
imshow(mask);
```



```
% 04 - Aplicando a máscara
nfimg = fimg .* mask;
imshow(log(abs(nfimg)+1),[]);
```



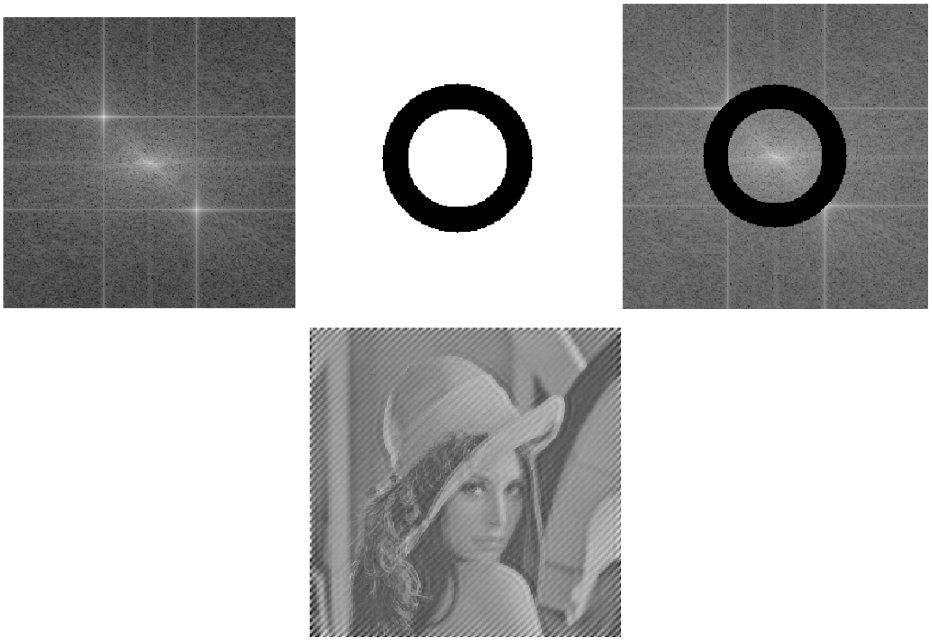
```
% 05 - Retornando no domínio espacial
nimg = ifft2(nfimg);
imshow(abs(nimg), []);
```

Questão 03) Repita o processo de remoção do ruído periódico da questão anterior utilizando os seguintes filtro passa-bandas: ideal, Butterworth e Gaussiano

Ideal

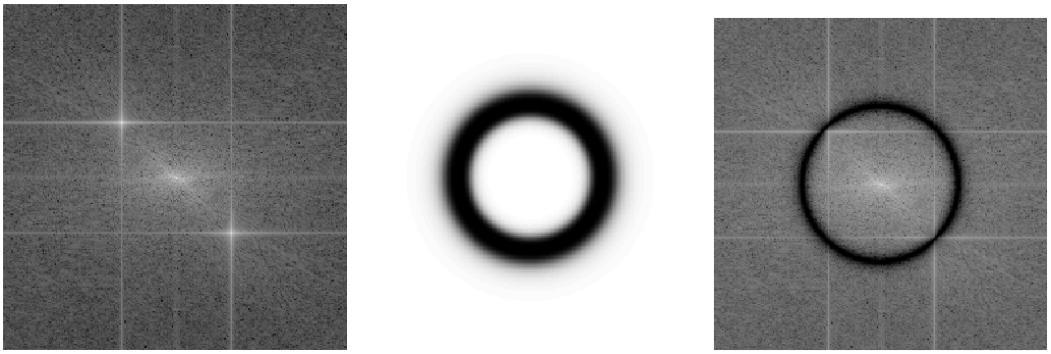
```
function mask = ideal_filter(img, w, d0)
  [rows, cols] = size(img);
  mask = zeros(rows, cols);

  for i = 1 : rows
    for j = 1 : cols
      d = sqrt((((i - rows/2))**2) + ((j - (cols/2))**2));
      v1 = d0 - (w/2);
      v2 = d0 + (w/2);

      if v1 > d || v2 < d
        mask(i,j) = 1;
      endif
    endfor
  endfor
```

Resultado:



Butterworth

```
function mask = butterworth_filter(img, w, d0, n)
   [rows, cols] = size(img);
   mask = zeros(rows, cols);

   for i = 1 : rows
     for j = 1 : cols
       d = sqrt((((i - rows/2))**2) + ((j - (cols/2))**2));
       h = 1/(1+ power((d*w/((d**2)-(d0**2))),2*n));
       mask(i,j) = h;
     endfor
   endfor
```



Gaussiano

```
function mask = gaussian_filter(img, w, d0, n)
  [rows, cols] = size(img);
  mask = zeros(rows, cols);

  for i = 1 : rows
    for j = 1 : cols
      d = sqrt((((i - rows/2))**2) + ((j - (cols/2))**2));
      h = 1 - power(e,-((((d**2) - (d0**2))/(d*w))**2));
      mask(i,j) = h;
    endfor
  endfor
```