3^{rd} tutorial in IVP

7 *March* 2024

Written by - Anurag Paul, 20EC01045.

```
% input image
i = imread('Subs/IVP/son.png');
imshow(i); title('\it{Original Image}', ...
    'interpreter', 'latex');
xlabel('\it The cameraman.', ...
    'interpreter', 'latex')
```

Original Image

The cameraman.

To grayscale the image.

```
g = rgb2gray(i); imshow(g);
title('\it{Greyed Image}','interpreter','latex');
```

Greyed Image

```
max(g, [], 'all')
```

```
ans = uint8

    255
```

```
size(g)
```

```
ans = 1×2
   121   236
```

Laplacian Filter.

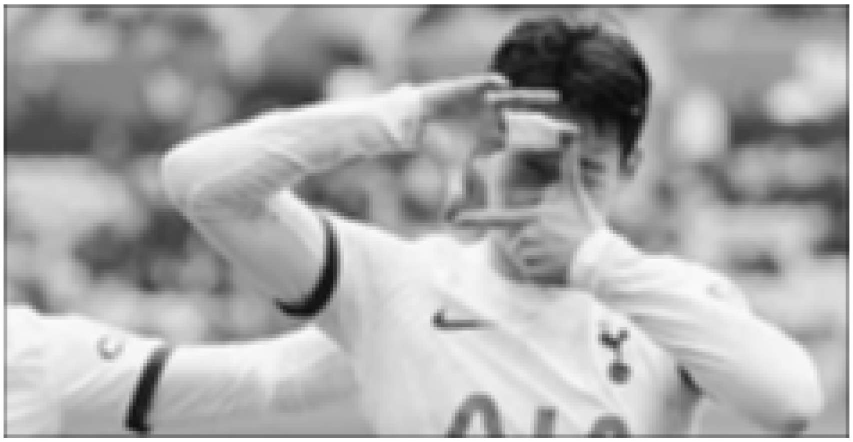
```
% Define the .
L=[0 -1 0;
  -1 4 -1;
   0 -1 0];
% fspecial('laplacian',0)
c=conv2(g, L, 'same'); imshow(c);
title('\it{Laplacianed Image}','interpreter', ...
    'latex');
```

Laplacianed Image

```
% r = imrotate(g,-45); imshow(r);
% Shrink Image By Factor of Two Using Default
```

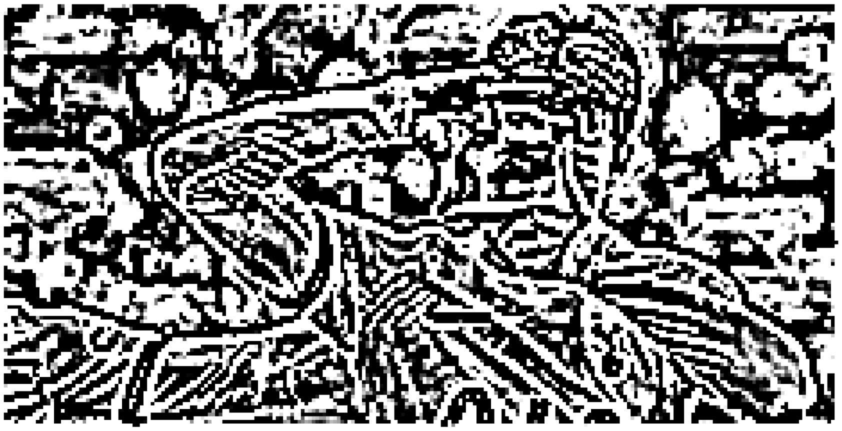
LoG

```
d = 3; G=fspecial('gaussian',d, 1);
gi=conv2(g, G, 'same');
imshow(uint8(gi)); s = num2str(d);
title(['$Gaussianed$ ',s,'$\times', s, ...
    ',$ $\sigma=1$'], 'interpreter', 'latex');
```



$Gaussianed\ 3{\times}3,\ \sigma = 1$

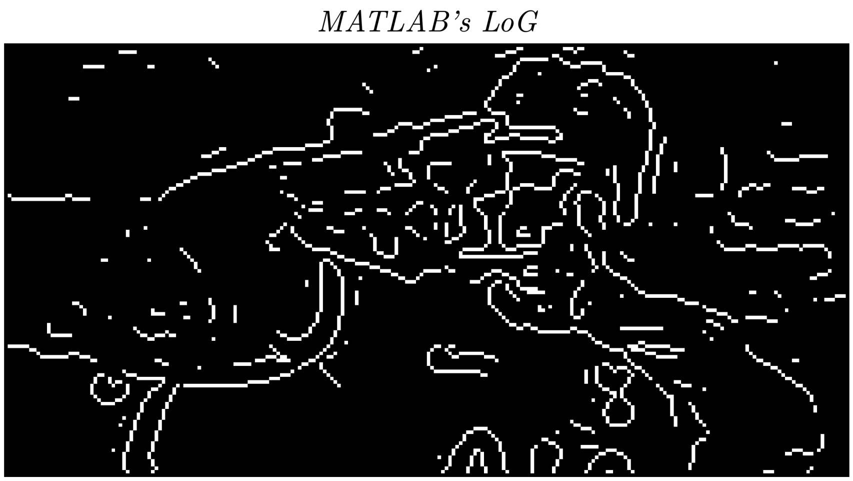
```
LoG=conv2(gi, L, 'same'); imshow(LoG);
title('$LoG$', 'interpreter', 'latex');
```



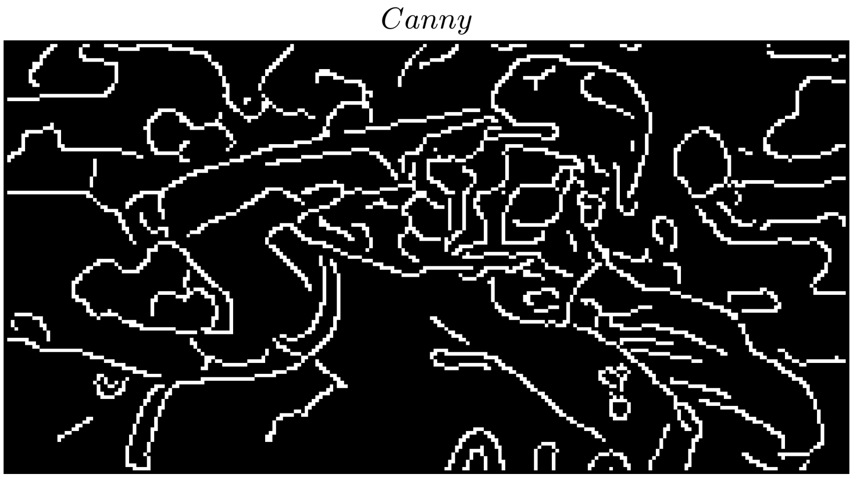
LoG

MATLAB's LoG

```
pL = edge(g,'log'); imshow(pL);
title('\it{MATLAB''s LoG}','interpreter','latex');
```



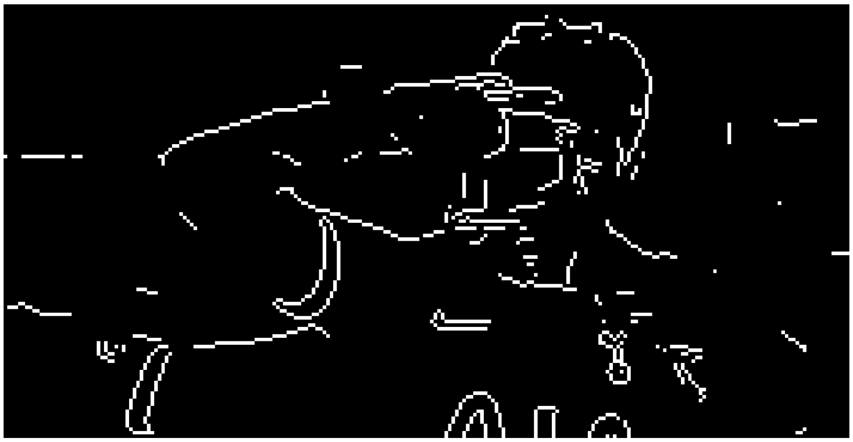
```
Ca = edge(g,'Canny'); imshow(Ca);
title('$Canny$','interpreter','latex');
```



Canny

```
% Prewitt method.
P = edge(g,'Prewitt'); imshow(P);
title('\it{Prewitt method}','interpreter','latex');
```

Prewitt method



```
% Display both results side-by-side.
% imshowpair(Ca,P,'montage')
% imshowpair(Ca,P)
```

Default Sobel gradient operator. vertical (*y*) direction, the weights are:

$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$, in the *x* direction, the weights are transposed.

```
% d = 20; G=fspecial('gaussian',d, 1);
% gi=conv2(g, G, 'same');
[Gx,Gy] = imgradientxy(gi); imshow(Gx); % g
title('$G_x(g)$','interpreter','latex');
```

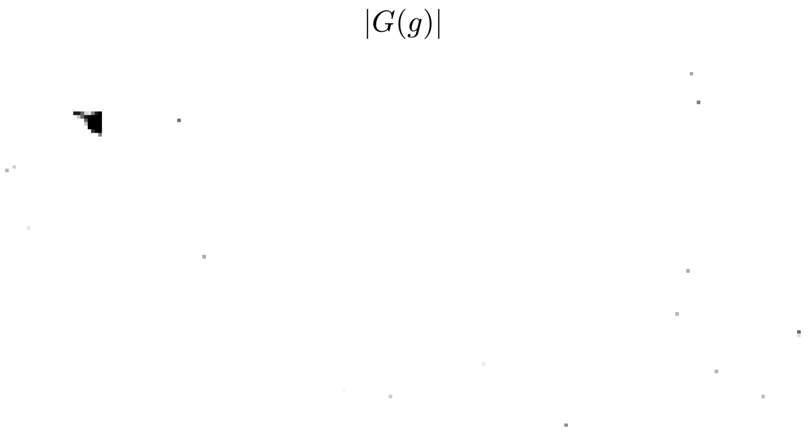
$G_x(g)$

```
imshow(Gy);
title('$G_y(g)$','interpreter','latex');
```

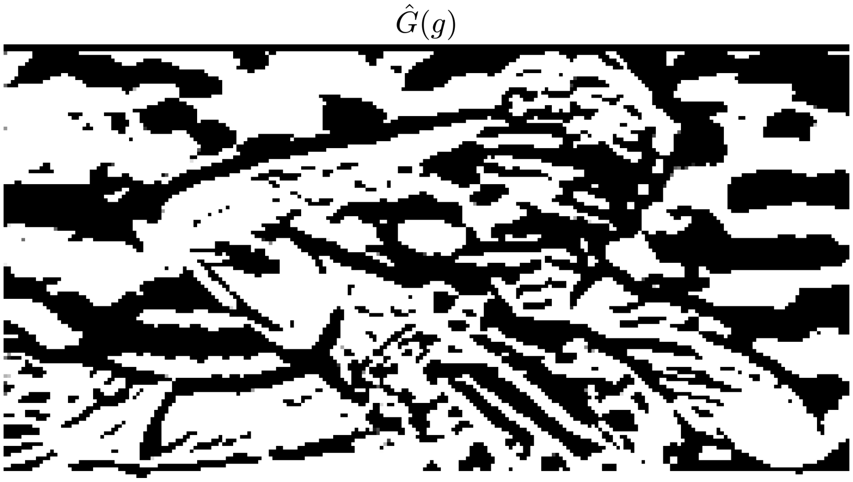
$G_y(g)$

Directions.

```
[Gm,d] = imgradient(Gx,Gy); imshow(Gm);
title('$|G(g)|$','interpreter','latex');
```



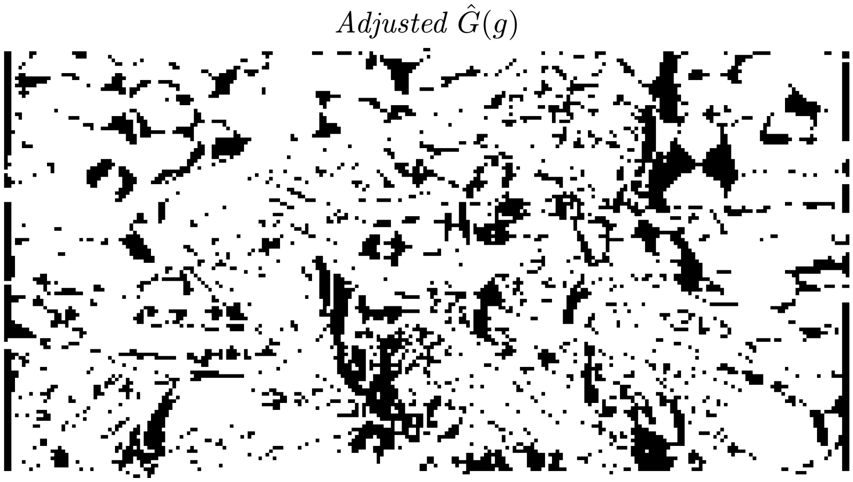
```
imshow(d); Gdir = d; Gmag = Gm;
title('$\hat G(g)$','interpreter','latex');
```



Adjusting directions to 0, 45, 90, or 135o

```
Gdir=Gdir.*~(-22.5<Gdir&Gdir<=22.5|Gdir>157.5|...
    Gdir<=-157.5);
te = (22.5<Gdir&Gdir<=67.5|-157.5<Gdir&Gdir<= ...
    -112.5); Gdir=Gdir.*~te+45*te;
te = (112.5>=Gdir&Gdir>67.5|-67.5>=Gdir&Gdir> ...
    -112.5); Gdir=Gdir.*~te+90*te;
te = (112.5<Gdir&Gdir<=157.5|-67.5<Gdir&Gdir<= ...
    -22.5); Gdir=Gdir.*~te+135*te;
```

```
imshow(Gdir);
title('\it Adjusted $\hat G(g)$','interpreter', ...
    'latex');
```

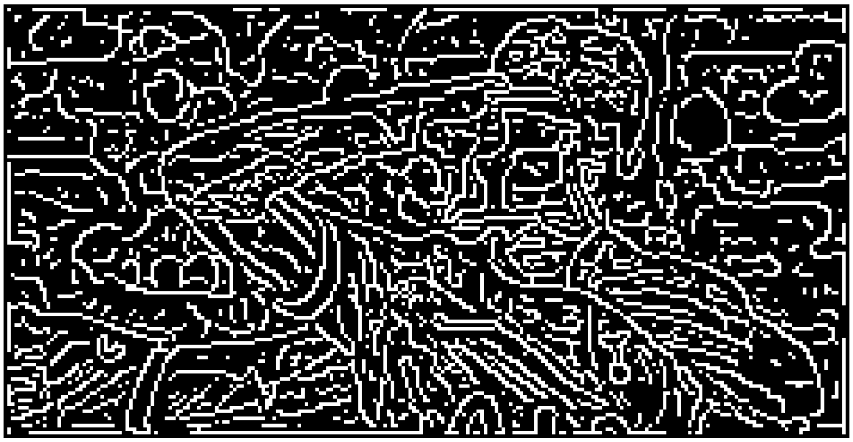


Non-Maximum Supression

```
% Gmag = Gm;
[r, c] = size(gi); ag = zeros(r, c);
for i=2:r-1
    for j=2:c-1
        if Gdir(i,j)==0
            ag(i,j) = Gmag(i,j)*(Gmag(i,j)>= ...
                Gmag(i,j+1)&Gmag(i,j)>= Gmag( ...
                i,j-1));
        elseif (Gdir(i,j)==45)
            ag(i,j) = Gmag(i,j)*(Gmag(i,j)>= ...
                Gmag(i+1,j-1)&Gmag(i,j)>= Gmag( ...
                i-1,j+1));
        elseif (Gdir(i,j)==90)
            ag(i,j) = Gmag(i,j)*(Gmag(i,j)>= ...
                Gmag(i+1,j)&Gmag(i,j)>= Gmag( ...
                i-1,j));
        elseif (Gdir(i,j)==135)
            ag(i,j) = Gmag(i,j)*(Gmag(i,j)>= ...
                Gmag(i+1,j+1)&Gmag(i,j)>= Gmag( ...
                i-1,j-1));
        end
    end
end
% n = Gmag;
```

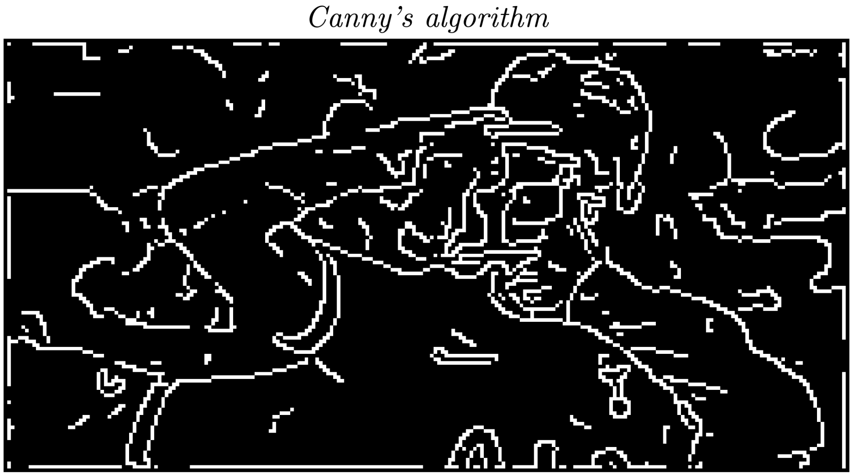
```
imshow(ag);
title('\it Non-Maxima Supressed |G(g)|', ...
    'interpreter', 'latex');
```

Non-Maxima Supressed —G(g)—



Hysteresis Thresholding

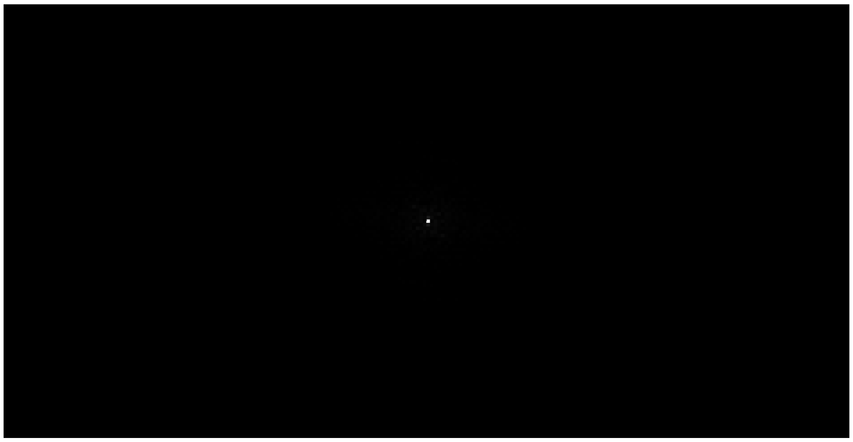
```
% Gmag = n;
m = max(ag(:));
l = 0.075*m; h = 0.175*m; f = 0;
for i = 1:r
    for j = 1:c
        if (ag(i, j) < l)
            ag(i, j) = 0;
        elseif 1
            f = 0;
            for k = -1*(i>1):1*(i<r) % 3 x 3
                for mk = -1*(j>1):1*(j<c)
                    if (ag(i+k, j+mk) > h)
                        ag(i, j) = 1; f = 1;
                        break
                    end
                end
                if f
                    break
                end
            end
        end
        if ~f
            ag(i, j) = 0;
        end
    end
end
imshow(uint8(ag.*255));
title('\it Canny''s algorithm' , ...
    'interpreter', 'latex');
```



Canny's algorithm

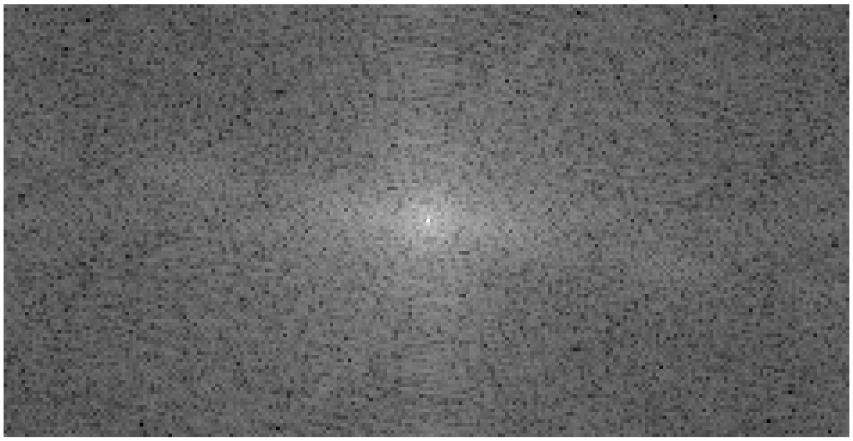
i

```
fuv=fft2(g);
imshow(abs(fftshift(fuv)), [])
title('\it Shifted spectrum' , ...
    'interpreter', 'latex');
```



Shifted spectrum

```
imshow(log(abs(fftshift(fuv))), [])
title('\it Shifted log spectrum' , ...
    'interpreter', 'latex');
```



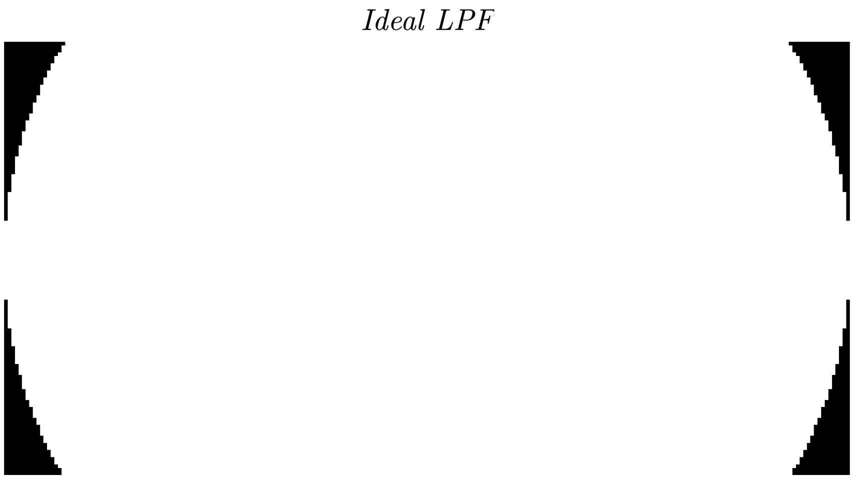
Shifted log spectrum

Designing filter

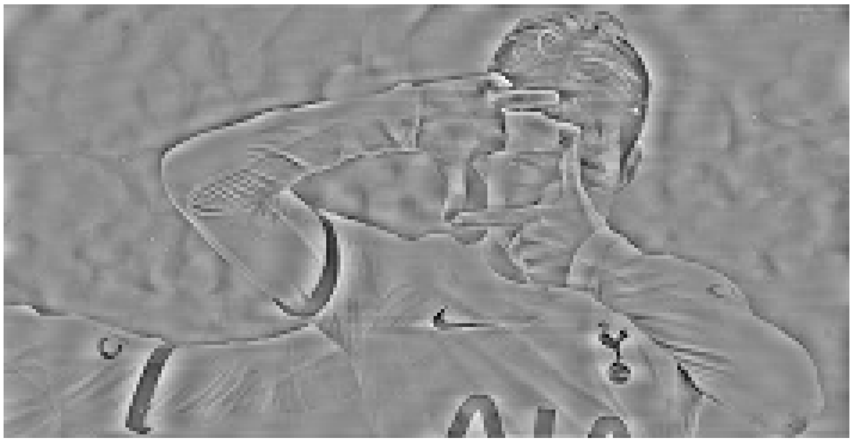
```
u = -r/2:r/2-1; v = -(c-1)/2:(c-1)/2;
```

Comparing with the cut-off frequency

```
[uu, vv] = meshgrid(u, v);
% Cut-off Frequency
H = double(sqrt(uu.^2+vv.^2) < max(r,c)/2); % 150
% H = double(sqrt(uu.^2+vv.^2) < min(r,c)/2); % 150
figure; imshow(H')
title('\it Ideal LPF' , ...
    'interpreter', 'latex');
```



```
% mask ,image
G = H'.*fuv;
% ifft2 (2D inverse fast fourier transform)
o = real(ifft2(G)); imshow(o, [ ]);
title('\it Result' , ...
    'interpreter', 'latex');
```

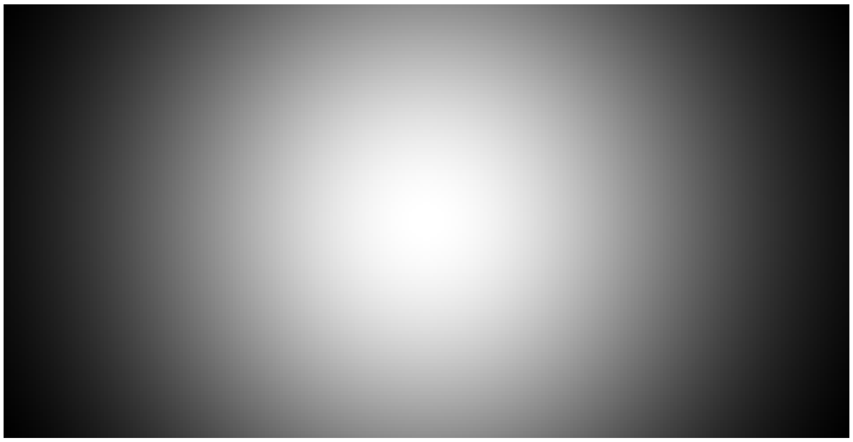


Result

mask ,image

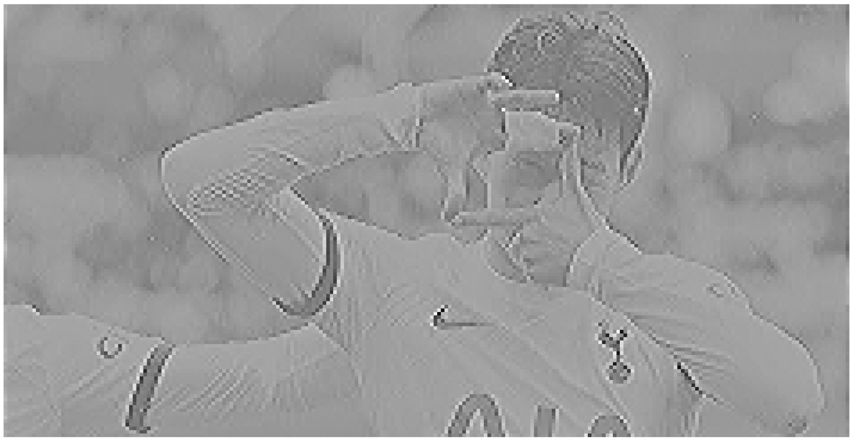
```
% m = fspecial('gaussian',[r c], 50); % min(r,c)/4
m = fspecial('gaussian',[r c], max(r,c)/4); % min
imshow(m, []);
title('\it Gaussian LPF' , ...
    'interpreter', 'latex'); G = m.*fuv;
```

Gaussian LPF



```
o = real(ifft2(G)); imshow(o, [ ]);
title('\it Result' , ...
    'interpreter', 'latex');
```

Result



The main cause of ringing artifacts is due to signal being bandlimited (not having high frequencies) when passed through a low-pass filter. Mathematically, this is called the Gibbs phenomenon.

One may distinguish overshoot (and undershoot), the output is higher than the input – from ringing.