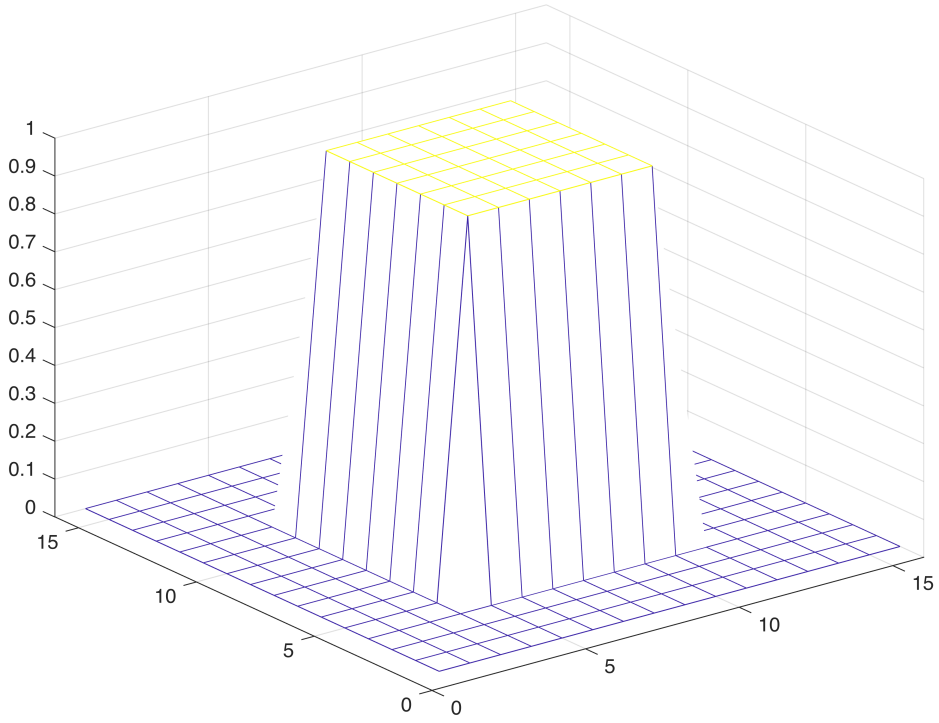
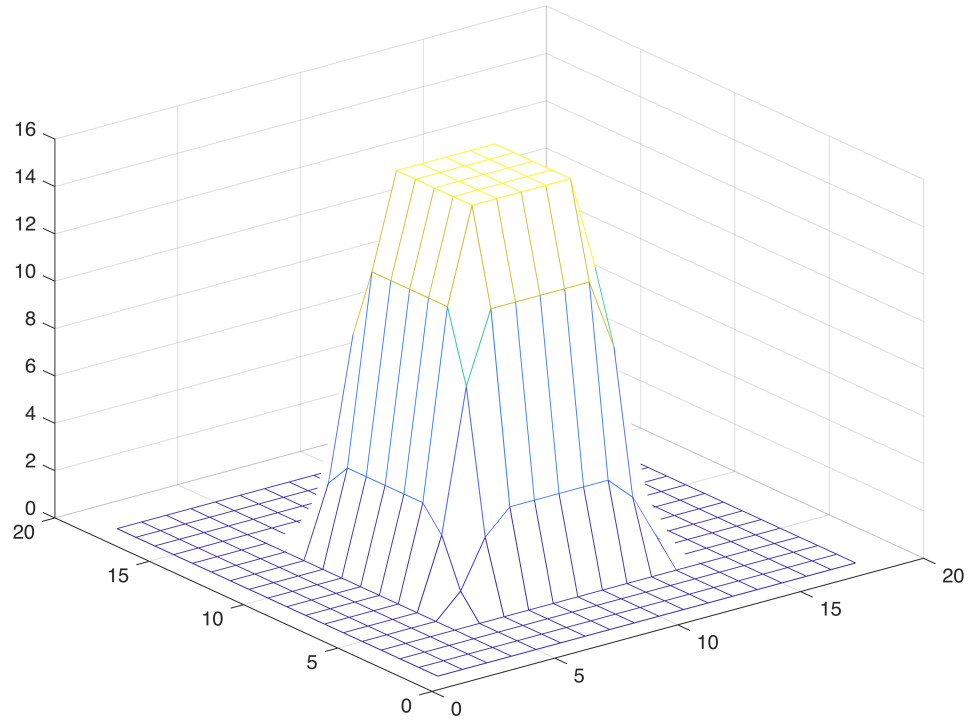
Step 1

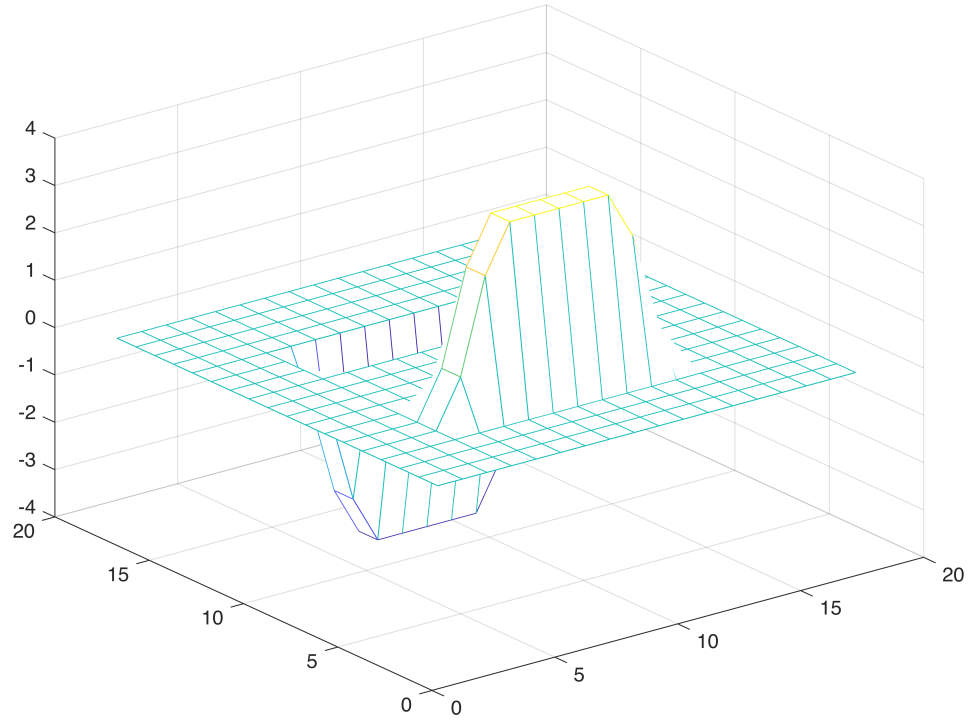
```
A1 = zeros(16);
A1(5:11,5:11) = ones(7);
mesh(A1)
```



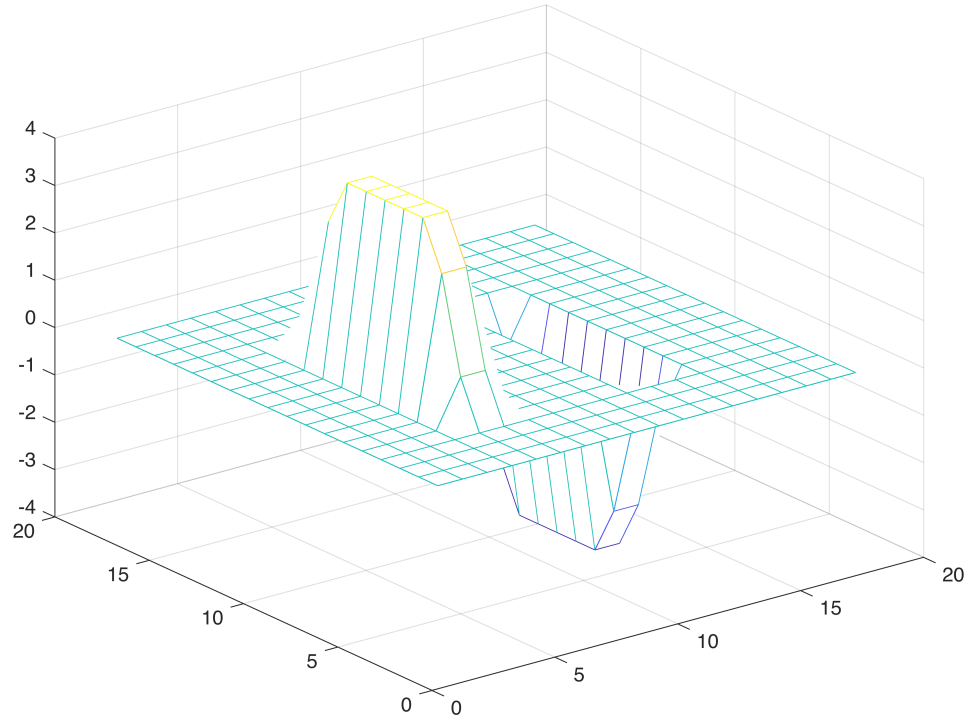

```
% the size of A1 is 16x16
h1 = [1 2 1];
h2 = [1 0 -1];
h3 = conv(h1, h1); % Convolution of h1 with itself

% Create two-dimensional filters using outer product
hA = h1' * h1;
hB = h2' * h1;
hC = h3' * h3;

Aout = conv2(hA,A1); % hA is the outer product of the vector h1 with itself
mesh(Aout);
```



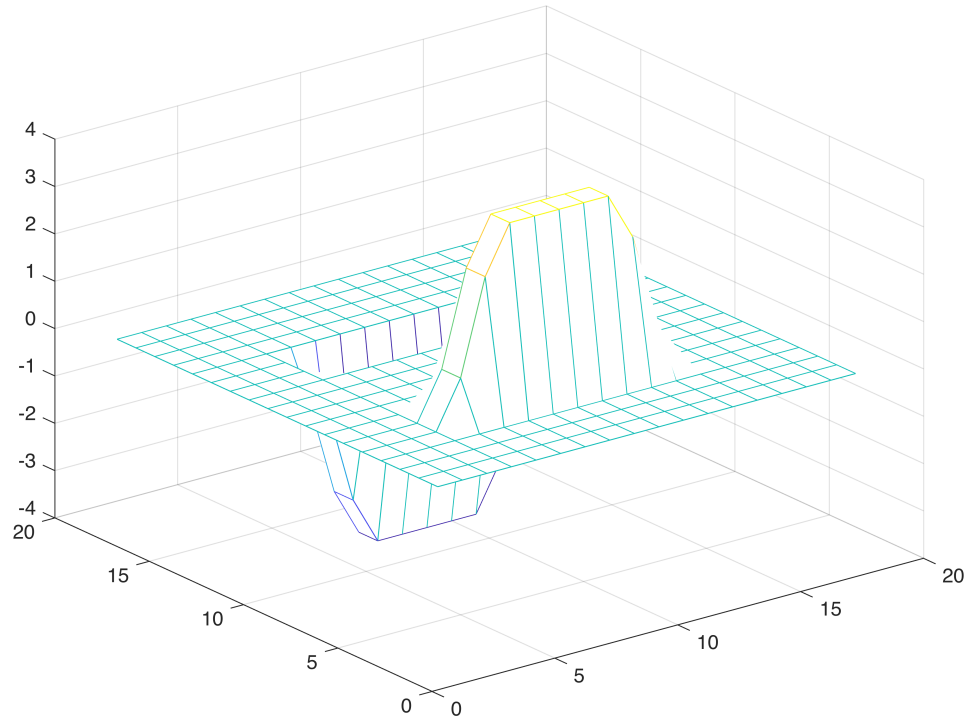

```
Aout = conv2(hB,A1); %hB highlights vertical edges
mesh(Aout);
```



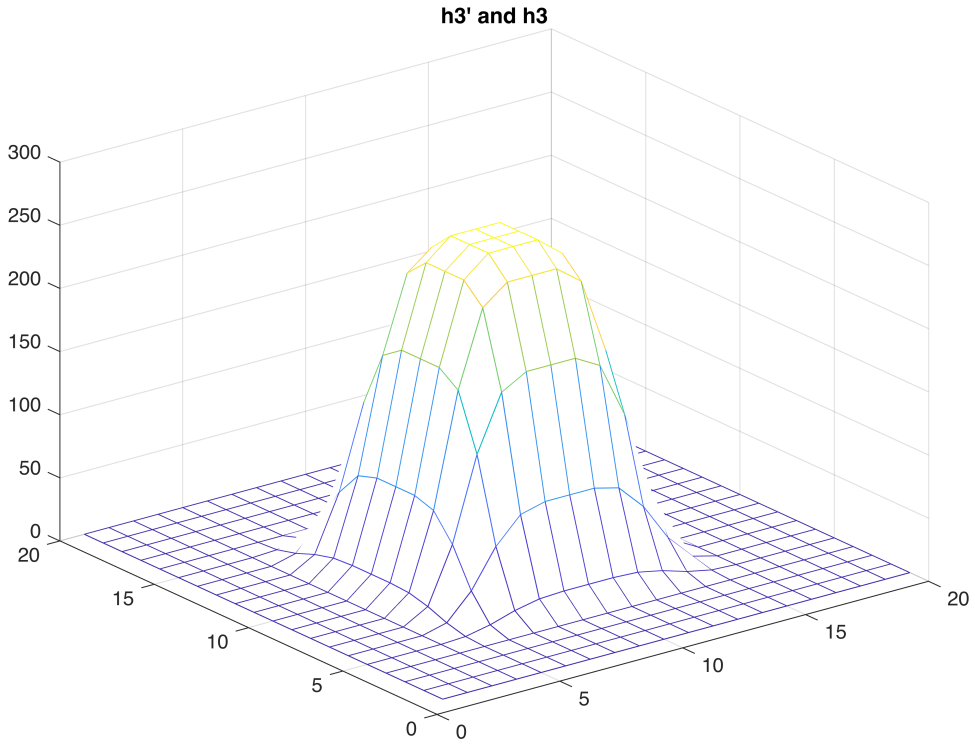
```
Aout = conv2(hB',A1); %inverse of hB
mesh(Aout);
```



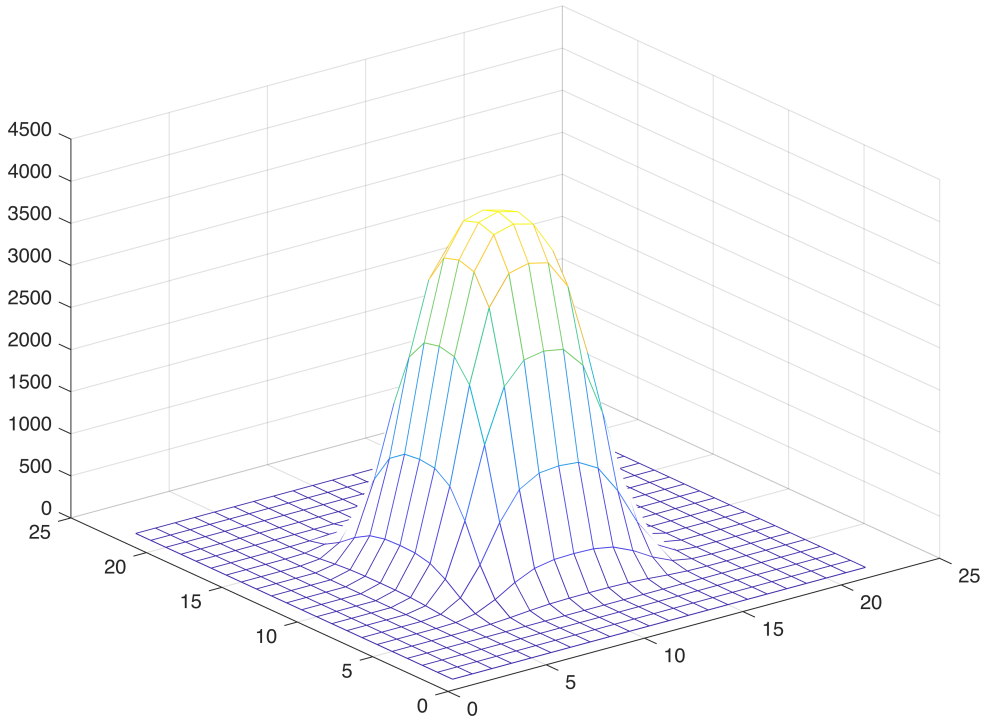

```
Bout = conv2(h2',h1,A1);
mesh(Bout); %same result as hB
```



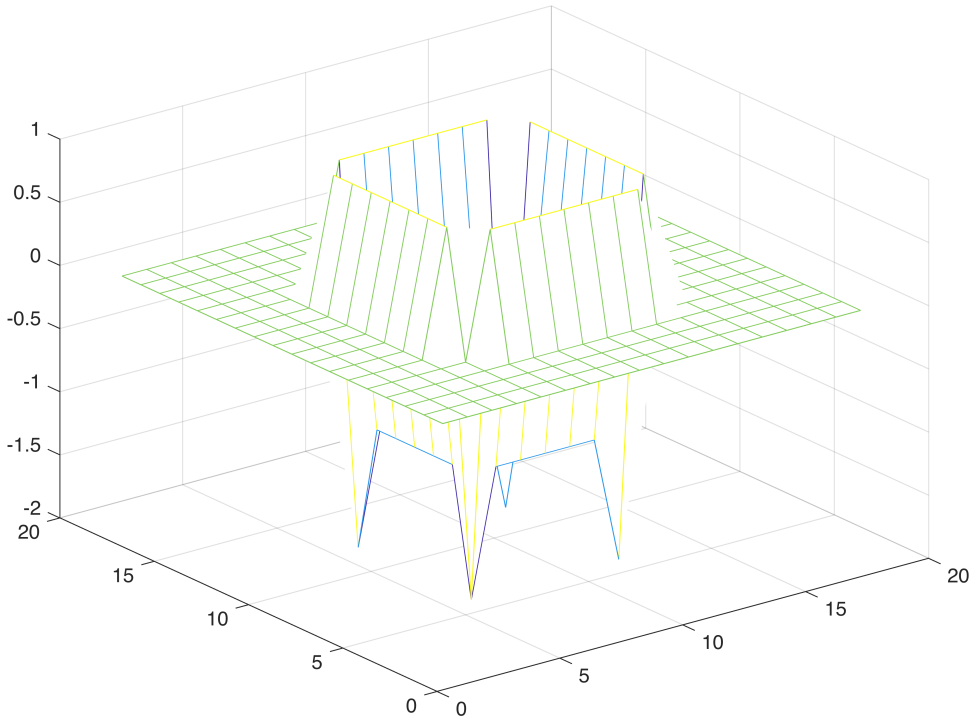

```
% output array's size is 18x18

Cout=conv2(h3',h3,A1);
mesh(Cout);
title("h3' and h3");
```




```
% h3, being convolved from h1 with itself, is likely a smoother and broader
version of a Gaussian-like filter
% with the peak being less sharp and the slopes more extended

h4 = conv(h1, h3, 'full');
Dout=conv2(h4',h4,A1);
mesh(Dout);
```




```
% comparing to h3' h3, the amplitude scale is 10 times greater

hL = fspecial('laplacian', 0.0);
% size is 3x3
% 0,1,0;
% 1,-4,1;
% 0,1,0;
Eout = conv2(hL,A1);
mesh(Eout);
```




```
% The Laplacian filter is generally more robust for edge detection because
% it emphasizes all abrupt changes
```

Step 2

```
i1 = imread('4.1.05.tiff');
imshow(i1);
```

```
i1grey = rgb2gray(i1);
imshow(i1grey);
```

```
s3 = h3/16;
filtered_image = conv2(s3, s3', double(i1grey), 'same'); % Convert i1grey
to double for processing
figure;
imshow(uint8(filtered_image));
```

```
% the filtered_image blured

hS = ones(2,40)/80;
smoothed_image = conv2(double(i1grey),hS, 'same'); % Convert to double for
processing
figure;
imshow(uint8(smoothed_image));  % Convert back to uint8 for display
title('Horizontally Smoothed Grayscale Image');
```

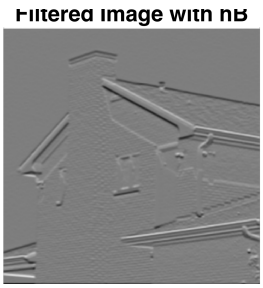

```
% a strong blurring effect horizontally while maintaining relatively more
detail vertically.
```

```
hM = fspecial('motion', 41, 30);
% size of hM is 21x37
motion=conv2(double(i1grey),hM,'same');
imshow(uint8(motion));
```

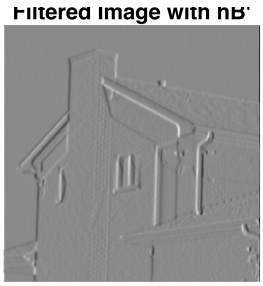
```
title('motion');
% this filter blur more comparing to previous one. This has
% a verticle  blurring effect.
```

Step 3

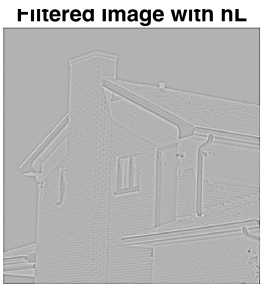
```
filter_hB=conv2(i1grey,hB,'same');
filter_hBt =conv2(i1grey,hB','same');
figure;
imshow(filter_hB, []); % Display image, automatically scaling the color
range
title('Filtered Image with hB');
```




```
figure;
imshow(filter_hBt, []); % Display image, automatically scaling the color
range
title('Filtered Image with hB''');
```




```
% hB' emphasizes vertical edges, hB emphasizes horizontal edges
% it's consistent

filter_hL=conv2(i1grey,hL,'same');
figure;
imshow(filter_hL, []); % Display image, automatically scaling the color
range
title('Filtered Image with hL');
```

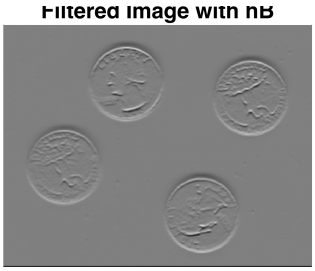



```
edge1 = edge(i1grey);
figure;
imshow(edge1,[]);
```

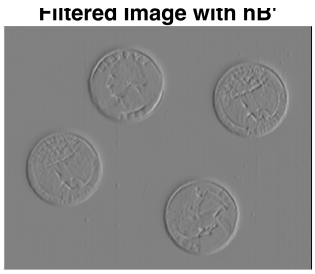
```
% The edge function combines the effects of both hB and hB' in essence. It
considers both vertical and horizontal gradients to detect all significant
edges.
```

repeat with a new image

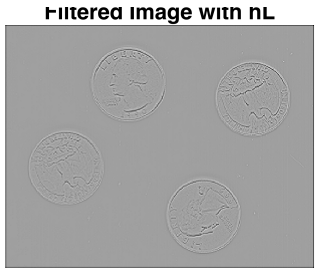
```
i2 = imread('eight.tif');
filter_hB=conv2(i2,hB,'same');
filter_hBt =conv2(i2,hB','same');
figure;
imshow(filter_hB, []); % Display image, automatically scaling the color
range
title('Filtered Image with hB');
```



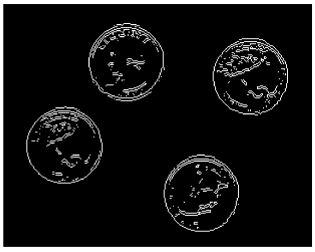

```


figure;
imshow(filter_hBt, []); % Display image, automatically scaling the color
range
title('Filtered Image with hB''');
```




```
% hB' emphasizes vertical edges, hB emphasizes horizontal edges
% it's consistent

filter_hL=conv2(i2,hL,'same');
figure;
imshow(filter_hL, []); % Display image, automatically scaling the color
range
title('Filtered Image with hL');
```

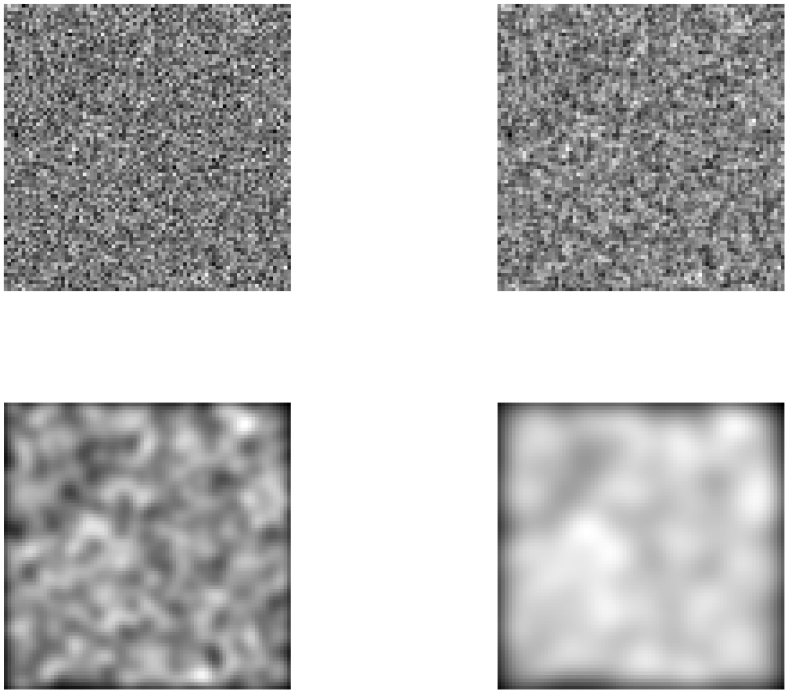



```
edge1 = edge(i2);
figure;
imshow(edge1,[]);
```



Step 4

```
inoise = 128 + 128*randn(80);
hG1 = fspecial('gaussian', 5, 0.5);
hG2 = fspecial('gaussian', 15, 2.0);
hG3 = fspecial('gaussian', 25, 4.0);
filtered_n11 = conv2(inoise, hG1, 'same');
filtered_n22 = conv2(inoise, hG2, 'same');
filtered_n33 = conv2(inoise, hG3, 'same');
figure;
subplot(2,2,1);
imshow(inoise,[]);
subplot(2,2,2);
imshow(filtered_n11,[]);
subplot(2,2,3);
imshow(filtered_n22,[]);
subplot(2,2,4);
imshow(filtered_n33,[]);
```



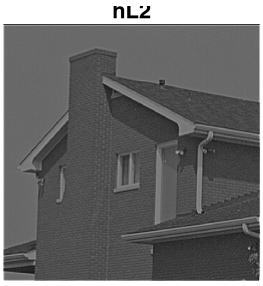
```
% As the size of the Gaussian filter and the standard deviation increase,
% The smoothing effect becomes more pronounced, reducing noise more
effectively.
```

For noise level V=0

```
V=0;
i1noise = double(i1grey) + V*randn(size(i1grey));
filtered_n1 = conv2(i1noise, hG1, 'same');
filtered_n2 = conv2(i1noise, hG2, 'same');
filtered_n3 = conv2(i1noise, hG3, 'same');
subplot(2,2,1);
imshow(i1noise,[]);
subplot(2,2,2);
imshow(filtered_n1,[]);
subplot(2,2,3);
imshow(filtered_n2,[]);
subplot(2,2,4);
imshow(filtered_n3,[]);
```




```
hL2 = [0, -1, 0; -1, 5, -1; 0, -1, 0];
filter_hL2=conv2(double(i1noise),hL2,'same');
figure;
imshow(filter_hL2,[]);
title('hL2');
```



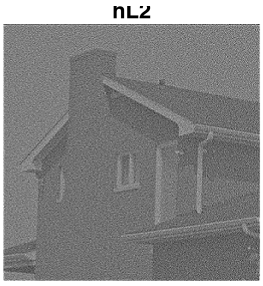

```
% when V=0, orginal and hG1 has the clearest images. As the size of
% Gaussian filter increase = blur increase

% The edges and corners in the sharpened output are define well
```

V=20

```
V=20;
i1noise = double(i1grey) + V*randn(size(i1grey));
filtered_n1 = conv2(i1noise, hG1, 'same');
filtered_n2 = conv2(i1noise, hG2, 'same');
filtered_n3 = conv2(i1noise, hG3, 'same');
subplot(2,2,1);
imshow(i1noise,[]);
subplot(2,2,2);
imshow(filtered_n1,[]);
subplot(2,2,3);
imshow(filtered_n2,[]);
subplot(2,2,4);
imshow(filtered_n3,[]);
```




```
hL2 = [0, -1, 0; -1, 5, -1; 0, -1, 0];
filter_hL2=conv2(double(i1noise),hL2,'same');
figure;
imshow(filter_hL2,[]);
title('hL2');
```

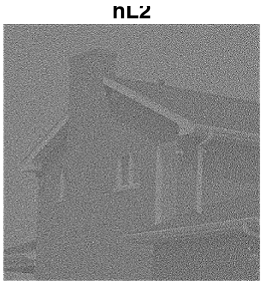


V=40

```
V=40;
i1noise = double(i1grey) + V*randn(size(i1grey));
filtered_n1 = conv2(i1noise, hG1, 'same');
filtered_n2 = conv2(i1noise, hG2, 'same');
filtered_n3 = conv2(i1noise, hG3, 'same');
subplot(2,2,1);
imshow(i1noise,[]);
subplot(2,2,2);
imshow(filtered_n1,[]);
subplot(2,2,3);
imshow(filtered_n2,[]);
subplot(2,2,4);
imshow(filtered_n3,[]);
```




```
hL2 = [0, -1, 0; -1, 5, -1; 0, -1, 0];
filter_hL2=conv2(double(i1noise),hL2,'same');
figure;
imshow(filter_hL2,[]);
title('hL2');
```




```
% For a moderate noise level like V = 20, a filter like hG2 (Gaussian,
15x15, σ = 2.0) is often the best choice. This filter provides a good
balance between reducing noise and maintaining image sharpness
%For a high noise level like V = 40, a larger and more robust filter such
as hG3 (Gaussian, 25x25, σ = 4.0) might be necessary.
% If an image contains noise, especially high-frequency noise like Gaussian
noise, sharpening can make this noise more pronounced
```