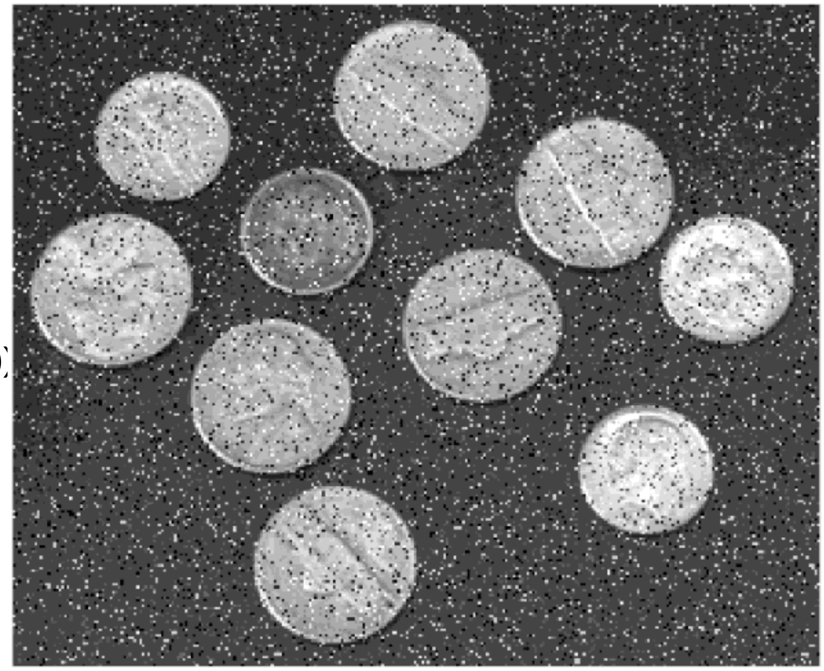
Lecture 18

Simulation of BSC on Matlab (cont'd)

```
r = 0.05;
I = imread('coins.png');
DIM = size(I);
noise_empty = uint8(zeros(size(I)));
X = rand(DIM(1), DIM(2), 8);
for i = 1: 8
    noise = bitset(noise_empty, i, (X(:,:,i) < r))
    noise_empty = noise;
end
J = bitxor(I, noise);
figure; imshow(J)
```



$\text{bitxor}(b, 1) = \bar{b}$: bit flipped

(↑ bit)

$\text{bitxor}(b, 0) = b$: No flipping

A XOR B =

A \ B	0	1
0	0	1
1	1	0

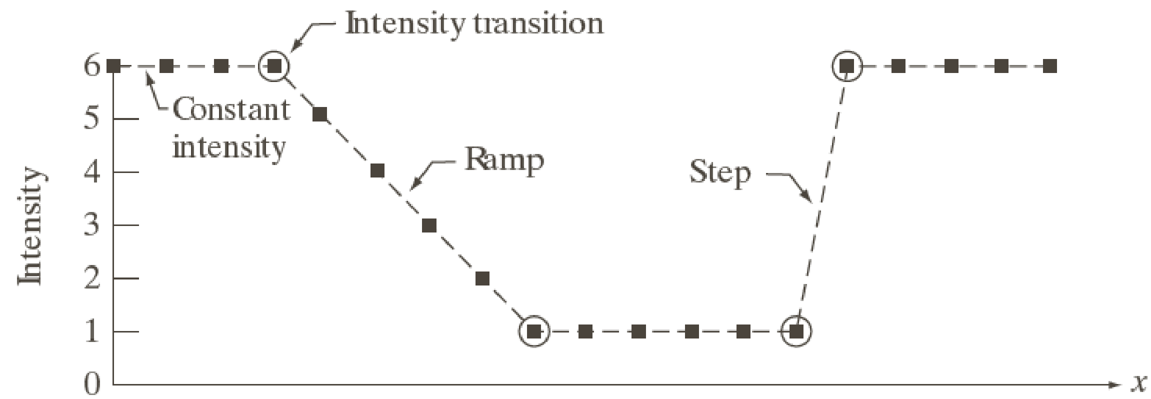
$A \text{ XOR } 0 = A$

$A \text{ XOR } 1 = \bar{A}$

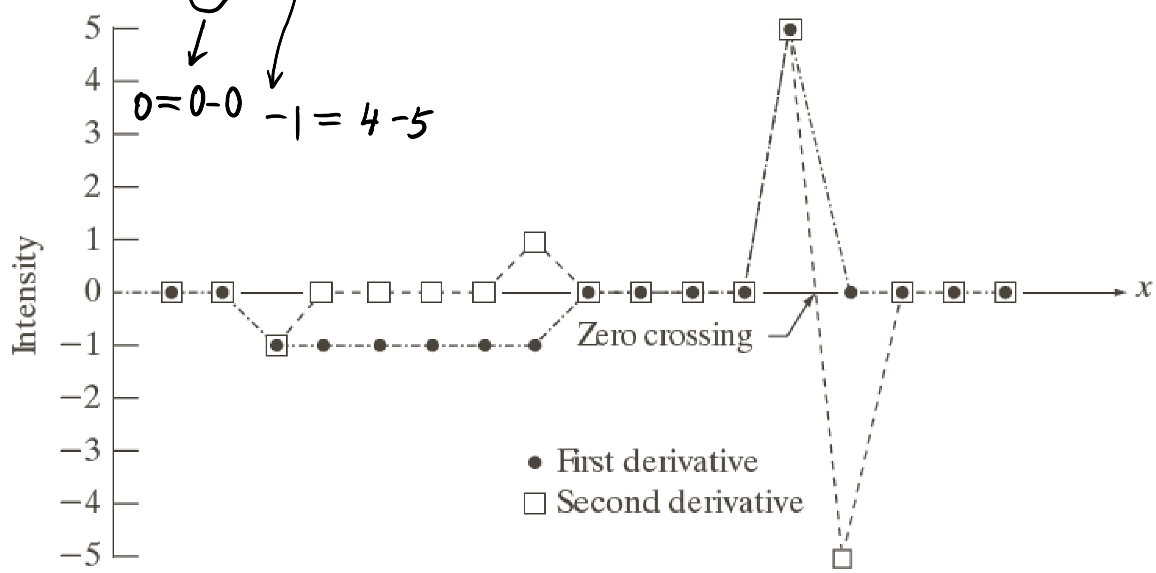
- Sharpening Spatial Filters
  - First-order derivative of a one-dimensional function is: $\frac{\partial f}{\partial x} = f(x+1) - f(x)$
  - Second-order derivative of the function is: $$\frac{\partial^2 f}{\partial^2 x} = f(x+1) + f(x-1) - 2f(x)$$

$$\frac{\partial^2 f}{\partial^2 x} = \underbrace{[f(x+1) - f(x)]}_{\text{1st-order derivative of the current pixel}} - \underbrace{[f(x) - f(x-1)]}_{\text{1st-order derivative of the previous pixel}}$$

$f(x-1)$ $f(x)$ $f(x+1)$



Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6	→ x
1st derivative		0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0		
2nd derivative		0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0		



Therefore, the 2nd order derivative enhances fine details much better than the1st order derivative -- a property that is ideally suited for sharpening images.

The Laplacian

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

$$\nabla^2 f(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

0	1	0
1	−4	1
0	1	0

Template for Laplacian filter

Image Sharpening Using Laplacian

- Background features can be "recovered" while still preserving the sharpening effect of the Laplacian simply by adding the Laplacian image to the original.
- $g(x, y) = f(x, y) - [\nabla^2 f(x, y)]$, if the center coefficient is negative.
- $g(x, y) = f(x, y) + [\nabla^2 f(x, y)]$, if the center coefficient is positive.

0	−1	0
−1	4	−1
0	−1	0

fspecial

Create predefined 2-D filter

h = fspecial('laplacian',alpha) returns a 3-by-3 filter approximating the shape of the two-dimensional Laplacian operator, alpha controls the shape of the Laplacian.

alpha — Shape of the Laplacian
0.2 (default) | number in the range [0, 1]

>> edit fspecial

```
case 'laplacian' % Laplacian filter
    alpha = p2;
    alpha = max(0, min(alpha,1));
    h1   = alpha/(alpha+1); h2 = (1-alpha)/(alpha+1);
    h    = [h1 h2 h1;h2 -4/(alpha+1) h2;h1 h2 h1];
```

$$h_1 = \frac{\alpha}{\alpha+1}, \quad h_2 = \frac{1-\alpha}{\alpha+1}$$

$$h = \begin{bmatrix} h_1 & h_2 & h_1 \\ h_2 & \frac{-4}{\alpha+1} & h_2 \\ h_1 & h_2 & h_1 \end{bmatrix} \qquad \sum h = 4h_1 + 4h_2 - \frac{4}{\alpha+1} = \frac{4\alpha}{\alpha+1} + \frac{4(1-\alpha)}{\alpha+1} - \frac{4}{\alpha+1} = 0$$

If $\alpha = 0$, then $h_1 = 0$, $h_2 = 1$, $h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$

```
>> h = fspecial('laplacian', 0)
h =
     0     1     0
     1    -4     1
     0     1     0
```

If $\alpha = 1$, then $h_1 = \frac{1}{2}$, $h_2 = 0$, $h = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & -2 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix}$

```
>> h = fspecial('laplacian')
h =
   0.1667    0.6667    0.1667
   0.6667   -3.3333    0.6667
   0.1667    0.6667    0.1667
```

default $\alpha = 0.2$

$$h_1 = \frac{0.2}{0.2+1} = \frac{1}{6}, \quad h_2 = \frac{1-0.2}{0.2+1} = \frac{0.8}{1.2} = \frac{2}{3}$$

$$\frac{-4}{\alpha+1} = \frac{-4}{0.2+1} = \frac{-4}{1.2} = -\frac{1}{0.3} = -\frac{10}{3}$$