Geometric Transformations

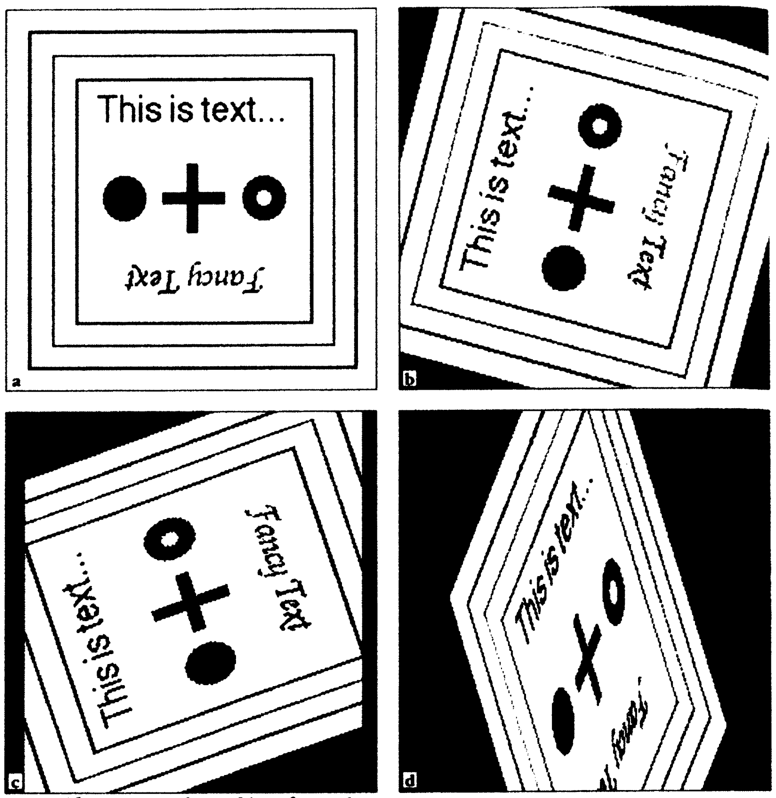
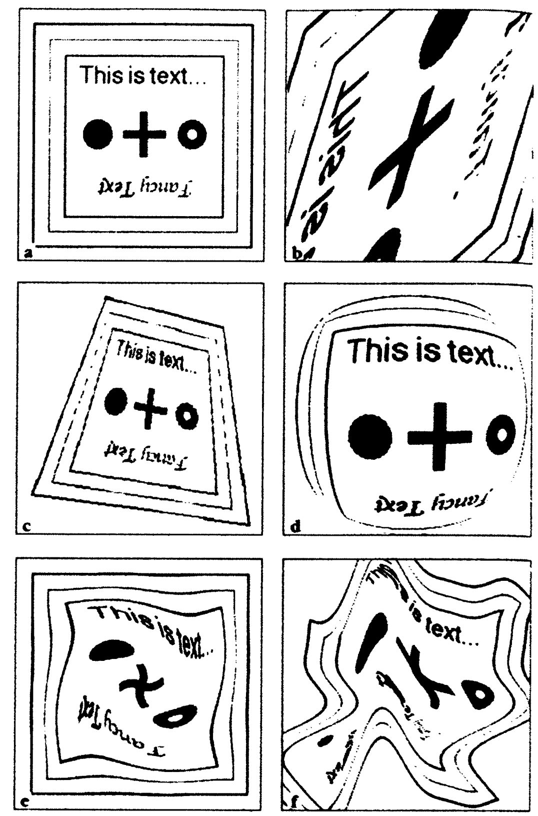


Figure 46. Rotation and stretching of a test image:
a) *original;*
b) *rotation only, no change in scale;*
c) *rotation and uniform stretching while maintaining angles;*
d) *general rotation and stretching.*

Figure 49. Some additional examples of image warping:
a) *original test image;*
b) *linear warping with reversal;*
c) *quadratic warping showing trapezoidal foreshortening (no interpolation);*
d) *cubic warping in which lines are curved (approximation here is to a spherical surface);*
e) *twisting the center of the field while holding the edges fixed (also cubic warping);*
f) *arbitrary warping in which higher order and trigonometric terms are required.*



1

Geometric transformations
Castleman,

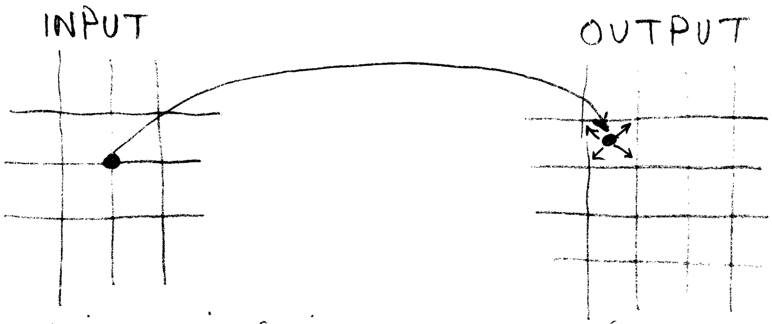
The general form of the spatial transformation is

$$g(x,y) = f(x',y') = f\left[a(x,y),\ b(x,y)\right]$$

This maps an original image $f(x,y)$ to a new image distribution $g(x,y)$ usually using polynomials

Gray-level interpolation will usually be necessary because the transformation of pixels tends to stretch and/or compress pixels.
Basically, you are mapping pixels on integer coordinates to fractional (non-integer coordinates).

forward mapping (pixel carry over)

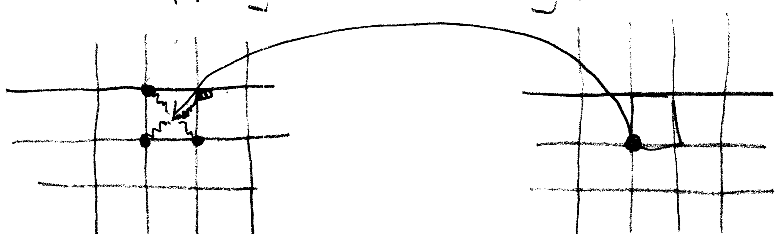


allocate gray scale value to pixels around transformed coordinates

start in original image (x',y') (x,y) transformed

problems
1. pixels mapping to locations outside image
2. multiple addressing of output pixels
3. missing output pixels ☆

backward mapping (pixel filling)



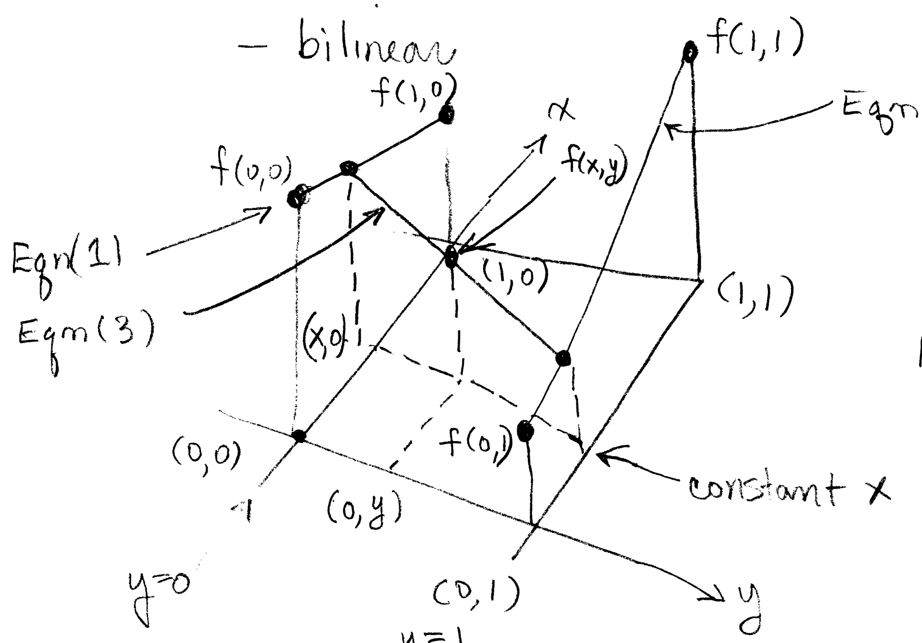
original (x',y') transformed (x,y)

types of interpolation

gray level interpolation

- nearest neighbor
  - assign gray level of output as that of nearest pixel in input image
  - can produce edge artifacts when gray levels change rapidly

- bilinear



Note: x, y are fractional coordinates

Can't fit plane through four points!
so fit hyperbolic paraboloid $f(x,y) = ax + by + cxy + d$

1) linearly interpolate along x-axis at $y=0$

$$f(x,0) = f(0,0) + x[f(1,0) - f(0,0)]$$

2) linearly interpolate along x-axis at $y=1$

$$f(x,1) = f(0,1) + x[f(1,1) - f(1,0)]$$

3) interpolate along y-axis at x (constant x)

$$f(x,y) = f(x,0) + y[f(x,1) - f(x,0)]$$

Combine all three equations to get the hyperbolic paraboloid

$$f(x,y) = [f(1,0) - f(0,0)]x + [f(0,1) - f(0,0)]y + [f(1,1) + f(0,0) - f(0,1) - f(1,0)]xy + f(0,0)$$

8 additions + 4 multiplications

Note: surfaces produced by bilinear interpolation are continuous in amplitude at boundaries of neighborhood but not in slope.

The slope discontinuities may cause undesirable effects
Can use other higher-order functions as well.

- cubic splines
- Legendre functions
- $\frac{\sin(\alpha x)}{\alpha x}$

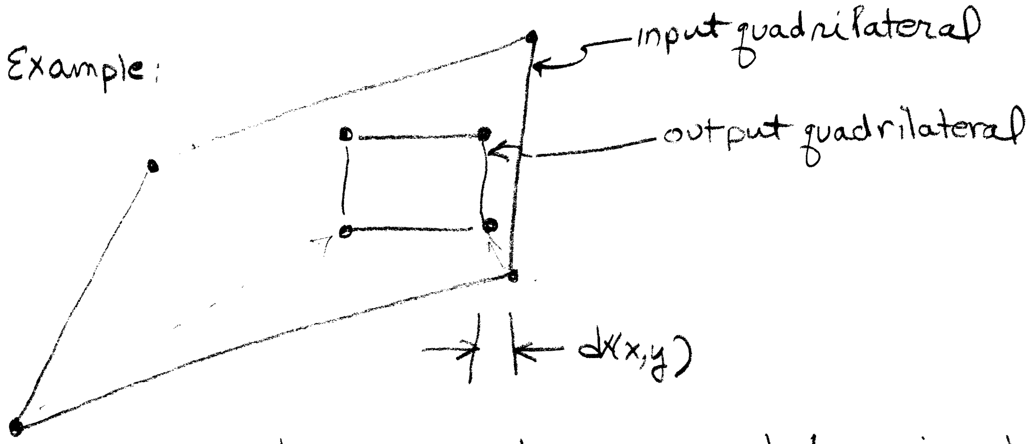
Image warping

Relate one image to another via mathematical transformations
Previously we considered gray scale.

$$G(x,y) = F\left[x + d(x,y),\ y + dy(x,y)\right]$$

restrict oneself to pixel displacements that are bilinear functions of x & y

Example:



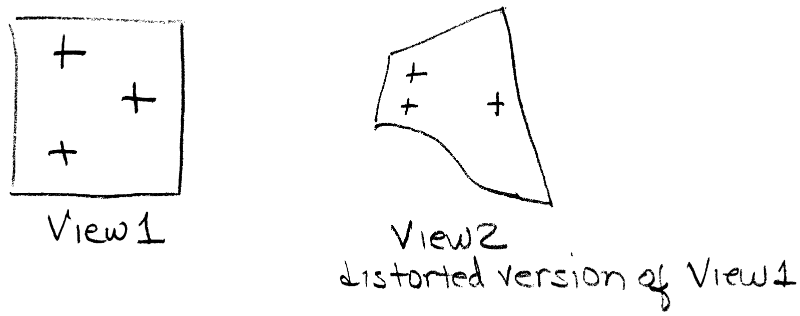
$dx(x,y)$ and $dy(x,y)$ are bilinear in x & y
$\Rightarrow$ linear in x <u>along</u> each horizontal line in output

for each output line $dx(x+1,y) = dx(x,y) + \Delta x$
where Δx varies for each line

image warping (in general)

- specify the spatial transform between two images as a series of displacement values for selected control points. in the image
- displacement of non-control points determined by interpolation

Geometric correction and registration – polynomial warping



Given two images of the same scene taken by sensors with different or time-dependent orientations, correct one of the images to the viewpoint of the other

(lots of industrial and tactical applications using geometrically distributed sensors, satellites with varying viewing angles, etc.)

Polynomial warp model

- mathematical model of the distortion (transformation)
- start with a set of points (called control points) whose location is known in both images

$\underline{x}$ is the vector location of the points in the undistorted image
$\underline{w}$ is the vector location of the points in the distorted image

A common application is to combine IR and visible images
image 1 $f_1(\underline{x})$ x_1, x_2 coordinates
image 2 $f_2(\underline{w})$ w_1, w_2 coordinates

Write
$$\left.\begin{aligned} x_1 &= g_1(w_1, w_2) \\ x_2 &= g_2(w_1, w_2) \end{aligned}\right\}$$
i.e., what is the transform from $\underline{w}$ to $\underline{x}$ for each coordinate

Approximate by N^{th} order 2-D polynomials

$$x_1 = \sum_{i=0}^{N} \sum_{j=0}^{N} c_{ij}^{(1)} w_1^i w_2^j$$ for the first variable

$$x_2 = \sum_{i=0}^{N} \sum_{j=0}^{N} c_{ij}^{(2)} w_1^i w_2^j$$ for the second variable

Given a set of M corresponding control points in each coordinate system

$$(X_{1k},\ X_{2k},\ W_{1k},\ W_{2k}) \qquad k = 1, 2, \ldots, M$$

only the k's need to be found

For N=2 (2nd order warp in each variable) we can write the following equation for the k-th control points

$$X_{1k} = c_{00}^{(1)} + c_{10}^{(1)} W_{1k} + c_{01}^{(1)} W_{2k} + c_{11}^{(1)} W_{1k} W_{2k} + c_{20}^{(1)} (W_{1k})^2$$

$$+ c_{02}^{(1)} (W_{2k})^2 + c_{21}^{(1)} (W_{1k})^2 W_{2k} + c_{12}^{(1)} W_{1k} (W_{2k})^2$$

$$+ c_{22}^{(1)} (W_{1k})^2 (W_{2k})^2$$

I can also write

$$X_{2k} = c_{00}^{(2)} + c_{10}^{(2)} W_{1k} + c_{01}^{(2)} W_{2k} + c_{11}^{(2)} W_{1k} W_{2k} + \cdots + c_{22}^{(2)} (W_{1k})^2 (W_{2k})^2$$

The X's and W's come from the known corresponding control points.
The C's are the unknown constants we want to find.

For N=2 there are 9 unknown coefficients in each equation for a total of 18 unknown coefficients

This means we need at least 9 control points

Notes

1. N-th order warp needs at least $2(N+1)^2$ control points

Just FYI:

For an exact number of needed control points.

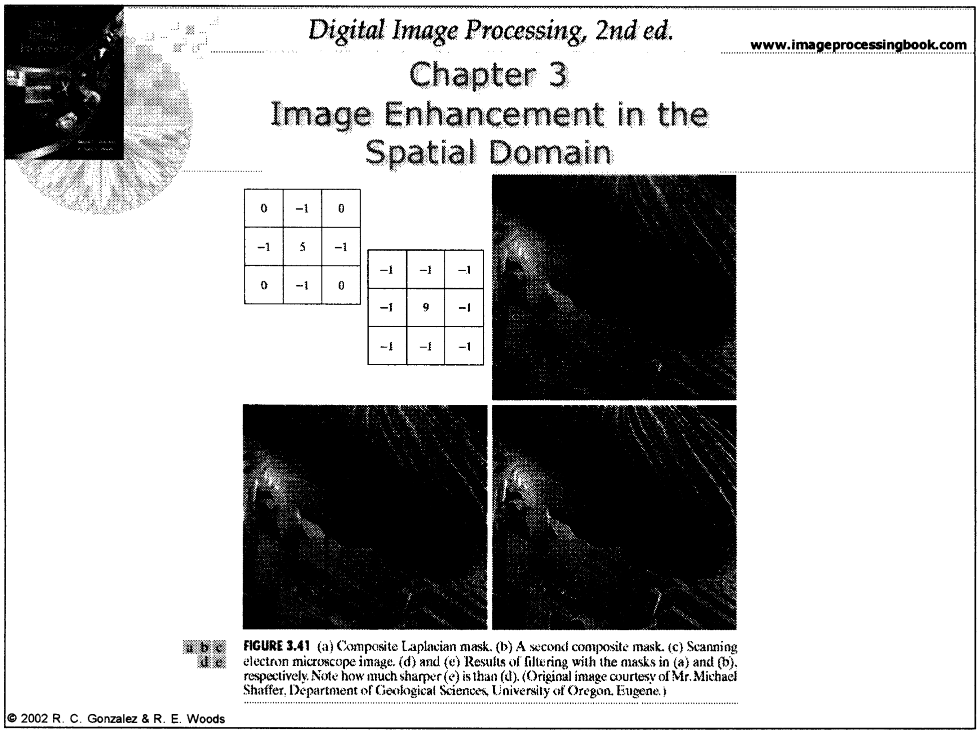
write as

$$\begin{bmatrix} 1 & W_{1k} & W_{2k} & W_{1k}W_{2k} & \cdots & 0 \\ 0 & \cdots & & & & (W_{1k})^2 (W_{2k})^2 \end{bmatrix} \begin{bmatrix} c_{00}^{(1)} \\ c_{10}^{(1)} \\ c_{01}^{(1)} \\ \vdots \\ c_{22}^{(1)} \\ \vdots \\ c_{22}^{(2)} \end{bmatrix} = \begin{bmatrix} X_{1k} \\ X_{2k} \end{bmatrix}$$

or $\underline{\underline{W}}\,\underline{c} = \underline{X}$

unknowns.

or $\underline{C} = \underline{\underline{W}}^{-1} \underline{X}$



Laplacian with diagonal terms (b) usually yields sharper detail.

(c) original SEM image

(d) image filtered with (a), Laplacian w/o diagonals

(e) image filtered with (e), Laplacian w/ diagonals

As expected (e) is much sharper than (d)

Chapter 3
Image Enhancement in the Spatial Domain

a b

0	−1	0
−1	$A+4$	−1
0	−1	0

−1	−1	−1
−1	$A+8$	−1
−1	−1	−1

FIGURE 3.42 The high-boost filtering technique can be implemented with either one of these masks, with $A \geq 1$.

unsharp masking $\qquad f_s(x,y) = f(x,y) - \underbrace{\bar{f}(x,y)}$

blurred (averaged) version of $f(x,y)$

high boost filtering

$$f_{hb}(x,y) = Af(x,y) - \bar{f}(x,y)$$

$$f_{hb}(x,y) = (A-1)f(x,y) + \underbrace{f(x,y) - \bar{f}(x,y)}_{f_s(x,y)}$$

which can come from Laplacian

$$f(x,y) \mp \nabla^2 f(x,y)$$

$$f_{hb} = \begin{cases} Af(x,y) - \nabla^2 f(x,y) \\ Af(x,y) + \nabla^2 f(x,y) \end{cases}$$

Digital Image Processing, 2nd ed. www.imageprocessingbook.com

Chapter 3
Image Enhancement in the Spatial Domain

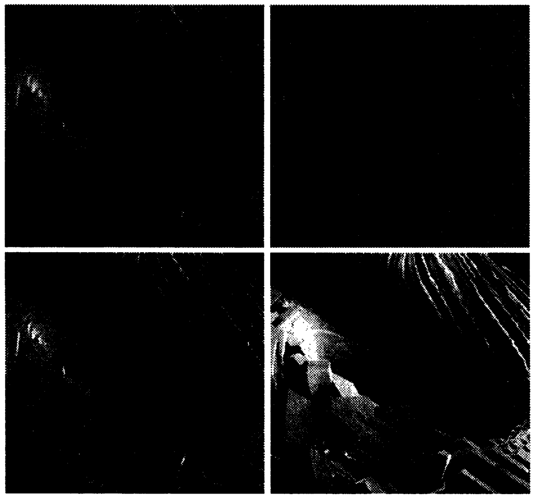


FIGURE 3.43 (a) Same as Fig. 3.41(c), but darker. (a) Laplacian of (a) computed with the mask in Fig. 3.42(b) using $A = 0$. (c) Laplacian enhanced image using the mask in Fig. 3.42(b) with $A = 1$. (d) Same as (c), but using $A = 1.7$.

(a) original image (dark in appearance)

(b) Laplacian computed with using A = 0

-1	-1	-1
-1	A+8	-1
-1	-1	-1

(c) boost using above mask with A=1 (brighter)

(d) boost using above mask with A=1.7 (even brighter)

Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

Chapter 3
Image Enhancement in the Spatial Domain

a
b c
d e

FIGURE 3.44 A 3 × 3 region of an image (the z's are gray-level values) and masks used to compute the gradient at point labeled z_5. All masks coefficients sum to zero, as expected of a derivative operator.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0
0	1

0	−1
1	0

Roberts cross-gradient operators

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Sobel operators (somewhat center weighted)

Gradient $\quad \underline{\nabla f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$

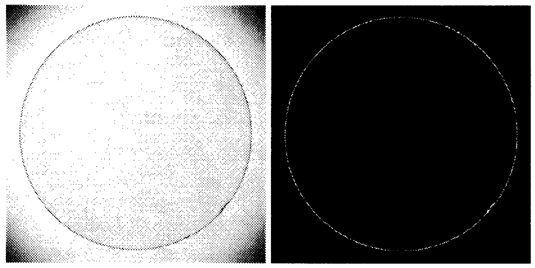
$$\nabla f = |\underline{\nabla f}| = \sqrt{G_x^2 + G_y^2} = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

This is computationally a lot of work so approximate

$$\nabla f \approx |G_x| + |G_y|$$

This loses some isotropy but is faster than computing squares and a square root.

Chapter 3
Image Enhancement in the Spatial Domain



a b

FIGURE 3.45 Optical image of contact lens (note defects on the boundary at 4 and 5 o'clock). (b) Sobel gradient. (Original image courtesy of Mr. Pete Sites, Perceptics Corporation.)

(a) original image with side and/or oblique lighting.

(b) Sobel gradient
- produces excellent boundaries for automated inspection analysis
- gradient also brings out small specks in "uniform" background areas

LIGHTING! LIGHTING! LIGHTING!

MATLAB/Image Processing Toolbox

LINEAR SPATIAL FILTERING

```
>> f=imread(fig3.15(a).jpg');         %load in checkerboard figure
% g=imfilter(f,w,filtering_mode, boundary_options,size_options)
% f is the input image
% w is the filter mask
% Filtering mode:
%    'corr' filtering is done using correlation
%    'conv' filtering is done using convolution -- flips mask 180 degrees
% Boundary options
%    P without quotes (default) - pad image with zeros
%   'replicate' - extend image by replicating border pixels
%   'symmetric' - extend image by mirroring it across its border
%   'circular' - extend image byrepeating it (one period of a periodic function)
% Size options
%   'full' - output is the same size as the padded image
%   'same' - output is the same size as the input

>> w=ones(9);                         % create a 9x9 filter (not normalized)
>> gd=imfilter(f,w);                  % filter using default values
>> imshow( gd, [ ])                   % [ ] causes MATLAB to display using low and high
                                      % gray levels of input image.
                                      %Good for low dynamic range
>> gr=imfilter(f,w,'replicate');      % pad using replication
>> figure, imshow(gr, [ ])            %
>> gs=imfilter(f,w,'symmetric');      % pad using symmetry
>> figure, imshow(gs, [ ])            % show this figure in a new window
```

SEE GWE, Section 3.4.1Linear Spatial Filtering

LINEAR SPATIAL FILTERING

```
>> f=imread(fig3.15(a).jpg');          %load in checkerboard figure
>> w=ones(9);                          % create a 9x9 filter (not normalized)

% f is of type double in [0,1] by default
>> f8=im2uint8(f);                     % converts image to uint8, i.e., integers in range [0,255]

>> g8r=imfilter(f8,w,'replicate');     % pad using replication
% imfilter creates an output of same data class as input, i.e., uint(8)
>> imshow( g8r, [ ])                   % clipping caused data loss since filter was not
                                       % normalized
```

SEE GWE, Section 3.4.1Linear Spatial Filtering

MATLAB/Image Processing Toolbox

MATLAB's built-in filters

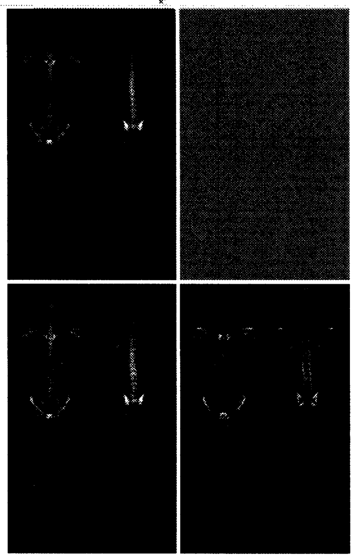
```
>> f=imread('fig3.15(a).jpg');        %load in checkerboard figure
>> w=fspecial('type', parameters); % create filter mask

% filter types:
%   'average', default is 3x3
%   'gaussian', default is 3x3 and sigma=0.5
%   'laplacian, default alpha=0.5
%   'prewitt', vertical gradient, default is 3x3.  Get horizontal by wh=w'
%   'sobel', vertical gradient, default is 3x3
%   'unsharp', default is 3x3 with alpha=0.2
```

SEE GWE, Section 3.5 Image processing Toolbox Standard Spatial Filters

Digital Image Processing, 2nd ed. www.imageprocessingbook.com

Chapter 3
Image Enhancement in the Spatial Domain



a b
c d

FIGURE 3.46 (a) Image of whole body bone scan. (b) Laplacian of (a). (c) Sharpened image obtained by adding (a) and (b). (d) Sobel of (a).

scaled for display

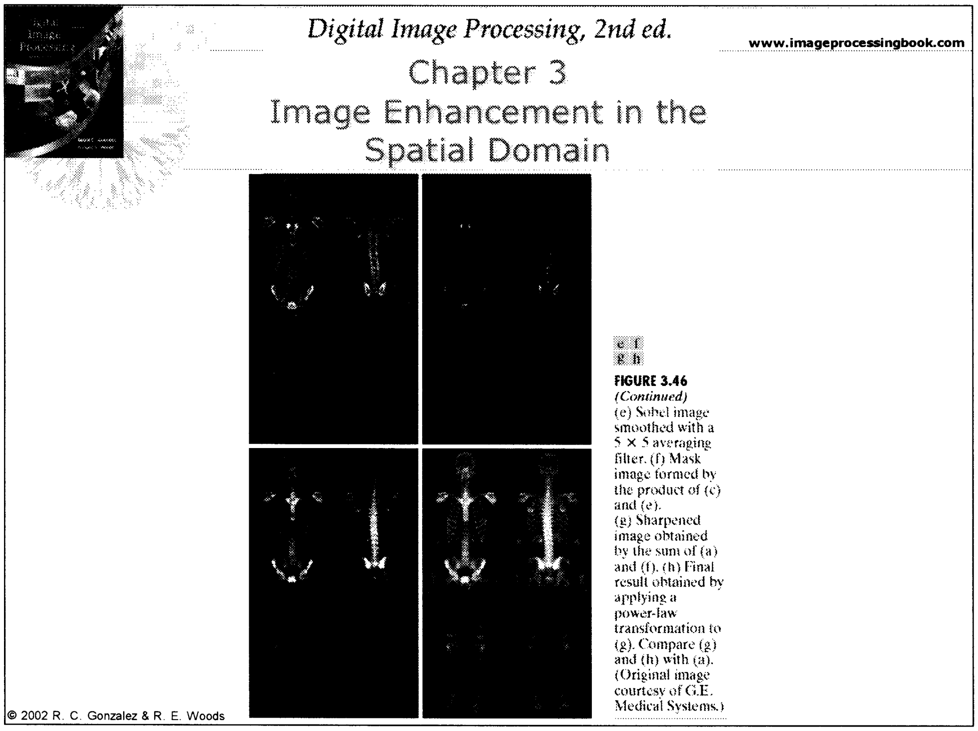
GOAL: enhance image and bring out skeletal detail

(a) original whole body bone scan

(b) Laplacian of (a). Used Laplacian with negative center

(c) sharpened image (a) + (b)
Shows noise but we can't use a median filter for medical images because they throw data away.
⇒ we use a smoother version of the gradient as well

(d) Sobel of original image (a).
Edges are clearly more dominant than in Laplacian.
Laplacian is superior for enhancing fine detail.



(e) Now smooth the Sobel using a 5×5 averaging filter

(d)&(e) are brighter than the Laplacian images because There are a lot of edges in the images.

(f) Multiply sobel smoothed image (e) by Laplacian (c) to get sharpened image.

(g) Add sharpened image (e) to original image (a) to emphasize detail in ribs, spinal cord, etc.

(h) use power law transformation to increase dynamic range of displayed image.