Digital Image Processing

Kenneth R. Castleman

2008/2/26

Presentation by S. Wang

Part One - 3

Fundamentals

Algebraic operations

Geometric transform

Chapter 7

Algebraic Operations

Introduction

- Algebraic operations produce an image which is pixel-by-pixel sum, difference, product, or quotient of two input images:

$$\boxed{C(x,y) = A(x,y) + B(x,y)}$$

$$\boxed{C(x,y) = A(x,y) - B(x,y)}$$

$$\boxed{C(x,y) = A(x,y) \times B(x,y)}$$

$$\boxed{C(x,y) = A(x,y) \div B(x,y)}$$

- The resulting pixel values must be integers within a given range, e.g., 0-255. Rounding and clipping is necessary.
- $B(x, y)$ may be a constant, which changes the overall brightness of $A(x, y)$.

Applications

- Addition
  - Produce multiple exposure effects
  - Average several images of the same scene to reduce noise.
- Subtraction
  - Remove unwanted contents
  - Detect moving objects
  - Calculate gradient to extract edges
- Multiplication
  - Correct digitizing errors or illumination non-linearity
  - Change color balance
  - Create special effects
- Division
  - Detect changes
  - Correct uneven illumination
  - Process multispectral images

Algebraic Operations and Histogram

- Find the histogram of sum images:

$$C(x,y) = A(x,y) + B(x,y)$$

- Assume *A* and *B* are uncorrelated. Joint 2-D histogram:

$$H(D_A, D_B) = H_A(D_A)\, H_B(D_B)$$

- To reduce the 2D histogram to a 1D marginal histogram:

$$H(D_A) = \int_{-\infty}^{\infty} H_{AB}(D_A, D_B)dD_B = \int_{-\infty}^{\infty} H_A(D_A)H_B(D_B)dD_B$$

- Taking into account relationship among D_A, D_B and D_C:

$$H(D) = \int_{-\infty}^{\infty} H_A(D_C - D_B)H_B(D_B)dD_B$$

- The right side is a function of D_C. Thus

$$H_C(D) = H_A(D) * H_B(D)$$

Histogram of Sum Images: an Example

- Supposing two images have Gaussian histograms, the sum image also has a Gaussian histogram.
  - The mean equal to the sum of individual means, and
  - The variance equal to the sum of individual variances.

$$H_i(x) = A_i \exp\left[-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right], \quad i = 1, 2$$

$$\Rightarrow \quad H_1(x) * H_2(x) = A_1 A_2 \sqrt{2\pi\sigma_1\sigma_2}\exp\left[-\frac{(x-(\mu_1+\mu_2))^2}{2(\sigma_1^2+\sigma_2^2)}\right]$$

- The convolution formula still holds as long as one of the images is changed to its negative.

Averaging for Noise Reduction

- Consider a number of exposures taken on the same scene, each with independent additive noise:

$$D_i(x,y) = S(x,y) + N_i(x,y), \qquad i = 1, 2, \ldots, M$$

- $N_i(x,y)$ is an independent sample of zero mean noise field:

$$\mathrm{E}\left[N_i(x,y)\right] = 0$$
$$\mathrm{E}\left[N_i(x,y)N_j(x,y)\right] = 0$$

- Signal-to-noise power ratio of individual images:

$$P(x,y) = S^2(x,y) / \mathrm{E}\left[N^2(x,y)\right]$$

Averaging for Noise Reduction (cont.)

- Average of *M* images:

$$\bar{D}(x,y)=\frac{1}{M}\sum_{i=1}^{M}\left[S(x,y)+N_i(x,y)\right]$$

Demo_Algebra1
Demo_NoiseAv

- Signal-to-noise power ratio *P*(*x*,*y*) is increased by a factor of *M*, or, SNR is increased by 10 logM dB:

$$\bar{P}(x,y)=M\ P(x,y)$$

- An example.
- Try to write a program to verify this conclusion.
- Demo: Demo_Algebra1. (Code)

Image Subtraction

- Background subtraction: see Fig.7-3 on p.109 (removing the uneven illumination field effect)
- Motion detection: see Fig.7-4 on p.110 (removing stationary freeway to show the moving vehicles)
- Gradient magnitude:
  - The gradient field: $\nabla f(x,y)=\vec{i}\,\frac{\partial f(x,y)}{\partial x}+\vec{j}\,\frac{\partial f(x,y)}{\partial y}$
  - The gradient magnitude: $\left|\nabla f(x,y)\right|=\sqrt{\left(\frac{\partial f}{\partial x}\right)^2+\left(\frac{\partial f}{\partial y}\right)^2}$
  - To avoid calculating square roots:

$$\left|\nabla f(x,y)\right|\approx\max\left[\left|f(x,y)-f(x+\Delta x,y)\right|,\left|f(x,y)-f(x,y+\Delta y)\right|\right]$$

Image subtraction

Summary of Algenraic Operations

- The histogram of a sum image is convolution of the histograms of the two component images.
- Convolution of two Gaussian functions produces a broader Gaussian, with the means and variances added.
- Averaging *M* images increases the SNR by 10 logM dB.
- Subtracting slightly offset images gives a partial derivative image.

Summary of Algenraic Operations (cont.)

- Image multiplication can be used to mask unwanted areas in an image by setting the multiplier to zero.
- Image division is useful in removing spatially varying digitizer sensitivity function.
- Multispectral image processing will be discussed later.
- Problems of Chapter 7: p.113, 4

Chapter 8

Geometric Operation

Introduction

- Geometric transform changes spatial relationships among objects in an image: Moving things around (like stretching a rubber sheet).
- Constraints: pixel order is preserved so that no random scrambling of pixels occurs.
- Two basic operations involved:
  - Spatial transformation
  - Gray level interpolation

Introduction: Spatial Transformation

- It is desirable to preserve continuity of curvilinear features and connectivity of objects.
- General definition of spatial transformation:

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

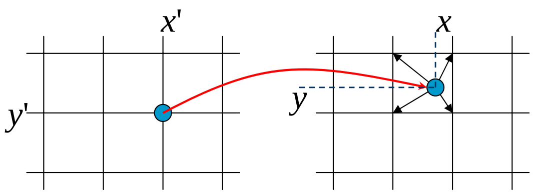
where $f(x,y)$ is the input image, and $g(x,y)$ is the output.

Spatial Transformation (cont.)

- Connectivity is preserved if $a(x,y)$ and $b(x,y)$ are continuous.
- Gray levels in the input image are defined at integer values of (x, y), while in the output image, $g(x,y)$ may be taken from $f(x,y)$ at **fractional** coordinate positions.
- Two possibilities:
  - Pixels in $f \rightarrow$ Positions between pixels in g
  - Positions between pixels in $f \rightarrow$ Pixels in g

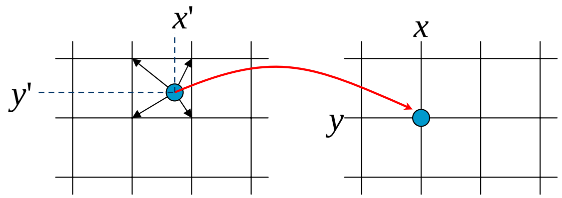
Introduction: Gray Level Interpolation

- Two approaches:



Forward mapping (pixel carryover)

A pixel value is divided into four parts.

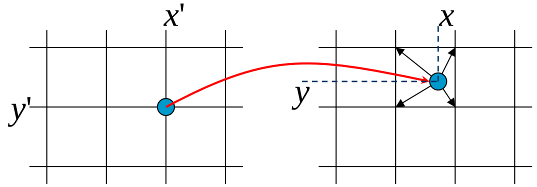


Backward mapping (pixel filling)

Four pixel values are combined into one.

Forward Mapping

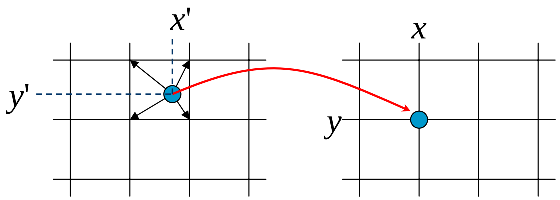
- Each output pixel may be addressed several times.
- Many input pixels may fall outside output range: a waste.
- If image is enlarged, some output pixels may be missed.
- If the size is reduced, more than 4 input pixels may contribute to an output pixel.



Forward mapping (pixel carryover)

Backward Mapping

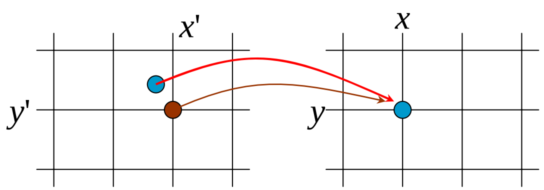
- Output pixels are generated line-by-line and point-by-point.
- Each output pixel is contributed by 4 input pixels.
- Input pixels are accessed randomly.
- It is a more practical way of doing interpolation.



Backward mapping (pixel filling)

Gray-Level Interpolation Methods

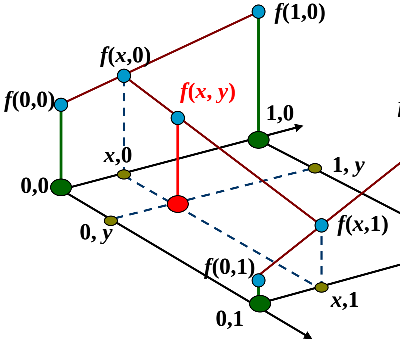
- Using backward mapping, output pixels generally map to somewhere between four input pixels.
- Zero-order technique: Nearest neighbor interpolation:
  - Computationally simple.
  - Artifacts may occur in areas with fine details (see Fig.8-2).



Nearest neighbor interpolation

Bilinear Interpolation (1)

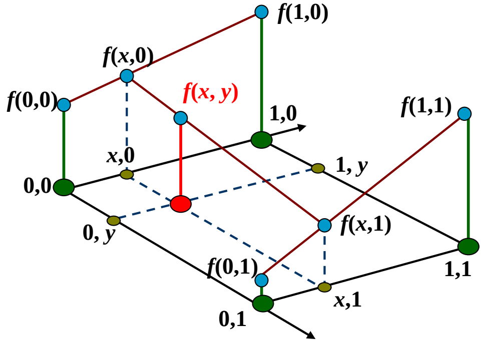
- It is a first-order technique, which gives a more desirable result with only slightly increased computation complexity.
- Fitting a plane from 4 points is **over-determined**. A bilinear function is required to interpolate on a rectangular grid.
- Problem: Find $f(x,y)$ given $f(0,0)$, $f(1,0)$, $f(0,1)$, and $f(1,1)$.



Bilinear Interpolation (2)

- Assume that $f(x,y)$ is on a hyperbolic paraboloid defined by the following bilinear equation:

$$\boxed{f(x,y) = ax + by + cxy + d}$$



Bilinear Interpolation (3)

- Linear interpolate from $f(0,0)$ and $f(1,0)$ to $f(x,0)$, and from $f(0,1)$ and $f(1,1)$ to $f(x,1)$:

$$\boxed{\begin{aligned} f(x,0) &= f(0,0) + x\left[f(1,0) - f(0,0)\right] \\ f(x,1) &= f(0,1) + x\left[f(1,1) - f(0,1)\right] \end{aligned}}$$

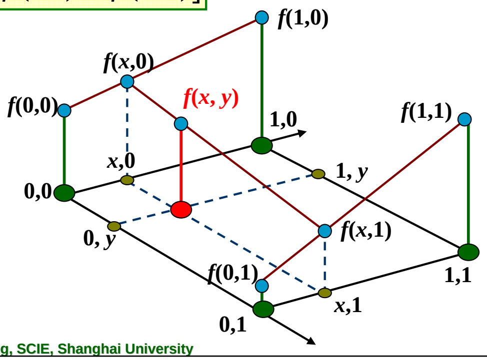
- Linear interpolate from $f(x,0)$ and $f(x,1)$ to $f(x,y)$:

$$\boxed{f(x,y) = f(x,0) + y\left[f(x,1) - f(x,0)\right]}$$

Computation Complexity:

6 Additions +

3 Multiplications



Bilinear Interpolation (4)

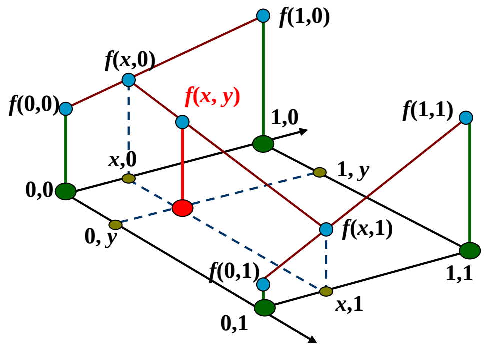
- Combine the above three equations into one:

$$\boxed{\begin{aligned} f(x,y) = {} & \left[f(1,0) - f(0,0)\right]x + \left[f(0,1) - f(0,0)\right]y \\ & + \left[f(1,1) + f(0,0) - f(0,1) - f(1,0)\right]xy + f(0,0) \end{aligned}}$$

Computation Complexity:

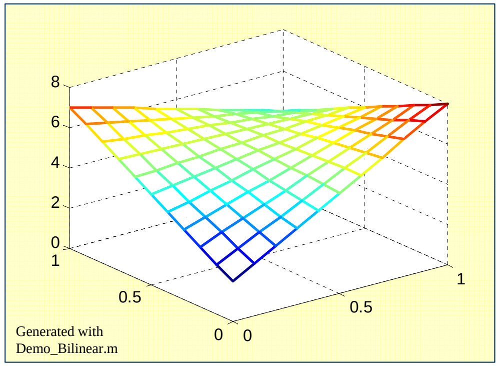
8 Additions +

4 Multiplications



Bilinear Interpolation (5)

- Intersection of the hyperbolic paraboloid in *x-z* and *y-z* planes are straight lines.
- Gray-levels across neighboring grids are continuous.
- Slopes (gradients) are not continuous.



Demo_Bilinear.m

High Order Interpolation

- High-order interpolation is used when
  - Higher resolution is required, or
  - Slope discontinuity is not tolerable.
- High-order interpolation can be done by solving an equation more complicated with more than 4 coefficients.
- Some high order techniques:
  - Cubic spline
  - Legendre centered function
  - Sinc function

Spatial Transformation

- The general form:

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

- **Simple transformations**
- Identity operation:

$$\begin{cases} a(x,y) = x \\ b(x,y) = y \end{cases}$$

- Translation:

$$\begin{cases} a(x,y) = x + x_0 \\ b(x,y) = y + y_0 \end{cases}$$

Or, in matrix form (in the homogeneous coordinates):

$$\begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Spatial Transformation (cont.)

- Magnification:

$$\begin{cases} a(x,y) = x / c \\ b(x,y) = y / d \end{cases}$$

or, in matrix:

$$\begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix} = \begin{bmatrix} 1/c & 0 & 0 \\ 0 & 1/d & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Reflection about y axis:

$$\begin{cases} a(x,y) = -x \\ b(x,y) = y \end{cases}$$

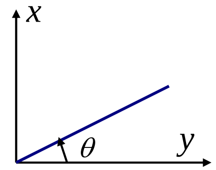
or:

$$\begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Spatial Transformation (cont.)

- Rotation by an angle θ:

$$\begin{cases} a(x,y) = x\cos\theta - y\sin\theta \\ b(x,y) = x\sin\theta + y\cos\theta \end{cases}$$



In the matrix form:

$$\begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Spatial Transformation (cont.)

- Rotation by θ about (x_0, y_0) $\Rightarrow$ combination of rotation and translation:
  - First, translate the image to make (x_0, y_0) the origin.
  - Rotate by θ, and
  - Translate back to restore the origin.

$$\begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta & -\sin\theta & -x_0\cos\theta + y_0\sin\theta + x_0 \\ \sin\theta & \cos\theta & -x_0\sin\theta - y_0\cos\theta + y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Separable Implementations

- It is easy to separate translation and magnification into two steps, for example, horizontal first, and then vertical.
- It is also possible to perform a rotation in two steps.

$$\begin{cases} a(x,y) = x\cos\theta - y\sin\theta \\ b(x,y) = x\sin\theta + y\cos\theta \end{cases}$$

Solving the first equation for *x*:

$$x = \frac{a(x,y) + y\sin\theta}{\cos\theta}$$

Substituting into the 2nd equation:

$$b(x,y) = \frac{a(x,y)\sin\theta + y}{\cos\theta}$$

Separable Implementations (cont.)

- Having obtained the equations

$$x = \frac{a(x,y) + y\sin\theta}{\cos\theta} \quad \& \quad b(x,y) = \frac{a(x,y)\sin\theta + y}{\cos\theta}$$

a two-step rotation can be performed as follows.

Step 1: keep y unchanged:

$$\begin{cases} a'(x,y) = x\cos\theta - y\sin\theta \\ b'(x,y) = y \end{cases}$$

Step 2: keep the obtained *a*(*x*,*y*):

$$\begin{cases} a(x,y) = a'(x,y) \\ b(x,y) = \dfrac{a'(x,y)\sin\theta + y}{\cos\theta} \end{cases}$$

Demo_Rotate.m

General Transformations

- Analytical expressions generally do not exist. Transform may be got from measurements. For example (Fig.8-4):
  - Take the first shot on a grid pattern,
  - Calculate the transformation numerically from the grid, and
  - Apply the obtained transformation to subsequent shots.
- Control point approach: Only a small number of pixels are specified, and most pixels are corrected by interpolation.
- Polynomial fitting:
  - Determined: No. of polynomial coefficients = No. of control points
  - Over-determined or under-determined: pseudo-inversion technique
- Piecewise bilinear interpolation

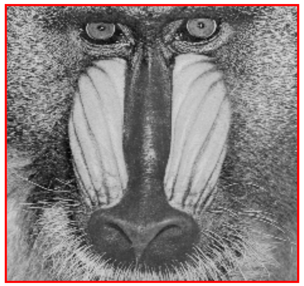
Applications of Geometrical Operations

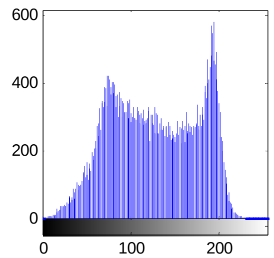
- Geometric calibration — To remove camera-induced geometric distortion.
- Image rectification — To correct non-rectangular pixel coordinates. (also see Fig.8-8, Fig.8-9)
- Image registration — To register images for comparison.
  - MRI image registration
  - Anti-attack watermark detection
  - Motion detection
- Image format conversion (Fig.8-10)
- Map projection (for self-reading)
- Morphing for special effects (Fig.8-17)
- Problems of Chapter 8: p.138, 1,4,

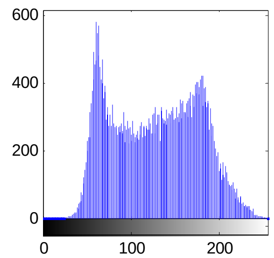
More examples

End of Part One

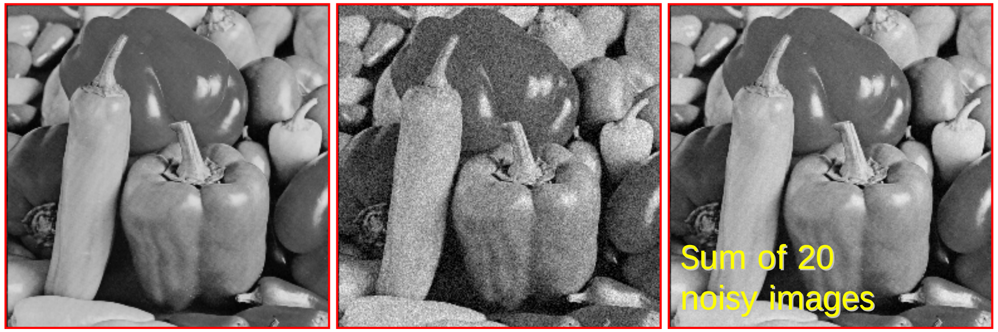
Histogram of Negative Image

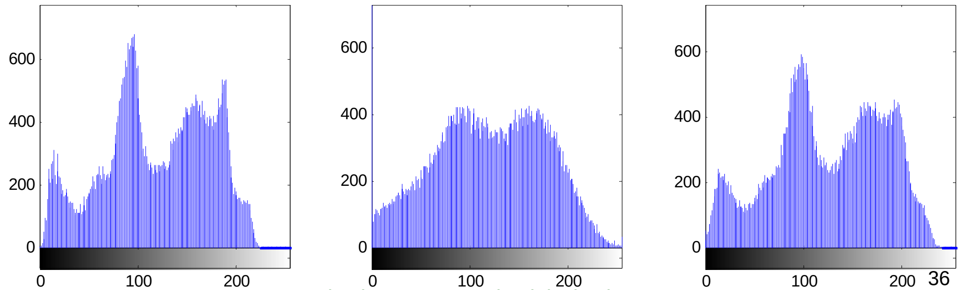






Noise Reduction by Image Averaging





36

Code for Noise Reduction by Averaging (1)

```
(get image F0, sized [M,N])
K=20;              % Number of images
S=20;              % Noise magnitude
F=zeros(M,N,K);
for k=1:K          % 20 noisy images
   F(:,:,k)=S*randn(M,N)+F0;
end
img4a(F0,[10 480],'Clean Image');
img4a(F(:,:,1),[20+M 480],'Noisy Image');
IM=img4a(F(:,:,1),[30+2*M 480]);
```


Code for Noise Reduction by Averaging (2)

```
% Histogram of clean image
figure('Pos',[10 400-N M N]);
imhist(uint8(F0),256);

% Histogram of noisy image
figure('Pos',[20+M 400-N M N]);
imhist(uint8(F(:,:,1)),256);

% Histogram of noisy image
FG=figure('Pos',[30+2*M 400-N M N]);
imhist(uint8(F(:,:,1)),256);
```

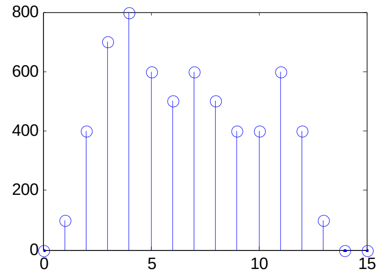

Code for Noise Reduction by Averaging (3)

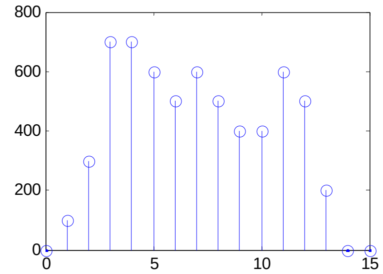
```
for k=1:K
   Fbar=F0;
   for l=1:k,              % averaging
     Fbar=Fbar+F(:,:,l);
   end
   Fbar=clip(round(Fbar/k),0,255);
   close(IM);
   IM=img4a(Fbar);
   close(FG);
   FG=figure;
   imhist(uint8(Fbar),256);
   pause;
end
```

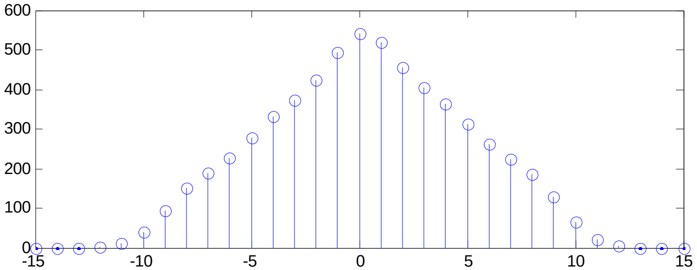
Exercise: P.113_4

```
H1=[0 100 400 700 800 600 500 600 500 400 400 600 400 100 0 0];
H2=[0 100 300 700 700 600 500 600 500 400 400 600 500 200 0 0];

% Histogram of the difference image: Im2-Im1:
H3=conv2(H2,H1(16:-1:1));

M=sum(H1(:));
figure('Pos',[10 300 300 220],'Units','norm');
axes('Pos',[.1 .1 .86 .86]);
stem(0:15,H1);
figure('Pos',[10 300 300 220],'Units','norm');
axes('Pos',[.1 .1 .86 .86]);
stem(0:15,H2);
figure('Pos',[10 300 600 220],'Units','norm');
axes('Pos',[.1 .1 .86 .86]);
stem(-15:15,H3/M);

H3=round(H3/M);        % Histogram of difference image
```


Exercise: P.113_4 (cont.)

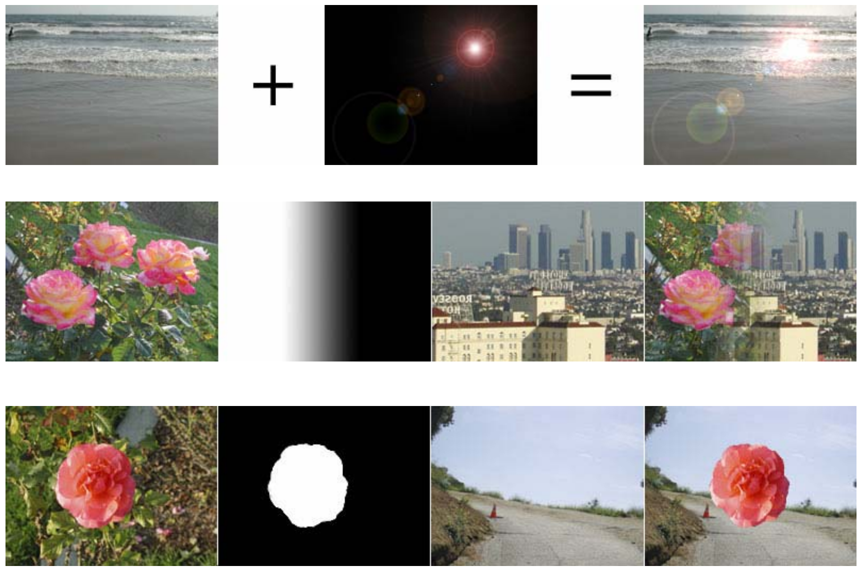






H3=[0 0 0 2 11 41 93 151 190 228 277 331 372 425 495 541 518 456 403 362 311 261 225 187 130 66 21 3 0 0 0]

Addition: Multiple Exposure



Addition: Multiple Exposure Effects

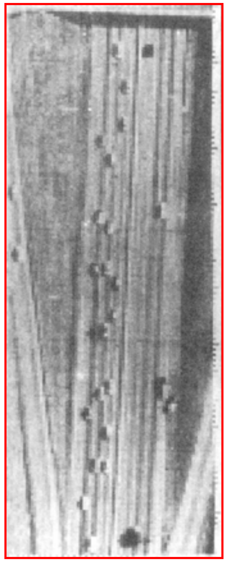
Go back

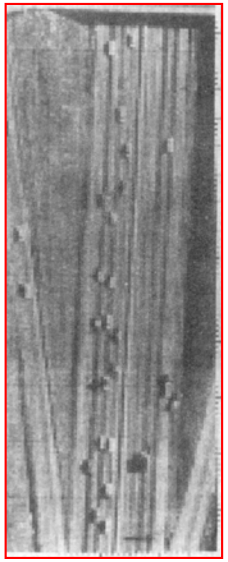
Image Subtraction

Subtraction: Detection of Moving Objects

Moving objects in a video sequence can be detected by pixel subtraction.

Go back





Subtraction: Background Removal

Subtraction: Edge Extraction

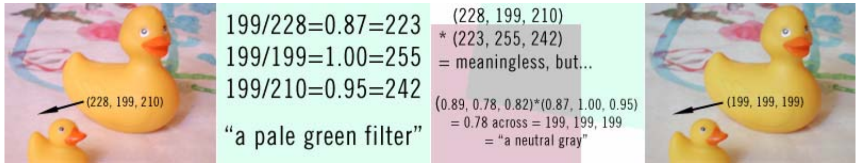
- Detection of motion or slight changes: difference of two nearly-identical and slightly offset images:

$$C(x,y) = A(x,y) - A(x+\Delta x, y) \approx \frac{\partial}{\partial x} A(x,y)\Delta x$$

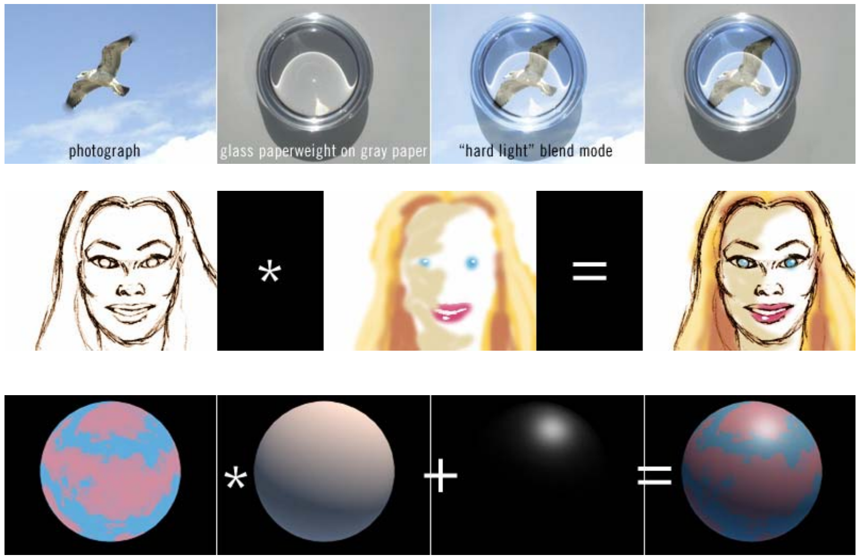
Multiplication: Correct Brightness

Multiplication: Color Balance Correction

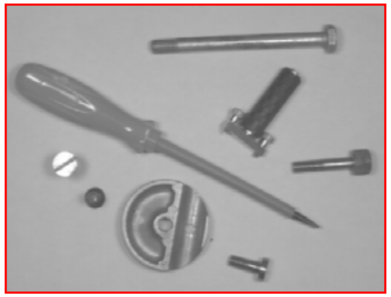
- Suppose an image is too pink (too red and too blue), but its green channel is just fine.
- To reduce red and blue, multiply them by a percentage less than 100%. Example: multiply it by (0.87, 1.00, 0.95).
- Multiplication by a color is analogous to looking through a filter of that color.
- Looking at the overly-magenta image through a "pale green" filter will correct the image.



Addition and Multiplication

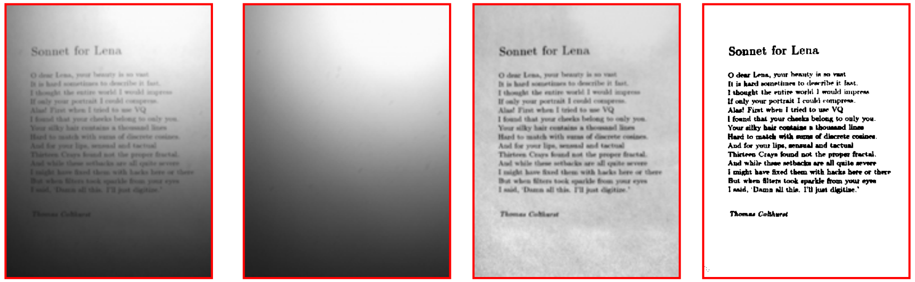


Division: Detection of Changes



Slight changes in image can be detected using pixel division.

Division: Correct Uneven Illumination



$$B(x,y) \propto I(x,y)\,R(x,y)$$

$$\frac{B_{orig}(x,y)}{B_{blank}(x,y)} \propto \frac{I_{orig}(x,y)\,R_{orig}(x,y)}{I_{blank}(x,y)\,R_{blank}(x,y)}$$

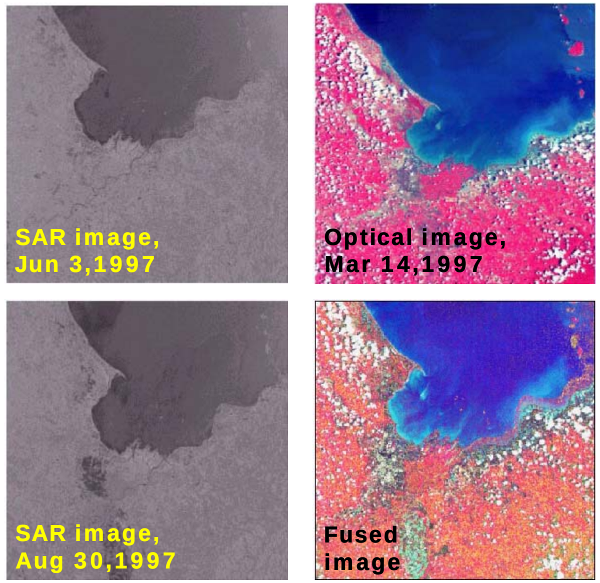
$$\frac{B_{orig}(x,y)}{B_{blank}(x,y)} \propto R_{orig}(x,y)$$

1. Same illumination: $I_{orig} = I_{blank}$
2. Reflectivity of blank paper is constant.
3. Even background obtained by division.

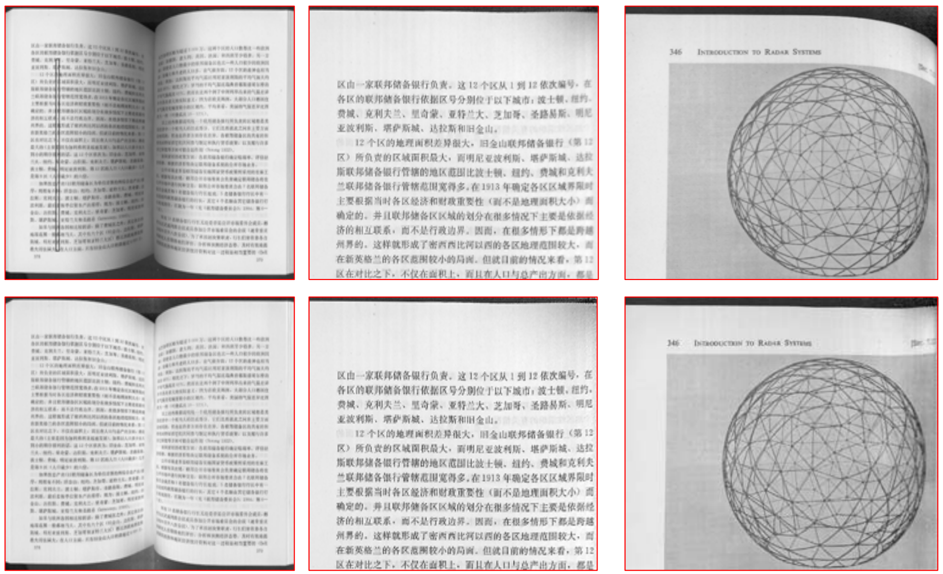
Multispectral Images: Flood Assessment

Flood assessment: Surat Thani, Thailand

Image arithmetic operations, neural networks and image fusion are used.



Correction of Geometric Distortion



Correction of Lens Distortion

Correction of Lens Distortion