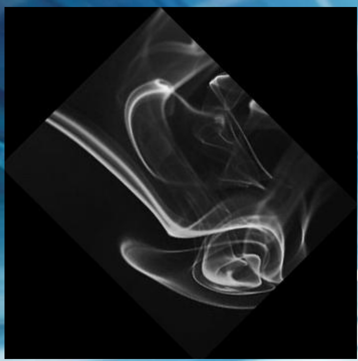
Introduction to Digital Image Processing with MATLAB® Asia Edition
McAndrew‧Wang‧Tseng



Chapter 6: Image Geometry

6.1 Interpolation of Data

- Suppose we have a collection of four values that we wish to enlarge to eight

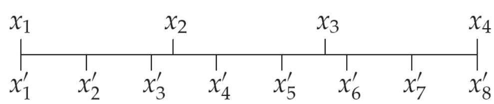


FIGURE 6.1 *Replacing four points with eight.*

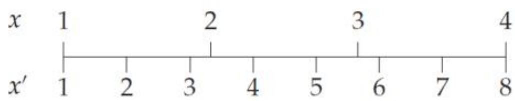


FIGURE 6.2 *Figure 6.1 slightly redrawn.*

Ch6-p.121-122

6.1 Interpolation of Data

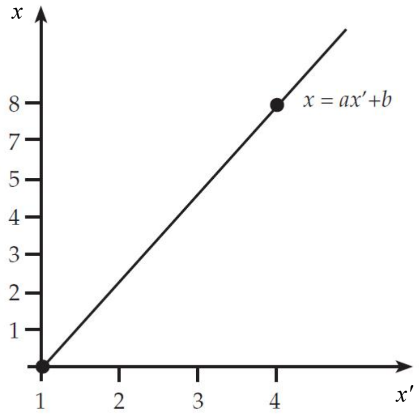


FIGURE 6.3 *Filled circles are the coincide points of x and x'.*

6.1 Interpolation of Data

- The a and b of the linear function can be solved by

$$\begin{cases} 1 = a + b \\ 4 = 8a + b \end{cases}$$

- Then we can obtain the linear function

$$x = \frac{3}{7}x' + \frac{4}{7} \qquad x' = \frac{1}{3}(7x - 4), \quad x = \frac{1}{7}(3x' + 4)$$

(continuous)

6.1 Interpolation of Data

- In digital (discrete), none of the x'_i points coincide exactly with an original x_j, except for the first and last
- We have to estimate function values $f(x'_i)$ based on the known values of nearby $f(x_j)$
- Such estimation of function values based on surrounding values is called **interpolation**

Ch6-p.122-123

CENGAGE Learning

5

FIGURE 6.4

- **Nearest-neighbor interpolation**

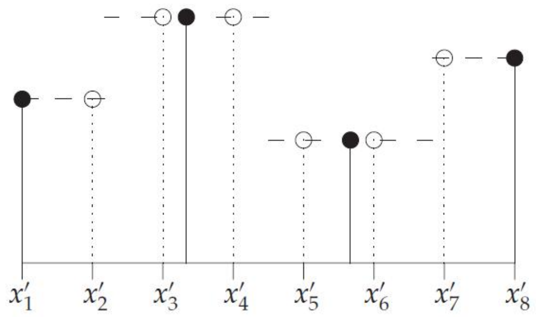


FIGURE 6.4 *Nearest-neighbor interpolation.*

Ch6-p.123

CENGAGE Learning

6

FIGURE 6.5

- **Linear interpolation**

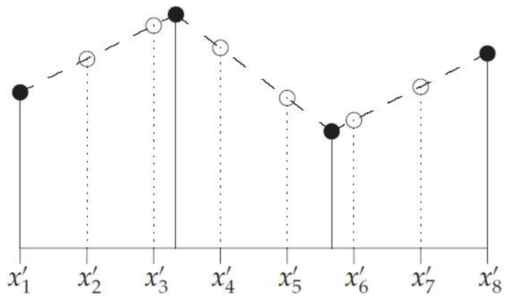
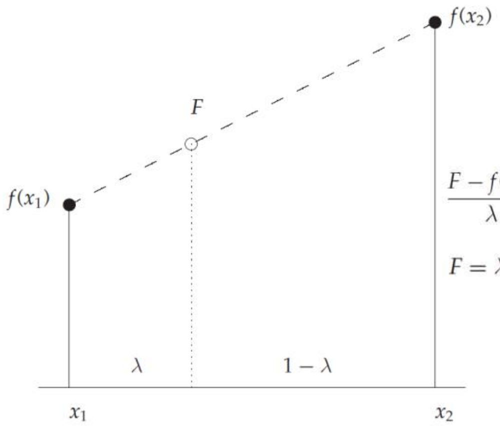


FIGURE 6.5 *Linear interpolation.*

7 **Ch6-p.123**

FIGURE 6.6



$$\frac{F - f(x_1)}{\lambda} = \frac{f(x_2) - f(x_1)}{1}$$

$$F = \lambda f(x_2) + (1-\lambda) f(x_1).$$

(Equation 6.1)

FIGURE 6.6 *Calculating linearly interpolated values.*

8 **Ch6-p.124**

6.2 Image Interpolation

- Using the formula given by Equation 6.1

$$f(x,y') = \mu f(x,y+1) + (1-\mu)f(x,y)$$

$$f(x+1,y') = \mu f(x+1,y+1) + (1-\mu)f(x+1,y)$$

$$f(x',y') = \lambda f(x+1,y') + (1-\lambda)f(x,y')$$

9 **Ch6-p.125**

CENGAGE Learning

FIGURE 6.7

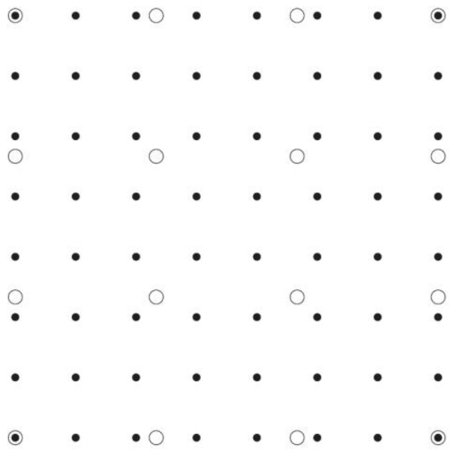


FIGURE 6.7 *Interpolation on an image.*

10 **Ch6-p.125**

CENGAGE Learning

FIGURE 6.8

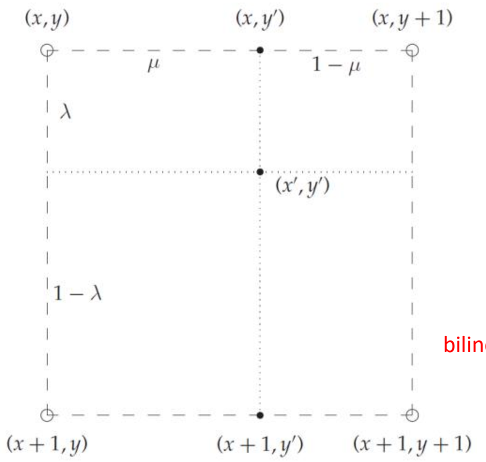


FIGURE 6.8 *Interpolation between four image points.*

bilinear interpolation

6.2 Image Interpolation

- Function `imresize`

  `imresize(A,k,'method')`

- Where `A` is an image of any type, `k` is a scaling factor, and `'method'` is either `'nearest'` or `'bilinear'`, etc.

```
>> c=imread('cameraman.tif');
>> head=c(33:96,90:153);
>> imshow(head)
>> head4n=imresize(head,4,'nearest');imshow(head4n)
>> head4b=imresize(head,4,'bilinear');imshow(head4b)
```

FIGURE 6.9 & 6.10

FIGURE 6.9 *The head.*

(a) (b)

FIGURE 6.10 *Scaling by interpolation. (a) Nearest neighbor scaling. (b) Bilinear interpolation.*

13 **Ch6-p.127-128**

CENGAGE Learning

6.3 General Interpolation

$$f(x') = R(-\lambda)f(x_1) + R(1-\lambda)f(x_2).$$ (Equation 6.2)

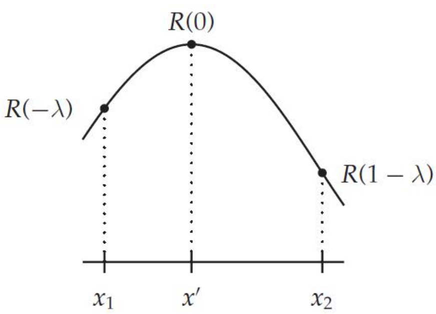


FIGURE 6.11 *Using a general interpolation function.*

14 **Ch6-p.129**

CENGAGE Learning

FIGURE 6.12

$$R_0(u) = \begin{cases} 0 & \text{if } u \leq -0.5, \\ 1 & \text{if } -0.5 < u \leq 0.5, \\ 0 & \text{if } u > 0.5, \end{cases}$$

$$R_1(u) = \begin{cases} 1+u & \text{if } u \leq 0, \\ 1-u & \text{if } u \geq 0. \end{cases}$$

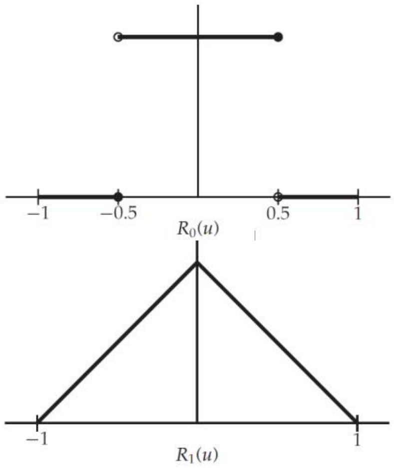


FIGURE 6.12 *Two interpolation functions.*

6.3 General Interpolation

- The functions $R_0(u)$ and $R_1(u)$ are just two members of a family of possible interpolation functions
- Another such function provides **cubic interpolation**

$$R_3(u) = \begin{cases} 1.5|u|^3 - 2.5|u|^2 + 1 & \text{if } |u| \leq 1, \\ -0.5|u|^3 + 2.5|u|^2 - 4|u| + 2 & \text{if } 1 < |u| \leq 2. \end{cases}$$

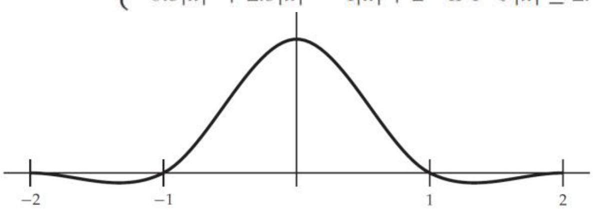


FIGURE 6.13 *The cubic interpolation function $R_3(u)$.*

FIGURE 6.14

$$f(x') = R_3(-1-\lambda)f(x_1) + R_3(-\lambda)f(x_2) + R_3(1-\lambda)f(x_3) + R_3(2-\lambda)f(x_4),$$

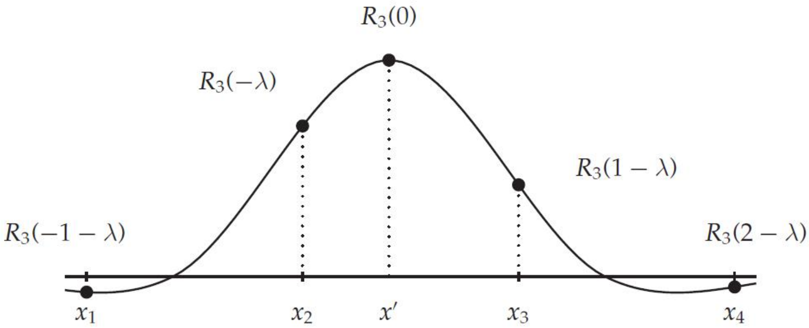


FIGURE 6.14 *Using $R_3(u)$ for interpolation.*

FIGURE 6.15

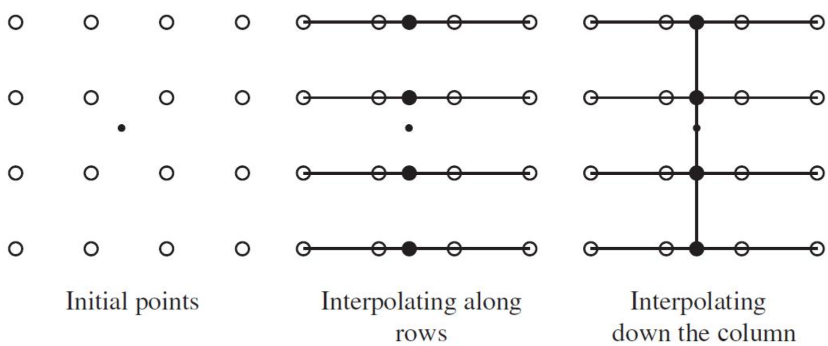


FIGURE 6.15 *How to apply bicubic interpolation.*

FIGURE 6.16

```
>> head4c=imresize(head,4,'bicubic');imshow(head4c)
```

Mathworks

FIGURE 6.16 *Enlargement using bicubic interpolation.*

6.4 Enlargement by Spatial Filtering

- If we merely wish to enlarge an image by a power of two, there is a quick and dirty method that uses linear filtering

e.g.

```
>> m=magic(4)

m =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

✓ **zero-interleaved**

$$m_2(i,j) = \begin{cases} m((i+1)/2,(j+1)/2) & \text{if } i \text{ and } j \text{ are both odd,} \\ 0 & \text{otherwise.} \end{cases}$$

FIGURE 6.17

This can be implemented with a simple function

```
function out=zeroint(a)
%
% ZEROINT(A) produces a zero-interleaved version of the matrix A.
% For example:
%
%   a=[1 2 3;4 5 6];
%   zeroint(a)
%
%     1     0     2     0     3
%     0     0     0     0     0
%     4     0     5     0     6
%
[m,n]=size(a); a2=reshape([a;zeros(m,n)],m,2*n);
out=reshape([a2';zeros(2*n,m)],2*n,2*m)';
```

FIGURE 6.17 *A simple function for implementing zero interleaving.*

6.4 Enlargement by Spatial Filtering

```
>> m2=zeroint(m)

m2 =

    16     0     2     0     3     0    13     0
     0     0     0     0     0     0     0     0
     5     0    11     0    10     0     8     0
     0     0     0     0     0     0     0     0
     9     0     7     0     6     0    12     0
     0     0     0     0     0     0     0     0
     4     0    14     0    15     0     1     0
     0     0     0     0     0     0     0     0
```

We can now replace the zeros by applying a spatial filter to this matrix

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad \frac{1}{64}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

nearest-neighbor bilinear bicubic

6.4 Enlargement by Spatial Filtering

```
>> filter2([1 1 0;1 1 0;0 0 0],m2)

ans =

    16    16     2     2     3     3    13    13
    16    16     2     2     3     3    13    13
     5     5    11    11    10    10     8     8
     5     5    11    11    10    10     8     8
     9     9     7     7     6     6    12    12
     9     9     7     7     6     6    12    12
     4     4    14    14    15    15     1     1
     4     4    14    14    15    15     1     1

>> filter2([1 2 1;2 4 2;1 2 1]/4,m2)

ans =

   16.0000    9.0000    2.0000    2.5000    3.0000    8.0000   13.0000    6.5000
   10.5000    8.5000    6.5000    6.5000    6.5000    8.5000   10.5000    5.2500
    5.0000    8.0000   11.0000   10.5000   10.0000    9.0000    8.0000    4.0000
    7.0000    8.0000    9.0000    8.5000    8.0000    9.0000   10.0000    5.0000
    9.0000    8.0000    7.0000    6.5000    6.0000    9.0000   12.0000    6.0000
    6.5000    8.5000   10.5000   10.5000   10.5000    8.5000    6.5000    3.2500
    4.0000    9.0000   14.0000   14.5000   15.0000    8.0000    1.0000    0.5000
    2.0000    4.5000    7.0000    7.2500    7.5000    4.0000    0.5000    0.2500
```


FIGURE 6.18

```
>> imshow(hz)
>> imshow(filter2([1 1 0;1 1 0;0 0 0],hz)/255)
>> imshow(filter2([1 2 1;2 4 2;1 2 1]/4,hz)/255)
>> bfilt=[1 4 6 4 1;4 16 24 16 4;6 24 36 24 6;4 16 24 16 4;1 4 6 4 1]/64;
>> imshow(filter2(bfilt,hz)/255)
```

Zero interleaving   Nearest neighbor   Bilinear   Bicubic

FIGURE 6.18 *Enlargement by spatial filtering.*

6.5 Scaling Smaller

- Making an image smaller is also called **image minimization**

- **Subsampling**

  e.g.

```
>> t=zeros(1024,1024);
>> for i=1:1024
    for j=1:1024
        t(i,j)=((255.5)^2<(i-512).^2+(j-512).^2)&((i-512)...
        ^2+(j-512).^2<(256.5)^2);
    end
   end
>> t=~t;
```

```
>> tr=imresize(t,0.25);
```

```
>> trc=imresize(t,0.25,'bicubic');
```

25 Ch6-p.135

FIGURE 6.19

(a) (b)

FIGURE 6.19 *Minimization. (a) Nearest-neighbor minimization. (b) Bicubic interpolation for minimization.*

26 Ch6-p.136

6.6 Rotation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}$$

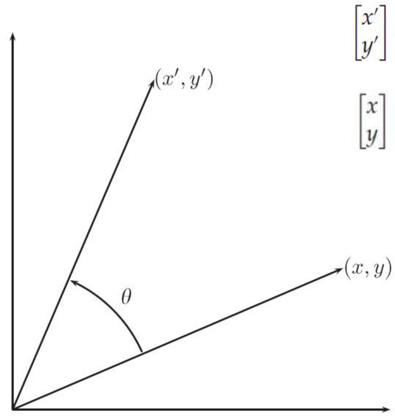


FIGURE 6.20 *Rotating a point through angle θ.*

27 Ch6-p.136

FIGURE 6.21

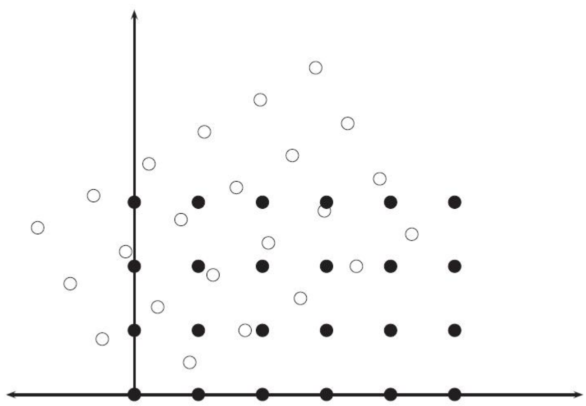


FIGURE 6.21 *Rotating a rectangle.*

28 Ch6-p.138

6.6 Rotation

- In Figure 6.21, the filled circles indicate the original position, and the open circles point their positions after rotation
- We must ensure that even after rotation, the points remain in that grid
- To do this we consider a rectangle that includes the rotated image, as shown in Figure 6.22

FIGURE 6.22

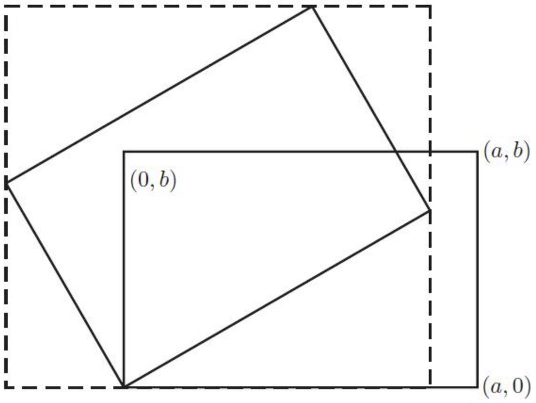


FIGURE 6.22 *A rectangle surrounding a rotated image.*

FIGURE 6.23

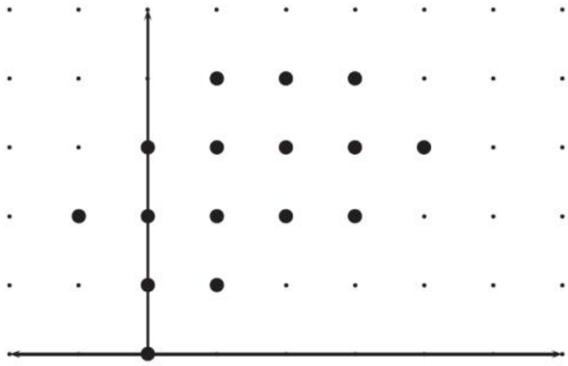


FIGURE 6.23 *The points on a grid after rotation.*

6.6 Rotation

The gray value at (x'', y'') can be found by interpolation, using surrounding gray values. This value is then the gray value for the pixel at (x', y') in the rotated image

FIGURE 6.24

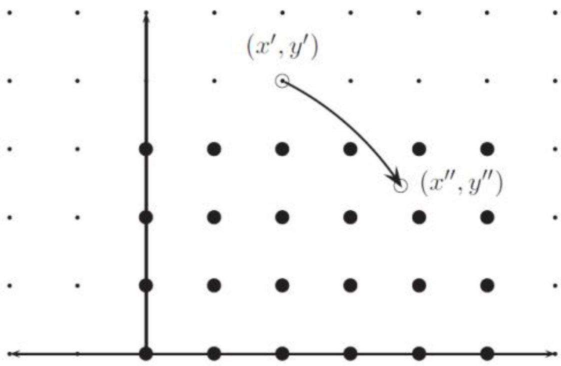


FIGURE 6.24 *Rotating a point back into the original image.*

FIGURE 6.25

```
>> cr=imrotate(c,60);
>> imshow(cr)
>> crc=imrotate(c,60,'bicubic');
>> imshow(crc)
```

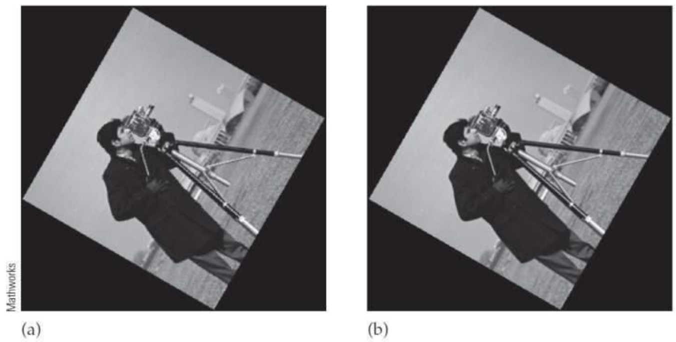


FIGURE 6.25 *Rotation with interpolation. (a) Nearest neighbor. (b) Bicubic interpolation.*

FIGURE 6.26

FIGURE 6.26 The Ambassadors *(1533) by Hans Holbein.*

FIGURE 6.27

```
>> a=imread('AMBASSADORS.JPG');
>> a=rgb2gray(a);
```

```
>> skull=a(566:743,157:586);
```

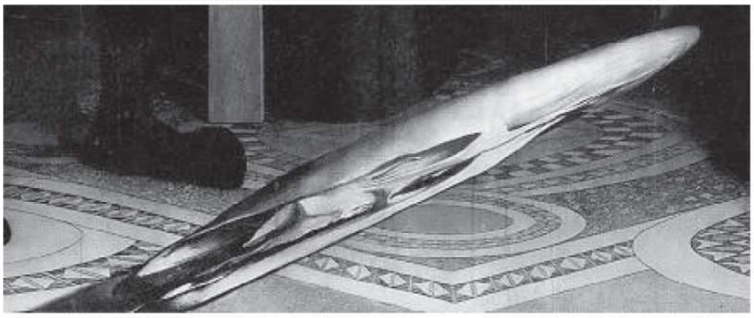


FIGURE 6.27 *The skull alone.*

FIGURE 6.28

```
>> skull2=imresize(imrotate(skull,-22,'bicubic'),[500,150],'bicubic');
```

```
>> imshow(skull2(200:350,:))
```

FIGURE 6.28 *The corrected skull.*

37 **Ch6-p.142**

CENGAGE Learning