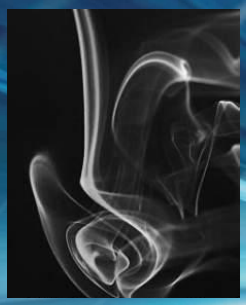
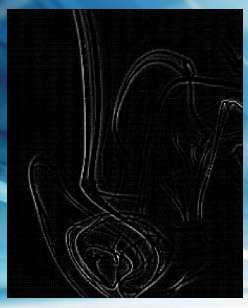
Introduction to Digital Image Processing with MATLAB® Asia Edition

McAndrew·Wang·Tseng





Chapter 5: Neighborhood Processing

CENGAGE Learning

5.1 Introduction

- Move a **mask**
  - ✓ A rectangle (usually with sides of odd length) or other shape over the given image

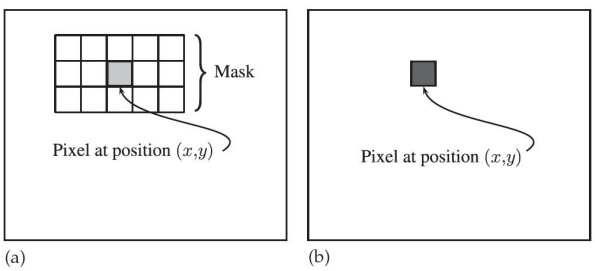


FIGURE 5.1 *Using a spatial mask on an image. (a) Original image. (b) Image after filtering.*

CENGAGE Learning

5.1 Introduction

- Mask values

$m(-1,-2)$	$m(-1,-1)$	$m(-1,0)$	$m(-1,1)$	$m(-1,2)$
$m(0,-2)$	$m(0,-1)$	$m(0,0)$	$m(0,1)$	$m(0,2)$
$m(1,-2)$	$m(1,-1)$	$m(1,0)$	$m(1,1)$	$m(1,2)$

5.1 Introduction

- Corresponding pixel values

$p(i-1,j-2)$	$p(i-1,j-1)$	$p(i-1,j)$	$p(i-1,j+1)$	$p(i-1,j+2)$
$p(i,j-2)$	$p(i,j-1)$	$p(i,j)$	$p(i,j+1)$	$p(i,j+2)$
$p(i+1,j-2)$	$p(i+1,j-1)$	$p(i+1,j)$	$p(i+1,j+1)$	$p(i+1,j+2)$

FIGURE 5.2

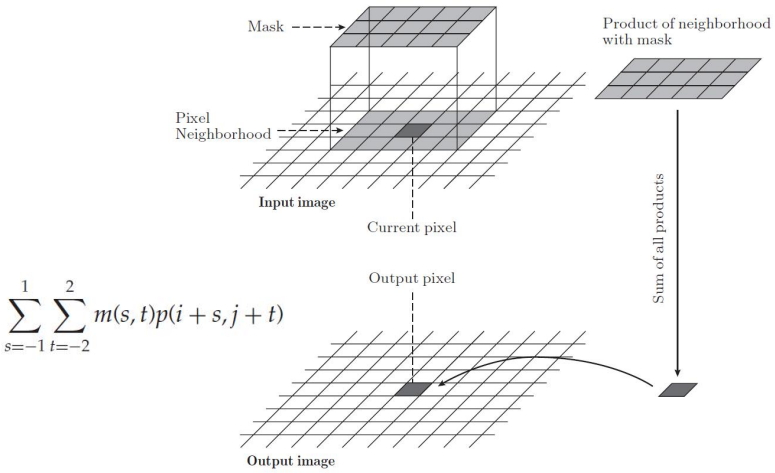


FIGURE 5.2 *Performing spatial filtering.*

Ch5-p.89

5

5.1 Introduction

- Allied to spatial filtering is spatial **convolution**
  - ✓ The filter must be rotated by 180° before multiplying and adding

$$\sum_{s=-1}^{1}\sum_{t=-2}^{2} m(-s,-t)p(i+s,j+t)$$

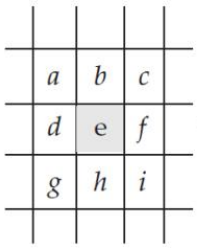
$$\sum_{s=-1}^{1}\sum_{t=-2}^{2} m(s,t)p(i-s+j-t)$$

Ch5-p.89

6

5.1 Introduction

- **EXAMPLE** One important linear filter is to use a 3×3 mask and take the average of all nine values within the mask



$$\longrightarrow \frac{1}{9}(a+b+c+d+e+f+g+h+i)$$

5.1 Introduction

```
>> x=uint8(10*magic(5))

x =

  170   240    10    80   150
  230    50    70   140   160
   40    60   130   200   220
  100   120   190   210    30
  110   180   250    20    90
```

```
>> mean2(x(1:3,2:4))

ans =

  108.8889
```

The result of filtering x with 3×3 averaging filter

```
111.1111  108.8889  128.8889
110.0000  130.0000  150.0000
131.1111  151.1111  148.8889
```

5.2 Notation

- It is convenient to describe a linear filter simply in terms of the coefficients of all the gray values of pixels within the mask
  - ✓ The averaging filter

$$\frac{1}{9}a+\frac{1}{9}b+\frac{1}{9}c+\frac{1}{9}d+\frac{1}{9}e+\frac{1}{9}f+\frac{1}{9}g+\frac{1}{9}h+\frac{1}{9}i$$

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

5.2 Notation

- ✓ **EXAMPLE** The filter

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

would operate on gray values as

	a	b	c
	d	e	f
	g	h	i

$$\longrightarrow a-2b+c-2d+4e-2f+g-2h+i$$

5.2.1 Edges of the Image

- What happens at the edge of the image, where the mask partly falls outside the image?

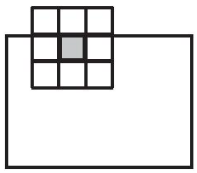
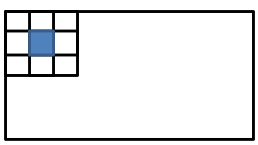


FIGURE 5.3 *A mask at the edge of an image.*

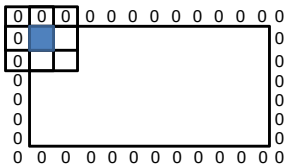
- There are a number of different approaches to dealing with this problem

5.2.1 Edges of the Image

- **Ignore the edges**

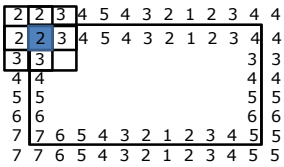


- **Pad with zeros**



5.2.1 Edges of the Image

- **Mirroring**



5.3 Filtering in MATLAB

- `filter2` function

```
filter2(filter,image,shape)
```

the result is a matrix of data type double!!

- `shape` is optional; it describes the method for dealing with the edges
  - ✓ `'same'` –**pad with zeros**
  - ✓ `'valid'` –**ignore the edges**

5.3 Filtering in MATLAB

```
>> a=ones(3,3)/9

a =

    0.1111    0.1111    0.1111
    0.1111    0.1111    0.1111
    0.1111    0.1111    0.1111

>> filter2(a,x,'same')

ans =

   76.6667   85.5556   65.5556   67.7778   58.8889
   87.7778  111.1111  108.8889  128.8889  105.5556
   66.6667  110.0000  130.0000  150.0000  106.6667
   67.7778  131.1111  151.1111  148.8889   85.5556
   56.6667  105.5556  107.7778   87.7778   38.8889
```


5.3 Filtering in MATLAB

```
>> filter2(a,x,'valid')

ans =

  111.1111  108.8889  128.8889
  110.0000  130.0000  150.0000
  131.1111  151.1111  148.8889
```

5.3 Filtering in MATLAB

- The result of `'same'` may also be obtained by padding with zeros and using `'valid'`:

```
>> x2=zeros(7,7);
>> x2(2:6,2:6)=x

x2 =

     0     0     0     0     0     0     0
     0   170   240    10    80   150     0
     0   230    50    70   140   160     0
     0    40    60   130   200   220     0
     0   100   120   190   210    30     0
     0   110   180   250    20    90     0
     0     0     0     0     0     0     0

>> filter2(a,x2,'valid')
```


5.3 Filtering in MATLAB

- `filter2(filter,image,'full')` returns a result larger than the original
- It does this by padding with zero and applying the filter at all places on and around the image where the mask intersects the image matrix

5.3 Filtering in MATLAB

```
>> filter2(a,x,'full')

ans =

   18.8889   45.5556   46.6667   36.6667   26.6667   25.5556   16.6667
   44.4444   76.6667   85.5556   65.5556   67.7778   58.8889   34.4444
   48.8889   87.7778  111.1111  108.8889  128.8889  105.5556   58.8889
   41.1111   66.6667  110.0000  130.0000  150.0000  106.6667   45.5556
   27.7778   67.7778  131.1111  151.1111  148.8889   85.5556   37.7778
   23.3333   56.6667  105.5556  107.7778   87.7778   38.8889   13.3333
   12.2222   32.2222   60.0000   50.0000   40.0000   12.2222   10.0000
```


5.3 Filtering in MATLAB

- `filter2` provides no mirroring option
- The mirroring approach can be realized by placing the following codes before `filter2(filter,image,'valid')`

```
m_x=[x(wr:-1:1,:); x; x(end:-1:end-(wr-1), :)];
m_x=[m_x(:, wc:-1:1), m_x, m_x(:, end:-1:end-(wc-1))];
```

5.3 Filtering in MATLAB

- Where matrix `x` is extended to `m_x`, `wr/wc` is defined as one half total column/row number of the mask (chopping the decimal)

```
>> filter2(a,m_x,'valid')
ans=
  185.5556  132.2222  102.2222   94.4444  135.5556
  136.6667  111.1111  108.8889  128.8889  164.4444
  107.7778  110.0000  130.0000  150.0000  152.2222
   95.5556  131.1111  151.1111  148.8889  123.3333
  124.4444  165.5556  157.7778  127.7778   74.4444
```


5.3 Filtering in MATLAB

- `fspecial` function
  - ✓ `h = fspecial(type, parameters)`

```
>> c=imread('cameraman.tif');
>> f1=fspecial('average');
>> cf1=filter2(f1,c);
```

```
>>imshow(uint8(cf1))
```

or

```
>>imshow(cf1/255)
```

FIGURE 5.4

FIGURE 5.4 *Average filtering. (a) Original image. (b) Average filtering. (c) Using a 9 × 9 filter. (d) Using a 25 × 25 filter. (e) Using a 25 × 25 filter with* ***Mirroring****.*

Ch5-p.97

CENGAGE Learning

23

5.4 Frequencies: Low- and High-Pass Filters

- Frequencies of an image are a measure of the amount by which gray values change with distance
  - ✓ **high-pass filter**

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

  - ✓ **low-pass filter**

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Ch5-p.98

CENGAGE Learning

24

5.4 Frequencies: Low- and High-Pass Filters

```
>> f=fspecial('laplacian')

f =

    0.1667    0.6667    0.1667
    0.6667   -3.3333    0.6667
    0.1667    0.6667    0.1667

>> cf=filter2(f,c);
>> imshow(cf/100)
>> f1=fspecial('log')

f1 =

    0.0448    0.0468    0.0564    0.0468    0.0448
    0.0468    0.3167    0.7146    0.3167    0.0468
    0.0564    0.7146   -4.9048    0.7146    0.0564
    0.0468    0.3167    0.7146    0.3167    0.0468
    0.0448    0.0468    0.0564    0.0468    0.0448

>> cf1=filter2(f1,c);
>> figure,imshow(cf1/100)
```


FIGURE 5.5

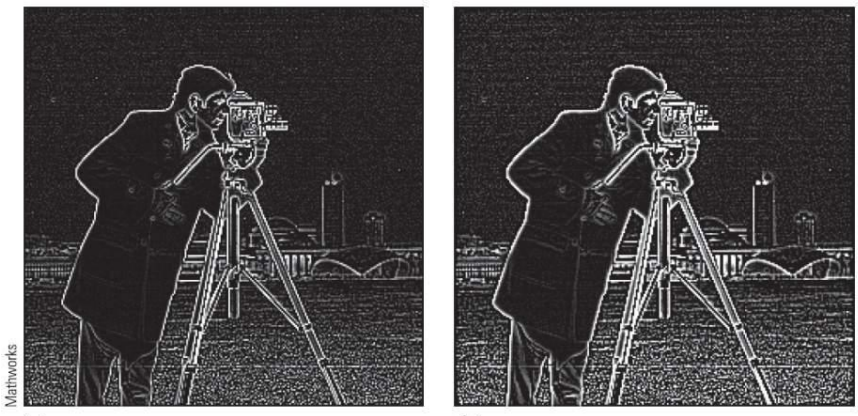


FIGURE 5.5 *High-pass filtering. (a) Laplacian filter. (b) Laplacian of Gaussian (log) filtering.*

5.4 Frequencies: Low- and High-Pass Filters

- **VALUES OUTSIDE THE RANGE 0–255**

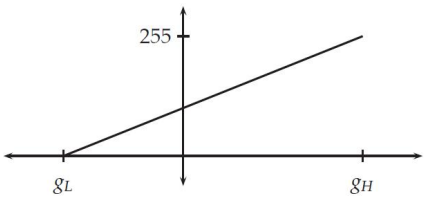
  ✓ **Make negative values positive**

  ✓ **Clip values**

$$y = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \le x \le 255 \\ 255 & \text{if } x > 255 \end{cases}$$

5.4 Frequencies: Low- and High-Pass Filters

✓ **0-255 Scaling transformation (uint8)**



$$y = 255\frac{x - g_L}{g_H - g_L}$$

5.4 Frequencies: Low- and High-Pass Filters

```
>> f2=[1 -2 1;-2 4 -2;1 -2 1];
>> cf2=filter2(f2,c);
```

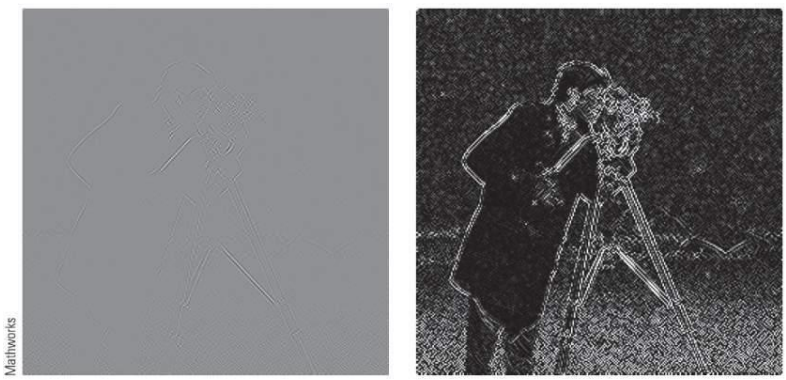
```
>> figure,imshow(mat2gray(cf2));
```

0-1 Scaling transformation (double)

```
>> maxcf2=max(cf2(:));
>> mincf2=min(cf2(:));
>> cf2g=(cf2-mincf2)/(maxcf2-mncf2);
```

```
>> figure,imshow(cf2/60)
```

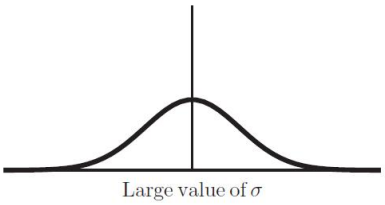

FIGURE 5.6



(a) (b)

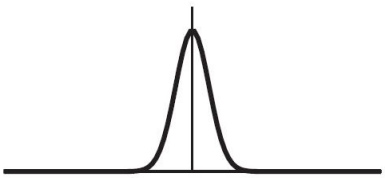
FIGURE 5.6 *Using a high-pass filter and displaying the result. (a) Using* `mat2gray`. *(b) Dividing by a constant.*

5.5 Gaussian Filters



Large value of σ

$$f(x) = e^{-\frac{x^2}{2\sigma^2}}$$

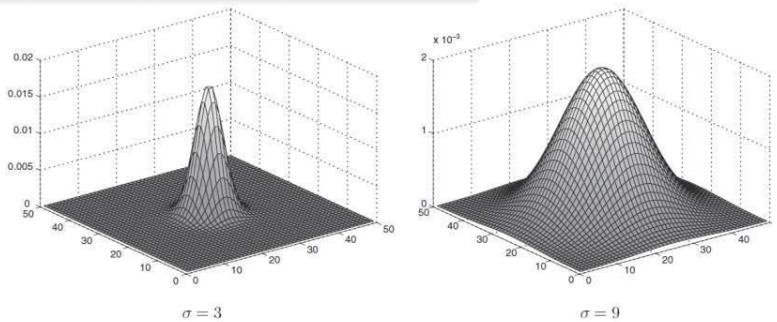


Small value of σ

FIGURE 5.7 *One-dimensional Gaussians.*

FIGURE 5.8

```
>> a=50;s=3;
>> g=fspecial('gaussian',[a a],s);
>> surf(1:a,1:a,g)
>> s=9;
>> g2=fspecial('gaussian',[a a],s);
>> figure,surf(1:a,1:a,g2)
```



$\sigma = 3$ $\sigma = 9$

FIGURE 5.8 *Two-dimensional Gaussians.*

5.5 Gaussian Filters

```
>> g1=fspecial('gaussian',[5,5]);
>> g2=fspecial('gaussian',[5,5],2);
>> g3=fspecial('gaussian',[11,11],1);
>> g4=fspecial('gaussian',[11,11],5);
```

```
>> imshow(filter2(g1,c)/256)
>> figure,imshow(filter2(g2,c)/256)
>> figure,imshow(filter2(g3,c)/256)
>> figure,imshow(filter2(g4,c)/256)
```


FIGURE 5.9

FIGURE 5.9 *Effects of different Gaussian filters on an image.*

5.6 Edge Sharpening

- **5.6.1 Unsharp Masking**

```
>> f=fspecial('average');
>> xf=filter2(f,x);
>> xu=double(x)-xf/1.5
>> imshow(xu/70)
```

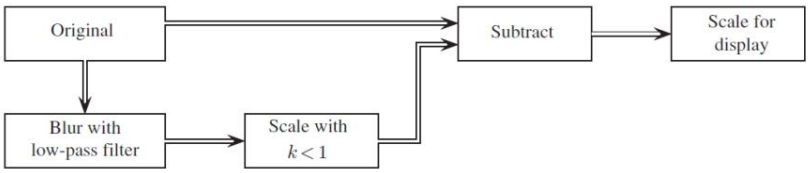


FIGURE 5.10 *Schema for unsharp masking.*

FIGURE 5.11

(a) (b)

FIGURE 5.11 *An example of unsharp masking. (a) Original image. (b) The image after unsharp masking.*

FIGURE 5.12

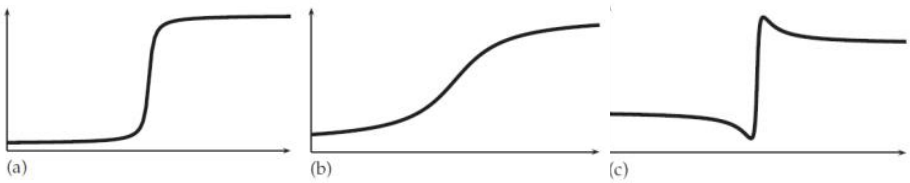


FIGURE 5.12 *Unsharp masking. (a) Pixel values over an edge. (b) The edge blurred. (c) (a) − k(b).*

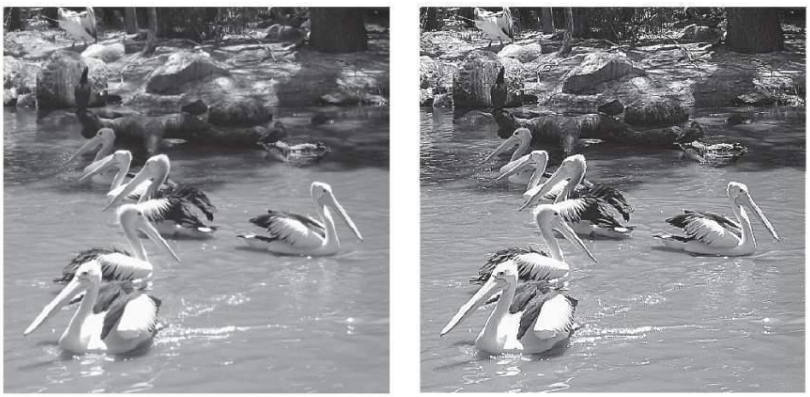
5.6.1 Unsharp Masking

- The `unsharp` option of `fspecial` produces such filters

$$\frac{1}{\alpha+1}\begin{bmatrix} -\alpha & \alpha-1 & -\alpha \\ \alpha-1 & \alpha+5 & \alpha-1 \\ -\alpha & \alpha-1 & -\alpha \end{bmatrix}$$

```
>> p=imread('pelicans.tif');
>> u=fspecial('unsharp',0.5);  α = 0.5
>> pu=filter2(u,p);
>> imshow(p),figure,imshow(pu/255)
```

FIGURE 5.13



(a) (b)

FIGURE 5.13 *Edge enhancement with unsharp masking. (a) The original. (b) After unsharp masking.*

5.6.2 High-Boost Filtering

- Allied to unsharp masking filters are the **high-boost** filters

$$\text{high boost} = A(\text{original}) - (\text{low pass})$$

  - ✓ where A is an amplification factor
  - ✓ If A = 1, then the high-boost filter becomes an ordinary high-pass filter

5.6.2 High-Boost Filtering

```
>> id=[0 0 0;0 1 0;0 0 0];
>> f=fspecial('average');
>> hb1=3*id-2*f

hb1 =

   -0.2222   -0.2222   -0.2222
   -0.2222    2.7778   -0.2222
   -0.2222   -0.2222   -0.2222

>> hb2=1.25*id-0.25*f

hb2 =

   -0.0278   -0.0278   -0.0278
   -0.0278    1.2222   -0.0278
   -0.0278   -0.0278   -0.0278
```


FIGURE 5.14

```
>> x1=filter2(hb1, x);
>> imshow(x1/255)
```

```
>> x2=filter2(hb2, x);
>> imshow(x2/255)
```

(a) (b)

FIGURE 5.14 *High-boost filtering. (a) High-boost filtering with* `hb1`*. (b) High-boost filtering with* `hb2`*.*

5.7 Nonlinear Filters

- Maximum filter

```
>> cmax=nlfilter(c,[3,3],'max(x(:))');
```

- Minimum filter

```
>> cmin=nlfilter(c,[3,3],'min(x(:))');
```


FIGURE 5.15

FIGURE 5.15 *Using nonlinear filters. (a) Using a maximum filter. (b) Using a minimum filter.*

5.8 Region of Interest Processing

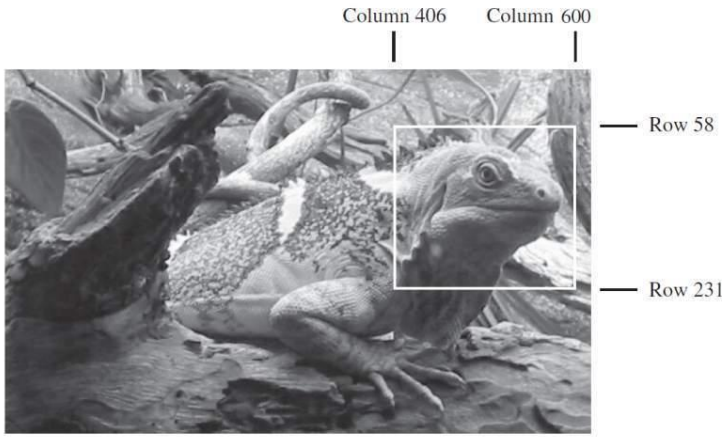


FIGURE 5.16 *An image with a ROI.*

Ch5-p.115

CENGAGE Learning

5.8.1 Regions of Interest in MATLAB

```
>> ig=imread('iguana.tif');
```

```
>> roi=roipoly(ig,[406 600 600 406],[58 58 231 231]);
```

FIGURE 5.17 *The mask corresponding to the ROI defined in Figure 5.16.*

Ch5-p.115-116

CENGAGE Learning

5.8.1 Regions of Interest in MATLAB

```
>> roi=roipoly(ig);
```

- This will bring up the iguana image (if it isn't shown already). Vertices of the ROI can be selected with the mouse

5.8.2 Region of Interest Filtering

```
>> a=fspecial('average',[15,15]);
>> iga=roifilt2(a,ig,roi);
>> imshow(iga)
>> u=fspecial('unsharp');
>> igu=roifilt2(u,ig,roi);
>> figure,imshow(igu)
>> l=fspecial('log');
>> igl=roifilt2(l,ig,roi);
>> figure,imshow(igl)
```

FIGURE 5.18

iga: Average filtering

igu: Unsharp masking

igl: Laplacian of Gaussian

FIGURE 5.18 *Examples of the use of* `roifilt2`.

CENGAGE Learning