$$w_{mn} = \begin{bmatrix} 0.0025 & 0.0125 & 0.0200 & 0.0125 & 0.0025 \\ 0.0125 & 0.0625 & 0.1000 & 0.0625 & 0.0125 \\ 0.0200 & 0.1000 & 0.1600 & 0.1000 & 0.0200 \\ 0.0125 & 0.0625 & 0.1000 & 0.0625 & 0.0125 \\ 0.0025 & 0.0125 & 0.0200 & 0.0125 & 0.0025 \end{bmatrix} . \qquad (6-32)$$

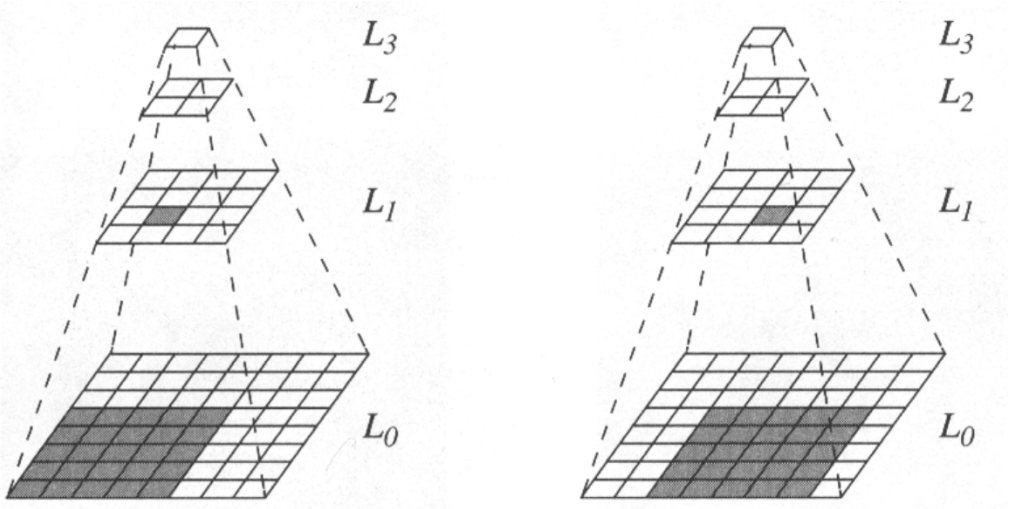


FIGURE 6-28. Gaussian pyramid construction with the 5 x 5 pixel weighting function of Eq. (6 – 32). On the left, the weighted average of 25 pixels in level 0 gives the shaded pixel in level 1. On the right, the weighting function is moved two pixels along a row and the weighted average of the corresponding pixels gives the next pixel in level 1. This process is repeated across the image in level 0, and then performed on level 1 to calculate level 2, and so forth. In this way, the linear size of the image is reduced by two from level to level. This is equivalent to a full convolution of level 0 with the weighting function, followed by a down-sample, but avoids unnecessary calculations. The border pixels require special attention as discussed earlier.

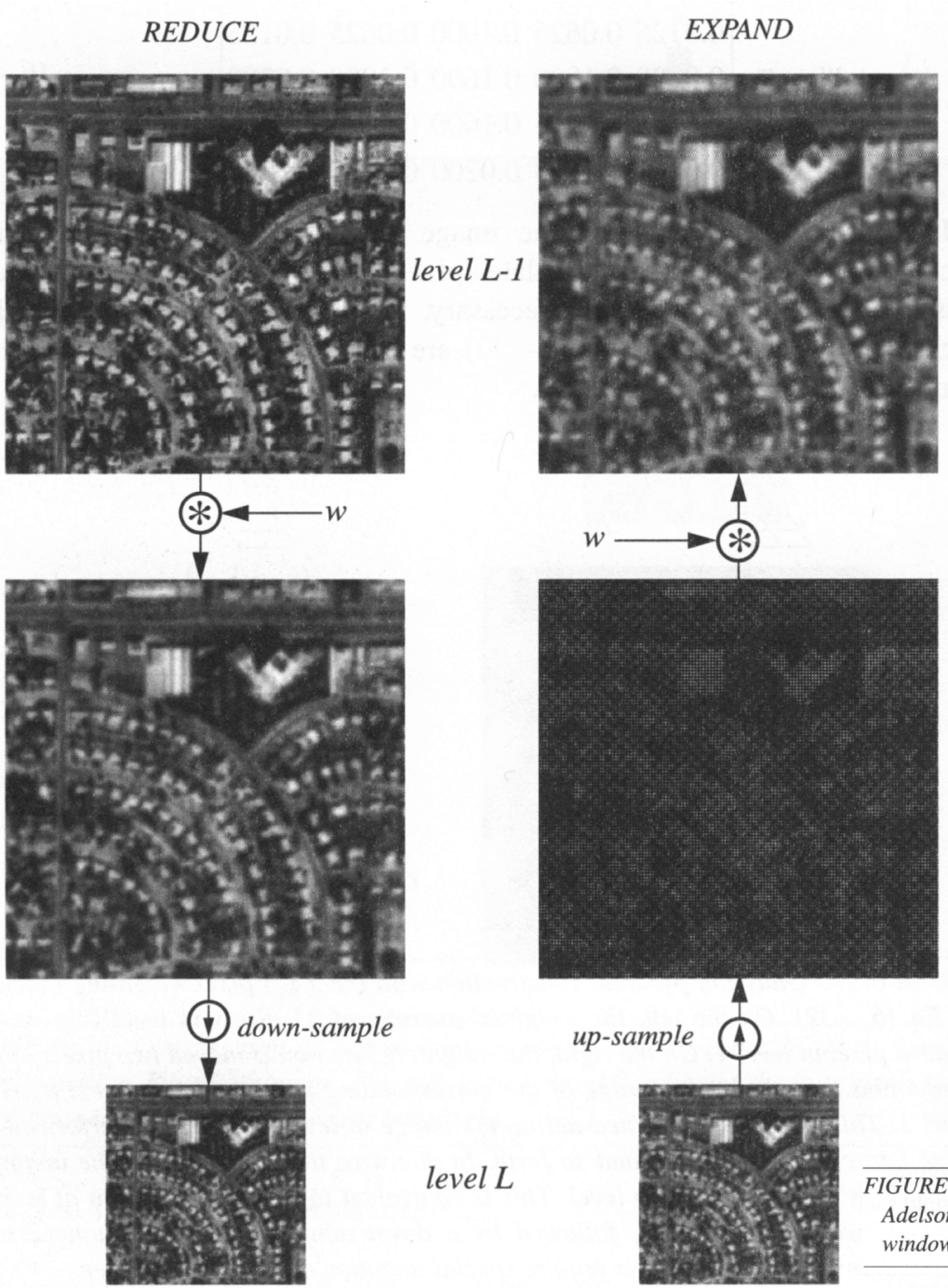


FIGURE 6-29. The REDUCE and EXPAND procedures as defined in (Burt and Adelson, 1983). Any spatial filter can be used, but these examples use the Gaussian window of Eq. (6 – 32).

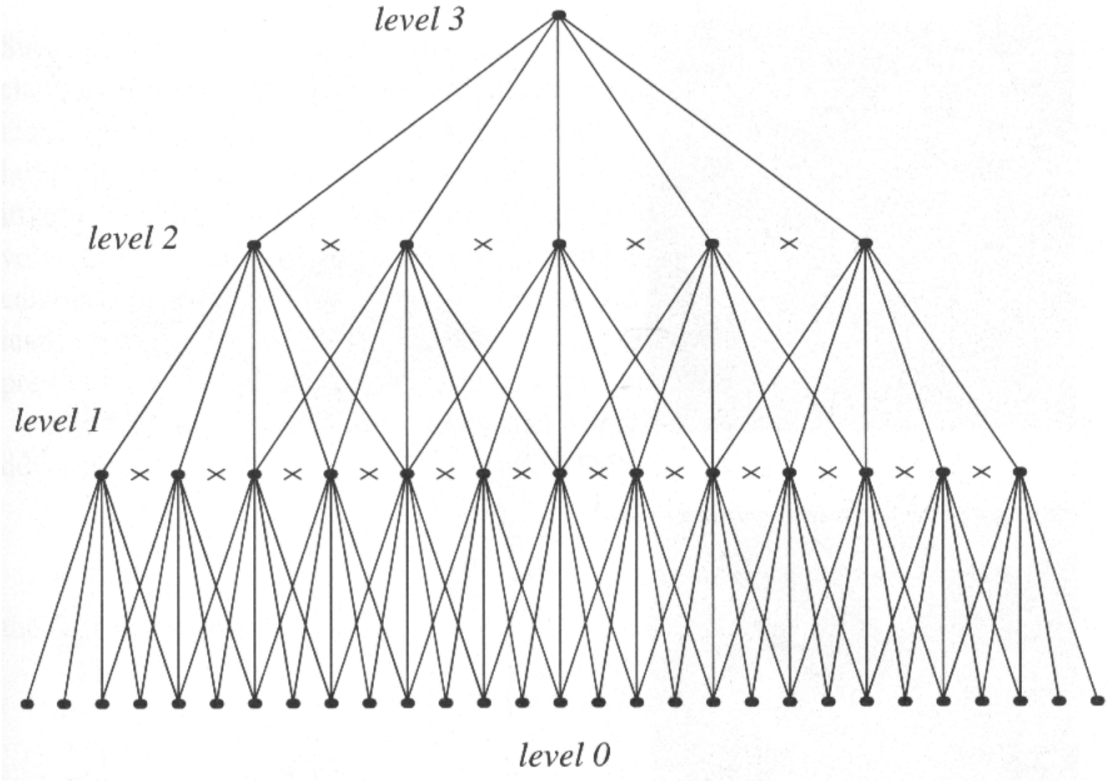


FIGURE 6-32. The links between a pixel in level 3 of the Gaussian pyramid and pixels at lower levels. The x-marks indicate pixels that are not included at each level because of downsampling. The effective convolution window size at the original level 0 for a pyramid that reduces by two at each level L is $4(2^L - 1) + 1$ *(Burt, 1981).*

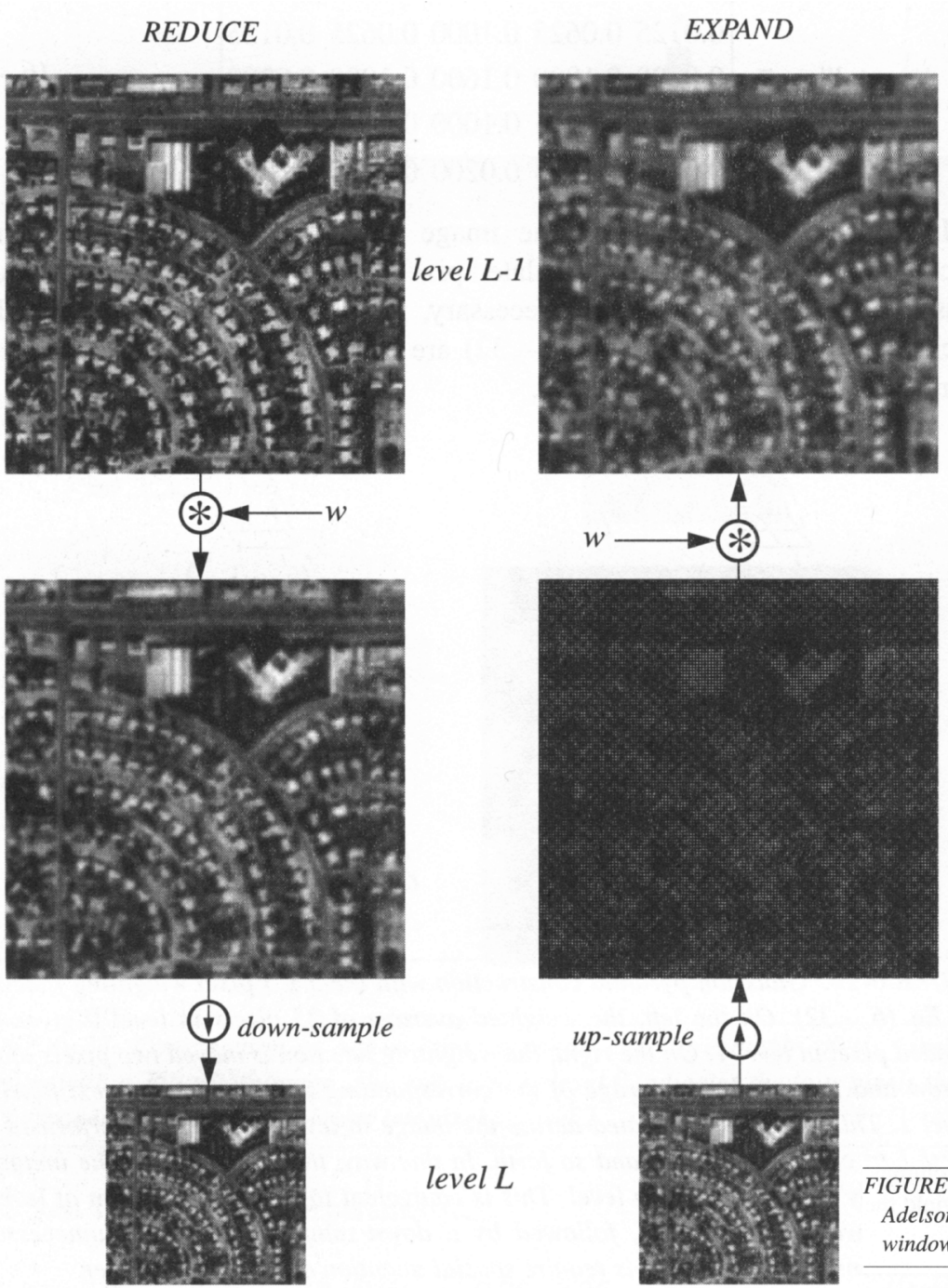


FIGURE 6-29. The REDUCE and EXPAND procedures as defined in (Burt and Adelson, 1983). Any spatial filter can be used, but these examples use the Gaussian window of Eq. (6 – 32).

0.0025	0.0125	0.0200	0.0125	0.0025		
0.0125	0.0625	0.1000	0.0625	0.0125		
0.0200	0.1000	0.1600	0.1000	0.0200		
0.0125	0.0625	0.1000	0.0625	0.0125		
0.0025	0.0125	0.0200	0.0125	0.0025		

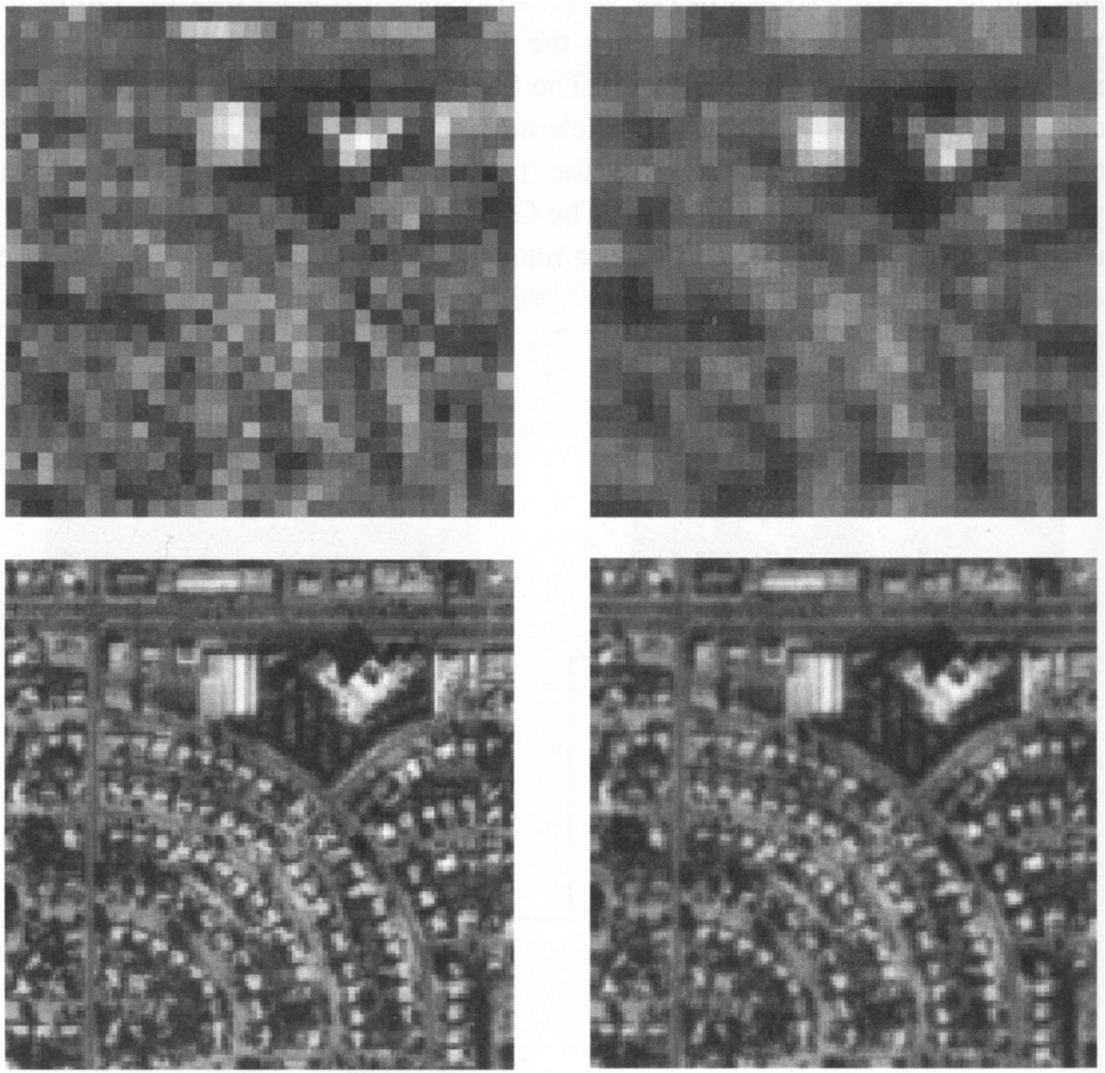
0.0025*4+0.02*4+0.16=0.25

0.0025	0.0125	0.0200	0.0125	0.0025		
0.0125	0.0625	0.1000	0.0625	0.0125		
0.0200	0.1000	0.1600	0.1000	0.0200		
0.0125	0.0625	0.1000	0.0625	0.0125		
0.0025	0.0125	0.0200	0.0125	0.0025		

0.0125*4+0.1*2=0.25

	0.0025	0.0125	0.0200	0.0125	0.0025	
	0.0125	0.0625	0.1000	0.0625	0.0125	
	0.0200	0.1000	0.1600	0.1000	0.0200	
	0.0125	0.0625	0.1000	0.0625	0.0125	
	0.0025	0.0125	0.0200	0.0125	0.0025	

0.0625*4=0.25



box pyramid *Gaussian pyramid*

FIGURE 6-31. Level 3 and level 1 images compared for the box and Gaussian pyramids. The level 3 images are magnified to the level 1 scale by pixel replication.

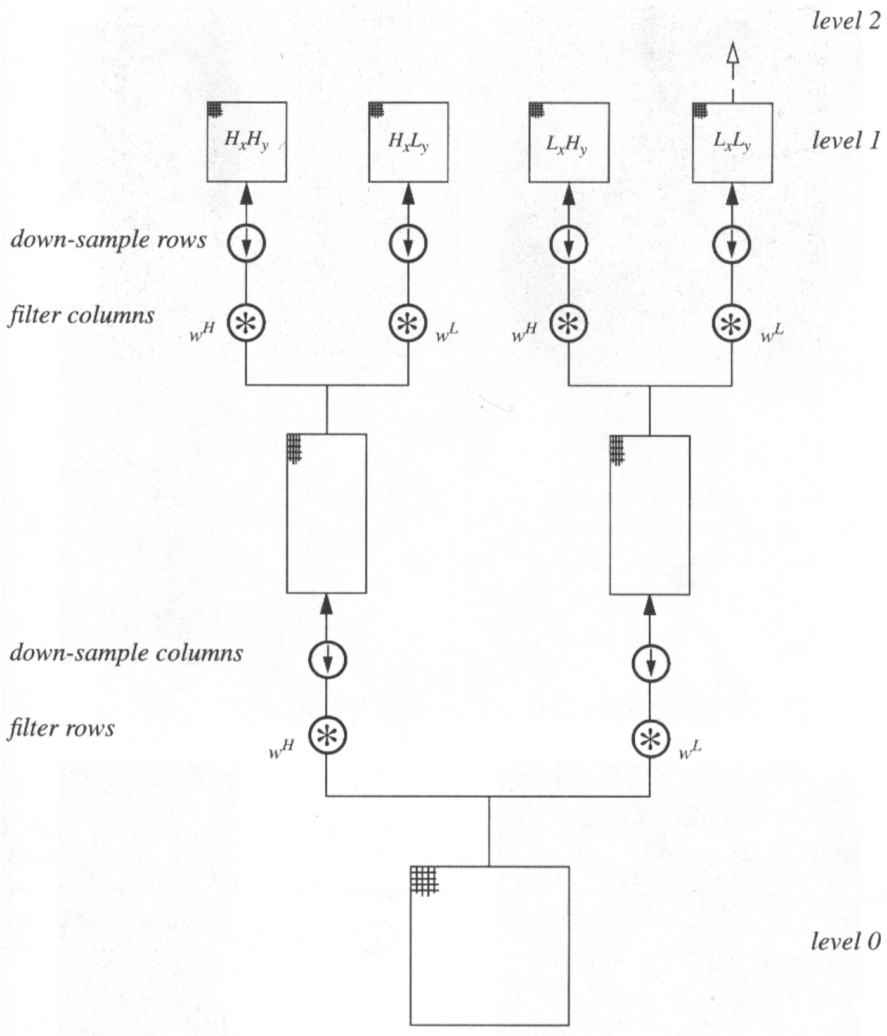
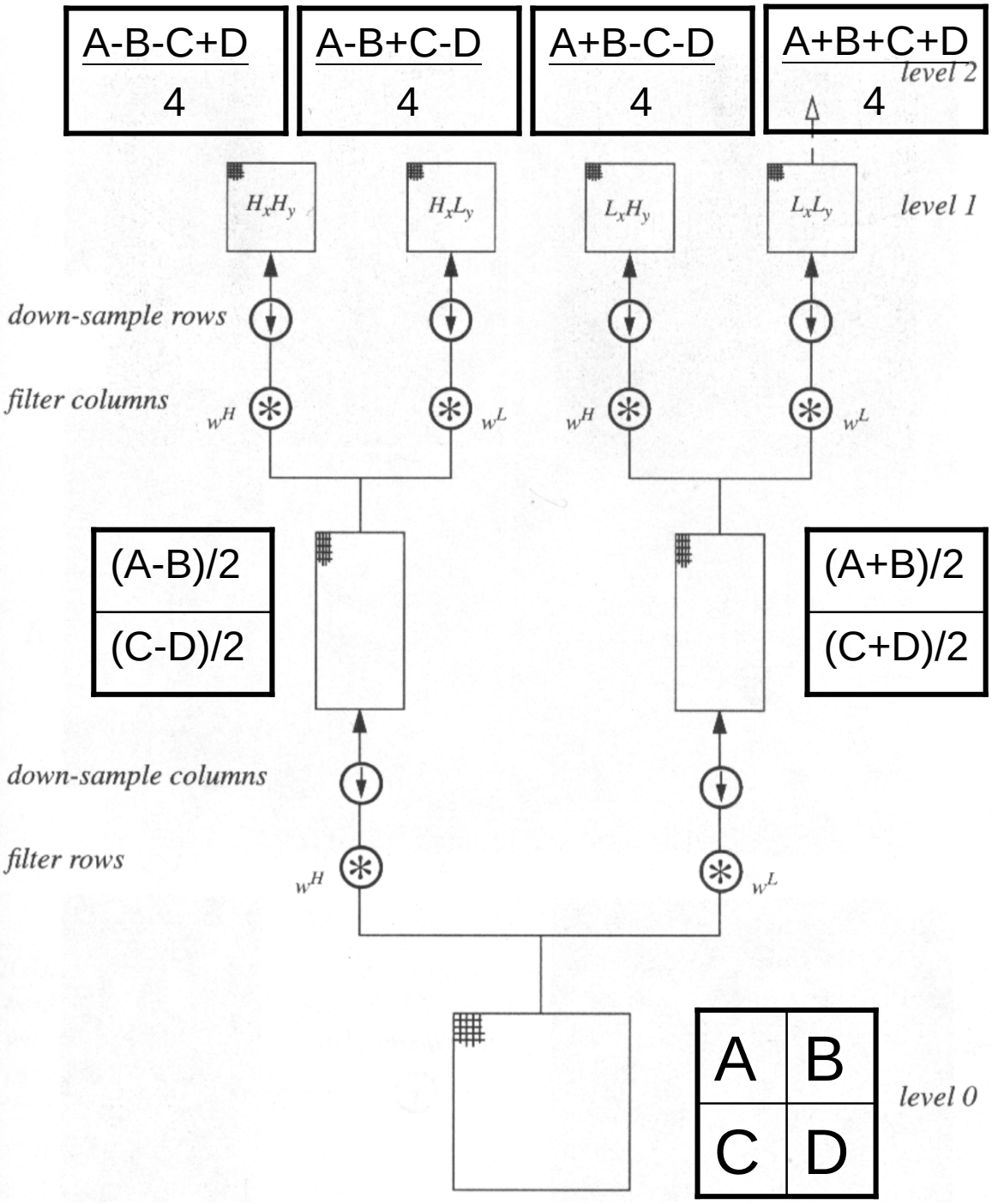


FIGURE 6-42. Wavelet decomposition from one pyramid level to the next. The letter L means low-pass and H means high-pass. Thus, L_xL_y means a low-pass filtered version that has been filtered in x (along rows) and y (along columns) and down-sampled by two. Likewise, H_xL_y means the image from the previous level is first high-pass filtered and down-sampled in x and then low-pass filtered and down-sampled in y. The L_xL_y component at each level is used as input to the calculations for the next level. This processing architecture is known as a filter bank.



Haar Wavelet

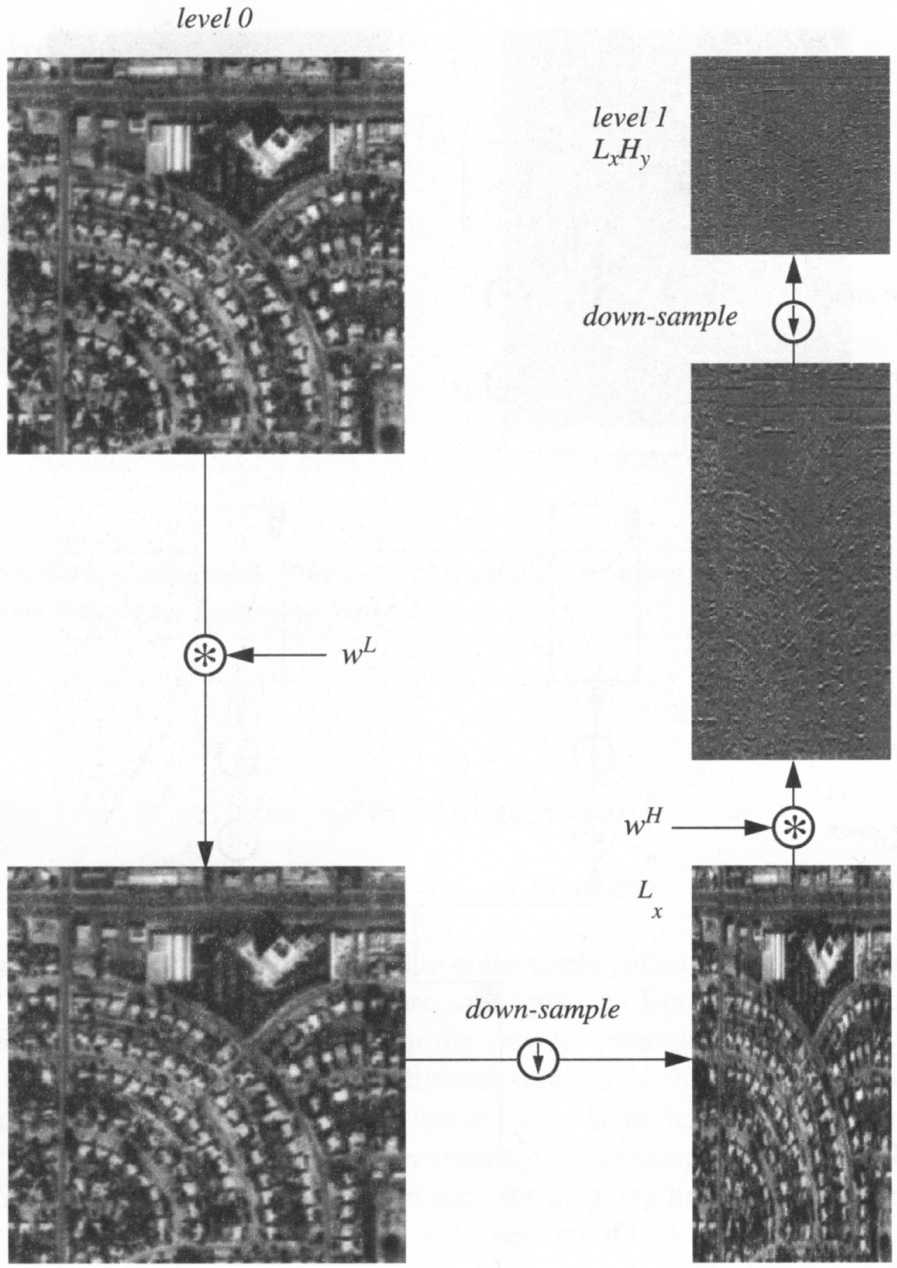


FIGURE 6-43. Calculation of one of the four wavelet components in level 1. The filtering and down-sampling combinations are similar to the REDUCE operation described earlier, except that here they are done one direction at a time.

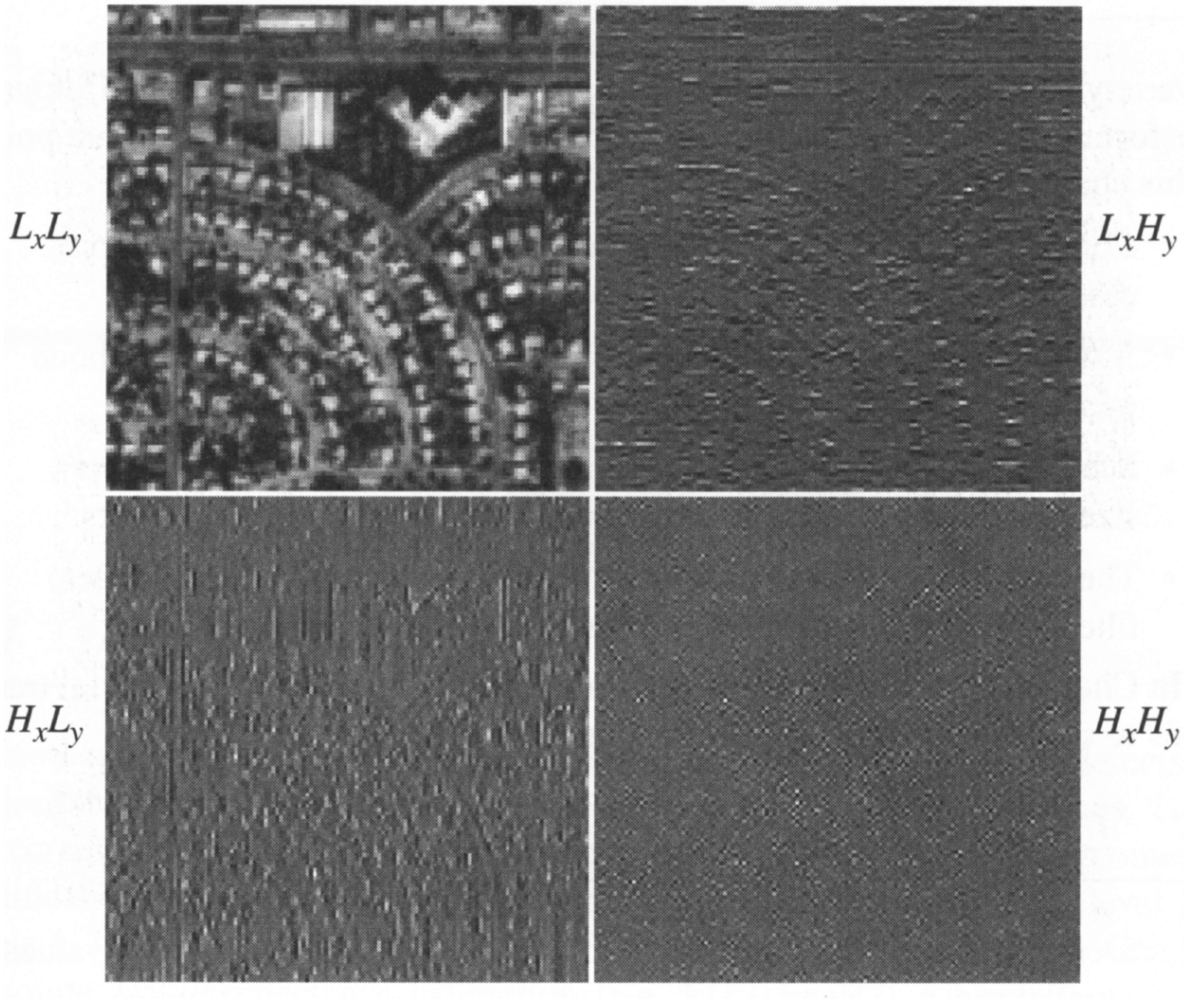
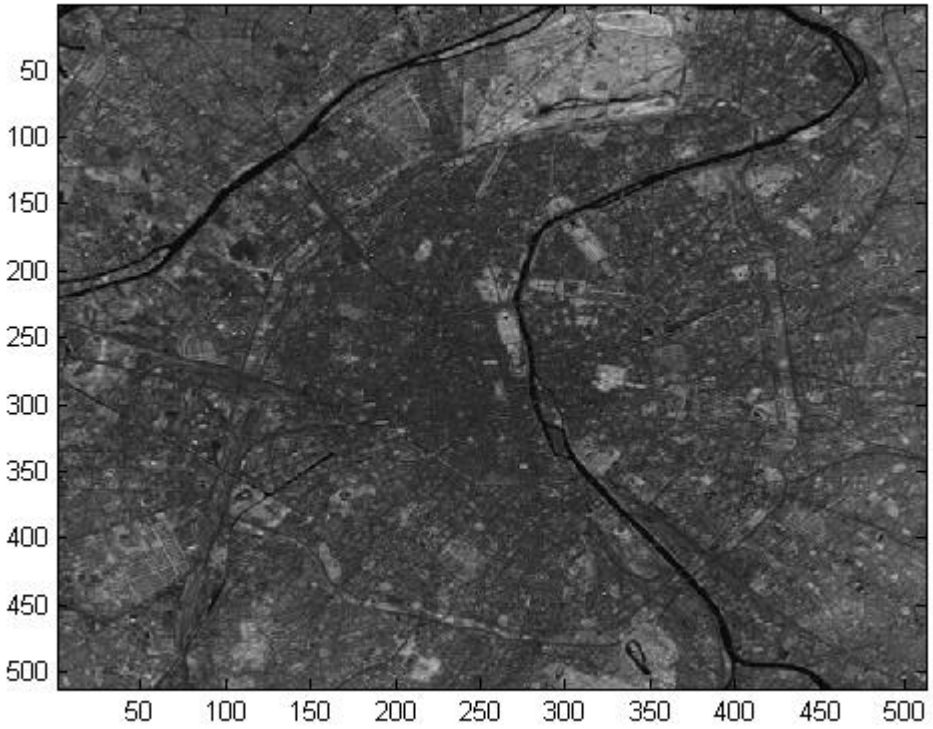


FIGURE 6-44. The four wavelet transform components of level 1 produced by the filter bank in Fig. 6-42. To create level 2, the upper left image, L_xL_y, is processed by the wavelet transform to produce a similar set of four images that are smaller by a factor of two in each direction.

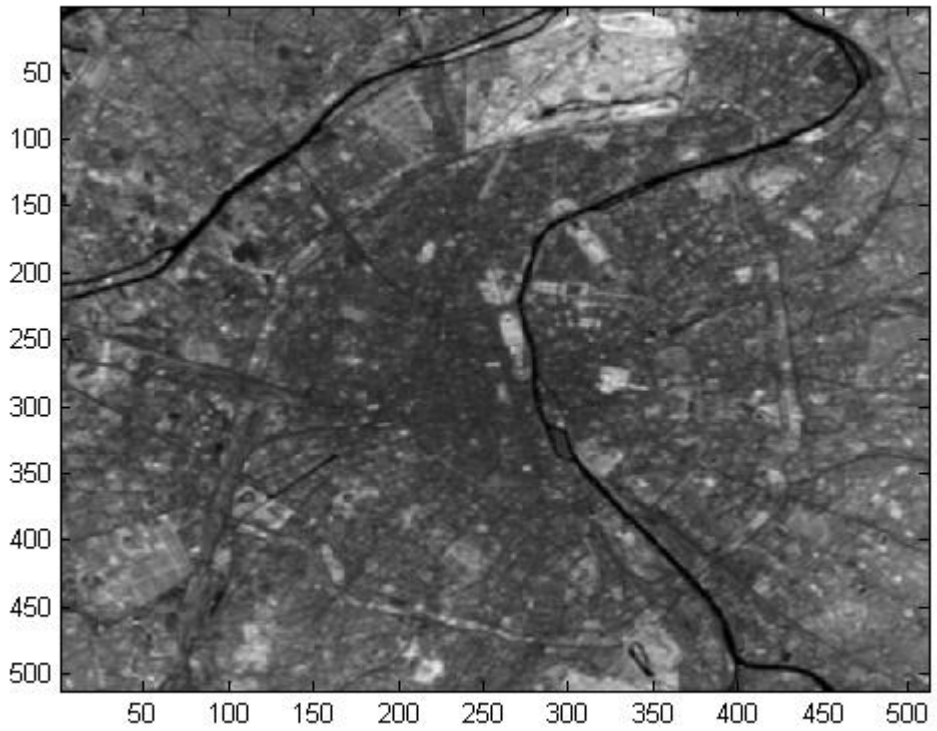
Gaussian pyramid

```
dir *.lan
fid=fopen('paris.lan','r');
fseek(fid,128,-1)
A=fread(fid,[512*7,512],'uint8');
B=A(512*3+1:512*4,:);
Imagesc (B)
```



Gaussian pyramid

```
w=[0.05 0.25 0.4 0.25 0.05];
W=w'*w;
C=ones(size(B));
D=conv2(B,W,'same')./conv2(C,W,'same');
F=D(1:2:end,1:2:end);
```



Gaussian expand

```
H=zeros(size(F)*2);
H(1:2:end,1:2:end)=F;
K=4*conv2(H,W,'same')./conv2(C,W,'same');
```