A good approach for filtering pepper noise is to use a contraharmonic filte with a positive value of Q. Figure 5.5(c) was generated using the statement

```
>> fp = spfilt(gp, 'chmean', 3, 3, 1.5);
```

Similarly, salt noise can be filtered using a contraharmonic filter with a negative value of Q:

```
>> fs = spfilt(gs, 'chmean', 3, 3, -1.5);
```

Figure 5.5(d) shows the result. Similar results can be obtained using max and min filters. For example, the images in Figs. 5.5(e) and (f) were generated from Figs. 5.5(a) and (b), respectively, with the following commands:

```
>> fpmax = spfilt(gp, 'max', 3, 3);
>> fsmin = spfilt(gs, 'min', 3, 3);
```

Other solutions using `spfilt` are implemented in a similar manner. ▪

5.3.2 Adaptive Spatial Filters

The filters discussed in the previous section are applied to an image without regard for how image characteristics vary from one location to another. In some applications, results can be improved by using filters capable of adapting their behavior depending on the characteristics of the image in the area being filtered. As an illustration of how to implement adaptive spatial filters in MATLAB, we consider in this section an adaptive median filter. As before, S_{xy} denotes a subimage centered at location (x, y) in the image being processed. The algorithm, which is explained in detail in Gonzalez and Woods [2002], is as follows: Let

$$
\begin{aligned}
z_{\min} &= \text{minimum intensity value in } S_{xy}\\
z_{\max} &= \text{maximum intensity value in } S_{xy}\\
z_{\text{med}} &= \text{median of the intensity values in } S_{xy}\\
z_{xy} &= \text{intensity value at coordinates } (x, y)
\end{aligned}
$$

The adaptive median filtering algorithm works in two levels, denoted level A and level B:

```
Level A:   If $z_{\min} < z_{\text{med}} < z_{\max}$, go to level B
           Else increase the window size
           If window size $\le S_{\max}$, repeat level A
           Else output $z_{\text{med}}$

Level B:   If $z_{\min} < z_{xy} < z_{\max}$, output $z_{xy}$
           Else output $z_{\text{med}}$
```

where $S_{\max}$ denotes the maximum allowed size of the adaptive filter window. Another option in the last step in Level A is to output z_{xy} instead of the median. This produces a slightly less blurred result but can fail to detect salt (pepper)

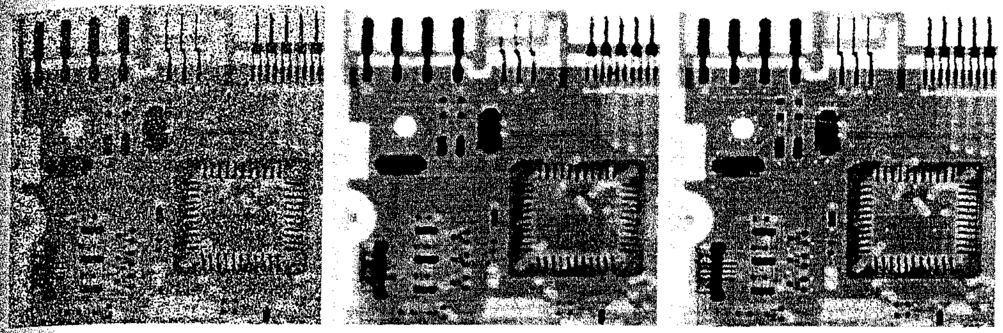

FIGURE 5.6 (a) Image corrupted by salt-and-pepper noise with density 0.25. (b) Result obtained using a median filter of size 7 × 7. (c) Result obtained using adaptive median filtering with $S_{\max} = 7$.

noise embedded in a constant background having the same value as pepper (salt) noise.

An M-function that implements this algorithm, which we call `adpmedian`, is included in Appendix C. The syntax is

```
f = adpmedian(g, Smax)
```

`adpmedian`

where `g` is the image to be filtered, and, as defined above, `Smax` is the maximum allowed size of the adaptive filter window.

EXAMPLE 5.6: Adaptive median filtering.

▪ Figure 5.6(a) shows the circuit board image, `f`, corrupted by salt-and-pepper noise, generated using the command

```
>> g = imnoise(f, 'salt & pepper', .25);
```

and Fig. 5.6(b) shows the result obtained using the command (see Section 3.5.2 regarding the use of `medfilt2`):

```
>> f1 = medfilt2(g, [7 7], 'symmetric');
```

This image is reasonably free of noise, but it is quite blurred and distorted (e.g., see the connector fingers in the top middle of the image). On the other hand, the command

```
>> f2 = adpmedian(g, 7);
```

yielded the image in Fig. 5.6(c), which is also reasonably free of noise, but is considerably less blurred and distorted than Fig. 5.6(b). ▪