

sets of samples. In a further aspect of the invention, its adaptive nature is obtained by generating a control signal from a spatially corresponding image difference between the images. The median-filtering of the samples then responds to the control signal.

One advantage of the hybrid median filter is due to its adaptive nature, which allows the filter to perform better than the standard median filter on fast-moving picture information of small spatial extent.

Several other advantages derive from inclusion of an IIR (recursive) term in the window of the median filter:

- (a) A high level of grain-suppression is achieved employing a smaller number of filter taps (frame stores), as compared with other median filter configurations. In stationary grain-only areas of the image the output of the median filter will generally be the IIR term, and hence in these circumstances the algorithm produces grain-suppression equivalent to motion-adaptive IIR algorithms;
- (b) Improved grain-suppression is achieved around areas of motion, as compared with an adaptive IIR filter; and
- (c) Sharp moving-picture information is preserved compared with finite (FIR) and infinite (IIR) response algorithms. The frequency response of the median filter is exact in the presence of a moving edge, since the output value is exactly equal to one of the local input values (although there may be a small phase shift).

#### BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be described with reference to the Figure, which shows a block diagram of an adaptive, hybrid median filter according to the invention.

#### DESCRIPTION OF THE PREFERRED EMBODIMENTS

Before proceeding to a hardware implementation of the filter according to the invention, it is helpful to examine the analytical basis of the filter. In particular, the filtered sequence for a median filter, designed according to the invention with a three pixel window, is

$$y(n) = \text{median} \quad (1)$$

where  $x(n)$  is the  $(i,j)$ th input pixel value of the current frame,  $x(n-1)$  is the  $(i,j)$ th pixel value of the previous frame,  $y(n)$  is the  $(i,j)$ th output, filtered pixel value in the current frame, and  $y'(n)$  is generated using a recursive filter, in particular

$$y'(n) = (1-a)x(n) + ay'(n-1) \quad (2)$$

In this specific example, the time constant "a" of the recursive filter is fixed and has no dependence on a motion-detection signal. However, in certain circumstances there may be an advantage in making "1" variable, that is, motion-adaptive. A value of "a" = 0.75 appears to provide a level of grain-suppression equivalent to the optimum IIR algorithms. In the preferred embodiment, a first-order IIR filter has been chosen, although a higher order filter could be used for higher performance at the cost of extra frame storage.

The algorithm also has an adaptive component, based on an interframe difference signal calculated for each pixel position,

$$k(n) = |x(n) - x(n-1)| \quad (3)$$

with  $x(n)$  and  $x(n-1)$  defined as before.

If  $k(n)$  has a value less than or equal to some predetermined threshold, T, then the algorithm is used in the standard form (1) described above. However, if  $k(n) > T$ , then impulsive picture information is assumed and hence:

$$y(n) = x(n) \quad (4)$$

Two frame stores are required to implement the filter algorithm in this form; the first to store pixel values of the delayed input frame for equations (1) and (3), and the second to implement the temporal recursive filter (2).

In the Figure, the input signal that is being filtered is applied to an input terminal 10, which is connected to a sample producer 20. The sample producer 20 samples the input signal and produces a current sample  $x(n)$  for a current frame of image signals. The current sample is coupled to an adaptive component generator 30, a sample sorter 40, a framestore 50, and a recursive component generator 60. The framestore 50 delays its input signal by a full image frame, thereby producing at its output a delayed sample  $x(n-1)$  for a preceding frame; sample  $x(n-1)$  thus spatially corresponds to the current sample  $x(n)$  in the current frame. The delayed sample  $x(n-1)$  is coupled to other inputs of the sample sorter 40 and the adaptive component generator 30. Equation (3) is implemented by the adaptive component generator 30, that is, an interframe difference between the current sample  $x(n)$  and the previous frame sample  $x(n-1)$  is obtained. Equation (2) is implemented by the recursive component generator 60 in combination with a further framestore 70, that is, a temporal smoothed value, proportioned by the value "a", is obtained for the input signal  $x(n)$ . While the value "a" that is input on a terminal 62 to the generator 60 is heretofore described as a constant (e.g., "a" = 0.25), it may be made adjustable for further motion-adaptive results. A feedback loop 64 through the framestore 70, which delays the output signal  $y'(n)$  by a full frame, and the current sample value  $x(n)$ , provide the necessary inputs for the recursive function.

The three values within the window of the median function— $x(n)$ ,  $x(n-1)$ , and  $y'(n)$ —are applied to inputs of the sample sorter 40. A typical sorter would include an array of comparators (not shown) which sort the input samples by magnitude. The sample sorter produces at respective output terminals 42a, 42b, 42c an ordered list of values of the input samples. For example, a sample having the greatest magnitude value is produced at topmost terminal 42a, a sample having the least magnitude value is produced at bottom-most terminal 42c, and the middle value is produced on the middle terminal 42b. A median selector 80 selects the middle sample value as the output  $y(n)$  of the median filter. The median selector 80 may be a multiplexing circuit having the middle sample output from the sorter 40 coupled to the output terminal of the multiplexing circuit. Thus, it can be seen that equation (1) is implemented by the combined operation of the sample sorter 40 and the median selector 80.

The filter is said to be hybrid because it uses the recursive output  $y'(n)$  in its input sample sequence. While described as a "sample value", this value  $y'(n)$  is not so much a distinct sample in the input sequence as a reevaluated, and smoothed, estimate of the current sample. As