FPGA IMPLEMENTATION OF ADAPTIVE TEMPORAL KALMAN FILTER FOR REAL TIME VIDEO FILTERING

March 15, 1999

Robert D. Turney[+], *Ali M. Reza*[*], *and Justin G. R. Delva*[*]

[+] CORE Solutions Group, Xilinx
San Jose, CA 95124-3450, USA

[*] Department of Electrical Engineering and Computer Science, UWM
Milwaukee, Wisconsin 53201-0784, USA

ABSTRACT

Filtering noise in real-time image sequences is required in some applications like medical imaging. The optimum approach in this case is in the form of adaptive 3-D spatial-temporal filter, which is generally very complex and prohibitive for real-time implementation. Independent processing of the image sequences, in spatial and temporal domains can resolve some of these implementation difficulties. Some of the existing spatial filters can easily be modified for real-time implementation. Adaptive temporal filters, however, are more involved. In this paper, an adaptive temporal filter is proposed that lend itself to hardware implementation for real-time temporal processing of image sequences. The proposed algorithm is based on adaptive Kalman filtering which is relatively simple and effective in its performance. Adaptation in this case is with respect to motion in the image sequence as well as variation of noise statistics. An efficient hardware implementation of this algorithm, based on FPGA technology, is proposed.

1. INTRODUCTION

The problem that is addressed in this study is the degradation of image sequences due to noise that is usually incurred during image acquisition process. Design of the optimum noise filter can only be based on the assumption that the image sequence and noise are stationary and their statistics are known. Statistics of image sequences and noise can be estimated if these signals are really stationary. In practice, however, this assumption is not valid and only adaptive algorithm can be utilized. General adaptive 3-D spatial-temporal filters are very complex and prohibitive for real-time implementation[1]. A more practical approach would be to process each frame independently by using an adaptive spatial filter and to filter each pixel in time domain by using an adaptive temporal algorithm. Some of the existing adaptive and/or robust (nonlinear) spatial filters can easily be modified for real-time implementation. Adaptive temporal filters, however, are more involved and require more hardware resources. In this case, application of a simple low-pass filter results into a peculiar lagging or ghosting effect that is due to the filtering of high frequency components (motion) in the temporal domain. Adaptation in time therefore not only requires estimation of the noise statistics but also a means to control the level of filtering when there is a motion associated to the corresponding pixel in the image sequence.

The general problem of adaptive filters is discussed in[2] and in particular some relevant algorithms for adaptive spatial-temporal noise filtering for video images are reviewed in [1]. Conventional approach in designing optimum filters is to use minimization of the mean-square-error (MSE) which results in Wiener filter for stationary signals and recursive Kalman filters for signals with time-varying statistics[3]. In this case, due to the real-time processing requirement and the fact that there is no linear phase restriction on temporal filters, recursive filters are the only feasible solutions. It is therefore of interest to develop an adaptive Kalman filter that can adjust its parameters based on the variations in the noise statistics and detection of motions in the image sequence. In the next section, a relatively simple but effective adaptive Kalman filtering algorithm for temporal filtering of image sequences is proposed which addresses some of the desired implementation requirements.

2. ADAPTIVE TEMPORAL FILTER

One important factor for adaptive filtering of video images is to find an algorithm that has acceptable performance and can lend itself to a real-time implementation. Wiener filter approach and general discrete recursive Kalman filtering both have same steady-state solution if the signal and noise are stationary. In the case of time varying statistics, Kalman filter would be the suitable approach. In practice, however, not only the signal and noise are non-stationary but also their statistics are unknown. Adaptive filters are based on dynamically adjusting the parameters of the supposedly optimum filter based on the estimates of the unknown parameters. Adaptive Kalman filter can be based on an on-line estimation of motion as well as the signal and noise statistics available data.

Let $x(k)$ represent a pixel grayscale on frame k. The ideal noise-free pixel value is represented by $s(k)$ which is assumed to be a first-order AR model. This is a more realistic and simple model that is usually used to represent the temporal behavior of pixels in video signals[1]. Under this assumption, the process and measurement equations are:

1. The process model is $s(k+1) = as(k) + w(k)$, in which a is a constant that depends on the signal statistics and $w(k)$ is the process noise (assumed to be a white independent zero mean Gaussian random process with variance of σ_w^2).
2. The measurement signal is $x(k) = s(k) + v(k)$ in which $v(k)$ is the independent additive zero mean Gaussian white noise with variance of σ_v^2.

Here it is implicitly assumed that the noise and signal are stationary random processes that are fully determined by their second-order statistics. Although this assumption is not true in practice, but for local statistics, it can safely be used with some approximation. The recursive Kalman filter is developed based on the following definitions:

1. The filter output is represented by $y(k)$ which is the estimate of the signal, at time k.
2. Variance of the estimation error is theoretically defined by $\sigma^2(k) = E\left\{[y(k) - s(k)]^2\right\}$, which is initially unknown.
3. Kalman filter gain is represented by $\mathrm{K}(k)$.

The overall Kalman filter algorithm is then given as follows:

Algorithm A

```
Let $y(-1) = 0$, and $\sigma^2(-1) = \sigma_v^2$, and start by
setting $k = 0$
    LOOP: for $k$ do the following operations:
        $\mathrm{K}(k) = \frac{a^2\sigma^2(k-1) + \sigma_w^2}{a^2\sigma^2(k-1) + \sigma_w^2 + \sigma_v^2}$
        $y(k) = \mathrm{K}(k) ? x(k) + a ? [1 - \mathrm{K}(k)] ? y(k-1)$
        $\sigma^2(k) = a^2 [1 - \mathrm{K}(k)] \sigma^2(k-1) + \sigma_w^2$
        Increment $k$, $k \Re k+1$, and go back to
      LOOP
    END
```

In this algorithm, there are several parameters, which are unknown in practice. These parameters are, σ_v^2, σ_w^2, and a. The algorithm can be made adaptive by properly estimating these parameters, using local information. Estimation of these parameters can be based on optimization of a criterion function like minimum mean-square error (MMSE). No matter which estimation technique is used, the quality or reliability of the estimates will always depend on the length of data used to estimate them. Based on the aforementioned assumptions, the following instantaneous estimates can be used to achieve fast and simple implementation:

1. Parameter a, defined by $E\{x(k)x(k-1)\} / E\{x^2(k)\}$, can be estimated by using $\hat{a} = x(k)x(k-1) / \frac{1}{2}\left(x^2(k) + x^2(k-1)\right)$. For stability reasons it is suggested to use some *a priori* information about the signal and keep this parameter constant.
2. Simple estimates of $\sigma_v^2 = E\left\{[x(k) - \hat{a}y(k-1)]^2\right\}$, as well as $\sigma_w^2 = E\left\{[y(k) - \hat{a}y(k-1)]^2\right\}$, are calculated, in turn, by $\hat{\sigma}_v^2 = [x(k) - \hat{a}y(k-1)]^2$ and $\hat{\sigma}_w^2 = [y(k) - \hat{a}y(k-1)]^2 = \mathrm{K}^2\hat{\sigma}_v^2$.

The advantage of the adaptive algorithm is that the Kalman gain starts with relatively large values, and gradually decreases when the temporal signal is stationary. This is a desired property, which produces more noise filtering as time advances. However, when there is a motion, estimates of the noise and process variances, $\hat{\sigma}_v^2$ and $\hat{\sigma}_w^2$, increase which results in less filtering of the signal. This will reduce the noise filtering so that it can better follow the motion with minimal lagging effect.

In this adaptation, there is still no explicit way of estimating the motion. One approach to motion estimation is to compare two consecutive temporal samples and use their magnitude difference to infer existence or nonexistence of motion. This estimation can be conducted by using a statistical test based on the assumption of white Gaussian noise[4]. For example, when there is a sudden change in the signal, the difference between $ay(k-1)$ and $x(k)$, with a given confidence level, goes beyond its statistical variation. In this case, a simple test can be used to estimate sudden changes (motion) in the signal. Assuming that σ_v^2 represents the variance of the aforementioned differences, then based on Gaussian noise distribution, it can be said that a motion is present if $\gamma = |x(k) - ay(k-1)| / \sigma_v ? \Gamma$. If the test is positive, then the gain calculation in the Kalman filter can be reinitiated by assuming $\sigma^2(k) = \sigma_v^2$. The new gain value significantly reduces the lagging effect while improves the noise filtering. Other modifications can also be utilized. For example, we can set the Kalman gain equal to 1 and reinitiate the filtering right after the motion. Or set both $\sigma^2(k)$ and σ_w^2 equal to σ_v^2 and initiate the Kalman gain to $\mathrm{K}(k) = \left(a^2 + 1\right) / \left(a^2 + 2\right)$ right after the motion.

The overall algorithm is represented is presented as follows:

Algorithm B

```
Let $y(-1)=0$, $\sigma_w^2=\sigma_v^2$, and $\sigma^2=\sigma_v^2$, and start the loop by setting $k=0$
LOOP: for $k$ do the following operations:
    $\mathrm{K}=\dfrac{\sigma^2+\sigma_w^2}{\sigma^2+\sigma_w^2+\sigma_v^2}$ ;
    $y(k)=\mathrm{K}\,?x(k)+(1-\mathrm{K})y(k-1)$ ; or
    $y(k)=y(k-1)+\mathrm{K}\lfloor x(k)-y(k-1)\rfloor$;
    if  $D=\dfrac{|x(k)-y(k-1)|}{\sigma_v}\ ?\ \Gamma$,
        $\sigma_w^2=\sigma_v^2$ ;
        $\sigma^2=\sigma_v^2$ ;
    else
        $\sigma_w^2=\mathrm{K}\sigma_v^2$ ;
        $\sigma^2\ \Re\ (1-\mathrm{K})\sigma^2+\sigma_w^2$ ;
    end-if
    Increment $k$, $k\ \Re\ k+1$ , and go back to LOOP
END
```

Selection of the threshold Γ is very important. In this case, it can be shown that γ^2 has χ^2 distribution with one degree of freedom[4]. For the purpose of statistical test of hypothesis, different values of Γ, for different confidence level, are tabulated in Table I. Proper use of these values will result in the same confidence level as stated in [4]. In other words, if the noise is white and independent from signal, using 95% confidence level will produce only 5% error, in average, when it is applied in motion detection. In Figure 1, the Kalman filter with motion detection is compared with the standard non-adaptive optimum Kalman filter by simulating a sudden change in the signal to represent motion. In this case the SNR is set to 20dB. In this simulation, the confidence level is kept at the highest level; namely 99.9% ($\Gamma=3.29$).

Table I: Threshold values for Motion detection

Confidence Level	Γ^2	Γ
99.90%	10.827	3.29
99%	6.635	2.576
98%	5.412	2.326
95%	3.841	1.96
90%	2.706	1.645

In any given application, the user should provide a reasonable value for σ_v^2 based on his or her *a priori* information. As it is evident, higher confidence level results in a better performance. It should be emphasized that this approach is sensitive to impulsive noise and assigns an impulsive noise to a motion. Using local spatial filtering can minimize the problem of impulsive noise. In other words, instead of using original frames, we can use spatially filtered version of that frame sequence just for the purpose of motion detection. The problem of impulsive noise can also be resolved by using some form of nonlinear filter.

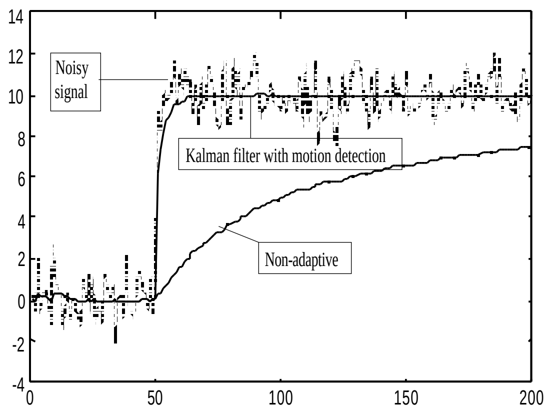

Figure 1- The result of Kalman filtering, using motion detection, in comparison with non-adaptive filtering when SNR is 20dB and Γ=3.29.

3. FPGA IMPLEMENTATION

The implementation of the adaptive Kalman filter is performed with standard memory components and a Xilinx FPGA. Memory components are used for frame and parameter buffers while the FPGA is used for pixel and parameter calculations. The system architecture shown in Figure 2 illustrates three input buffers holding $y(k-1)$, σ^2, σ_w^2, and a Xilinx implementation to calculate updated values for these buffers in addition to generating the proper output $y(k)$. The system can easily operate at a 66 Mhz clock enabling 1024x1024 60 Frames/s operation. System clock rates of 80 Mhz and 100 Mhz can also be achieved for more aggressive system bandwidth requirements.

The pixel calculation data path for $y(k)$ is straight forward once the K parameter is calculated. The parallel pipelined structure used for the sample processing algorithm involves pre-subtraction, two input variable multiplier and post-addition as shown in Figure 3. After the initial latency a sample $y(k)$ is output every system clock. Hardware resources to perform this data path operation is approximately 458 Logic Cells.

The parameter calculation is more involved and also requires a parallel pipelined structure for sample processing since each pixel in the frame also has parameters σ^2 and

σ_w^2. The algorithm requires pre-addition and division for the K parameter calculation. To perform the comparison for the inequality we can normalize for σ_v and define D' and Γ'.

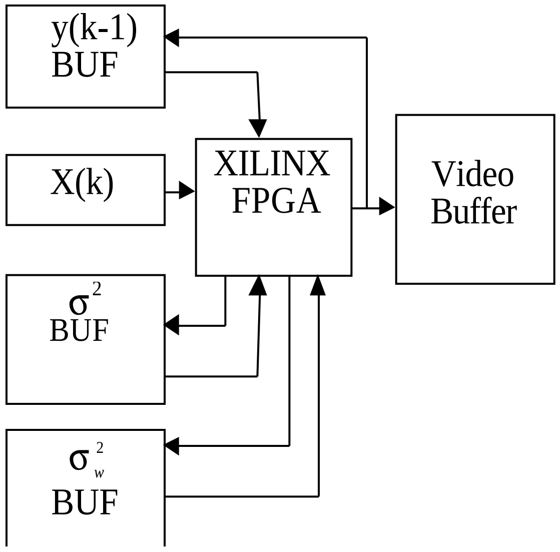

Figure 2- System level implementation for Kalman filtering.

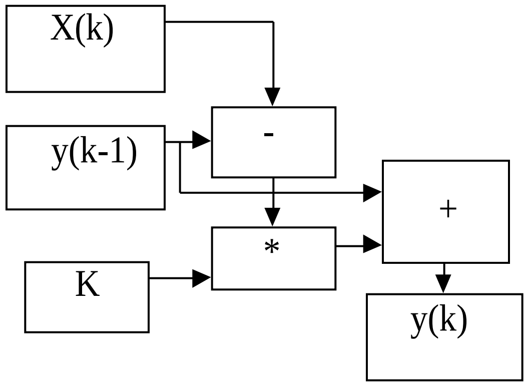

Figure 3- Pixel Calculation implementation for Kalman filtering.

Using a 2's complement function on $x(k)-y(k-1)$ from the pixel calculation, we derive the necessary signal for the 2x1 mux parameter selection. The calculation of new parameters involves an adder, subtracter, variable multiplier and loadable constant coefficient multiplier as shown in Figure 4. After initial latency a new updated parameter is given every system clock. The pixel and parameter calculation blocks are latency synchronized such that K and $x(k)-y(k-1)$ are properly aligned. The output and parameters are aligned such that one memory controller can handle reads and writes to input buffers. Hardware resources for the parameter calculation is approximately 1664 Logic Cells.

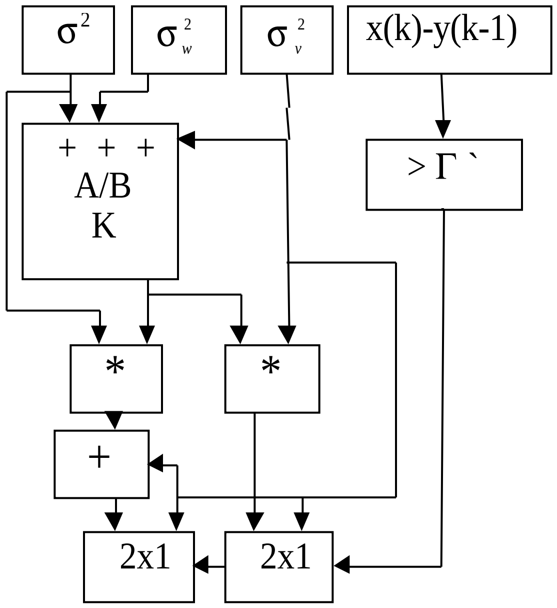

Figure 4- Parameter Calculation implementation for Kalman filtering.

The total hardware resources for the FPGA is 2122 Logic Cells allowing for a Xilinx XC4036XLA or XCV50 to be used for implementation of the Kalman filtering operation. Extensive use of Xilinx Smart-IP available in the COREGEN tool allowed for optimal performance and shortened design cycle.

4. CONCLUSION

General problem of video image filtering has been discussed and some simulation results have been presented. VLSI implementation of the developed algorithm, using Xilinx FPGA has been presented. It should be understood that if there is an impulsive noise in the image sequence, this Kalman filter algorithm should be used in conjunction with pre-spatially non-linear filtered frames.

5. REFERENCES

[1] J. C. Braileam, R.P. Kleihorst, S. Efstratiadis, A. K. Katsaggelos, and R.L. Lagendijk, "Noise reduction filters for dynamic image sequences: A review," *Proceedings of The IEEE,* vol. 83, no. 9, September 1995.

[2] S. Haykin, *Adaptive Filter Theory,* 3rd Ed., Prentice Hall, New Jersey, 1996.

[3] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering,* 3rd Ed., John Wiley and Sons, New York, 1997.

[4] R. E. Kirk, *Statistics, An Introduction,* 3rd Ed., Holt, Rinehart and Winston, Inc., Texas, 1990.