PRACTICAL ADAPTIVE FILTERING ON FPGA

Spirov R. P.

Technical University- Varna
1, Studentska str., 9010 Varna, Bulgaria
Tel: +359 52383618; E-mail: rosexel@abv.bg

Abstract

In this paper the author is presented the hardware application of the adaptive Kalman filter algorithm for pattern recognition with the FPGA. The adaptive temporal filter is proposed that lend itself to hardware implementation for real-time temporal processing of image sequences. Adaptation in this case is with respect to motion in the image sequence as well as variation of noise statistics. The project is used the Altera DE2 board to implement a simple hardware design. For describing its behavior use the VHDL language and the Altera's Quartus tools to synthesize and Altera DE2 board to implement a simple hardware FPGA design

1. INTRODUCTION

In this paper is presented basic hardware architecture, using extended Kalman filter-based method for calculating a trajectory by tracking features at an unknown location on Earth's surface, provided the topography is known, is given in [1]. The proposed model is implemented using VHDL and simulated and synthesized into an FPGA. The hardware design was implemented on an Altera Quartus II board. The practical method for using FPGA to realize VHDL implementation of the developed algorithm using Altera De2 FPGA. The board is shown in fig. 1.

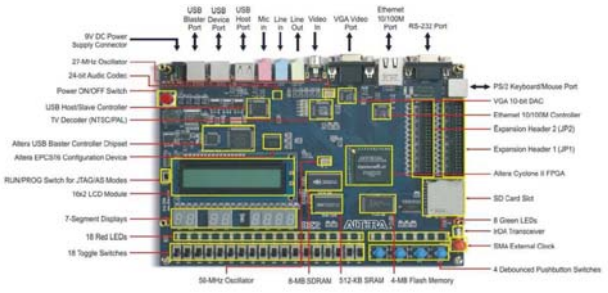

Fig. 1. The Altera DE2 board

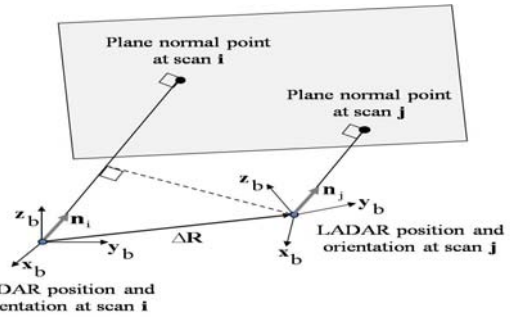

Fig. 2. Navigation-changes in perceived location of the normal point between scan I and scan j are applied to estimate position changes

In Fig. 2 ΔR is the delta position vector (displacement vector between scans i, at time t_i, and scan j, at time t_j, in this case); n_i is the plane normal vector whose components are resolved in the Ladar body frame at scan epoch t_i; n_j is the plane normal vector whose components are. Note that in the navigation frame, the planar surface normal vectors at epoch's t_i and t_j are equal since resolved in the Ladar body frame at scan epoch t_j ; and, ρ_i and ρ_j are the shortest distances from the Ladar to the plane at epochs t_i and t_j, respectively. stationary planar surfaces are assumed. However, expressed in the Ladar body frame both normal vectors are likely to be unequal due to the body frame rotation between epoch's t_i and t_j. From the geometry presented in Fig.1, a relationship can be derived between the projection of the displacement vector (between epoch's t_i and t_j) onto the planar surface normal vector and the change in the normal point range between scans i and j is shown in Eq.1:

$$\mathbf{\Delta R} \cdot \mathbf{n}_i = \rho_i - \rho_j \qquad (1)$$

Given *M* associated planar surfaces, a set of linear equations like (7) can be set up in matrix form is given in Eq.2:

$$\mathbf{H} \cdot \mathbf{\Delta R} = \mathbf{\Delta \rho} \qquad (2)$$

Were:

$$\mathbf{H} = \begin{bmatrix} \mathbf{n}_{i,1}^T \\ \mathrm{M} \\ \mathbf{n}_{i,M}^T \end{bmatrix}, \quad \mathbf{\Delta r} = \begin{bmatrix} \rho_{i,1} - \rho_{j,1} \\ \mathrm{M} \\ \rho_{i,M} - \rho_{j,M} \end{bmatrix} \qquad (3)$$

Note that a minimum of three non-collinear planar surfaces is required for the observation matrix, H, to be non-singular and thus allowing for a unique solution of Eq.2. The estimation process is based on a complementary Kalman filter methodology [2] which employs differences between INS and laser scanner observables as filter measurements. Changes in planar surface ranges between consecutive scans are used as laser observables. Correspondingly, laser scanner observables of the Kalman filter are formulated as follows for the scan at time epoch t_m in Eq.4:

$$\Delta\boldsymbol{\rho}_{LS}(t_m)=\begin{bmatrix}\rho_1(t_{m-1})-\rho_1(t_m)\\ \cdots\\ \rho_N(t_{m-1})-\rho_N(t_m)\end{bmatrix} \quad (4)$$

where N is the number of features for which is match is found time epoch t_m and t_{m-1}. Equivalent observables can be synthesized from INS measurements by transformation of the INS displacement vector into the range domain as follows in Eq.5 and Eq.6:

$$\Delta\boldsymbol{\rho}_{INS}(t_m)=\mathbf{H}(t_{m-1})\left(\Delta\mathbf{R}_{INS}(t_m)+\Delta\mathbf{C}_{\mathrm{b}}^{\mathrm{n}}(t_m)\mathbf{l}_{\mathrm{b}}\right) \quad (5)$$

$$\Delta\mathbf{C}_{\mathrm{b}}^{\mathrm{n}}(t_m)=\mathbf{C}_{\mathrm{b}}^{\mathrm{n}}(t_m)-\mathbf{C}_{\mathrm{b}}^{\mathrm{n}}(t_{m-1}) \quad (6)$$

The differences between inertial and laser scanner observables Eq.7:

$$\mathbf{y}_{Kalman}(t_m)=\Delta\boldsymbol{\rho}_{INS}(t_m)-\Delta\boldsymbol{\rho}_{LS}(t_m) \quad (7)$$

Particular filter states include: errors in position changes between consecutive scans, velocity errors, attitude errors, gyro biases, and accelerometer biases in Eq.8:

$$\delta\mathbf{x}=\left[\delta\Delta\mathbf{R}_n^T \quad \delta\mathbf{v}_n^T \quad \boldsymbol{\psi}^T \quad \mathbf{a}_b^T \quad \mathbf{b}_b^T\right]^T \quad (8)$$

For this state vector, the observation matrix H_{Kalman} can be derived directly by augmenting the geometry matrix of Eq.3 with zero elements is shown in Eq.9:

$$\mathbf{H}_{Kalman}(t_m)=\begin{bmatrix}\mathbf{n}_1^T(t_{m-1}) & 0 & \mathrm{L} & 0\\ \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M}\\ \mathbf{n}_M^T(t_{m-1}) & 0 & \mathrm{L} & 0\end{bmatrix} \quad (9)$$

The measurement noise matrix R_{Kalman} is derived from the line and planar surface estimation processes performing a comprehensive covariance analysis of the feature extraction method. Derivation of the filter state transition matrix and the system noise matrix for the filter states chosen employs a standard Kalman filter formulation. Errors in the position estimate are thus transformed into errors in line parameters and add up to line extraction errors. As a result, the current position error contributes to the position error for the next scan where the new line is used for navigation.

2. ADAPTIVE FILTERING

The 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 apixel 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[3]. Under this assumption, the process and measurement equations:

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.

The noise and signal are stationary random processes that are fully determined by their second-order statistics. The recursive Kalman filter is developed based on the following definitions:

1. The filter output is $y(k)$ which is the estimate of the signal, at time *k* .

2. The estimation error is defined by:

$\sigma^2(k) = E\{[y(k) - s(k)]^2\}$, which is initially unknown.

3. Kalman filter gain is presented by $K(k)$.

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:

$$K(k)=\frac{a^2\sigma^2(k-1)+\sigma_w^2}{a^2\sigma^2(k-1)+\sigma_w^2+\sigma_v^2}$$

$$y(k)=K(k).x(k)+a.[1-K(k)].y(k-1)$$

$$\sigma^2(k)=a^2[1-K(k)].\sigma^2(k-1)+\sigma_w^2$$

Increment k, $k=(k+1)$, 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. 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 $\hat{a} = x(k)x(k-1) - (x^2(k) + x^2(k-1))$. For stability reasons it is suggested to use some a priori information about the signal and keep this parameter constant.

2. Simple estimates of:

$\sigma_v^2 = E\{[x(k) - \hat{a}y(k-1)]^2$ well as

$\sigma_w^2 = E\{[y(k) - \hat{a}y(k-1)]^2$, are calculated, in turn by

$\hat{\sigma}_v^2 = [x(k) - \hat{a}y(k-1)]^2$ and

$\hat{\sigma}_w^2 = [y(k) - \hat{a}y(k-1)]^2 = K^2\hat{\sigma}_v^2$

Assuming 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. 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]. Kalman filter by simulating a sudden change in the signal to represent motion [5]. In this case the SNR is set to 20dB, highest level, namely 99,9% ($\Gamma = 3.29$). One approach to motion estimation is to compare two consecutive temporal samples and use their magnitude difference to infer existence or non existence of motion. 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. Assuming that σ_v^2 represents the variance of the aforementioned differences, then based on Gaussian noise [4] 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. Or set both $\sigma^2(k)$ and σ_w^2 equal to σ_v^2 and initiate the Kalman gain to $K(k) = (a^2+1)(a^2+2)$ right after the motion. The overall algorithm is Algorithm B.

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:

$$K=\frac{\sigma^2+\sigma_w^2}{\sigma^2+\sigma_w^2+\sigma_v^2};$$

$$y(k)=K.x(k)+(1-K)y(k-1);\ \text{or}$$

$$y(k)=y(k-1)+K[x(k)-y(k-1)];$$

$$\text{if } D=\frac{|x(k)-y(k-1)|}{\sigma_v}<\Gamma,$$

$\sigma_w^2=\sigma_v^2$;

$\sigma^2=\sigma_v^2$;

else

$\sigma_w^2=K\sigma_v^2$;

$\sigma^2=(1-K).\sigma^2+\sigma_w^2$;

end-if

Increment k, $k=(k+1)$, and go back to LOOP

END

3. FPGA IMPLEMENTATION

Memory components are used for frame and parameter buffers while the FPGA is used for pixel and parameter calculations. The system architecture shown in Fig.3 illustrates 3 input buffers holding $y(k-1)$, and σ^2, σw^2 and a Altera implementation to calculate updated values for these buffers in addition to generating the output $y(k)$.

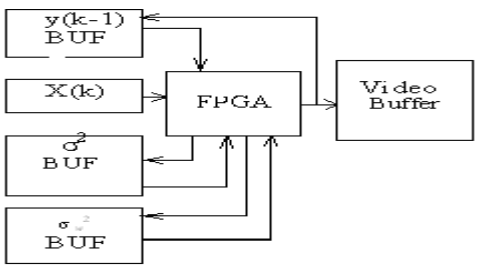

Fig. 3. Levels of implementation for Kalman filtering

The system can easily operate at 66 MHz clock enabling 1024 X 1024 60 Frames 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-subtractions, two input variable multiplier and post-addition as shown in Fig.4.

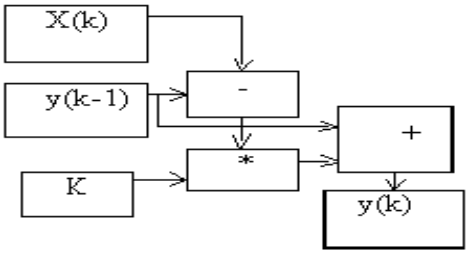

Fig. 4. Pixel Calculation

After the initial latency a sample $y(k)$ is output every system clock. Hardware resources to perform this data path operation are 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 σw^2.

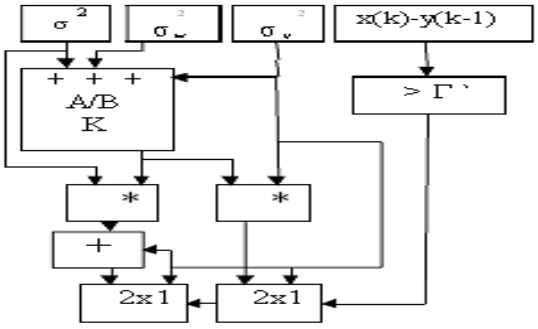

Fig. 5. Implementation for filtering

The algorithm requires pre-addition and division for the K parameter calculation. To perform the comparison for the in quality it can requires pre-addition and division for the K parameter calculation. To perform the comparison for the in quality it can normalize for $\sigma_{,}$, define D' and Γ'. Using second compliment 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 Fig. 6.

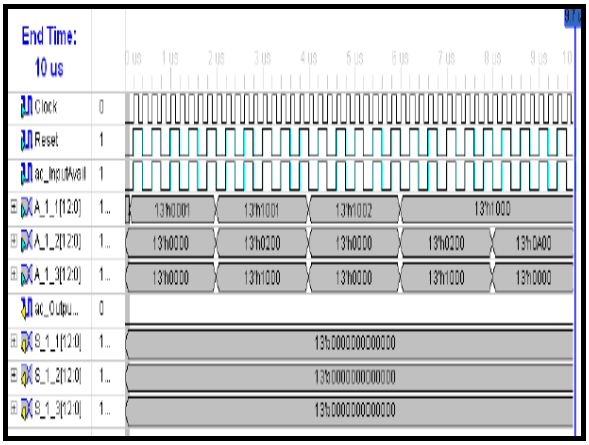

Fig. 6. Behavioral simulation of the Kalman Filter

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 property aligned.

The output and parameters are aligner such that one memory controller can handle reads and writes to input buffers. Hardware resources for the parameter calculation is approximately 1664 Logic Cells.

Table 1. Synthesis Results

No.	Information	Count	% use
1	No of slice	2145 of 32640	7%
2	Slice LUTs	3626 of 32640	14%
3	Slice LUTs Used as logic	3626 of 32640	14%
4	LUT Flip-Fl.pair used	4168	
5	LUT Flip-Fl. pairs withan unused FF	2023 of 4168	49%
6	LUT flip-flop an unsed LUT	542 of 4168	17%
7	Fully used FF pairs	1603 of 4168	41%
8	Bonded IOBs	82 of 480	21%
9	DSP48Es	16 of 288	7%

The project includes the creation of parallel models respectively Matlab environment and secondly in Quartus tools. The Altera Cyclone II FPGA connected to a variety of peripherals including 512K of SRAM, 4MB of Flash, 8MB of SDRAM, VGA output Ethernet, audio input and output, and USB ports.

4. CONCLUSION

The performance of this implementation can be attributed to the parallel hardware blocks used in performing the necessary calculations for the algorithm [7]. Further to this, the design can be scaled for larger databases by simply adding more processing elements in parallel The above hardware design was implemented on an Altera Quartus II board (clocked at 100 MHz) and was able to operation time is about 60 clock cycle, which about 0.6us at 100MHz clock pulse, so the operation speed can be up to 1.5MHz. The whole design requires 4168 ALUTs and 241 registers (occupancy of resources is about 49%).The advantage of parallel processing in FPGA leads to a substantial increase in performance and accuracy in processing, extraction of information than in the simulation in Matlab. 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. The main contribution of the work is design and implementation of a physically feasible hardware system to accelerate the processing speed of the operations required for real time face recognition. The proposed models are implemented using VHDL, and simulated and synthesized into a single FPGA. It is demonstrates that this technology can produce effective and powerful applications systems.

References

[1] Hagen, E., Navigation by Optical Flow, In Proc. of IAPR Intern.Conf.-Patt. Recgn.. Vol.1, 1992, pp. 700–703.

[2] Maybeck, Peter S. *Stochastic Models Estimation and Control, Vol I.* Academic Press, Inc., Orlando, 32887.

[3] Lazarov, A. D. *Spatial correlation algorithm for ISAR image reconstruction*- 2000 IEEE - Rad.Conf. Virginia, USA, 7-12

[4] S. Haykin, *Adaptive Filter Theory*, 3rd Ed., Jersey, 1996

[5] Brown and P. Y. C. Hwang, Introduction to Random Signals and Applied Kalman Filtering, 3 rd Ed. 2006.

[6] Pellerin D. and S. Thibault. Practical FPGA programming in C. *Prentice Hall PTR*, ISBN: 0-13-154318-0.

[7] Wang Y., Osterman J. and Zhang Y. Video Processing and Communications. C.*Prentice Hall PTR*, ISBN: 0-13-017547-1.

[8] Parhi K. VLSI Digital Signal Processing System Design and Implementation Wiley Inter Since 0471241865.