REAL TIME IMAGE ROTATION USING B-SPLINE INTERPOLATION ON FPGA's BOARD

C. Berthaud, E. Bourennane, M. Paindavoine and C. Milan
Laboratoire Le2i, Université de Bourgogne,
Aile des Sciences de l'Ingénieur,
BP 400, 21011 Dijon, FRANCE
Tel/fax: +33-3-80-39-60-43
e-mail: berthaud@satie.u-bourgogne.fr

ABSTRACT

The aim of our work is to realize the implementation of a real-time image rotation on FPGA's board. The method we used is based on a B-spline interpolator. The integration capicity of FPGAs is relatively weak, so the difficulty in this problem is to determine the right coding of the rotation filter while keeping a good accuracy on filtering outpût. In this article, we remind a few definitions about B-spline functions and we present how we use B-spline interpolation for the image rotation problem. Then, we describe the way we calculate probability density function of the output error in order to determine the filter data coding.

1 B-SPLINE FUNCTION

1.1 Introduction

A B-spline is a continous function, noted $B_n(x)$, continously derivable up to the $(n-1)^{th}$ order, polynomial of degree n and defined by pieces [1]-[2].

A B-spline function is defined by the recursive equation:

$$B_n(x) = B_{n-1}(x) * B_0(x) \quad (1)$$

where $B_0(x)$ is the B-spline of order 0 defined by:

$$B_0(x) = \begin{cases} 1 & pour\ -0,5 < x < 0,5 \\ 0 & sinon \end{cases} \quad (2)$$

So we can deduce the function:

$$B_p(x) = \sum_{j=0}^{p+1} C_{p+1}^{j}(x + \frac{p+1}{2} - j)^p . u(x + \frac{p+1}{2} - j) \quad (3)$$

with $u(x)$, the Heaviside's function

and $C_n^p = \frac{n!}{(n-p)!.p!}$

1.2 Interpolation

With a B-spline of the third order, we approximate the grey scale $s(k)$ by :

$$s(x) = \sum_j C_j B_3(x-j) \quad (4)$$

For $x = k$, $k \in Z$, we have:

$$s(k) = \sum_j C_j B_3(k-j) \quad (5)$$

We know the sequence $s(k)$ and also the values of $B_3(k-j)$ (formula 3). So interpolate the signal $s(k)$ consists in determining the weighting factors C_j of the B-spline function.
With (4), we can deduce that:

$$s(k) = \frac{1}{6} \cdot C_{k-1} + \frac{4}{6} \cdot C_k + \frac{1}{6} \cdot C_{k+1} \quad (6)$$

After a Z-transform, we obtain:

$$C_(z) = \frac{6}{z^{-1} + 4 + z} . s(z) \quad (7)$$

The calculation of the weighting factors C_k can also be done by using a non causal recursive filter.

1.3 Application of B-spline interpolation : translation by a non entire value

In the next paragraph, we will need to realize translations by a non entire value. We make such translations by using the following formula :

$$s(k-\Delta) = \sum_j C_j B_3(k - \Delta - j) \quad (8)$$

Where Δ is the non entire translation value to apply on the signal $s(k)$.

2 IMAGE ROTATION

Let $R(\theta)$ be the rotation matrix :

$$R(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \tag{9}$$

A simple way to realize an image rotation is to factorize the rotation matrix into a sequence of one-dimensional transformations along the x and y axis. To solve this problem, a factorization of the rotation matrix has been proposed by several authors [3,4,5]:

$$R(\theta) = A \; . \; B \; . \; A \tag{10}$$

with

$$A \;=\; \begin{pmatrix} 1 & -\tan(\theta/2) \\ 0 & 1 \end{pmatrix} \tag{11}$$

and

$$B \;=\; \begin{pmatrix} 1 & 0 \\ \sin(\theta) & 1 \end{pmatrix} \tag{12}$$

The A and B matrices are translation matrices respectively along the x and y axis.
Image rotation is also a serie of translation by non entire values. As we have seen in the paragraph 1.3, we will use B-spline interpolation to realize these translations.

3 ALGORITHM OF TRANSLATION BY NON ENTIRE VALUE

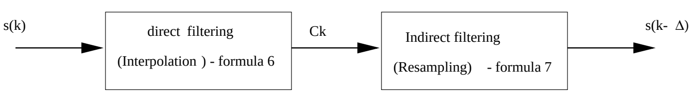

Figure 1: *Principle of the algorithm of translation by non entire value*

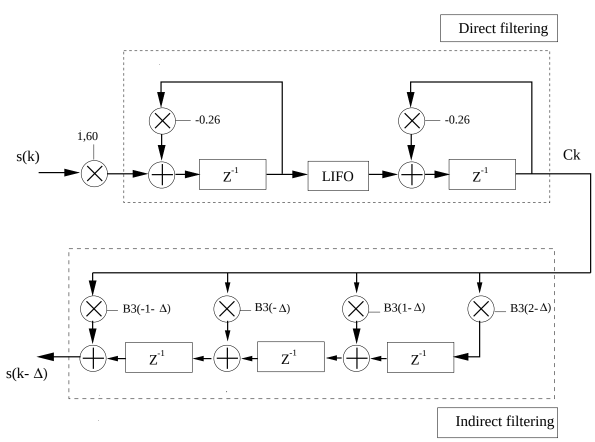

Figure 2: *Algorithm of translation by non entire value*

4 FILTER DATA CODING

Before we implement the rotation algorithm on a FPGA's board, we must calculate the number of bits necessary to code the data and the multiplying coefficients of the filter. Several methods have been proposed to estimate the filter accuracy by determining the calculation noise variance of the filter output. However, the calculation noise variance is not a sufficient information to code the filter. Indeed, we need to know the range of the error we make in function of the wordlength. So we choose to solve this problem in an original way by calculating the probability density function of the filter output error.

The effects of wordlength limitation can be shared in two separated categories. The roundoff of multiplying coefficients has the effect of altering the transfer function. This effect is purely deterministic and will not be studied in this article [6][7]. The second category is due to the multiplication results roundoff (data roundoff). In the following section, we will study this effect by considering that each multiplication is the source of uniform white noise and we will calculate the probability density function of filter output.

4.1 Property

4.1.1 Sum of two stationary processes

Let Y be a stationary process of which the realisations y are the sum of realisations of two statistically independent stationary processes X_1 and X_2 with probability densities $p_{e1}(x)$ and $p_{e2}(x)$. So the probability density $p_s(y)$ of Y will be:

$$p_s(y) = \int_{-\infty}^{+\infty} p_{e1}(x) p_{e2}(y-x) dx \tag{13}$$

Then

$$p(y) = (p_{e1} * p_{e2})(y) \tag{14}$$

4.1.2 Multiplication d'un processus stationnaire par une constante

Let Y be a stationary process of which the realisations $y = ax$ are proportional to the realisations x of a stationary process X with probability density $p_e(x)$. So the probability density function of Y will be $p_s(y)$:

$$p_s(y) = \frac{1}{|a|} p_e(\frac{y}{a}) \qquad \forall a \in \mathbf{R}^* \tag{15}$$

4.2 Range and variance

Let X be a stochastic signal of which we know the probability density function p(X). We can then determine:

- its range, *i.e. the maximal amplitude of the signal X. It is directly determined from the probabilty density function $p(X)$ and corresponds to the maximal value of X for which $p(X > X_{max}) = 0$.*

- *its variance with the following formula:*

$$\sigma^2 \quad = \quad \int_{-\infty}^{+\infty} X^2 \ p(X) \ dX \qquad (16)$$

5 PROBABILTY DENSITY CALCULATION OF DATA ROUNDOFF IN A DIGITAL FILTER

5.1 General case

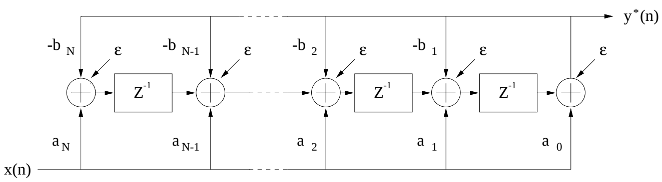

Figure 3: *IIR Filter*

The general expression for a digital filter with an impulse response $g(n)$ is:

$$y(n) = x(n) * g(n) = \sum_{i=0}^{N} a_i x(n-i) + \sum_{i=1}^{M} b_i y(n-i) \qquad (17)$$

Thus,

$$y(n) = \left(\sum_{i=0}^{N} a_i x(n-i) \right) * h(n) \qquad (18)$$

With h(n) the impulse response of the recursive part of the filter g.

For an implementation, the wordlength limitation involves necessarily an error $\epsilon_i(n)$ in each multiplication. So the expression of the implemented filter is:

$$\begin{aligned} y^*(n) &= \sum_{i=0}^{N} a_i x(n-i) + \sum_{i=1}^{M} b_i y^*(n-i) + \sum_{i=0}^{N+M+1} \epsilon_i(n) \\ &= \left(\sum_{i=0}^{N} a_i x(n-i) + \sum_{i=0}^{N+M+1} \epsilon_i(n) \right) * h(n) \qquad (19) \end{aligned}$$

Then we can deduce the expression of the data roundoff error:

$$\begin{aligned} e(n) &= y(n) - y^*(n) && (20) \\ &= \left(\sum_{i=1}^{N+M+1} \epsilon_i(n) \right) * h(n) && (21) \\ &= \sum_{i=1}^{N+M+1} \sum_{j=0}^{+\infty} h(j) \epsilon_i(n-j) && (22) \end{aligned}$$

We can notice that the data roundoff error $e(n)$ depends, of course, on the data coding in the filter but also on the number of multiplication (N+M+1) and on the impulse response h(n) of the recursive part of the filter.

We suppose that the set of realisations $\epsilon_i(n)$ with probability density $p_i(x)$ are independent. With this hypothesis, it is possible to apply the relationship (14) in order to determine the probability density function $p_e(x)$ of the filter output error:

$$p_e(x) \quad = \quad \coprod_{i=1}^{N+M+1} \coprod_{j=0}^{+\infty} K_i^j \qquad (23)$$

With

$$K_i^j = \begin{cases} p_i\left(\frac{x}{h(j)}\right) \cdot \frac{1}{|h(j)|} & for\ h(j) \neq 0 \\ \delta(x) & else \end{cases}$$

And $\coprod$ is defined as a serie of convolution products:

$$\coprod_{i=0}^{N} m_i(x) = m_0(x) * m_1(x) * \cdots * m_{N-1}(x) * m_N(x)$$

6 ERROR PROPAGATION IN A FILTER CHAIN

6.1 parallel filters

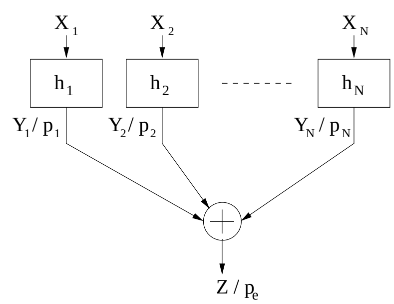

Figure 4: *Parallel structure*

By applying directly the relationship (14), we obtain the expression of the output error probability density for parallel filters:

$$p_e(x) \quad = \quad \coprod_{i=1}^{N} p_i(x) \qquad (24)$$

N	0	1	2	3	4	5	6
e_{max}	3.30468	1.165234	0.82617	0.41308	0.20654	0.10327	0.05163
σ^2	0.41689	0.10422	0.02605	0.00651	1.628e-3	4.07e-4	1.02e-4

Table 1: *Maximal value and variance of translation filter output error*

6.2 cascaded filters

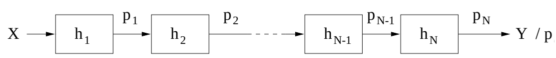

Figure 5: *Cascaded structure*

Let's consider the error $\epsilon_i(n)$ produced by the i^{th} chain stage of cacaded digital filters. The error generated at the end of the chain is:

$$e_(n) = \epsilon_i(n) * (\prod_{j=i+1}^{N} h_j(n)) \tag{25}$$

Where $h_i(n)$ is the impulse response of the i^{th} chain stage of cascaded filters and N is the number of filters.

If each stage produces an error $\epsilon_i(n)$, then we will have at the end of the filter chain:

$$e(n) = \epsilon_N(n) + \sum_{i=1}^{N} \epsilon_i(n) * (\prod_{j=i+1}^{N} h_j(n)) \tag{26}$$

With $H_i^N(n) = \prod_{j=i+1}^{N} h_j(n)$, we obtain:

$$\begin{aligned} e(n) &= \epsilon_N(n) + \sum_{i=1}^{N} \epsilon_i(n) * H_i^N(n) && (27) \\ &= \epsilon_N(n) + \sum_{i=1}^{N} \sum_{j=0}^{+\infty} \epsilon_i(j).H_i^N(n-j) && (28) \end{aligned}$$

If we suppose that the errors produced in each stage are independent, we can write then:

$$p_e(x) = p_N(x) * \prod_{i=1}^{N} \prod_{j=0}^{+\infty} G_i^j \tag{29}$$

With

$$G_i^j = \begin{cases} p_i(\frac{x}{H_i^N(j)}).\frac{1}{|H_i^N(j)|} & for\ H_i^N(j) \neq 0 \\ \delta(x) & else \end{cases}$$

6.3 Error produced by the translation filter

With the previous formulas, we can calculate the output error probability density function of the translation filter in function of N, the number of bits coding the data decimal part. With this probability density, it is possible to determine the variance σ^2 and the maximal value e_{max} of the filter output error in function of N.

We state, as a constraint, that the error due to data roundoff must not be greater than 0.5 grey scale. According to the table above, we have chosen to code the data decimal part with 3 bits.