Distortion Correction Scheme for Multiresolution Camera Images

M. Mohankumar¹, T. Thamaraimanalan² and N. Sanjeev³

¹Assistant Professor, Department of Electronics and Communication Engineering, Sri Eshwar College of Engineering, Coimbatore, India.
²Assistant Professor, Department of Electronics and Communication Engineering, Sri Eshwar College of Engineering, Coimbatore, India.
³Senior Technical Analyst, Caliber Embedded Technologies, Coimbatore, India. Email: sanjeev.n@caliber.net

ABSTRACT

An efficient VLSI architecture implementation for barrel distortion correction in surveillance camera images is presented. The distortion correction model is based on least squares estimation method. To reduce the computing complexity, an odd-order polynomial to approximate the back-mapping expansion polynomial is used. By algebraic transformation, the approximated polynomial becomes a monomial form which can be solved by Horner’s algorithm. The proposed VLSI architecture can achieve frequency 218MHz with 1490 logic elements by using 0.18µm technology. Compared with previous techniques, the circuit reduces the hardware cost and the requirement of memory usage.

Keywords: Barrel distortion correction, Least squares estimation, Surveillance and Horner’s algorithm.

1. INTRODUCTION

Nowadays, surveillance camera is commonly used in public and private places such as government buildings, military bases, car parks, and banks, and so forth. Surveillance camera can capture the entire region of interest with cameras as few as possible if cameras with large field of view are adopted. Surveillance cameras are video cameras used for the purpose of observing an area. It is connected to a recording device or IP network, and may be watched by a guard. Cameras and recording equipment used to be relatively expensive and required human personnel to monitor camera footage, but analysis of footage has been made easier by automated software that organizes digital video footage into a searchable database, and by video analysis software. The amount of footage is also drastically reduced by motion sensors which only record when motion is detected. With cheaper production techniques, surveillance cameras are simple and inexpensive enough to be used in home security systems, and for everyday surveillance.

The distortion correction circuit may be included in end-user camera equipment’s, so how to implement it with lower hardware cost is an important issue. To reduce more hardware, algebraic transformation is used to replace all the vector magnitude square by a new variable. It makes the new combined polynomial becomes a monomial form. Therefore, Horner’s algorithm [6] is able to efficiently evaluate the results of back-mapping and polar to Cartesian coordinate transformation.

2. OVERVIEW OF DISTORTION CORRECTION TECHNIQUE

In this section, least-square estimation method is used for distortion correction technique. Barrel distortion can be corrected by two main tasks: 1) back mapping of all pixels in CIS onto DIS, and 2) calculating the intensity of every pixel in CIS by linear interpolation [7]. The block diagram of the distortion correction procedure is shown in Fig1. First the transformation mapping from rectangular coordinate to polar coordinate for all pixels is done. Then, a back mapping procedure is introduced to decrease the computing complexity and polar to rectangular coordinate steps by eliminating the angle θ and reducing the square root operation for p. Finally, a basic algebraic manipulation is used to decrease the arithmetic resource of the linear interpolation.

The first step of the proposed distortion correction technique is transforming all pixels in the distorted image space (DIS) onto the corrected image space (CIS) i.e., this step is to convert rectangular to polar coordinate. The distortion center is (u_c, v_c) in DIS and the correction center is (u_c, v_c) in CIS. In DIS, (u', v') is the Cartesian coordinate and (ρ', θ') is the polar coordinate. The distance ρ' from distortion center (u_c, v_c) to an image pixel (u', v') and the angle θ' between the pixel and distortion center are given by

\[ ρ' = \sqrt{(u' - u_c)^2 + (v' - v_c)^2} \]  \hspace{1cm} (1)

\[ θ' = \arctan\left(\frac{v' - v_c}{u' - u_c}\right) \]  \hspace{1cm} (2)

Fig. 1. Block diagram for distortion correction procedure.
The distance $\rho$ from distortion center $(u_c, v_c)$ to an image pixel $(u, v)$ and the angle $\theta$ between the pixel and distortion center are given by,

$$\rho = \sqrt{(u - u_c)^2 + (v - v_c)^2}$$

(3)

$$\theta = \arctan\left(\frac{v - v_c}{u - u_c}\right)$$

(4)

The second step is back-mapping process which will map the pixel $(\rho, \theta)$ in CIS to $(\rho', \theta')$ in DIS. The back mapping procedure can be defined by a back mapping expansion polynomial of degree $N$ as,

$$\rho' = \sum_{n=1}^{N} b_n \rho^n$$

(5)

$$\theta' = \theta$$

(6)

Where $b_n$ is the back mapping coefficient which can be obtained by the least-squares estimation method.

The third step is to perform the polar to rectangular coordinate conversion. The location $(u', v')$ in DIS can be calculated as,

$$u' = u_c' + \rho' \cdot \frac{u-u_c}{\rho}$$

$$v' = v_c' + \rho' \cdot \frac{v-v_c}{\rho}$$

(11)

(12)

Thus, the complex square-root calculation is eliminated because both $u'$ and $v'$ are calculated with $\rho'$ rather than with $\rho$.

2.1 Simplified Back-Mapping Procedure

As mentioned above, the odd order polynomial is high-approximation to the back mapping expansion polynomial is used. The back mapping expansion polynomial can be approximated to odd-order polynomial as,

$$\rho' = c_0 \rho + c_1 \rho^3 + c_2 \rho^5 + c_3 \rho^7 + ....$$  

(8)

Where $c_0, c_1, c_2, c_3, ...$ are back-mapping coefficients of the odd-order back-mapping polynomial.

There are two steps to reduce the computing resources of back-mapping procedure. The first is eliminating the calculation of $\theta$. According to (7), the $\theta$ and $\theta'$ are the same. Thus, the $\cos \theta$ and $\sin \theta$ can be obtained as,

$$\sin \theta = \sin \theta' = \frac{v - v_c}{\rho}$$

(9)

$$\cos \theta = \cos \theta' = \frac{u - u_c}{\rho}$$

(10)

In below equations mentioned, here no odd power of $\rho$ exists, so the square-root calculation for $\rho$ can be removed.

3. HORNER'S ALGORITHM

The main purpose of Horner’s algorithm [16] is to efficiently evaluate the polynomials in monomial form.

Given the polynomial,

$$p(x) = \sum_{i=0}^{n} a_i x^i$$

(13)

$$p(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n$$

Where $a_0, a_1, a_2, ..., a_n$ real numbers, then to evaluate the polynomial at a specific value of $x$, say $x_0$. A new sequence of constants is defined as follows,

$$b_n := a_n$$

$$b_{n-1} := a_{n-1} + b_n x_0$$

$$...$$

$$b_0 := a_0 + b_1 x$$

(14)

Then $b_n$ is the value of $p(x_0)$. Then the polynomial can be written as,

$$p[x] = a_0 + x [a_1 + x_0 (a_2 + x_0 (a_3 + x_0 (a_4 + x_0(...))))].$$

(15)

Thus, by iteratively substituting the $b_i$ into the expression,

$$p(x_0) = a_0 + x_0 (a_1 + x_0 (a_2 + ... + x_0 (a_{n-1} + b_n x_0) ...))$$

(16)

As described above, an iteration function is used to rewrite the polynomial. The function $p^n$ which is the $n$th iteration of $p$ can be represented as $p^np^{n-1}$ and it is defined as

$$p^n = p \cdot p^{n-1}$$

(17)
To rewrite the polynomial as iteration function, the evaluated polynomial at the specified value of $x = x_0$ can be represented as,

$$p(x_0) = (c_{n-1} + c_n x_0) \cdot p^{n-1}(x_0)$$  \hspace{1cm} (18)

The polynomial is evaluated to compute $p(x_0)$ where $x_0$ is a constant.

The Horner’s algorithm is executed by following the below steps:

**Step 1.** Set $u = n$ (where $n$ is the degree of the polynomial)

**Step 2.** Set Result = $C_n$.

**Step 3.** If $u = 0$ stop. Answer is Result.

**Step 4.** Compute Result = Result $\times x_0 + Cu$-1.

**Step 5.** $u = u - 1$.

**Step 6.** Go to step 3.

By Horner’s algorithm, the computing complexity of the evaluating polynomial can be decreased. The complexity of back mapping and polar to rectangular coordinate steps can be reduced in the same way.

### 4. PROPOSED VLSI ARCHITECTURE

Fig.2. shows the block diagram of proposed architecture. In this design, when the start signal is enabled, the circuit will output the intensity value of the first pixel after 21 clock cycles. Then, it can process one pixel in CIS per clock cycle. Totally, it would take $(1024 \times 728 + 21)$ cycles to process an image with $1024 \times 728$ pixels. This design consists of four main modules: mapping unit, memory bank, linear interpolation unit and controller. The memory bank is composed of solid state drives and RAM. The interpolation unit linearly interpolates the final intensity value. The controller provides the control signals of each state to other units and handles the whole correction procedure. The detailed circuits of mapping and linear interpolation units are described as follows.

![Mapping Unit](image1)

![Linear Interpolation Unit](image2)

**Fig.2. Proposed VLSI Architecture**

For each pixel $(u, v)$ in CIS, the mapping unit performs the operations needed to calculate $(u', v')$. Most multipliers and adders are realized with 24-bit width as adopted in [6]. According to the simulation results obtained from XILINX, it was found that the propagation delay of a 24-bit multiplier is quite long. Hence, the 15-bit two-stage pipelined multiplier is adopted in the design to get better pipeline scheduling. Fig 3. shows the 15-stage pipelined architecture of the mapping unit. The linear interpolation unit obtains the intensity values of the four neighboring pixels around $(u', v')$ and calculates the final intensity value $I(u, v)$. Fig.4. shows the 6-stage pipelined architecture of the linear interpolation unit which performs the operations needed to calculate $I(u,v)$.

![15-stage pipelined architecture of mapping unit](image3)

![6-stage pipelined architecture of linear interpolation unit](image4)

**Fig.3. 15-stage pipelined architecture of mapping unit**

**Fig.4. 6-stage pipelined architecture of linear interpolation unit**

### 5. RESULTS AND DISCUSSION

The VLSI architecture of the proposed design was implemented by using Verilog HDL. Here MATLAB is used to convert the image file to grey scale image. Table 1 shows the Simulation Results of two designs using Altera EP20K600EBC652-1X FPGA the size the design with TSMC’s 0.18 m cell library. The results show that the design contains 1490 logic elements. It works with a clock period of 5 ns and operates at a clock rate of 218 MHz.

<table>
<thead>
<tr>
<th>Paper</th>
<th>Logic Elements</th>
<th>Clock rate</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>[13]</td>
<td>1686</td>
<td>200MHz</td>
<td>14Mpixels/s</td>
</tr>
<tr>
<td>Proposed</td>
<td>1320</td>
<td>210MHz</td>
<td>21Mpixels/s</td>
</tr>
</tbody>
</table>

![Table 1. Comparative Analysis](image5)
6. CONCLUSION
In this paper, a low-cost, low-power, and low memory requirement VLSI architecture of distortion correcting circuit was presented for surveillance images. The computing complexity of correcting functions is reduced by Horner’s algorithm and the algebraic manipulation of the linear interpolation. This provides an efficient and effective approach to eliminate the distortion. It provides a logic element of about 1490, clock rate of 218MHz and throughput of 21MPixels/s. From this it is proved that this algorithm has high performance than other existing VLSI correcting designs.

REFERENCES


