Scientia Iranica B (2015) 22(6), 2150–2162

SCIENTIA IRANICA

Sharif University of Technology
**Scientia Iranica**
*Transactions B: Mechanical Engineering*
www.scientiairanica.com

# Neuron-based VLSI architecture for real-time camera distortion correction

**C.-H. Chen*** **and T.-K. Yao**

*Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan (R.O.C).*

Received 22 May 2014; accepted 17 August 2015

**KEYWORDS**
Distortion correction;
Neural network;
Wide-angle view;
VLSI.

**Abstract.** This paper proposes the efficient VLSI architecture of camera distortion correction, based on a Neural Camera Distortion Model (NCDM). Conventional imaging methods use over two kinds of models to correct the camera and lens distortions, but the NCDM uses a single model to immediately correct the geometry distortion and unsymmetrical manufacturing errors. The NCDM, with four neurons, performs a wide-angle distortion correction. The results show that the maximal corrected error in a whole image is less than 1.1705 pixels, and the MSE approaches 0.1743 between corrected and ideal results. The distortion correction by NCDM is 429× more accurate than the conventional approach. The chip size of NCDM is $1.51 \times 1.51$ mm$^2$ and contains 126 K gates using the TSMC 90 nm CMOS technology process. Working at 240 Mhz, this architecture can correct 30 frames and a Full-HD resolution video per second. Results show that the maximal corrected error in a whole image is less than 1.4 pixels, and the mean square error approaches 0.0376 between corrected and ideal results.

© 2015 Sharif University of Technology. All rights reserved.

## 1. Introduction

A low-price camera has many kinds of manufacturing flaws that exist in the lens groups, the holder and the sensor board. To correct the lower quality cameras, Brown et al. [1] presented a lens distortion model that includes the decentering lens distortion concept. Then, Wing et al. [2] explained decentering and thin prism distortions in the camera. He mathematically modeled these errors and used a calibration board to obtain the distortion pattern that is used to calculate the parameters of the lens distortion model. They use various different mathematical models implemented by software to, respectively, correct camera distortions.

For correcting wide-angle lens distortion in real time, Ansari et al. [3] presented a similar distortion polynomial model based on a Nonlinear Least Squares Curve Fitting (NLSCF) scheme to obtain the coefficients of the distortion polynomial. Different to the wide-angle lens distortion model of Wing et al. [2], Ansari[s polynomial model is easier for implementing VLSI architecture. So, most recent studies [4-8] have presented hardware accelerator designs based on Ansari[s method. However, the low-price camera still has others distortions that cannot be corrected by VLSI architecture, which means that for medical [9] or measuring purposes, the wide-angle camera cannot correct higher resolution video in real time.

This study proposes the efficient VLSI architecture of a camera distortion correction method using a Neural Camera Distortion Model (NCDM) to rapidly correct various lenses and manufacturing flaws of low-price cameras. The NCDM consists of an off-line calibration, a Multilayer Feed-Forward Neural Network (MFFNN) [10], and bi-linear interpolation. Because the calibration of NCDM is an off-line preprocess, it is

*. *Corresponding author. Tel.: +886-3-422-7151#35211; Fax: +886-3-422-2681*
*E-mail address: pierre@csie.ncu.edu.tw (C.-H. Chen)*

better that the calibration is implemented by software. In the correction process of NCDM, the MFFNN back maps Distortion Image Space (DIS) coordinates from Correction Image Space (CIS) coordinates. The bilinear interpolation resamples the image pixel if the back mapping DIS coordinates are located in the middle of two pixels.

The MFFNN uses parallel neuron calculating to effectively solve a variety of nonlinear problems for function fitting, data classification, and pattern recognition. The multilayer structure of the MFFNN can increase the nonlinear modeling capability, but the complex arithmetic of the network needs more calculating time, which limits MFFNN application in real time. To accelerate operation of the MFFNN by hardware, the active function of the neurons using the tan-sigmoid is a bottleneck for implementation, which is the exponential and dividing arithmetic.

The hardware arithmetic of the tan-sigmoid function can be implemented using analog [11,12] or digital circuits. The analog circuit can use a few CMOS gates to generate the analog signal to approximate the tan-sigmoid function. However, the tan-sigmoid uses a digital logic circuit to implement, and has several advantages, including better repeatability, flexibility, compatibility, testability, and anti-noise. Hence, for implementing the hardware arithmetic of the MFFNN, many researchers based their studies on digital implementations and have proposed various design approaches for the tan-sigmoid, and applications to obtain the optimum for nonlinear problems. To design hardware arithmetic that can generate an accurate tan-sigmoid function is a critical route in implementing a neural network.

Recently, implementation of a hardware accelerator for the tan-sigmoid can be classed by four methods; the look-up table, approximation, the Taylor series, and the coordinate rotation digital computer (CORDIC).

(i) **Look-up table:** For application in an industrial controller, Orlowska-Kowalska et al. [13] used a software program to automatically generate the LUT of the activation function of the NN for Field-Programmable Gate Array (FPGA) implementation. However, to generate a more accurate activation function, the LUT needs more memory space to memorize the sample points on the curve of the activation function. Then, Himavathi [14] analyzed the activation function, and found the best size of LUT was in the neural network (NN). Increasing the accuracy of the LUT output will increase memory size exponentially. Generally, the LUT approach is suitable for applications that do not require higher accuracy.

(ii) **Approximation:** Savich et al. [15] and Hariprasath and Prabakar [16] use several simple linear functions to fit the tan-sigmoid, which use the lowest resources to implement tan-sigmoid arithmetic without memory. Using a linear function makes the approximation approach difficult for increasing the precision of the result. The approximation approach is suitable for applications with lower accuracy than the LUT methods.

(iii) **Taylor series:** Zhou et al. [17] used a 7 order Taylor series and a Look-Up-Table (LUT) to generate the exponential function, which is operated in the floating point number. In this method, if the orders of the Taylor series are enough, the exponential arithmetic can generate accurate results, as well as the software program. However, the Taylor series approach uses many resources to implement the factorial arithmetic.

(iV) **CORDIC:** Zhu and Chen [18] used a CORDIC approach to implement the exponential arithmetic, which generates more accurate results than Taylor series methods with the same resources. CORDIC uses addition and subtraction to gain exponential results. Its arithmetic is simple and regular, and it is suitable for implementing hardware with Very-Large-Scale Integration (VLSI) technology. Zhu's approach operates the exponential function between 0 and 10, which applies to the probabilistic neural network. In this study, the tan-sigmoid of the MFFNN operates the exponential arithmetic from -16 to 16. For this requirement, this study analyzes the output characteristics of the CORDIC arithmetic, and proposes a modified design of the CORDIC arithmetic for generating higher accuracy results of the exponential.

## 2. Neural camera distortion model

The Back Propagation Neural Network (BPNN) [19] consists of the feed-forward stage, which is the Multilayer Feed-Forward Neural Network (MFFNN), and the error back-propagation stage. Because most problems in the real world are nonlinear, the MFFNN uses a multilayered structure to enhance the adjustability of the nonlinear output results. The MFFNN learns from the supervised learning method, which is an Error Back-Propagation (EBP) method for error correction learning. The neuron bias modification method is similar to that of the neuron weight. In the training epoch, all the training patterns use a feed-forward operation to calculate the results. Then, the back-propagation operation estimates the errors between the results and the target patterns. According to the errors, the weights and biases of the neural network are modified. The BPNN continues using the training

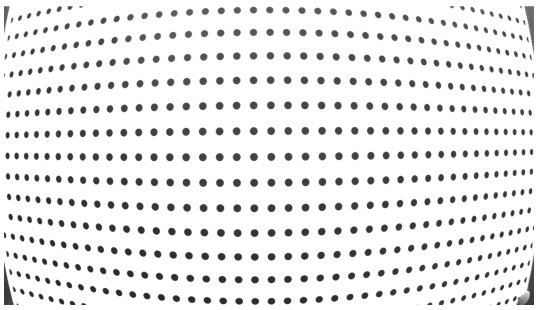

**Figure 1.** Calibration image of 120° wide-angle lens.

epoch of the EBP method to train until the stopping condition is met. Upon completing the EBP method training, the BPNN stops back-propagation and only uses feed-forward to back map the camera distortion surface.

NCDM models various lens distortions with a MFFNN, which can simplify the correction processing of lens distortions. The various mathematical models of lens distortion are integrated into one model. However, if the MFFNN directly models the 3-D distortion curve surface, computations of the MFFNN that have more than three-layers or too many neurons in the hidden layer are very complex and slow. Researchers do not popularly use the neural network to model lens distortion.

This study uses a 105° and a 120° wide-angle lens to build the NCDM. Distortion of a low-price 120° wide-angle lens consists of many convex and concave lenses, and the optical characteristics of this wide-angle lens are more complex. Figure 1 shows a calibration image that uses a $1920 \times 1080$ pixels resolution camera with a 120° wide-angle lens to capture the calibration board. Because of distortion, the distance between adjacent dots is different between the most peripheral and the center area of the Detected Calibration Dots (DCD), which has little distortion. The average distance is estimated using the nine adjacent dots in the center, and the Virtual Calibration Dots (VCD) use the average distance to build the grid dots. The built grid dots have the same distance between the neighboring dots.

Figure 2(a) shows the VCD and the DCD of a $1280 \times 1024$ resolution camera with a 105° wide-angle lens, which uses the estimated center from the DCD to align with each other in the image space. Figure 2(b) shows the VCD and DCD of a $1920 \times 1080$ resolution camera with a 120° wide-angle lens.

Figure 2(c) shows the distortion depth of a 105° wide-angle lens that estimates the difference between the DCD and the VCD, and the maximum distortion is more than 150 pixels in the most peripheral region. Figure 2(d) shows that the maximum distortion of the 120° wide-angle lens is more than 100 pixels in the most peripheral region. From the distortion surface, the quality of the 105° wide-angle lens is better than that of the 120°lens.

To reduce the network size of the MFFNN, the training pattern of the MFFNN is very important. For generating the training pattern, the VCD is sequentially inputted into the input neurons of the MFFNN. According to the calculated errors of the output results between the results of the MFFNN and the DCD, the EPB training method modifies the weights and biases of the MFFNN.

In this paper, we propose a method that reduces the complexity of the MFFNN using a method to reduce the complex 3-D distortion surface. The best operating range of the MFFNN is the region that includes positive and negative floating numbers, and between 1 and -1. Generally, researchers use the lens center to calculate the radial distortion vectors. However, in low-price cameras, the lens center is not a known parameter, and it is very hard to estimate the right location coordinates. Differentiating between the lens center and the point of the DIS is not a good training pattern because it is too big for a simple MFFNN.

Hence, the NCDM uses the difference between the points of the DIS and the CIS to train the MFFNN, which is given by:

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \end{bmatrix} - \begin{bmatrix} x_c \\ y_c \end{bmatrix}, \tag{1}$$

where $\begin{bmatrix} x_d & y_d \end{bmatrix}^T$ is the point in the DIS, and $\begin{bmatrix} x_c & y_c \end{bmatrix}^T$ is the corresponding point in the CIS. The curved surface in Figure 3(a) is the $x$-axis direction differentiation of the surface in Figure 2 between the distortion of the $x$-axis coordinates and their correction. In Figure 3(b), the curved surface of $dy$, which is the difference between the distortion and correction of $y$-axis coordinates, is similar to the curved surface of Figure 3(a). Hence, the curved surface of Figure 2 decomposes into two different directions of the curved surface.

After reducing the distortion surface, the neural based back mapping uses an efficient NCDM to model the differentiation of the $x$- and $y$-directions. In Figure 4, besides the input and output layer, the structure of the NCDM has a single hidden layer, which is a single hidden layer MFFNN. In the hidden layer, the neurons use the tan-sigmoid function to normalize the output range of the hidden neuron. The neuron in the output layer gathers the results of the hidden layer, and are, respectively, multiplied by the weight of the output layer. Then, the neuron outputs sum up all the multiplied results, which is a distortion differentiation of the camera distortion surface.

The NCDM uses 4 neurons to model the distortion surfaces of Figure 5. After the BP training, the correction error of a 105° wide-angle lens in Figure 5(a)

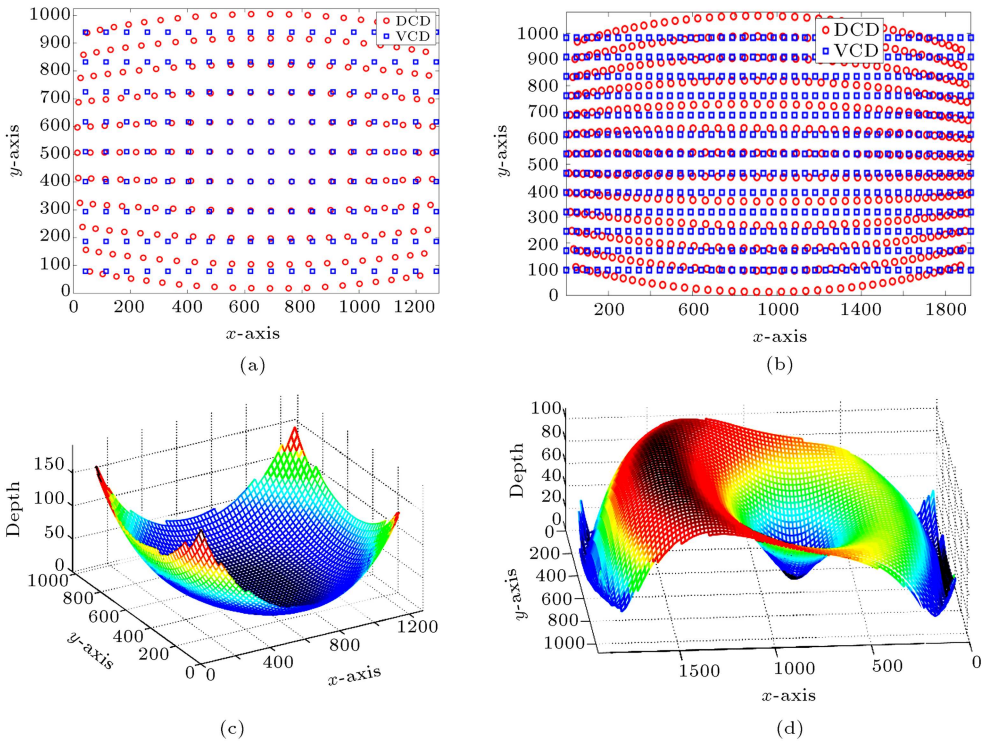


**Figure 2.** Virtual calibration and distortion surface: (a) Calibration dots pattern for 1280 × 1024 resolution camera with 105° lens; (b) calibration dots pattern for 1920 × 1080 resolution camera with 120° lens; (c) distortion surface for 105° lens; and (d) distortion surface for 120° lens.

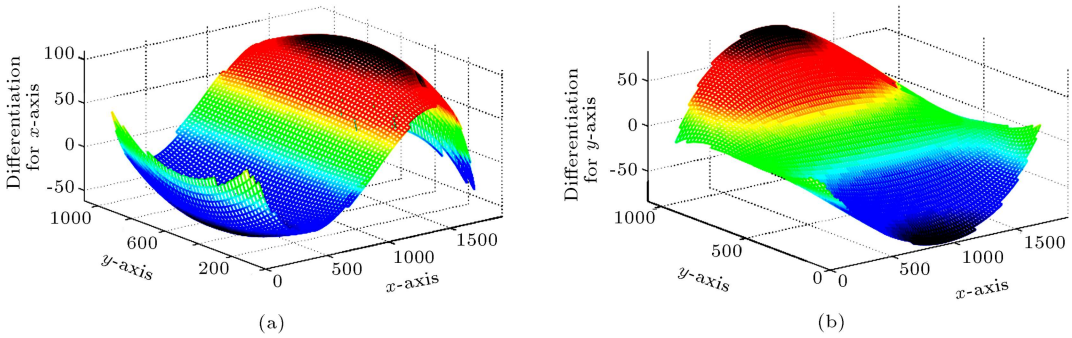


**Figure 3.** Differential result of distortion surface for (a) $x$-axis and (b) $y$-axis.

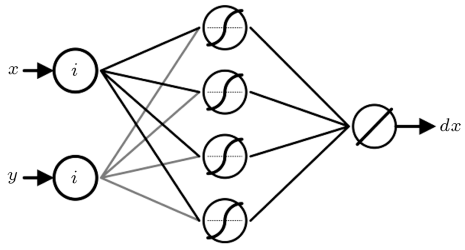


**Figure 4.** Neural camera distortion model.

is under 1.6 pixels. Figure 5(b) shows the maximum correction error of a 120° wide-angle lens in the surround arrives at 1.4 pixels.

In Figure 6, the trained NCDM with 4 neurons is used to correct the distortion image, and the human eye is unable to observe distortion in the surround.

## 3. VLSI architecture of neural processor

### *3.1. Arithmetic circuit design*

Figure 7 is the arithmetic circuit of a neural processor for the NCDM that consists of 2-input hidden neurons and 4-input output neurons. Because the input data of the correction neural network are 2-D coordinates $(x, y)$, the hidden layer in the MFFNN consists of four 2-input neurons. To operate the 2-input neurons at higher operating frequency, the hidden neuron is designed using pipeline architecture. After the input value is multiplied by the weight, the pipe registers store the multiplied results. Then, the summed result of the multiplied results, and the bias store in the next stage pipe register is the net signal. The MFFNN in this study uses the register to store the descriptor of

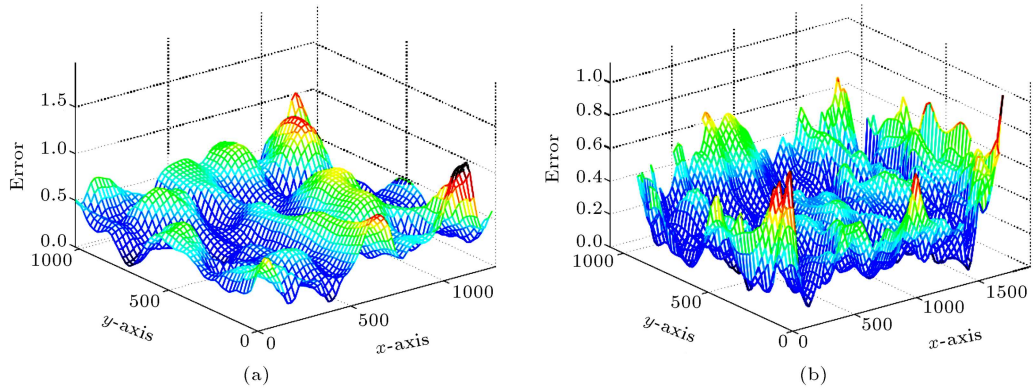


**Figure 5.** Training error of (a) 105° and (b) 120° wide-angle lens with 4 neurons.

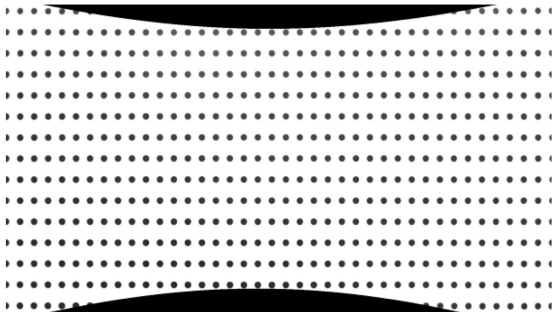

**Figure 6.** Corrected image of 120° wide-angle lens with 4 neurons.

the neural network that controls the weights and biases of the hidden neurons in the neural network. In the memory mapping of the bus system, the bus interface operates the descriptor register to write a value to register, or to read a value by the microprocessor.

The arithmetic structure of the output neuron is similar to the 2-input neuron. The pipe register stores the multiplied results of $a_{0\sim3}^{\mathrm{HN}}$ and $w_{0\sim3}^{\mathrm{ON}}$. Every two multiplied results in the pipe register use an adder to sum, and the summed results store in the second stage pipe register. Then, the third stage pipe register preserves the summed result that sums up the summed results of the second stage and the bias. To make it simple to connect the neuron at the next stage, the input value of the neuron is limited to a range between 1 and -1. Thus, the neurons can constitute any structure of the neural network for any purposes.

When the wide-angle lens is changed to others, the neural network can use the same neuron to correct the distortion. The input pixel coordinates of other cameras only need to normalize to a range between 1 and -1. The neural network uses new normalized data to train the weights and biases for a new camera. The normalized design of the neuron is a re-use structure. For different applications, the neural network only redesigns the normalization arithmetic.

The arithmetic operation in the neuron uses the fixed point to calculate. The input and output widths of the neuron in the neural network for camera correction are 24 bits. The normalized value can completely utilize the 24-bit width of the multiplier and the adder in the neuron, which can output most calculations with precision. By controlling the weights and biases to a fixed point value within a 24-bit width, the limited range can ensure that the multiplied and summed operations in the neuron do not overflow. Then, the output range of the neuron is limited to a fixed range,

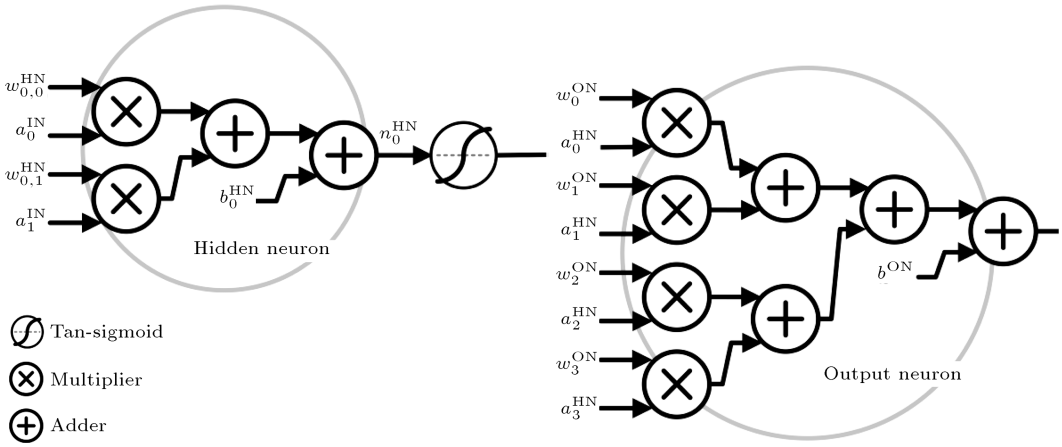


**Figure 7.** Arithmetic circuit of neural processor.

which simplifies the effective operating range of the tan-sigmoid arithmetic.

### 3.2. Tan-sigmoid acceleration

The tan-sigmoid function:

$$f(n)_{\text{tansig}} = \frac{2}{1+e^{-\gamma n}} - 1, \tag{2}$$

is the most critical function in the neural network for VLSI implementation, which consists of an exponential and a divider. According to the review in Section 1, implementation of the tan-sigmoid function has four types:

(i) The operating range of the tan-sigmoid is divided into several regions, and the relation between the input value and corresponding output are stored in a look-up table. If the applications do not need higher output precision, the look-up table is a rapid approach;

(ii) The operating range of the tan-sigmoid is also divided into several regions, and every region of the tan-sigmoid uses an approximate linear polynomial to fit the output result. This approach can provide a more similar output precision, and is a lower cost implementation than the look-up table;

(iii) The above approaches do not appropriately use the same application and need higher precision. To accurately correct camera distortion, we propose a method to improve the output characteristics of CORDIC arithmetic.

It is difficult for the exponential function of the tan-sigmoid to generate the exponential function using hardware arithmetic. In this study, a coordinate rotation digital computer (CORDIC) is used to implement the hardware arithmetic of the exponential function. However, CORDIC cannot directly generate the value of the exponential, which is operated in rotational mode to generate the results of the $\cosh(\theta)$ and $\sinh(\theta)$. In Eq. (3), the exponential value of $\theta$ is the summed result of $\cosh(\theta)$ and $\sinh(\theta)$:

$$e^{\theta} = \cosh(\theta) + \sinh(\theta). \tag{3}$$

Eq. (4) is the iterative function of CORDIC based on a pseudo-rotation method. To generate the values of $\cosh(\theta)$ and $\sinh(\theta)$, the CORDIC is operated in rotational mode to calculate the hyperbolic function:

$$\begin{cases} x_{k+1} = x_k - m\delta_k y_k 2^{-k} \\ y_{k+1} = y_k + \delta_k x_k 2^{-k} \\ z_{k+1} = z_k - \delta_k \varepsilon_k \end{cases} \tag{4}$$

where $m = -1$ for the hyperbolic function. In the rotational mode, $\delta_k$ in Eq. (5) uses the sign of $z_{k+1}$ to decide if the next iteration is positive or negative rotation:

$$\delta_k = \begin{cases} -1, & \text{if } z_k < 0 \\ 1, & \text{otherwise.} \end{cases} \tag{5}$$

where $k > 0$. In (6), $\varepsilon_k$ is the rotating angle at the $k$ stage:

$$\varepsilon_k = \tanh^{-1}(2^{-k}), \tag{6}$$

where $k > 0$. After several iterative operations, $z_{k+1}$ almost closes to 0. $x_{k+1}$ and $y_{k+1}$ converge in a hyperbolic term that can be written as:

$$\begin{cases} x_{k+1} = K_h(x_0\cosh(z_0) + y_0\sinh(z_0) \\ y_{k+1} = K_h(y_0\cosh(z_0) + x_0\sinh(z_0)) \end{cases} \tag{7}$$

where $k > 0$. $K_h$ is produced from each pseudo-rotation operation, which is given by:

$$K_h = \prod_{i=0}^{N} \sqrt{1 - 2^{-2i}}, \tag{8}$$

where $N$ is defined as the times of the pseudo-rotation operation. $K_h$ is a constant, as $N$ is greater than 25. Figure 8 shows the result of the exponential using the HBCORDIC and standard C/C++ library, and the obvious difference is between 0 and 0.04.

Figure 9 is the output error between the HBC and the standard C/C++ library (STD), which has a two part obvious difference. Between 0 and 0.04, the HBC cannot converge to the objective, which is the result of the STD. The second difference is generated from the front of several of the rotation operations. The summed angle of the twice rotating operation cannot use the rotating operation to compensate, which is an over rotating error.

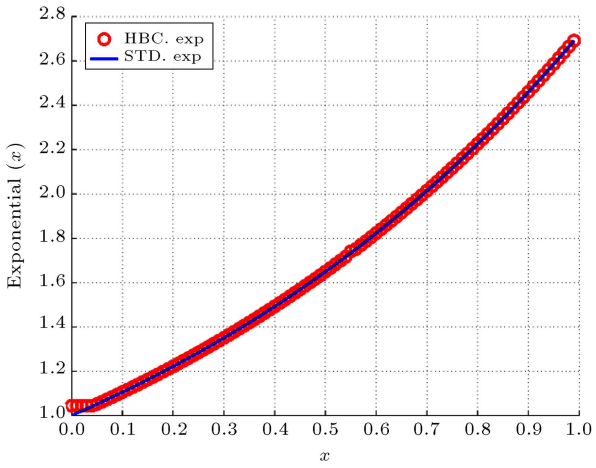


**Figure 8.** Output characteristics of exponential using hyperbolic CORDIC.

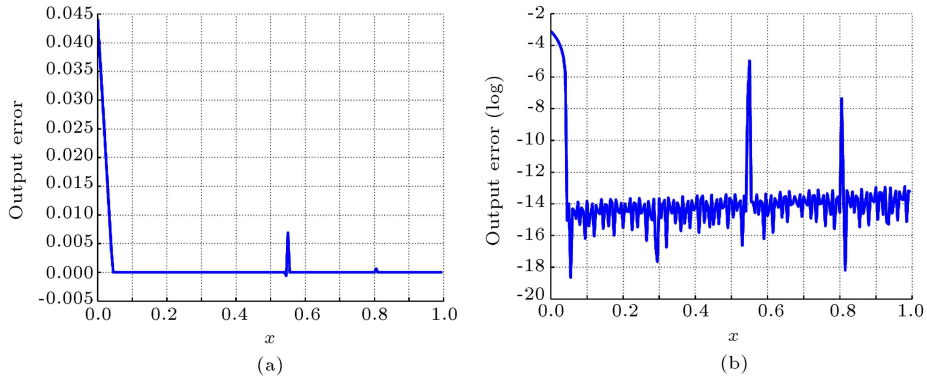


**Figure 9.** Output error of tan-sigmoid circuit: (a) The subtraction output error; and (b) the logarithm output error between HBC and STD.

Figure 9(b) is the log result of the output error, and the error in most areas is between $10^{-15}$ and $10^{-14}$. Because the operating range of the tan-sigmoid is from 16 to -16, this study proposes a method to modify the HBC, which can improve the error between 0 and 0.04.

The error of the HBC between 0 and 0.04 is generated from the rotating operation of HBC not starting from 0, and thus, the result of the HBC cannot converge to the objective. In Eq. (9), $\varepsilon_k$ is an unlimited number when $k$ is 0. Hence, an added negative rotating operation [20] is used to change the starting angle of the HBC:

$$\varepsilon_k = \tanh^{-1}(1 - 2^{k-2}), \tag{9}$$

where $k = 0$. A new negative pseudo-rotation (10) uses the rotating angle of $\varepsilon_0$ to modify the error of the HBC between 0 and 0.04:

$$\begin{cases} x_{k+1} = x_k - m\delta_k y_k(1 - 2^{k-2}) \\ y_{k+1} = y_k + \delta_k x_k(1 - 2^{k-2}) \\ z_{k+1} = z_k - \delta_k \varepsilon_k \end{cases} \tag{10}$$

where $k = 0$ and $m = -1$. Because of adding a negative rotation, the scale-factor $K_h$ must also be multiplied by a scaling value for the added negative pseudo-rotation, and is given by:

$$K_h = \sqrt{1 - (1 - 2^{(i-2)})^2} \prod_{i=0}^{N} \sqrt{-2^{-2i}}. \tag{11}$$

Figure 10 is the result of the exponential using Modified HBC (MHBC) and the STD. Because the MHBC added a negative pseudo-rotation, the convergence range of the MHBC expands to about ±2.

Figure 11(a) is the output error of the exponential between the MHBC and the STD. Because of adding a negative pseudo-rotation, the region of the over rotating error in Figure 9(a) between 0.5 and 0.6 moves close to 0.4 in Figure 11(a). The maximum error in Figure 11(a) is close to ±0.12. To amplify the few

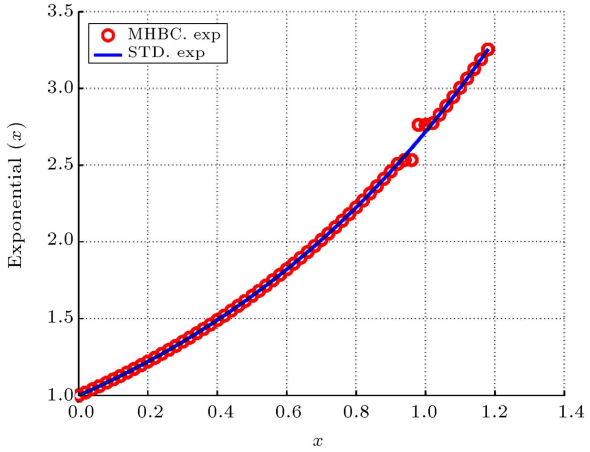


**Figure 10.** Exponential output characteristics using modified hyperbolic CORDIC.

errors, for the most part, in Figure 11(a), the results use the logarithm function to calculate their negative exponent, which is shown in Figure 11(b). Most errors in Figure 11(a) are less than $10^{-14}$, besides the region at $0.4 \sim 0.5$ and $0.9 \sim 1.0$.

However, the convergence range in Figure 11(a) has a critical section between 0.4 and 0.45.

The neuron in the ANN needs the tan-sigmoid (2) to converge the multiple-accumulated results of the weights and biases at a range between 1 and −1. The calculating range of the multiple-accumulate of the neuron is from -15 to 15.

To expand the operating range of the hyperbolic CORDIC, most studies [18,21] based on CORDIC divide the operating range of the EXP into several regions. Ansari [21] divided the $z_0$ in Eq. (3) into an integer and fraction, and preprocesses the integer results of the EXP that is stored in the lookup table. The expanded $\exp(-z_0)$ is calculated from:

$$\exp(-z_0) = \exp(-z_{\text{integer}}) * \exp(-z_{\text{fraction}}), \tag{12}$$

where $-z_{\text{integer}}$ is the negative integer part of $-z_0$, and $-z_{\text{fraction}}$ is the fraction part of $-z_0$, that is

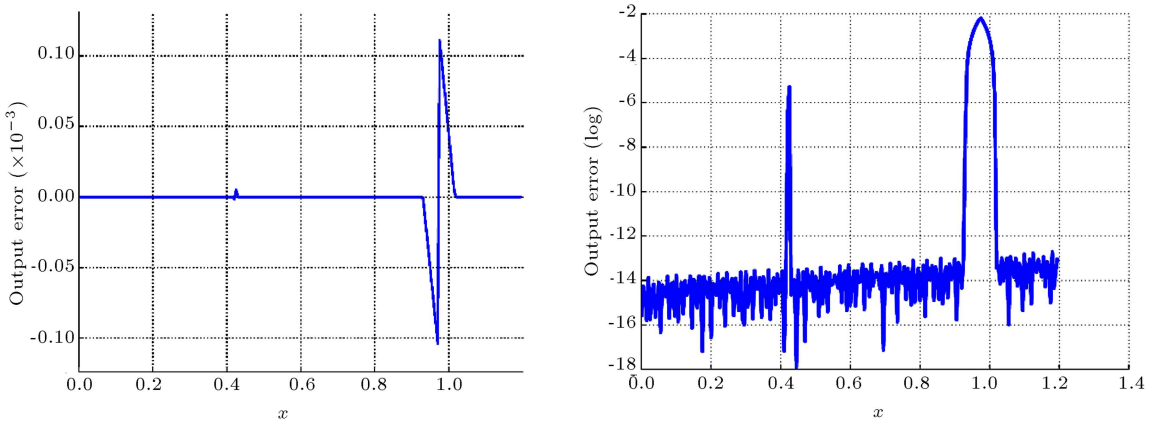


**Figure 11.** (a) The subtraction error, and (b) the logarithm error of the exponential between the MHBC and STD.

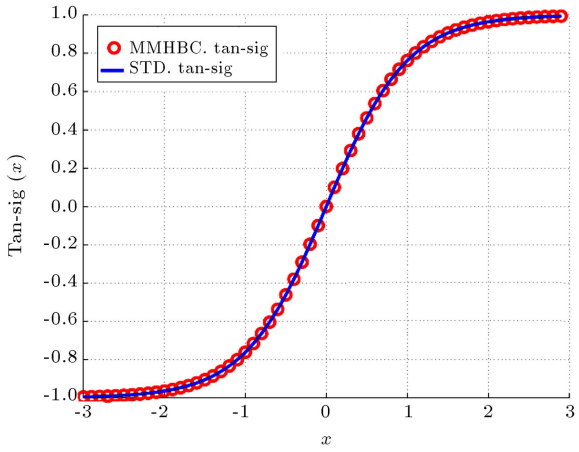


**Figure 12.** Tan-sigmoid output characteristics using mixed MHBC and STD.

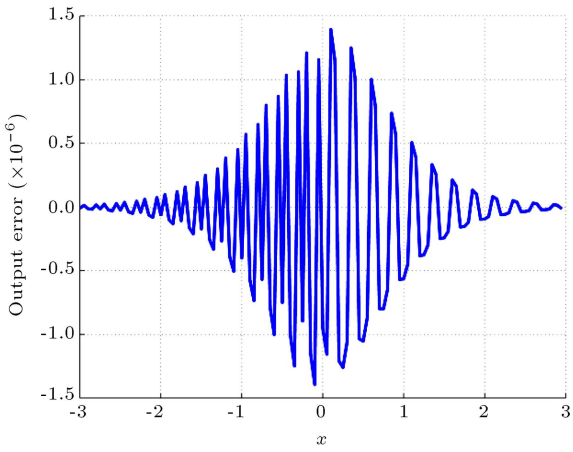


**Figure 13.** Output error using MMHBC.

$0 \leq -z_{\text{fraction}} < -1$. The HBC in the [21] is used to generate the $\exp(-z_{\text{fraction}})$, and the preprocessing $\exp(-z_{\text{integer}})$ multiplied by the result of the HBC is $\exp(-z_0)$, which effectively expands by more than $\pm 1$.

However, the MHBC generates the error regions as 0.25. To keep out error regions, the exponential rewrites to:

$$\exp(-z_0) = exp(-z_{\text{integer}}) * \exp(-z_{\text{half}}) * \exp(-z_{\text{quad}}) * \exp(-z_{\text{fraction}}). \quad (13)$$

Figure 12 shows the output characteristics using Mixed MHBC (MMHBC), and that the results of the MHBC are very like the result of the standard tan-sigmoid.

Figure 13 shows the output error of the tan-sigmoid between the MMHBC and the standard; this figure shows that the error is less than $10^{-5}$.

Figure 14 shows the MSE results of three different length fixed points. It shows that as the length of the fixed point is more than 20-bit, the increased ratio of the MSE is less than the fixed point using 20-bit.

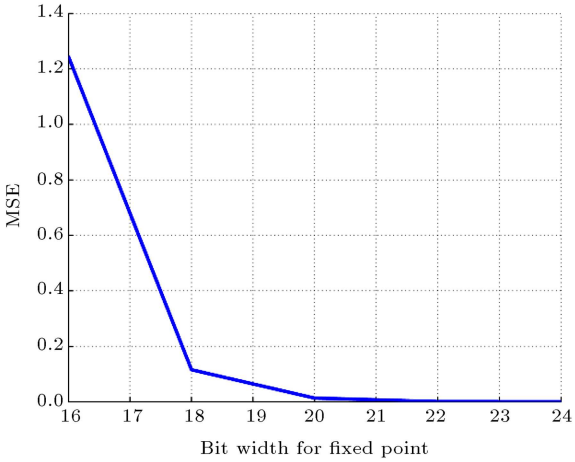


**Figure 14.** MSE of the tan-sigmoid using three different widths of fixed point implementation.

According to this result, the tan-sigmoid of this study uses a 20-bit fixed point to implement.

In Eq. (2), after calculating the exponential, the result must be inversed. However, if using a divider to inverse the exponential result, the hardware arithmetic

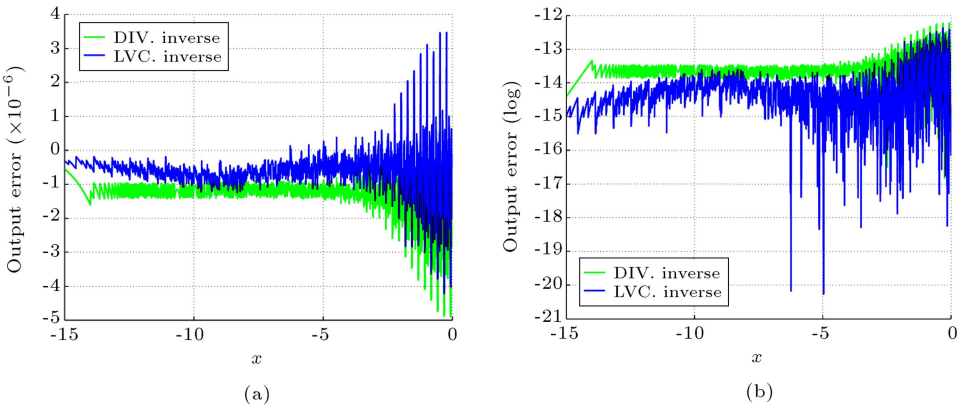


**Figure 15.** (a) Output error and (b) logarithm of output error using LVC and DIV.

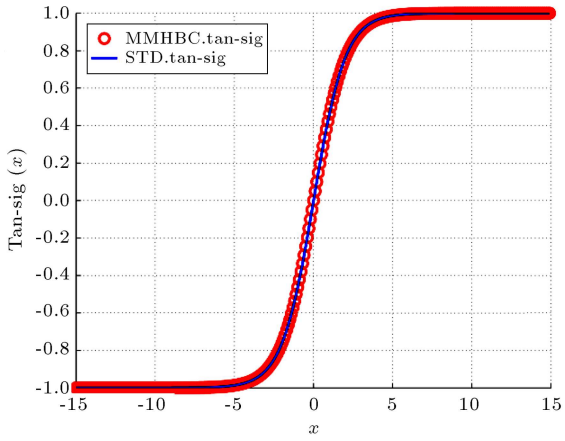


**Figure 16.** Output characteristics of the tan-sigmoid using MMHBC and STD in full range.

will need a very large area to implement the divider. For less cost in implementing the inverse arithmetic, this study uses a linear CORDIC arithmetic to replace the divider.

Figure 15(a) shows the results of the tan-sigmoid using linear CORDIC and a general divider, and the two methods are operating at a fixed point domain. Figure 15(b) shows the logarithmic output error of the LVC and the general divider.

Figure 16 shows the output characteristics of the tan-sigmoid using MMHBC and LVC from -15 to 15, and shows that the result is very like the standard tan-sigmoid using a floating point.

Figure 17(a) shows the output error of the tan-sigmoid using MMHBC and LVC from -15 to 15. Figure 17(b) shows the logarithm of the error in full using a range, and shows that the output of the MMHBC is very accurate.

Figure 18 is the arithmetic structure of the tan-sigmoid function. The bit-sequence separator is used to separate the input value into sign bit, region bit, and least fraction bit from the net of the neuron.

Because the output of the tan-sigmoid is symmetric, with respect to the zero-axis, the sign bit of the net is stored to a shift buffer, and is used to decide if the output of the tan-sigmoid is positive or negative in the finial pipe-line stage. Hence, the tan-sigmoid can be calculated from $e^z$ and $e^{-z}$.

Because the arithmetic of the tan-sigmoid uses a divider to invert the term of $1 + e^{-z}$ in Eq. (2), the MHBC in this study is only used to calculate $e^{-z}$. The result is 1, as $z$ is 0. Then, as z increases in a negative direction, the result of the EXP is closer and closer to 0.

So, the result ranges of $1 + e^{-z}$ are from 2 to 1. The complex of the divider arithmetic in Eq. (2) can be controlled, as the divisor and dividend ranges are fixed, and the dividend is less than or equal to twice the divisor, which is important when using the linear vector CORDIC to implement the dividing arithmetic in Eq. (2).

## 4. Experimental result

Recently, researchers have paid interest to camera distortion correction, because cameras on mobile devices and smart phones have become smaller and smaller. However, most methods are too complex and it is very hard to implement the hardware arithmetic. Ansari [7] firstly proposed a pipe-line hardware arithmetic based on a radial lens model and a least squares method to correct wide-angle distortion. According to Ansari[s method, Chen [4] proposed a more optimum implementation method that sacrifices correction precision to reduce the area of the chip. Table 1 shows the comparison of implementation resources.

Although the gates count of the NCDM in this study is higher than others, the NCDM can very accurately correct various errors in low-price cameras in one process. The hardware arithmetic of the others only corrects wide-angle distortion, which means the corrected precision is influenced by manufacturing flaws.

Table 2 shows an error comparison of the results of the NCDM with different neuron numbers in hidden

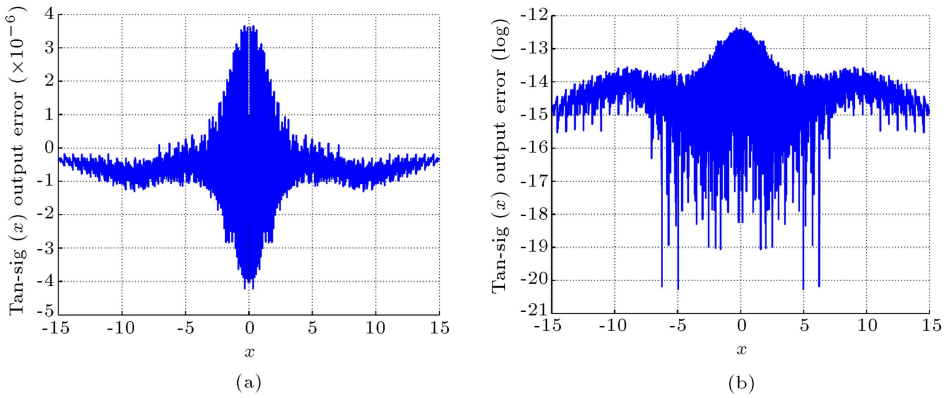


**Figure 17.** (a) Output error and (b) logarithm of the tan-sigmoid using MMHBC in full range.

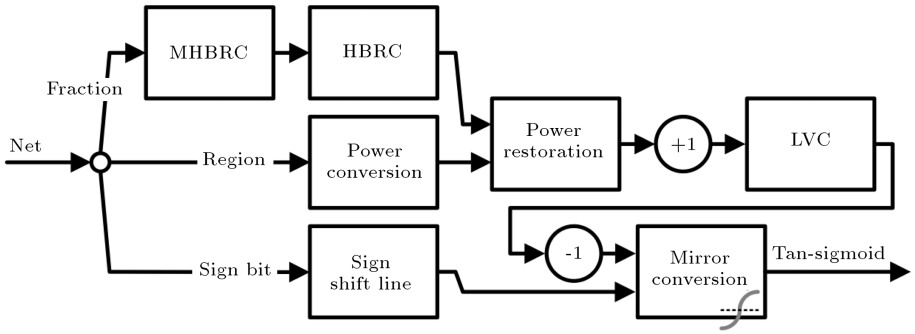


**Figure 18.** The tan-sigmoid arithmetic using MHBRC.

**Table 1.** VLSI implementations comparison.

| | [7] | This work | [7] | [4] | This Work |
|---|---|---|---|---|---|
| FPGA/ASIC process | Altera Apex EP20K600EBC652 | Altera Cyclone V 5CEFA7F31C8 | 0.18 $\mu$m | 0.18 $\mu$m | TSMC 90 nm |
| Logic elements | 18,344 | 20,218 | - | - | - |
| Gates | - | - | 44,992 | 28,662 | 263,080 |
| Frequency | 40 Mhz | 98.63 | 200 | 200 | 300 |
| Throughput | 30 Mpixels/s | 98.63 | 30 | 160 | 300 |

**Table 2.** Correction precision error comparison with different degree wide-angle lens.

| PBM | Order (N) | Lens | MSE | Min. error | Avg. error | Max. error |
|---|---|---|---|---|---|---|
| | 6 | 105° | 87.9597 | 0.1038 | 7.8584 | 21.1068 |
| | 7 | | 87.9452 | 0.0389 | 7.8571 | 20.9047 |
| | 8 | | 87.9316 | 0.0308 | 7.8566 | 20.8914 |
| **NCDM** | **Neurons (HN)** | | | | | |
| | 4 | 105° | 0.205017 | 0.000726 | 0.252331 | 4.621188 |
| | 5 | | 0.021510 | 0.001243 | 0.104972 | 0.573920 |
| | 6 | | 0.005172 | 0.000013 | 0.047855 | 0.357827 |
| | 4 | 120° | 0.037571 | 0.000074 | 0.122573 | 1.402213 |
| | 5 | | 0.007127 | 0.000023 | 0.056211 | 0.435714 |
| | 6 | | 0.004582 | 0.000368 | 0.044143 | 0.353196 |
| **This work** | **Neurons (HN)** | | | | | |
| | 4 | 120° | 0.150788 | 0.009167 | 0.336822 | 1.170525 |

layers, and the PBM with different polynomial order. In software using a floating point with a 120° wide-angle, the results show that MSE achieves 0.037571, as the number of the neurons in NCDM is 4, and the maximum error is 1.40221260 pixels. When the number of the neurons is 7, the maximum error in the whole image is under 0.32898513 pixels. However, in this camera sensor, the pixel width is 2.8 $\mu$m. If the number of neurons is greater than 4, the difference in the corrected image cannot be observed by the human eye. So, the optimum neuron number is 4 for the 120-degree wide-angle lens and the $1920 \times 1080$ resolution camera. Although the chip area of the NCDM is 10 times bigger than that in Ansari[s work [7], the NCDM is 429 times more accurate than the 6 order PBM that is based on Ansari[s method with the 105° wide-angle lenses.

Because the hardware arithmetic uses a fixed point with 120° wide-angle lenses, part of the correction errors are from the transformation errors of the floating point coordinates. The operating range of the NCDM using a floating point is -1∼1. However, to efficiently utilize the pipe-line register width of VLSI architecture, the range of the NCDM using a fixed point is $-0.9999 \sim 0.9999$. The result shows that MSE still achieves 0.150788011 as the NCDM hardware arithmetic with 4 neurons, and the maximum error is 1.170525455 pixels.

Figure 19 shows the precision errors of the neural processor, which are different errors of the whole image between post-simulation and ideal results. The hardware arithmetic of the neural processor uses a fixed point to implement, and the ideal software is implemented by a floating point numeric domain. Results show that most errors are lower than $10^{-6}$, and some errors are greater than $10^{-5}$ in the whole distortion image. Because the NCDM back maps to a distortion curve surface, the very small difference errors in the curve critical region cause the result of the software to map to the next curve, and the result of the hardware maps to the current curve, which

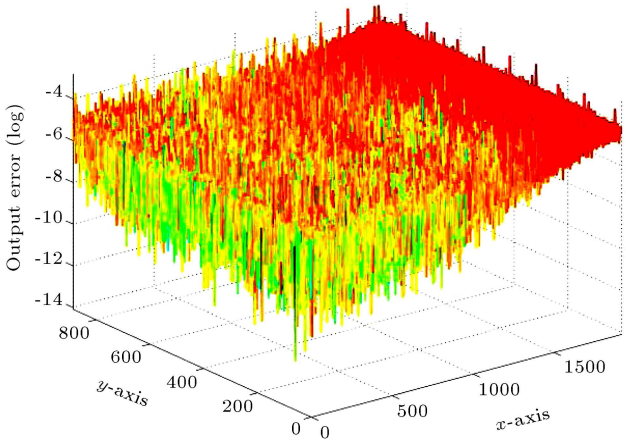


**Figure 19.** Neurons map on tape-out chip.

**Figure 20.** Captured distortion image from $1920 \times 1080$ resolution camera with 120°.

**Figure 21.** Corrected image obtained by the proposed neural correction chip.

generate the protruding points at every fixed distance. Following the reason, the MSE of the VLSI NCDM architecture is more than the NCDM using a floating point. As the case stands, the protruding points are not wrong events, because the CIS coordinates used for back mapping have transformation errors between the floating point and the fixed point.

Figure 20 shows a distortion image captured from a $1920 \times 1080$ resolution camera with a 120° wide-angle lens in which one can observe the clear wide-angle distortion.

Figure 21 is the corrected image of Figure 20, which shows the distortion region has corrected by the NCDM.

The chip size of NCDM is $1.51 \times 1.51$ mm$^2$, containing 126 K gates using the TSMC 90 nm CMOS technology process. Figure 22 shows a neuron map diagram illustrating the place and routing location of the Read Only Memory (ROM), the hidden neurons and output neurons. The ROM is used to store the power region of the exponential, which selects the shift times of the exponential arithmetic output. The utilization rate of the area surrounded by the pad is 79.22%.

## 5. Conclusion

In this paper, we design and implement an efficient VLSI architecture of a camera distortion correction

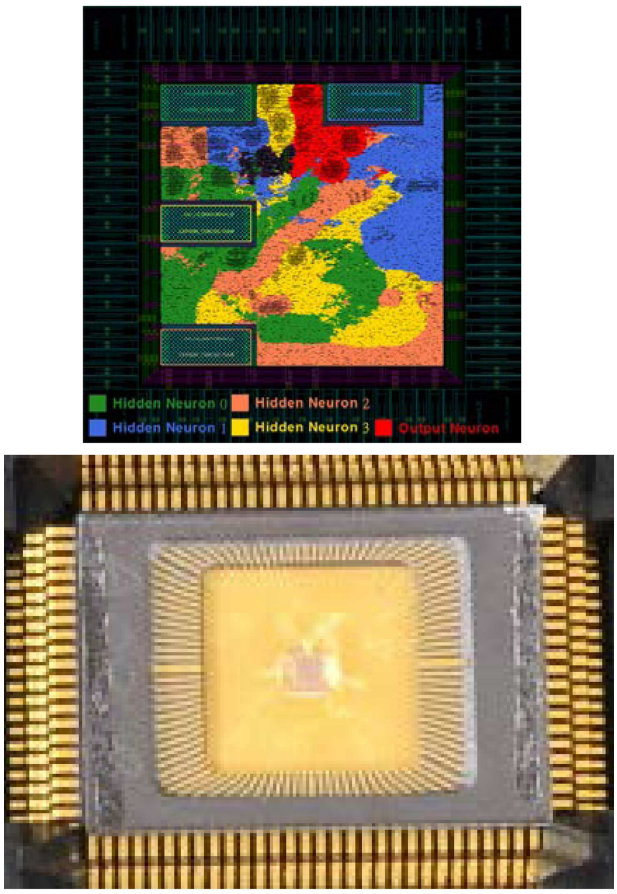


**Figure 22.** Neurons map on tape-out chip.

method based on the NCDM (Neural Camera Distortion Model). It can rapidly and accurately correct various lenses and the manufacturing flaws of low-price cameras. In the off-line calibration processing, the NCDM only captures a single calibration image to calculate the distortion vector, and does not need to estimate the optical center. The NCDM with four neurons performs the wide-angle distortion correction. Results show that the maximal corrected error in a whole image is less than 1.1705 pixels, and the MSE approaches 0.1743 between the corrected and ideal results. The distortion correction by NCDM is 429 times more accurate than the conventional approach. The implementation of a NCDM chip use a TSMC 90 nm CMOS technology process. The chip size is $1.51 \times 1.51$ mm$^2$ and contains 126 K gates. While the chip works at 240 Mhz, the NCDM can correct over 30 frames in a full HD resolution video per second, which meets real-time distortion corrections for streaming video applications.

## References

1. Brown, D.C. "Close-range camera calibration", *Photogramm. Eng. Remote Sens.*, **8**(37), pp. 855-866 (1971).
2. Wing, J., Cohen, P. and Herniou, M. "Camera calibration with distortion models and accuracy evaluation", *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **14**(10), pp. 965-980 (1992).
3. Ansari, K.V., Kumar, S. and Radhakrishnan, D. "A new approach for nonlinear distortion correction in endoscopic images based on least squares estimation", *IEEE Transactions on Medical Imaging*, **18**(4), pp. 345-354 (1999).
4. Chen, S.-L., Huang, H.-Y. and Luo, C.-H. "Time multiplexed VLSI architecture for real-time barrel distortion correction in video-endoscopic images", *Circuits and Systems for Video Technology, IEEE Transactions on*, **21**(11), pp. 1612-1621 (2011).
5. Chen, P.Y., Huang, C.C., Shiau, Y.H. and Chen, Y.T. "A VLSI implementation of barrel distortion correction for wide-angle camera images", *IEEE Transactions on Circuits and Systems Ii-Express Briefs*, **56**(1), pp. 51-55 (2009)
6. Qiang, L. and Allinson, N.M. "FPGA implementation of pipelined architecture for optical imaging distortion correction", in *Signal Processing Systems Design and Implementation, 2006. SIPS '06. IEEE Workshop on*, Banff, Canada, pp. 182-187 (2006).
7. Ngo, H.T. and Ansari, V.K. "A pipelined architecture for real-time correction of barrel distortion in wide-angle camera images", *IEEE Transactions on Circuits and Systems for Video Technology*, **15**(3), pp. 436-444 (2005).
8. Ansari, K.V. "Design of an efficient VLSI architecture for non-linear spatial warping of wide-angle camera images", *Journal of Systems Architecture*, **50**(12), pp. 743-755 (2004).
9. Melo, R., Barreto, J.P. and Falcao, G. "A new solution for camera calibration and real-time image distortion correction in medical endoscopy initial technical evaluation", *Biomedical Engineering, IEEE Transactions on*, **59**(3), pp. 634-644 (2012).
10. Hornik, K., Stinchcombe, M. and White, H. "Multilayer feedforward networks are universal approximators", *Neural Netw.*, **2**(5), pp. 359-366 (1989).
11. Choi, J., Bang, S.H. and Sheu, B.J. "A programmable analog VLSI neural network processor for communication receivers", *Neural Networks, IEEE Transactions on*, **4**(3), pp. 484-495 (1993).
12. Satyanarayana, S., Tsividis, Y.P. and Graf, H.P. "A reconfigurable VLSI neural network", *Solid-State Circuits, IEEE Journal of*, **27**(1), pp. 67-81 (1992).
13. Orlowska-Kowalska, T. and Kaminski, M. "FPGA implementation of the multilayer neural network for the speed estimation of the two-mass drive system", *Industrial Informatics, IEEE Transactions on*, **7**(3), pp. 436-445 (2011).
14. Himavathi, S., Anitha, D. and Muthuramalingam, A. "Feedforward neural network implementation in FPGA using layer multiplexing for effective resource

utilization", *IEEE Trans. Neural Netw.*, **18**(3), pp. 880-888 (2007).

15. Savich, A.W., Moussa, M. and Areibi, S. "The impact of arithmetic representation on implementing MLP-BP on FPGAs: A study", *Neural Networks, IEEE Transactions on*, **18**(3), pp. 240-252 (2007).

16. Hariprasath, S. and Prabakar, T.N. "FPGA implementation of multilayer feed forward neural network architecture using VHDL", in *2012 International Conference on Computing, Communication and Applications (ICCCA)*, pp. 1-6 (2012).

17. Zhou, F., Liu, J., Yu, Y., Tian, X., Liu, H., Hao, Y., et al. "Field-programmable gate array implementation of a probabilistic neural network for motor cortical decoding in rats", *Journal of Neuroscience Methods*, **185**(2), pp. 299-306 (2010).

18. Zhu, X.-P. and Chen, Y.-W. "Improved FPGA implementation of probabilistic neural network for neural decoding", in *2010 International Conference on Apperceiving Computing and Intelligence Analysis (ICACIA)*, pp. 198-202 (2010).

19. Rumelhart, D.E., Hinton, G.E. and Williams, R.J. "Learning internal representations by error propagation", in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, **1**, MIT Press, Ed., pp. 318-362 (1986).

20. Hu, X., Harber, R.G. and Bass, S.C. "Expanding the range of convergence of the CORDIC algorithm", *IEEE Transactions on Computers*, **40**(1), pp. 13-21 (1991).

21. Gisutham, B., Srikanthan, T. and Ansari, K.V. "A high speed flat CORDIC based neuron with multi-level activation function for robust pattern recognition", in *Fifth IEEE International Workshop on Computer Architectures for Machine Perception, 2000*, pp. 87-94 (2000).

## Biographies

**Ching-Han Chen** was born in Kinmen, Taiwan, in 1963. He received his BS and MS degrees in Geophysics from the National Central University, Taiwan, in 1986 and 1988, respectively, a DEA (Diplome d'Etudes Approfondies) in Informatique, Automatique et Productique, in 1992, and a PhD degree from Franche-Comte University, France, in 1995. He is currently Associate Professor in the Department of Computer Science and Information Engineering at the National Central University, Taoyuan County, Taiwan. His research interests include embedded system design, multimedia signal processing, and robotics.

**Tun-Kai Yao** received a BS degree in Electrical Engineering from I-Shou University, Taiwan, in 2005, and an MS degree in Electrical Engineering from the National University of Kaohsiung, Taiwan, in 2007. He is currently pursuing a PhD degree in the Department of Computer Science and Information Engineering at National Central University, Taiwan. His research interests include embedded systems design, VLSI design, image processing, and computer vision.