A Multi-Level Accumulation-Based Rectification Method and Its Circuit Implementation

Hyeon-Sik Son and Byungin Moon
School of Electronics Engineering, Kyungpook National University
Buk-gu, Daegu 41566, Korea
[e-mail: soc_shs1984@knu.ac.kr, bihmoon@knu.ac.kr]
*Corresponding author: Byungin Moon

Received September 19, 2016; revised December 15, 2016; accepted April 5, 2017; published June 30, 2017

Abstract

Rectification is an essential procedure for simplifying the disparity extraction of stereo matching algorithms by removing vertical mismatches between left and right images. To support real-time stereo matching, studies have introduced several look-up table (LUT)- and computational logic (CL)-based rectification approaches. However, to support high-resolution images, the LUT-based approach requires considerable memory resources, and the CL-based approach requires numerous hardware resources for its circuit implementation. Thus, this paper proposes a multi-level accumulation-based rectification method as a simple CL-based method and its circuit implementation. The proposed method, which includes distortion correction, reduces addition operations by 29%, and removes multiplication operations by replacing the complex matrix computations and high-degree polynomial calculations of the conventional rectification with simple multi-level accumulations. The proposed rectification circuit can rectify 1,280 × 720 stereo images at a frame rate of 135 fps at a clock frequency of 125 MHz. Because the circuit is fully pipelined, it continuously generates a pair of left and right rectified pixels every cycle after 13-cycle latency plus initial image buffering time. Experimental results show that the proposed method requires significantly fewer hardware resources than the conventional method while the differences between the results of the proposed and conventional full rectifications are negligible.

Keywords: Rectification, multi-level accumulation, distortion correction, stereo vision, stereo matching

A preliminary version of this paper appeared in EEECS 2016, January 20-22, Phuket, Thailand. This version includes the improvement of the method and its concrete analysis wth FPGA implementation results. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1D1A3B01015379).

https://doi.org/10.3837/tiis.2017.06.022 **ISSN : 1976-7277**

1. Introduction

In recent years, stereo vision has been the focus of considerable attention because of its use in various applications such as autonomous vehicles, intelligent robots, three-dimensional (3D) broadcasting systems, and mobile devices [1-5]. The main mechanism of stereo vision is the reconstruction of 3D information of a scene captured from two points of view. To acquire 3D distance, stereo vision finds disparity between corresponding points of a stereo image pair taken from two viewpoints of the same scene. However, it is a so-called "correspondence problem," requiring a very time-consuming procedure. To simplify the procedure of a time-consuming matching point search, rectification, which aligns epipolar lines in parallel with the x-axis using parameters obtained from calibration [6], should be performed as a preprocessing step in the stereo vision. With rectification, the search space for finding the corresponding pixel pair between a pair of stereo images can decrease from two to one dimension. Because every stereo vision algorithm requires rectification of input images, the cost of rectification is important [7].

The process of rectification finds the pixel coordinates of the input image corresponding to the pixel coordinates of the rectified image and then determines the pixel values of the rectified image through interpolation using the pixel values of the input image [8]. To find correspondence between the pixel coordinates of the input and rectified images, rectification can use two mapping schemes: forward or inverse mapping. Forward mapping computes the rectified target pixel locations from given pixel locations in the input image [9], and inverse mapping computes the pixel locations of the input image from given pixel locations in the rectified image [10, 11]. Forward mapping often produces undesired holes or overlaps [12] and requires a more complex interpolation process than inverse mapping [9]. Hence, inverse mapping is usually used in rectification. Also, the Camera Calibration Toolbox for MATLAB [13], widely used for camera calibration and rectification, uses the inverse mapping scheme because of its simplicity.

Inverse mapping rectification, which maps (x_r, y_r), the coordinates of the rectified image, onto (x, y), the coordinates of the input image, entails the following procedure: First, (x_n, y_n), undistorted coordinates in the normalized camera coordinate system, which corresponds to the input image, are calculated as

$$x_n = \frac{x_c}{z_c},\ y_n = \frac{y_c}{z_c},\ \text{and} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = R^T \times KK_{rect}^{-1} \begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix}, \tag{1}$$

where (x_c, y_c, z_c) are coordinates in the camera coordinate system that corresponds to the input image, KK_{rect} is the 3 × 3 intrinsic parameter matrix of the ideally aligned camera, and R is the 3 × 3 rectification rotation matrix. Second, for distortion correction, (x_{dist}, y_{dist}), the distorted coordinates in the normalized camera coordinate system, are calculated as

$$\begin{bmatrix} x_{radial} \\ y_{radial} \end{bmatrix} = \begin{bmatrix} (1 + kc_1 \cdot q^2 + kc_2 \cdot q^4 + kc_5 \cdot q^6) \cdot x_n \\ (1 + kc_1 \cdot q^2 + kc_2 \cdot q^4 + kc_5 \cdot q^6) \cdot y_n \end{bmatrix}, \tag{2}$$

$$\begin{bmatrix} x_{tangential} \\ y_{tangential} \end{bmatrix} = \begin{bmatrix} 2 \cdot kc_3 \cdot x_n \cdot y_n + kc_4 \cdot (q^2 + 2 \cdot x_n{}^2) \\ kc_3 \cdot (q^2 + 2 \cdot y_n{}^2) + 2 \cdot kc_4 \cdot x_n \cdot y_n \end{bmatrix}, \tag{3}$$

and

$$\begin{bmatrix} x_{dist} \\ y_{dist} \end{bmatrix} = \begin{bmatrix} x_{radial} + x_{tangential} \\ y_{radial} + y_{tangential} \end{bmatrix}, \tag{4}$$

where q^2 is equal to $(x_n^2 + y_n^2)$, kc_1, kc_2, kc_3, kc_4, and kc_5 are distortion coefficients determined by camera calibration, and (x_{radial}, y_{radial}) and $(x_{tangential}, y_{tangential})$ are coordinates subject to radial distortion and tangential distortion, respectively. Finally, (x, y), the pixel coordinates of the input image, are calculated as

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = KK \times \begin{bmatrix} x_{dist} \\ y_{dist} \\ 1 \end{bmatrix} = \begin{bmatrix} fc_{ix} & 0 & cc_{ix} \\ 0 & fc_{iy} & cc_{iy} \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_{dist} \\ y_{dist} \\ 1 \end{bmatrix} = \begin{bmatrix} fc_{ix} \cdot x_{dist} + cc_{ix} \\ fc_{iy} \cdot y_{dist} + cc_{iy} \\ 1 \end{bmatrix}, \tag{5}$$

where KK is the 3 × 3 intrinsic parameter matrix of the input camera. Elements fc_{ix} and fc_{iy} of KK are the focal length parameters of the camera, and the other elements cc_{ix} and cc_{iy} of KK are the principal point parameters of the camera. Rectification entails complex matrix calculations for coordinate transformation, shown in (1) and (5), and requires the calculations of seventh-degree polynomials for distortion correction, shown in (2), (3), and (4). To reduce the computational complexity of CL-based rectification, this paper[1] proposes a multi-level accumulation-based rectification method and its circuit implementation. The proposed rectification method replaces complex matrix multiplications and high-degree polynomial calculations of the conventional rectification with multi-level accumulations by modifying conventional rectification equations to difference sequences.

The rest of this paper is organized as follows. Section 2 discusses previous rectification methods and proposes a multi-level accumulation-based rectification method, and Section 3 proposes the circuit implementation of the proposed rectification method. Section 4 presents experimental results and analysis, and Section 5 concludes the paper.

2. Rectification Method

2.1 Related Work

Fusiello, Trucco, and Verri proposed a compact rectification algorithm that reduces computational overhead [14]. Jin et al. adopted an inverse-mapping rectification method that uses the transformation matrix [10], which presents corresponding relationships between pixels in the input image and those in the rectified image using

[1] This paper is fully extended from a conference paper [19] published in the Proceedings of the EEECS 2016.

$$x = \frac{x_h}{z_h},\ y = \frac{y_h}{z_h},\ \text{and} \begin{bmatrix} x_h \\ y_h \\ z_h \end{bmatrix} = H^{-1} \times \begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix}, \tag{6}$$

where (x_h, y_h, z_h) are homogeneous coordinates in the input image and

$$H^{-1} = KK \times R^T \times KK_{rect}^{-1} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}. \tag{7}$$

However, these rectification methods, unfortunately, cover only coordinate transformation without addressing distortion correction.

For real-time rectification, researchers have proposed two kinds of rectification hardware circuits. One is the look-up table (LUT)-based rectification circuit [8, 9, 15, 16], and the other is the computational logic (CL)-based rectification circuit [10, 11, 17]. Because of their relatively simple implementation, LUT-based hardware circuits are commonly used. However, the LUT-based approach, used with high-resolution images, requires significant memory resources [16]. Although CL-based rectification circuits do not require large memory resources, they have significantly higher computational overhead and latency than LUT-based circuits. This is because they require complex matrix multiplications for coordinate transformation and high-degree polynomial calculations for distortion correction.

To reduce memory usage for the look-up table, Jawed et al. adopted a look-up table reduction method using offline computation. As a result, the 1,024 × 768 entries of the original look-up table declined to 65 × 65 entries [8]. Because the method proposed by Jawed et al. is oriented to a specific camera system, it cannot be generally applied. Akin et al. introduced compressed LUT-based rectification, which solves the problem of using external memory blocks to store the large amount of coordinate mapping information [9]. However, this rectification method still requires considerable memory resources. In addition, because it ignores the fractional precision of computed coordinates, accuracy loss occurs in the disparity estimation. Vancea and Nedevschi proposed a parameterized rectification circuit design based on a hardware description language that can generate diverse hardware configurations based on adjustable parameters such as the resolution of images and the number of bits to store sub-pixel precision [15]. When the resolution is 640 × 512, however, it requires two 64-MB SDRAM blocks. Zicari proposed a CL-based rectification architecture with radial and tangential distortion correction [17]. However, to support complex matrix multiplications for coordinate transformation and high-degree polynomial calculations for distortion correction, this architecture requires numerous hardware resources.

To reduce the computational overhead of CL-based rectification, Hyun and Moon proposed a simplified rectification method [18]. When the rectified image has a resolution of $M \times N$, shown in **Fig. 1**, rectification processes pixels in the order shown in **Fig. 2**. Using the linearity of the pixel-unit calculation, Hyun and Moon dramatically reduced the complexity of mathematical calculations for CL-based rectification by replacing matrix multiplications with simple accumulations, shown in the following equation:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix} + \begin{cases} \begin{matrix} h_{11} \\ h_{21} \end{matrix} & \text{, when the column coordinate increases} \\ \begin{bmatrix} h_{12} - (M-1) \cdot h_{11} \\ h_{22} - (M-1) \cdot h_{21} \end{bmatrix} & \text{, when the row coordinate increases} \end{cases}, \quad (8)$$

where i is an index equal to $(M \times y_r + x_r)$, (x_i, y_i), referred to as index i, are pixel coordinates in the input image corresponding to the pixel coordinates (x_r, y_r) in the rectified image, and h_{11}, h_{12}, h_{21}, and h_{22} are the elements of H^{-1}, defined in (7). However, this method neglects distortion correction because it assumes that the camera is calibrated. Thus, it is not suitable for common uncalibrated cameras with lens distortion.

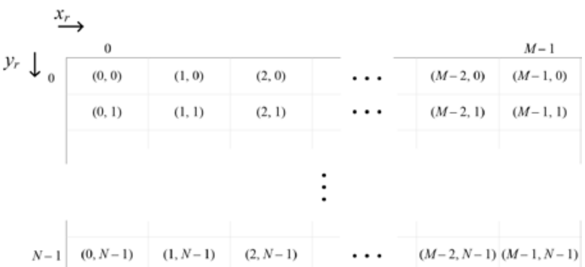


Fig. 1. Coordinates in the rectified image with a resolution of $M \times N$ (adapted from [18])

Fig. 2. Sequence of the pixel-unit calculation with a resolution of $M \times N$ (adapted from [18])

For the rectification of distorted stereo images, Son and Moon proposed accumulation-based rectification with radial distortion correction [19]. However, to reduce computational overhead, this method simplifies the radial distortion correction process, shown as

$$\begin{bmatrix} x_{radial} \\ y_{radial} \end{bmatrix} = \begin{bmatrix} (1 + kc_1 \cdot q^2 + kc_2) \cdot x_n \\ (1 + kc_1 \cdot q^2 + kc_2) \cdot y_n \end{bmatrix}. \quad (9)$$

Because this method erroneously assumes that q^4 in (2) is nearly equal to 1, it could not correct radial distortion properly and lost pixel information at image borders.

2.2 Proposed Rectification Method

The proposed rectification method uses the inverse mapping scheme to map rectified image coordinates (x_r, y_r) onto input image coordinates (x, y). To reduce computational complexity, we applied reasonable approximations to distortion correction. Tangential distortion occurs when the lens and the image sensor are not parallel. Nowadays, because of improvements in manufacturing technology, the effects of tangential distortion are negligible [20]. In addition, two coefficients, kc_1 and kc_2, are typically sufficient for distortion correction, and kc_5 is meaningful only in cases of severe distortion, such as in wide-angle lenses [21]. **Table 1** shows the distortion coefficients of the camera examples provided by BoofCV [22] and Camera Calibration Toolbox for MATLAB of Caltech [13], and the camera used in this paper. We extracted the distortion coefficients using the stereo camera calibrator APP in MATLAB [23] with two radial distortion models. One model uses two radial distortion coefficients, kc_1 and kc_2, while the other model uses three coefficients, kc_1, kc_2, and kc_5. **Table 2** shows tangential distortions calculated from the coefficients of **Table 1**. As shown in **Table 2**, the tangential distortions are small enough to be neglected. **Table 3** shows coordinate differences between the rectification results from the two radial distortion models. In the BoofCV and Caltech camera examples, even though the maximum coordinate differences are significant, the average differences are less than 0.32, and the percentages of x and y coordinates whose differences are greater than one are less than 8.4% and 4.43%, respectively. Furthermore, in the case of the camera used in this paper, the average x and y coordinate differences fall below 0.084, and the percentages of x and y coordinates whose differences are over one are less than 1.6%. These results confirm that the radial distortion model using two coefficients is sufficient for radial distortion correction.

Table 1. Extracted distortion coefficients of stereo cameras

Camera		Coefficient values			
		Two radial distortion coefficients		Three radial distortion coefficients	
		Left camera	**Right camera**	**Left camera**	**Right camera**
BoofCV	kc_1	-0.3548	-0.3545	-0.3693	-0.3545
	kc_2	0.1608	0.1488	0.2458	0.1517
	kc_3	0.0002	0.0004	0.0002	0.0003
	kc_4	-0.0011	-0.0018	-0.0008	-0.0020
	kc_5	0	0	-0.1529	-0.0247
Caltech	kc_1	-0.2915	-0.2883	-0.2765	-0.2974
	kc_2	0.1143	0.1045	-0.0052	0.1545
	kc_3	0.0009	-0.0002	0.0009	-0.0002
	kc_4	-0.0003	0.0002	-0.0002	0.0002
	kc_5	0	0	0.2524	-0.0746
Used in this paper	kc_1	-0.1232	-0.1356	-0.1281	-0.1411
	kc_2	0.5464	0.8740	0.7613	1.2750
	kc_3	0.0003	-0.0049	0.0003	-0.0049
	kc_4	0.0020	-0.0041	0.0020	-0.0041
	kc_5	0	0	-6.6711	-7.6942

Table 2. Tangential distortions in stereo cameras

Camera		Tangential distortions with the model of two radial distortion coefficients			Tangential distortions with the model of three radial distortion coefficients		
		Maximum	**Minimum**	**Average**	**Maximum**	**Minimum**	**Average**
BoofCV	$x_{tangential}$	-6.46e-10	-0.001832	-0.000490	-4.66e-10	-0.001437	-0.000371
	$y_{tangential}$	0.000863	0.000443	0.000706	0.000739	-0.000242	0.000081
Caltech	$x_{tangential}$	0.000119	-0.001011	-0.000133	0.000168	-0.000918	-0.000107
	$y_{tangential}$	0.001134	3.00e-11	0.000293	0.001091	2.92e-10	0.000289
Used in this paper	$x_{tangential}$	0.000475	4.29e-11	0.000130	0.000477	4.30e-11	0.000130
	$y_{tangential}$	0.000192	-0.000001	0.000012	0.000194	-0.000099	0.000012

Table 3. Coordinate differences of rectification results from two radial distortion models

Camera		Maximum difference		Average difference		Percentage of difference > 1	
		Left	**Right**	**Left**	**Right**	**Left**	**Right**
BoofCV	x	7.1202	2.4299	0.2791	0.1078	6.90%	1.55%
	y	4.9533	1.7144	0.1700	0.0592	2.84%	0.42%
Caltech	x	11.1673	2.1692	0.3163	0.0856	8.38%	0.57%
	y	7.3509	1.4598	0.1827	0.0608	4.43%	0.10%
Used in this paper	x	0.4662	2.6390	0.0171	0.0839	≒ 0%	1.59%
	y	0.2635	1.4752	0.0082	0.0321	≒ 0%	0.15%

Therefore, conventional rectifications commonly use only two coefficients, kc_1 and kc_2 for radial distortion correction [24, 25]. Thus, (2), (3), and (4) can be simplified as

$$\begin{bmatrix} x_{radial} \\ y_{radial} \end{bmatrix} = \begin{bmatrix} (1+kc_1 \cdot q^2 + kc_2 \cdot q^4) \cdot x_n \\ (1+kc_1 \cdot q^2 + kc_2 \cdot q^4) \cdot y_n \end{bmatrix} \tag{10}$$

and

$$\begin{bmatrix} x_{dist} \\ y_{dist} \end{bmatrix} = \begin{bmatrix} x_{radial} \\ y_{radial} \end{bmatrix}. \tag{11}$$

In addition, (1) can be rewritten as

$$x_n = \frac{x_c}{z_c},\ y_n = \frac{y_c}{z_c}, \text{ and } \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \times \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{bmatrix} c_{11} \cdot x_r + c_{12} \cdot y_r + c_{13} \\ c_{21} \cdot x_r + c_{22} \cdot y_r + c_{23} \\ c_{31} \cdot x_r + c_{32} \cdot y_r + c_{33} \end{bmatrix}, \tag{12}$$

and

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} = R^T \times KK_{rect}^{-1} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}^T \times \begin{bmatrix} fc_{rx} & 0 & cc_{rx} \\ 0 & fc_{ry} & cc_{ry} \\ 0 & 0 & 1 \end{bmatrix}^{-1}, \tag{13}$$

where elements fc_{rx} and fc_{ry} of KK_{rect} are the focal length parameters of the rectified image, and the other elements cc_{rx} and cc_{ry} of KK_{rect} are the principal point parameters of the rectified image. Because c_{31} and c_{32} are almost equal to 0 and c_{33} is very close to 1, z_c of (12) can be approximated to 1 [18]. This approximation is justified by **Table 4**, which shows the third-row vectors of the coefficient matrix of (13) for the camera examples provided by BoofCV [22] and Camera Calibration Toolbox for MATLAB [13], and the camera used in this paper. For this reason, by removing division by z_c, (12) can be simplified as

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} c_{11} \cdot x_r + c_{12} \cdot y_r + c_{13} \\ c_{21} \cdot x_r + c_{22} \cdot y_r + c_{23} \end{bmatrix} = \begin{bmatrix} c_{11} \cdot x_r + g(y_r) \\ c_{21} \cdot x_r + h(y_r) \end{bmatrix}, \tag{14}$$

where

$$\begin{bmatrix} g(y_r) \\ h(y_r) \end{bmatrix} = \begin{bmatrix} c_{12} \cdot y_r + c_{13} \\ c_{22} \cdot y_r + c_{23} \end{bmatrix}. \tag{15}$$

Table 4. Third-row vectors of the coefficient matrix of actual stereo cameras (updated from [18])

Camera		c_{31}	c_{32}	c_{33}
BoofCV	Left	−0.00000651	0.00000082	1.00183437
	Right	0.00000919	−0.00000082	0.99730203
Caltech	Left	0.00000913	−0.00000742	0.99865517
	Right	−0.00000113	0.00000745	0.99855221
Used in this paper	Left	0.00000532	-0.00000191	0.99772009
	Right	0.00000952	0.00000196	0.99401412

According to the pixel-processing sequence for rectification, shown in **Fig. 2**, column coordinate x_r of the rectified image increases by 1 for every pixel calculation while inverse mapping rectification is carried out within one row with fixed row coordinate y_r. By contrast, row coordinate y_r of the rectified image increases by 1 only after the last column pixel calculation. This procedure is iterated from pixel (0, 0) to the last pixel (M – 1, N – 1) in the rectified image with a resolution of $M \times N$. Because y_r is fixed during the rectification processing in one row, $g(y_r)$ and $h(y_r)$ are constants. Using this feature, (10), (11), (13), (14), and (15), (5) can be rewritten as

$$\begin{aligned} x_{(x_r,y_r)} &= a_x \cdot x_r^5 + b_x(y_r) \cdot x_r^4 + c_x(y_r) \cdot x_r^3 + d_x(y_r) \cdot x_r^2 + e_x(y_r) \cdot x_r + f_x(y_r) \\ y_{(x_r,y_r)} &= a_y \cdot x_r^5 + b_y(y_r) \cdot x_r^4 + c_y(y_r) \cdot x_r^3 + d_y(y_r) \cdot x_r^2 + e_y(y_r) \cdot x_r + f_y(y_r), \end{aligned} \tag{16}$$

where $(x_{(x_r,y_r)}, y_{(x_r,y_r)})$ are the coordinates of the input image that correspond to rectified pixel coordinates (x_r, y_r),

$$
\begin{aligned}
a_x &= fc_{ix} \cdot (kc_2 \cdot c_{11}^5 + 2 \cdot kc_2 \cdot c_{11}^3 \cdot c_{21}^2 + kc_2 \cdot c_{11} \cdot c_{21}^4) \\
b_x(y_r) &= fc_{ix} \cdot (5 \cdot kc_2 \cdot c_{11}^4 \cdot g(y_r) + 6 \cdot kc_2 \cdot c_{11}^2 \cdot c_{21}^2 \cdot g(y_r) + 4 \cdot kc_2 \cdot c_{11}^3 \cdot c_{21} \cdot h(y_r) \\
&\quad + 4 \cdot kc_2 \cdot c_{11} \cdot c_{21}^3 \cdot h(y_r) + kc_2 \cdot c_{21}^4 \cdot g(y_r)) \\
c_x(y_r) &= fc_{ix} \cdot (kc_1 \cdot c_{11}^3 + kc_1 \cdot c_{11} \cdot c_{21}^2 + 10 \cdot kc_2 \cdot c_{11}^3 \cdot g(y_r)^2 \\
&\quad + 6 \cdot kc_2 \cdot c_{11} \cdot c_{21}^2 \cdot g(y_r)^2 + 12 \cdot kc_2 \cdot c_{11}^2 \cdot c_{21} \cdot g(y_r) \cdot h(y_r) + 2 \cdot kc_2 \cdot c_{11}^3 \cdot h(y_r)^2 \\
&\quad + 6 \cdot kc_2 \cdot c_{11} \cdot c_{21}^2 \cdot h(y_r)^2 + 4 \cdot kc_2 \cdot c_{21}^3 \cdot g(y_r) \cdot h(y_r)) \\
d_x(y_r) &= fc_{ix} \cdot (3 \cdot kc_1 \cdot c_{11}^2 \cdot g(y_r) + 2 \cdot kc_1 \cdot c_{11} \cdot c_{21} \cdot h(y_r) + kc_1 \cdot c_{21}^2 \cdot g(y_r) \\
&\quad + 10 \cdot kc_2 \cdot c_{11}^2 \cdot g(y_r)^3 + 2 \cdot kc_2 \cdot c_{21}^2 \cdot g(y_r)^3 + 6 \cdot kc_2 \cdot c_{11}^2 \cdot g(y_r) \cdot h(y_r)^2 \\
&\quad + 12 \cdot kc_2 \cdot c_{11} \cdot c_{21} \cdot g(y_r)^2 \cdot h(y_r) + 4 \cdot kc_2 \cdot c_{11} \cdot c_{21} \cdot h(y_r)^3 \\
&\quad + 6 \cdot kc_2 \cdot c_{21}^2 \cdot g(y_r) \cdot h(y_r)^2) \\
e_x(y_r) &= fc_{ix} \cdot (3 \cdot kc_1 \cdot c_{11} \cdot g(y_r)^2 + kc_1 \cdot c_{11} \cdot h(y_r)^2 + 2 \cdot kc_1 \cdot c_{21} \cdot g(y_r) \cdot h(y_r) \\
&\quad + 5 \cdot kc_2 \cdot c_{11} \cdot g(y_r)^4 + 4 \cdot kc_2 \cdot c_{21} \cdot g(y_r)^3 \cdot h(y_r) + 6 \cdot kc_2 \cdot c_{11} \cdot g(y_r)^2 \cdot h(y_r)^2 \\
&\quad + kc_2 \cdot c_{11} \cdot h(y_r)^4 + 4 \cdot kc_2 \cdot c_{21} \cdot g(y_r) \cdot h(y_r)^3 + c_{11}) \\
f_x(y_r) &= fc_{ix} \cdot (kc_1 \cdot g(y_r) \cdot h(y_r)^2 + kc_1 \cdot g(y_r)^3 + kc_2 \cdot g(y_r) \cdot h(y_r)^4 \\
&\quad + 2 \cdot kc_2 \cdot g(y_r)^3 \cdot h(y_r)^2 + kc_2 \cdot g(y_r)^5 + g(y_r)) + cc_{ix},
\end{aligned} \tag{17}
$$

and

$$
\begin{aligned}
a_y &= fc_{iy} \cdot (kc_2 \cdot c_{21}^5 + 2 \cdot kc_2 \cdot c_{11}^2 \cdot c_{21}^3 + kc_2 \cdot c_{11}^4 \cdot c_{21}) \\
b_y(y_r) &= fc_{iy} \cdot (5 \cdot kc_2 \cdot c_{21}^4 \cdot h(y_r) + 6 \cdot kc_2 \cdot c_{11}^2 \cdot c_{21}^2 \cdot h(y_r) + 4 \cdot kc_2 \cdot c_{11}^3 \cdot c_{21} \cdot g(y_r) \\
&\quad + 4 \cdot kc_2 \cdot c_{11} \cdot c_{21}^3 \cdot g(y_r) + kc_2 \cdot c_{11}^4 \cdot h(y_r)) \\
c_y(y_r) &= fc_{iy} \cdot (kc_1 \cdot c_{21}^3 + kc_1 \cdot c_{11}^2 \cdot c_{21} + 10 \cdot kc_2 \cdot c_{21}^3 \cdot h(y_r)^2 \\
&\quad + 6 \cdot kc_2 \cdot c_{11}^2 \cdot c_{21} \cdot g(y_r)^2 + 12 \cdot kc_2 \cdot c_{11} \cdot c_{21}^2 \cdot g(y_r) \cdot h(y_r) + 2 \cdot kc_2 \cdot c_{21}^3 \cdot g(y_r)^2 \\
&\quad + 6 \cdot kc_2 \cdot c_{11}^2 \cdot c_{21} \cdot h(y_r)^2 + 4 \cdot kc_2 \cdot c_{11}^3 \cdot g(y_r) \cdot h(y_r)) \\
d_y(y_r) &= fc_{iy} \cdot (3 \cdot kc_1 \cdot c_{21}^2 \cdot h(y_r) + 2 \cdot kc_1 \cdot c_{11} \cdot c_{21} \cdot g(y_r) + kc_1 \cdot c_{11}^2 \cdot h(y_r) \\
&\quad + 10 \cdot kc_2 \cdot c_{21}^2 \cdot h(y_r)^3 + 2 \cdot kc_2 \cdot c_{11}^2 \cdot h(y_r)^3 + 6 \cdot kc_2 \cdot c_{21}^2 \cdot g(y_r)^2 \cdot h(y_r) \\
&\quad + 12 \cdot kc_2 \cdot c_{11} \cdot c_{21} \cdot g(y_r) \cdot h(y_r)^2 + 4 \cdot kc_2 \cdot c_{11} \cdot c_{21} \cdot g(y_r)^3 \\
&\quad + 6 \cdot kc_2 \cdot c_{11}^2 \cdot g(y_r)^2 \cdot h(y_r)) \\
e_y(y_r) &= fc_{iy} \cdot (3 \cdot kc_1 \cdot c_{21} \cdot h(y_r)^2 + kc_1 \cdot c_{21} \cdot g(y_r)^2 + 2 \cdot kc_1 \cdot c_{11} \cdot g(y_r) \cdot h(y_r) \\
&\quad + 5 \cdot kc_2 \cdot c_{21} \cdot h(y_r)^4 + 4 \cdot kc_2 \cdot c_{11} \cdot g(y_r)^3 \cdot h(y_r) + 6 \cdot kc_2 \cdot c_{21} \cdot g(y_r)^2 \cdot h(y_r)^2 \\
&\quad + kc_2 \cdot c_{21} \cdot g(y_r)^4 + 4 \cdot kc_2 \cdot c_{11} \cdot g(y_r) \cdot h(y_r)^3 + c_{21}) \\
f_y(y_r) &= fc_{iy} \cdot (kc_1 \cdot g(y_r)^2 \cdot h(y_r) + kc_1 \cdot h(y_r)^3 + kc_2 \cdot g(y_r)^4 \cdot h(y_r) \\
&\quad + 2 \cdot kc_2 \cdot g(y_r)^2 \cdot h(y_r)^3 + kc_2 \cdot h(y_r)^5 + h(y_r)) + cc_{iy}.
\end{aligned} \tag{18}
$$

Within one row, because column coordinate x_r increases by 1 for every pixel calculation, x_r can be used as an index of sequence. Thus, (16) can be converted to difference sequences whose general forms are fifth-degree polynomials, shown in **Fig. 3**. Thus, we propose a rectification method that uses multi-level accumulations defined as

$$\begin{aligned} x_{(x_r,y_r)} &= x_{(x_r-1,y_r)} + \Delta x_{(x_r-1,y_r)},\ x_{(0,y_r)} = f_x(y_r) \\ y_{(x_r,y_r)} &= y_{(x_r-1,y_r)} + \Delta y_{(x_r-1,y_r)},\ y_{(0,y_r)} = f_y(y_r), \end{aligned} \tag{19}$$

$$\begin{aligned} \Delta x_{(x_r,y_r)} &= \Delta x_{(x_r-1,y_r)} + \Delta^2 x_{(x_r-1,y_r)},\ \Delta x_{(0,y_r)} = a_x + b_x(y_r) + c_x(y_r) + d_x(y_r) + e_x(y_r) \\ \Delta y_{(x_r,y_r)} &= \Delta y_{(x_r-1,y_r)} + \Delta^2 y_{(x_r-1,y_r)},\ \Delta y_{(0,y_r)} = a_y + b_y(y_r) + c_y(y_r) + d_y(y_r) + e_y(y_r), \end{aligned} \tag{20}$$

$$\begin{aligned} \Delta^2 x_{(x_r,y_r)} &= \Delta^2 x_{(x_r-1,y_r)} + \Delta^3 x_{(x_r-1,y_r)},\ \Delta^2 x_{(0,y_r)} = 30a_x + 14b_x(y_r) + 6c_x(y_r) + 2d_x(y_r) \\ \Delta^2 y_{(x_r,y_r)} &= \Delta^2 y_{(x_r-1,y_r)} + \Delta^3 y_{(x_r-1,y_r)},\ \Delta^2 y_{(0,y_r)} = 30a_y + 14b_y(y_r) + 6c_y(y_r) + 2d_y(y_r), \end{aligned} \tag{21}$$

$$\begin{aligned} \Delta^3 x_{(x_r,y_r)} &= \Delta^3 x_{(x_r-1,y_r)} + \Delta^4 x_{(x_r-1,y_r)},\ \Delta^3 x_{(0,y_r)} = 150a_x + 36b_x(y_r) + 6c_x(y_r) \\ \Delta^3 y_{(x_r,y_r)} &= \Delta^3 y_{(x_r-1,y_r)} + \Delta^4 y_{(x_r-1,y_r)},\ \Delta^3 y_{(0,y_r)} = 150a_y + 36b_y(y_r) + 6c_y(y_r), \end{aligned} \tag{22}$$

$$\begin{aligned} \Delta^4 x_{(x_r,y_r)} &= \Delta^4 x_{(x_r-1,y_r)} + 120a_x,\ \Delta^4 x_{(0,y_r)} = 240a_x + 24b_x(y_r) \\ \Delta^4 y_{(x_r,y_r)} &= \Delta^4 y_{(x_r-1,y_r)} + 120a_y,\ \Delta^4 y_{(0,y_r)} = 240a_y + 24b_y(y_r). \end{aligned} \tag{23}$$

where $\Delta^n x_{(x_r,y_r)}$ and $\Delta^n y_{(x_r,y_r)}$ are the nth-order difference sequences.

$$x_{(x_r,y_r)} = a_x \cdot x_r^5 + b_x(y_r)\cdot x_r^4 + c_x(y_r)\cdot x_r^3 + d_x(y_r)\cdot x_r^2 + e_x(y_r)\cdot x_r + f_x(y_r)$$

	$x_{(x_r,y_r)} = x_{(x_r-1,y_r)} + \Delta x_{(x_r-1,y_r)}$	$\Delta x_{(x_r,y_r)} = \Delta x_{(x_r-1,y_r)} + \Delta^2 x_{(x_r-1,y_r)}$	$\Delta^2 x_{(x_r,y_r)} = \Delta^2 x_{(x_r-1,y_r)} + \Delta^3 x_{(x_r-1,y_r)}$
x_r	$x_{(x_r,y_r)}$	$\Delta x_{(x_r,y_r)}$	$\Delta^2 x_{(x_r,y_r)}$
0	$f_x(y_r)$	$a_x + b_x(y_r) + c_x(y_r) + d_x(y_r) + e_x(y_r)$	$30a_x + 14b_x(y_r) + 6c_x(y_r) + 2d_x(y_r)$
1	$a_x + b_x(y_r) + c_x(y_r) + d_x(y_r) + e_x(y_r) + f_x(y_r)$	$31a_x + 15b_x(y_r) + 7c_x(y_r) + 3d_x(y_r) + e_x(y_r)$	$180a_x + 50b_x(y_r) + 12c_x(y_r) + 2d_x(y_r)$
2	$32a_x + 16b_x(y_r) + 8c_x(y_r) + 4d_x(y_r) + 2e_x(y_r) + f_x(y_r)$	$211a_x + 65b_x(y_r) + 19c_x(y_r) + 5d_x(y_r) + e_x(y_r)$	$570a_x + 110b_x(y_r) + 18c_x(y_r) + 2d_x(y_r)$
3	$243a_x + 81b_x(y_r) + 27c_x(y_r) + 9d_x(y_r) + 3e_x(y_r) + f_x(y_r)$	$781a_x + 175b_x(y_r) + 37c_x(y_r) + 7d_x(y_r) + e_x(y_r)$	$1320a_x + 194b_x(y_r) + 24c_x(y_r) + 2d_x(y_r)$
4	$1024a_x + 256b_x(y_r) + 64c_x(y_r) + 16d_x(y_r) + 4e_x(y_r) + f_x(y_r)$	$2101a_x + 369b_x(y_r) + 61c_x(y_r) + 9d_x(y_r) + e_x(y_r)$	
5	$3125a_x + 625b_x(y_r) + 125c_x(y_r) + 25d_x(y_r) + 5e_x(y_r) + f_x(y_r)$		
	⋮	⋮	⋮

	$\Delta^3 x_{(x_r,y_r)} = \Delta^3 x_{(x_r-1,y_r)} + \Delta^4 x_{(x_r-1,y_r)}$	$\Delta^4 x_{(x_r,y_r)} = \Delta^4 x_{(x_r-1,y_r)} + \Delta^5 x_{(x_r-1,y_r)}$	
x_r	$\Delta^3 x_{(x_r,y_r)}$	$\Delta^4 x_{(x_r,y_r)}$	$\Delta^5 x_{(x_r,y_r)}$
0	$150a_x + 36b_x(y_r) + 6c_x(y_r)$	$240a_x + 24b_x(y_r)$	$120a_x$
1	$390a_x + 60b_x(y_r) + 6c_x(y_r)$	$360a_x + 24b_x(y_r)$	$120a_x$
2	$750a_x + 84b_x(y_r) + 6c_x(y_r)$		
	⋮	⋮	⋮

Fig. 3. Multi-level accumulations of the proposed rectification for the column coordinate

Because fc_{ix}, fc_{iy}, kc_2, c_{11}, and c_{21} are constants, a_x and a_y in (17) and (18) are also constants. In addition, during rectification within one row, because $g(y_r)$ and $h(y_r)$ are uniquely determined as constants, all of the coefficients, $b_x(y_r)$, $c_x(y_r)$, $d_x(y_r)$, $e_x(y_r)$, $f_x(y_r)$, $b_y(y_r)$, $c_y(y_r)$, $d_y(y_r)$, $e_y(y_r)$, and $f_y(y_r)$, are also fixed as constants according to (17) and (18). As a result, the initial values of sequences, $x_{(0,y_r)}$, $\Delta x_{(0,y_r)}$, $\Delta^2 x_{(0,y_r)}$, $\Delta^3 x_{(0,y_r)}$, $\Delta^4 x_{(0,y_r)}$, $y_{(0,y_r)}$, $\Delta y_{(0,y_r)}$, $\Delta^2 y_{(0,y_r)}$, $\Delta^3 y_{(0,y_r)}$, and $\Delta^4 y_{(0,y_r)}$ are also fixed as constants within one row according to (19) to (23). Thus, the proposed method carries out rectification by simple multi-level accumulations of constants when column coordinate x_r increases with row coordinate y_r fixed within one row.

Whenever row coordinate y_r increases by 1 after the last column pixel calculation, $g(y_r)$ and $h(y_r)$ change, shown as

$$\begin{aligned} g(y_r) &= g(y_r-1)+c_{12}, \quad g(0)=c_{13} \\ h(y_r) &= h(y_r-1)+c_{22}, \quad h(0)=c_{23}. \end{aligned} \tag{24}$$

When $g(y_r)$ and $h(y_r)$ change, the coefficients defined in (17) and (18), which depend on $g(y_r)$ and $h(y_r)$, must be recalculated. Hence, the initial values of the sequences defined in (19) to (23), which depend on the coefficients defined in (17) and (18), must be also recalculated. These recalculations, which occur at every row change, may cause critical performance degradation in the proposed method because the coefficient calculations of (17) and (18) and the initial value calculations from (19) to (23) are very complex. Fortunately, these values can be precalculated because y_r is an integer predetermined for every row, so we precalculate them and save the initial values of sequences in the memory, shown in Section 3.

As a result, using only simple multi-level accumulations, the proposed method produces the same rectified image with rectification using (5), (10), (11), (14), and (15). As shown in **Table 5**, by replacing complex matrix multiplications and high-degree polynomial calculations with multi-level accumulations, the proposed rectification method, compared to rectification using (5), (10), (11), (14), and (15), reduces addition operations by 29% and removes multiplication operations.

Table 5. Comparison between the numbers of operations of the rectification method using (5), (10), (11), (14), and (15) and those of the proposed one

Method	Image resolution	Number of operations	
		Addition	Multiplication
Using (5), (10), (11), (14), and (15)	640 × 480	4,300,800	4,147,200
	1280 × 720	12,902,400	18,432,000
	1920 × 1080	29,030,400	41,472,000
Using the proposed multi-level accumulations	640 × 480	3,067,200	-
	1280 × 720	9,208,800	-
	1920 × 1080	20,725,200	-

3. Proposed Rectification Circuit

This section presents a hardware circuit of the proposed rectification method based on multi-level accumulations explained in Section 2. The proposed rectification hardware circuit, shown in **Fig. 4**, includes four main components: an address counter, image line buffers, a coordinate calculator, and a bilinear interpolator.

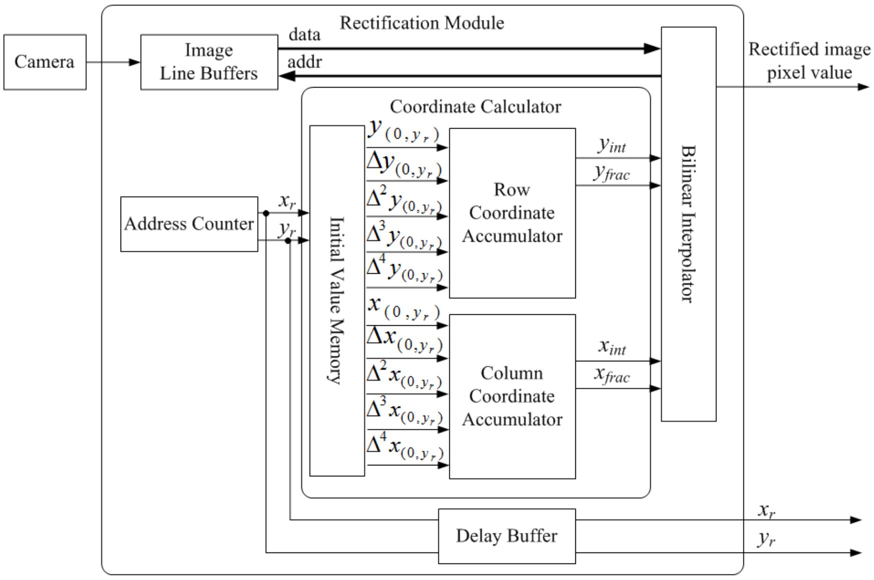
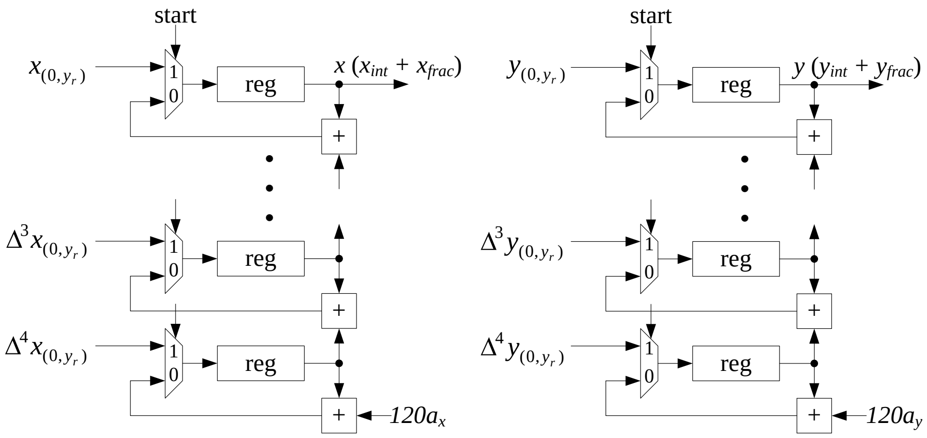


Fig. 4. Proposed rectification hardware architecture

The address counter generates rectified image coordinates (x_r, y_r) in the order of the pixel processing sequence shown in **Fig. 2**. The coordinate calculator, which consists of row and column coordinate accumulators and an initial value memory, computes corresponding input image coordinates (x, y) for each rectified image coordinate pair (x_r, y_r). The outputs of the coordinate calculator, (x, y), which are non-integers and separated into an integral part, (x_{int}, y_{int}), and a fractional part, (x_{frac}, y_{frac}), are sent to the bilinear interpolator. The row and column coordinate accumulators, with the same architecture, perform multi-level accumulations of (19) to (23), shown in **Fig. 5**. When column coordinate x_r of the rectified image equals 0, that is, at the first column of each row, the start signal in **Fig. 5** is asserted and the initial values of (19) to (23) are loaded into the registers in **Fig. 5**. For x_r other than 0, the row and column coordinate accumulators perform multi-level accumulations using the previous results in the registers. The initial values of sequences vary according to the rectified row coordinate y_r, described in Section 2, so we precalculated all of the initial values and stored them in the initial value memory shown in **Fig. 6**. These precalculated initial values, which are read from the initial value memory by using y_r as an address whenever the first column pixel of each row in the rectified image is processed, are sent to the row and column coordinate accumulators.



(a) Column coordinate accumulator (b) Row coordinate accumulator

Fig. 5. Row and column coordinate accumulators

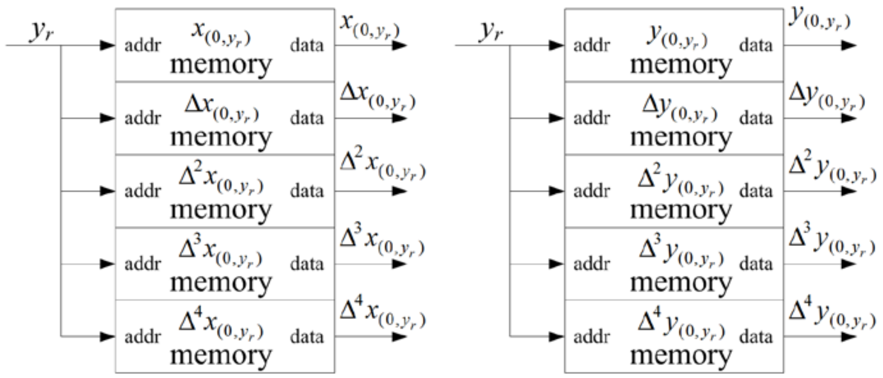


Fig. 6. Initial value memory

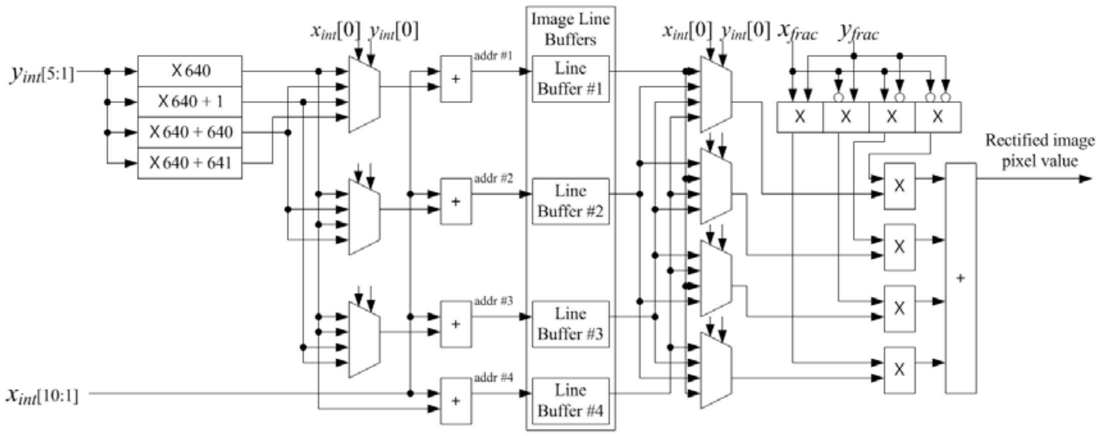


Fig. 7. Bilinear interpolator

The value of the rectified image pixel (x_r, y_r) is equal to that of corresponding input image pixel (x, y). However, because the computed coordinates (x, y) are non-integers, the pixel value of (x, y) should be calculated by interpolation with the values of the four neighboring pixels, (x_{int}, y_{int}), $(x_{int}+1, y_{int})$, $(x_{int}, y_{int}+1)$, and $(x_{int}+1, y_{int}+1)$ of the input image

[18]. The bilinear interpolator reads these four values from the image line buffers, which store the pixel values of the input images, and conducts bilinear interpolation with interpolation weights calculated from fractional coordinates (x_{frac}, y_{frac}), shown in **Fig. 7**. The bilinear interpolator generates the value of rectified image pixel (x_r, y_r), as the final output of the proposed rectification circuit.

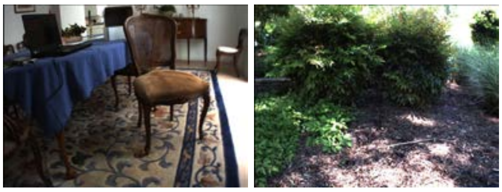
4. Experimental Results and Analysis

To evaluate the proposed rectification method, we used MATLAB to model five rectification methods, described in **Table 6**. Methods 1 and 2 are rectification methods including both radial and tangential distortion corrections, while method 4 is one without any distortion correction. Method 1 uses three radial distortion coefficients, kc_1, kc_2, and kc_5, while method 2 uses only two radial distortion coefficients, kc_1 and kc_2. Method 3 and the proposed method are similar, for both include radial distortion correction using two radial distortion coefficients but omit tangential distortion correction, but they also differ. That is, while method 3 uses the matrix multiplications of (1) and (5) and the polynomial calculations of (10) and includes division by z_c for normalization, the proposed method uses the multi-level accumulations of (19) to (23) and skips division by z_c.

We extracted camera calibration parameters for rectification using the stereo camera calibrator APP in MATLAB [23] and precalculated the initial values of the sequences in (19) to (23) for the proposed method using these parameters. Then we performed rectifications of the five methods in **Table 6** using the example stereo images with a resolution of 640 × 480 provided by BoofCV, shown in **Fig. 8**, as input images. In addition, we extracted depth maps from the rectified images of the five methods using the semi-global matching function provided by MATLAB [26].

Table 6. Rectification methods for the experiments

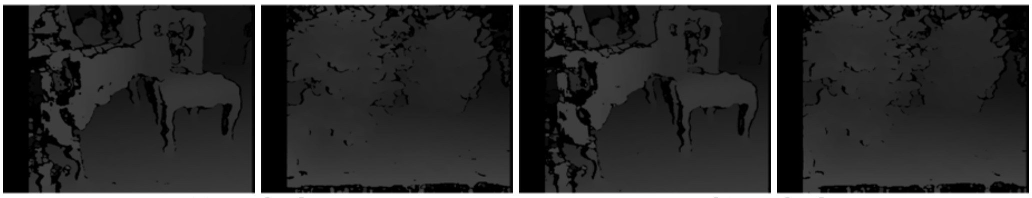
Method	Description	Use of techniques
1	Rectification with radial and tangential distortion corrections using three radial distortion coefficients	(1), (2), (3), (4), (5)
2	Rectification with radial and tangential distortion corrections using two radial distortion coefficients	(1), (3), (4), (5), (10)
3	Rectification with radial distortion correction using two radial distortion coefficients	(1), (5), (10), (11)
4	Rectification without any distortion correction	(6)
Proposed	Rectification using multi-level accumulations	(19), (20), (21), (22), (23)



(a) Chair01 (b) Garden02

Fig. 8. Example images of BoofCV

To analyze the influence of the radial distortion model on rectification results, we extracted depth maps from the rectification results of methods 1 and 2. **Fig. 9** illustrates depth maps extracted from the rectified images of methods 1 and 2, and **Fig. 10** shows differences between depth maps from methods 1 and 2. In **Fig. 10**, the larger the difference is, the brighter it appears, so black represents no difference. Shown in **Fig. 10**, depth maps from methods 1 and 2 exhibit small differences only at the boundaries of objects. Again, this confirms that the radial distortion model using two coefficients is sufficient for radial distortion correction.



(a) Method 1 (b) Method 2

Fig. 9. Depth maps extracted from the rectified images of methods 1 and 2

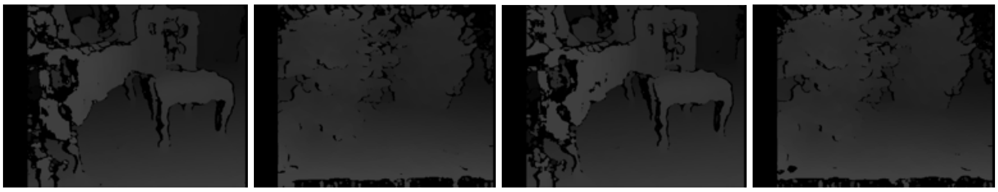
Fig. 10. Differences between the depth maps of methods 1 and 2

To analyze the effects of tangential distortion on rectification, we compared the rectification results of methods 3 and 4 with those of method 2. **Table 7** shows the differences between the coordinates of methods 2 and 3 and methods 2 and 4. These differences are calculated as those between the input image coordinates (x, y) derived from the same coordinates (x_r, y_r) of the left rectified image by the compared methods. The differences between methods 2 and 3 are minor, as shown in **Table 7**. Method 3 differs from method 2 only because it excludes tangential distortion correction from rectification. The minor differences between methods 2 and 3 confirm that tangential distortion correction can be omitted. By contrast, the differences between methods 2 and 4 are significant, showing the considerable effects of radial distortion.

Table 7. Coordinate differences between rectification methods

Coordinate	Difference between methods 2 and 3		Difference between methods 2 and 4	
	Maximum	Average	Maximum	Average
x	0.9602	0.2568	58.9331	12.7432
y	0.4541	0.0821	41.7381	8.6591

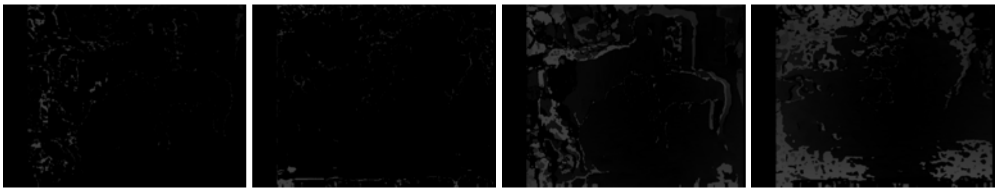
Fig. 11 illustrates depth maps extracted from the rectified images of all of the methods except method 1. The differences between the depth maps of method 2 and the other methods are shown in **Fig. 12**. Compared to the depth maps of method 2, those of method 3 and the proposed method exhibit minor differences at the boundaries of objects. Again, this finding confirms that tangential distortion correction can be omitted. By contrast, differences between the depth maps of methods 2 and 4 are considerably large in the entire area, indicating that radial distortion correction should be included in the rectification process for accurate stereo matching.



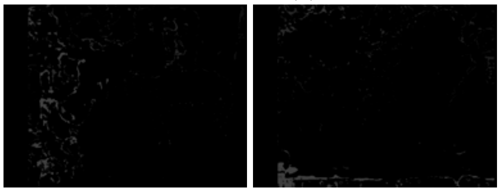
(a) Method 2 (b) Method 3

(c) Method 4 (d) Proposed

Fig. 11. Depth maps extracted from the rectified images of methods 2, 3, 4, and the proposed method



(a) Between methods 2 and 3 (b) Between methods 2 and 4



(c) Between method 2 and the proposed method

Fig. 12. Differences between the depth maps of method 2 and the other methods

The differences between the rectification results of method 3 and those of the proposed method result from the fact that method 3 includes division by z_c for normalization while the proposed method skips it. Therefore, to confirm that division by z_c is negligible, we compared the results of method 3 to those of the proposed method. As shown in **Table 8**, the differences between the input image coordinates (x, y) derived from the same rectified coordinates (x_r, y_r) by the two methods are negligible. Also, **Figs. 11(b)**, **11(d)**, and **13** show that the depth maps extracted from the rectified images of both method 3 and the proposed method are almost the same. These results of the comparison confirm that division by z_c in (12) can be skipped.

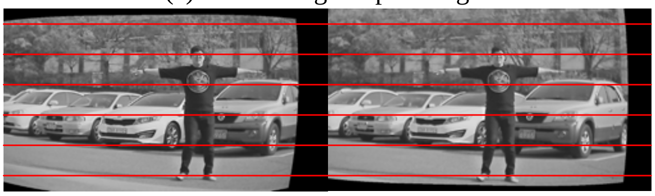
Table 8. Coordinate differences between method 3 and the proposed method

Coordinate	**Maximum**	**Average**
x	0.6322	0.1888
y	0.4051	0.1051

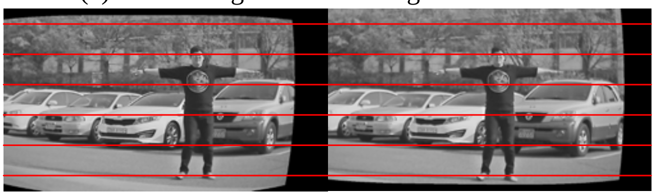
Fig. 13. Differences between the depth maps of method 3 and the proposed method

To evaluate the rectification performance of the proposed method in a real environment, we used a real camera to compare the rectification results of method 2 to those of the proposed method. Not only does method 2 produce well-rectified images, shown in **Fig. 14**, but the proposed method also produces such images. **Table 9** shows that differences between input image coordinates (x, y) derived by the two methods for the same rectified coordinates (x_r, y_r) are negligible. The results of comparison show that the proposed method, which reduces computational overhead by excluding tangential distortion correction and skipping division by z_c, produces almost the same rectification results as method 2, which performs full rectification. Besides excluding tangential distortion correction and skipping division by z_c, the proposed method additionally reduces computational overhead by replacing the matrix computations and the high-degree polynomial calculations of conventional rectification with simple multi-level accumulations.

(a) Left and right input images



(b) Left and right rectified images of method 2



(c) Left and right rectified images of the proposed method

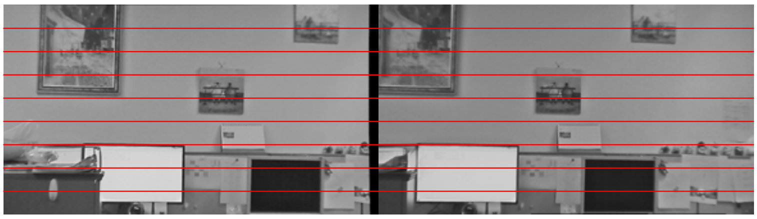
Fig. 14. Rectification results of method 2 and the proposed method

Table 9. Differences between the coordinates of method 2 and the proposed method when real scene images are used

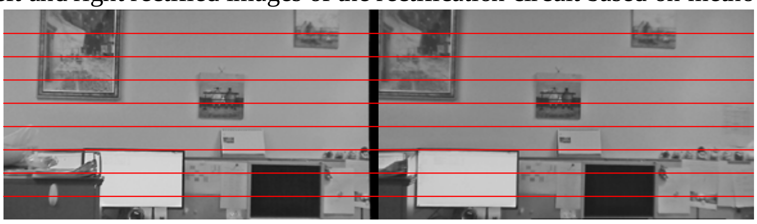
Coordinate	Maximum	Average
x	0.9174	0.6102
y	0.6438	0.1377

To evaluate the effects of the proposed multi-level accumulations, we compared the proposed method and the method using (5), (10), (11), (14), and (15), denoted as method 5 below. Method 5 corresponds exactly to method 3, except for skipping division by z_c. It differs from the proposed method only by using conventional matrix multiplications and polynomial calculations for rectification instead of the proposed multi-level accumulations. Thus, the comparison between the proposed method and method 5 is suitable for the evaluation of the effects of multi-level accumulations. For the comparison, we implemented the two methods in hardware circuits using Xilinx FPGA Virtex-7 XC7V2000T. The two circuits have the same architecture except for their coordinate calculators. The coordinate calculator of the proposed rectification circuit uses multi-level accumulations for coordinate calculations while that of the method 5 circuit uses conventional matrix multiplications and polynomial calculations. The stereo camera used in these hardware implementations has a resolution of 1,280 × 720, a frame rate of 30 fps and a pixel clock of 37.125 MHz, with which the two rectification circuits are synchronized.

Fig. 15 shows the rectification results of the circuit based on method 5 and the proposed circuit, which confirm that both circuits generate the same results. The two circuits were implemented using only slices and block RAMs of the FPGA, and **Fig. 16** shows the hardware use of the two rectification circuits. While the proposed circuit, shown in **Fig. 16**, consumes only 68%, 38%, and 72% of slice LUTs, slice registers, and slices required by the circuit of method 5, respectively, the two circuits require the same number of block RAMs. The two circuits exhibit similar performance, continuously generating a pair of left and right rectified pixels at every clock cycle because they are fully pipelined. However, while the circuit based on method 5 requires additional latency of 145 cycles in addition to the initial image buffering time, the proposed circuit requires additional latency of only 13 cycles. This is because the proposed method, compared to method 5, reduces additions by 29% and removes multiplications by replacing complex matrix multiplications and high-degree polynomial calculations with simple multi-level accumulations.



(a) Left and right rectified images of the rectification circuit based on method 5



(b) Left and right rectified images of the proposed rectification circuit

Fig. 15. Rectification results of the rectification hardware circuits

(a) Slice LUTs (b) Slice registers (c) Slices (d) BRAM36s

Fig. 16. Hardware usage of method 5 and proposed rectification circuits

Table 10. Results of the comparison of the most recent rectification hardware implementations

System	Image size	Device	# DSP	# BRAM	External memory	# Slice	Max. fps	Max. frequency (MHz)
Proposed	1280 × 720	Xilinx XC7V2000T	0	40	no	5588	135	125
Method 5	1280 × 720	Xilinx XC7V2000T	0	40	no	7738	217	200
[8]	1024 × 768	Altera Stratix III	2 × 23	2 × 1,342,776 bits	no	# LE = 2 × 901 # Reg = 2 × 259	30	90
[10]	640 × 480	Xilinx XC4VLX200	n.a	64	no	4035	230	93
[15]	640 × 512	Xilinx V600EFG680	0	72	2 × 64-MB SDRAMs	6912	85	90
[17]	1280 × 720	Xilinx XC4VLX60	0	64	no	9977	120	111

Table 10 shows the results of the comparison of the most recent rectification hardware implementations. Because the maximum frame rates of all the implementations, except for [8] in **Table 10**, are greater than 60 fps, which the frame rate of the camera input images is usually less than or equal to, the maximum frame rate comparison is meaningless. Direct comparison of hardware costs is also difficult because the implementations of **Table 10** use different FPGA devices, image sizes, and distortion corrections. However, it is easily shown that the proposed rectification circuit, compared with the others, requires the lowest hardware costs, considering that the proposed circuit supports the highest resolution and corrects image distortion. Overall, the effect of the proposed multi-level accumulations is significant, so the proposed rectification circuit is the most practical with the lowest hardware costs.

5. Conclusion

This paper proposed a multi-level accumulation-based rectification method and its optimized hardware circuit for the stereo matching system. The proposed method reduces computational overhead by excluding tangential distortion correction from the rectification process and skipping division for translating 3D coordinates into 2D coordinates, inspired by previous studies, while producing almost the same results with full rectification. Furthermore, we also reduced computational overhead by replacing the complex matrix multiplications and the high-degree polynomial calculations of conventional rectification with simple multi-level accumulations. The proposed multi-level accumulations reduced the number of addition operations by 29% and removed multiplication operations. In addition, we implemented the proposed method in a hardware circuit using Xilinx FPGA Virtex-7 XC7V2000T, which can process 1,280 × 720 images at a frame rate of 135 fps at a maximum clock frequency of 125 MHz. Results of the implementation showed that the proposed multi-level accumulations have the effect of saving slice LUTs, slice registers, and slices of the FPGA by 32%, 62%, and 28%, respectively. The overall results showed that the proposed rectification circuit, compared to conventional rectification circuits, reduces computational overhead and hardware costs while maintaining rectification performance. Future work will cover the implementation of rectification circuits in an ASIC and an analysis of the implementation results for an accurate cost comparison.

References

[1] Kohtaro Sabe, Masaki Fukuchi, Jens-Steffen Gutmann, Takeshi Ohashi, Kenta Kawamoto and Takayuki Yoshigahara, "Obstacle avoidance and path planning for humanoid robots using stereo vision," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 592-597, April 26-May 1, 2004. Article (CrossRef Link)

[2] Kunsoo Huh, Jaehak Park, Junyeon Hwang and Daegun Hong, "A stereo vision-based obstacle detection system in vehicles," *Optics and Laser in Engineering*, vol. 46, no. 2, pp. 168-178, February, 2008. Article (CrossRef Link)

[3] Annett Chilian and Heiko Hirschmuller, "Stereo camera based navigation of mobile robots on rough terrain," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 4571-4576, October 11-15, 2009. Article (CrossRef Link)

[4] Martin Humenberger, Tobias Engelke and Wilfried Kubinger, "A census-based stereo vision algorithm using modified semi-global matching and plane fitting to improve matching quality," in *Proc. of the 2010 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops*, pp. 77-84, June 13-18, 2010. Article (CrossRef Link)

[5] Wannes van der Mark and Dariu M. Gavrila, "Real-time dense stereo for intelligent vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 38-50, March, 2006. Article (CrossRef Link)

[6] Charles Loop and Zhengyou Zhang, "Computing rectifying homographies for stereo vision," in *Proc. of IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, pp. 125-131, June 23-25, 1999. Article (CrossRef Link)

[7] Joshua Gluckman and Shree K. Nayar, "Rectified catadioptric stereo sensors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp.224-236, February, 2002. Article (CrossRef Link)

[8] Khurram Jawed, John Morris, Tariq Khan and Georgy Gimel'farb, "Real time rectification for stereo correspondence," in *Proc. of IEEE Int. Conf. on Computational Science and Engineering*, pp. 277-284, August 29-31, 2009. Article (CrossRef Link)

[9] Abdulkadir Akin, Ipek Baz, Luis Manuel Gaemperle, Alexandre Schmid and Yusuf Leblebici, "Compressed look-up-table based real-time rectification hardware," in *Proc. of 2013 IFIP/IEEE 21st Int. Conf. on Very Large Scale Integration*, pp. 272-277, October 7-9, 2013. Article (CrossRef Link)

[10] S. Jin, J. Cho, X. D. Pham, K. M. Lee, S.-K. Park, M. Kim and J. W. Jeon, "FPGA design and implementation of a real-time stereo vision system," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 1, pp.15-26, January, 2010. Article (CrossRef Link)

[11] Sang-Kyo Han, SeongHoon Woo, Mun-Ho Jeong and Bum-Jae You, "Improved-quality real-time stereo vision processor," in *Proc. of 22nd Int. Conf. on VLSI Design*, pp. 287-292, January 5-9, 2009. Article (CrossRef Link)

[12] Hongru Zhang, Zaifeng Shi, Ke Pang, Qingjie Cao, Tao Luo and Suying Yao, "Seam-based variable-step Bresenham blending method for real-time video mosaicking," *Journal of Electronic Imaging*, vol. 25, no. 5, pp. 053008, September, 2016. Article (CrossRef Link)
[13] Jean-Yves Bouguet, "Camera calibration toolbox for Matlab," http://www.vision.caltech.edu/ bouguetj/calib_doc/.
[14] Andrea Fusiello, Emanuele Trucco and Alessandro Verri, "A compact algorithm for rectification of stereo pairs," *Machine Vision and Applications*, vol. 12, no. 1, pp. 16-22, July, 2000. Article (CrossRef Link)
[15] Cristian Vancea and Sergiu Nedevschi, "LUT-based image rectification module implemented in FPGA," in *Proc. of IEEE Int. Conf. on Intelligent Computer Communication and Processing*, pp. 147-154, September 6-8, 2007. Article (CrossRef Link)
[16] Deuk Hyun Park, Hyoung Seok Ko, Jae Gon Kim and Jun Dong Cho, "Real time rectification using differentially encoded lookup table," in *Proc. of Int. Conf. on Ubiquitous Information Management and Communication*, February 21-23, 2011. Article (CrossRef Link)
[17] Paolo Zicari, "Efficient and high performance FPGA-based rectification architecture for stereo vision," *Microprocessors and Microsystems*, vol. 37, no. 8, pp. 1144-1154, November, 2013. Article (CrossRef Link)
[18] Jongkil Hyun and Byungin Moon, "A simplified rectification method and its hardware architecture for embedded multimedia systems," *Multimedia Tools and Applications*, published online, April, 2016. Article (CrossRef Link)
[19] Hyeon-Sik Son and Byungin Moon, "An accumulation-based rectification and distortion correction method," in *Proc. of Int. Conf. on Electronics, Electrical Engineering, Computer Science*, pp.1-6, January 20-22, 2016. Article (CrossRef Link)
[20] Roger Y. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf TV cameras and Lenses," *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 4, pp. 323-343, August, 1987. Article (CrossRef Link)
[21] Matlab documentation, "What is camera calibration?," http://www.mathworks.com/help/vision/ug /camera-calibration.html.
[22] Peter Abeles. "Boofcv," http://boofcv.org/.
[23] Matlab documentation, "Stereo calibration app," http://www.mathworks.com/help/vision/ug/ stereo-camera-calibrator-app.html.
[24] Zhengyou Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330-1334, November, 2000. Article (CrossRef Link)
[25] Janne Heikkila and Olli Silven, "A four-step camera calibration procedure with implicit image correction," in *Proc. of IEEE Int. Conf. on Computer Vision and Pattern Recognition*. pp. 1106-1112, June 17-19, 1997. Article (CrossRef Link)
[26] Heiko Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *Proc. of IEEE Int. Conf. on Computer Vision and Pattern Recognition*. pp. 807-814, June 20-25, 2005. Article (CrossRef Link)

Hyeon-Sik Son is a Ph.D. student in the School of Electronics Engineering, Kyungpook National University at Daegu in Korea. He received the B.S. degree and M.S. degree in Electrical Engineering & Computer Science in Kyungpook National University in 2010 and 2012, respectively. His research interests include SoC, computer architecture, and stereo vision.

Byungin Moon is currently an associate professor in the School of Electronics Engineering, Kyungpook National University at Daegu in Korea. He received the B.S. degree and M.S. degree in Electronic Engineering in Yonsei University in 1995 and 1997, respectively. He received the Ph.D. degree in Electrical & Electronic Engineering in the same university in 2002. He spent two years as a senior researcher in Hynix Semiconductor Inc., and also worked as a research professor in Yonsei University for one year. His current research fields include SoC, computer architecture, and vision processor.