Updated Version 17 February 2021

FPGA-based lens undistortion and image rectification for stereo vision applications

Christina Junger[a], Albrecht Heß[a], Maik Rosenberger[a], and Gunther Notni[a,b]

[a]Technische Universität Ilmenau, Department of Mechanical Engineering, Group for Quality Assurance and Industrial Image Processing, 98693 Ilmenau, Germany
[b]Fraunhofer Institute for Applied Optics and Precision Engineering, 07745 Jena, Germany

ABSTRACT

Lens undistortion and image rectification is a commonly used pre-processing, e.g. for active or passive stereo vision to reduce the complexity of the search for matching points. The undistortion and rectification is implemented in a field programmable gate array (FPGA). The algorithm is performed pixel by pixel. The challenges of the implementation are the synchronisation of the data streams and the limited memory bandwidth. Due to the memory constraints, the algorithm utilises a pre-computed lossy compression of the rectification maps by a ratio of eight. The compressed maps occupy less space by ignoring the pixel indexes, sub-sampling both maps, and reducing repeated information in a row by forming differences to adjacent pixels. Undistorted and rectified images are calculated once without and once with the compressed transformation map. The deviation between the different computed images is minimal and negligible. The functionality of the hardware module, the decompression algorithm and the processing pipeline are described. The algorithm is validated on a Xilinx Zynq-7020 SoC. The stereo setup has a baseline with 46 mm and non-converged optical axis between the cameras. The cameras are configured at 1.3 Mpix @ 60 fps and distortion correction and rectification is performed in real time during image capture. With a camera resolution of 1280 pixels $\times$ 960 pixels and a maximum vertical shift of $\pm$ 20 pixels, the efficient hardware implementation utilizes 12 % of available block RAM resources.

Keywords: lens undistortion, image rectification, computer stereo vision, FPGA, Xilinx Zynq 7020 SoC

1. INTRODUCTION

Three-dimensional reconstruction using passive or active stereo vision have been used in applications such as industrial environment,[1,2] medical imaging[3] and other fields. For the high-speed inline measurement systems, an acceleration of the image processing algorithms is required. One solution for this is the use of FGPAs.[4,5] Real time preprocessing, such as lens distortion and image rectification, near the sensor is therefore an advantage.

2. THEORETICAL BACKGROUND

The physical inhomogeneity of the camera sensor causes non-linear lens effects (radial distortion) and the relative positioning offsets of the camera causes tangential distortion. For industrial image processing it is necessary to correct the lens distortion and to rectify the images. After rectification, the epipolar lines in both stereoscopic images run parallel to the image rows [6, pp. 239]. Thus, dense matching takes place in a 1D search area instead of in a 2D search area (both stereo images are line-correspondent).

The following describes the reverse transformation of lens distortion and image correction (as used in FPGAs[4,7,8]). The reverse transformation (see Figure 1) uses per camera two lens undistortion and image rectification transformation maps. In this maps, horizontal and vertical pixel positions in the raw image are defined. The grey value at the pixel position is usually an intermediate value within a two-dimensional rectangular grid. This is calculated by a bilinear interpolation. The four neighboring pixels at the pixel position (see Figure 1, neighboring pixels $G0 - G3$) and the decimal places of the pixel position are required for this. The interpolated grey value corresponds to the grey value of the rectified image to be calculated. [9, p. 273]

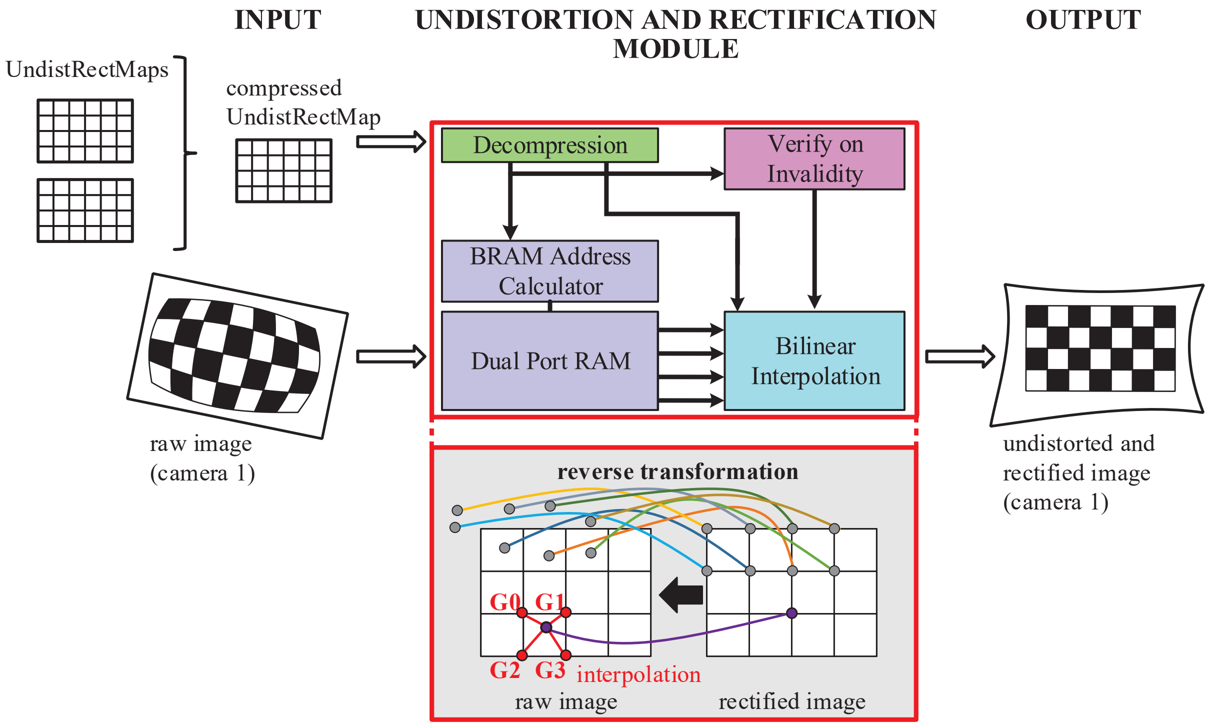

Figure 1. Overview of the lens undistortion and image recftification module.

3. LENS UNDISTORTION AND IMAGE RECTIFICATION MODULE

Figure 1 show an overview of the lens undistortion and rectification module. One module per camera is required. This module calculates the rectified images by reverse transformation (see section 2). This means that each integer pixel in the rectified image corresponds to one pixel coordinate in the raw image. Since the corresponding pixel position in the raw image is not an integer, a bilinear interpolation is performed from the nearby integer pixel locations [10, pp. 437]. Two data must be available at the input of the module - the raw image of the camera and the pre-computed compressed rectification map (see sections 2 and 3.1.1). The module consists of four subsystems - Dual Port RAM with address calculator, decompression of the compressed rectification map, an bilinear interpolation and the verification of invalidity values. The following subsections describe the two necessary preparatory steps (calibration and decompression of rectification maps), the four subsystems and the processing pipeline.

3.1 Preparatory steps

Before using the lens undistortion and image rectification module, a camera calibration and the generation of a compressed rectification map per camera (camera setup k, $k \in [0, 1]$) is required.

3.1.1 Calibration

For the camera calibration a calibre plate with a circle grid pattern is used [5, 10, pp. 428]. At least 20 images per camera are taken with different positions of the calibre plate. The four rectification maps are calculated using these calibration images. The following OpenCV functions are used to calculate rectification maps:

Further author information: (Send correspondence to Christina Junger)
Christina Junger: E-mail: christina.junger@tu-ilmenau.de, Telephone: +49 (0)3677 69 3838

1. `findCirclesGrid()`: Determination of the image points of the circle grid.[11]
2. `stereoCalibrate()`: Using for camera calibration to calculate the camera matrices, the distortion coefficients of both cameras and the rotation and translation vector between them.[11]
3. `stereoRectify()`: Calculation of the rectification transformation matrices for both cameras and the 4×4 reprojection matrix Q (necessary for 3D reconstruction[5]).[11]
4. `initUndistortRectifyMap()`: Generation of rectification maps using the parameters calculated by step 3. Two maps (x and y) for each camera containing the floating point pixel positions in the raw images.[11]

3.1.2 Generation of compressed rectification maps

To reduce bandwidth utilization, it is required to compress the rectification maps. Algorithm 1 shows the necessary steps how to compress the two UndistRectMaps map_{k_x} and map_{k_y} of one camera k into one compressed map map_{k_c}. Redundant information is removed using lossy compression. The resulting map map_{k_c} has the size of $N \times M$ bytes. N is the image width (or number of columns) and M is the image height (or number of rows). In the first step, the pixel index in both UndistRectMaps are subtracted so the maps contain only the relative pixel offsets.[8] Then each UndistRectMap is subsampled. Subsampling is done according to a chequered pattern. For map_{k_y}, the index of this checkerboard pattern is offset by one compared to map_{k_x}. That way the missing value only has to be interpolated once per clock during decompression. Both subsampled UndistRectMaps are merged to map_{k_c}. The last step is reinterpretation of the values of this map_{k_c}. The offset values from the second column are quantized to seven binary decimal bits. Moreover, the offset deviations are calculated by the subtraction of the value from the value located two columns before.[8] The calculated offset deviations are stored with a sign bit in addition to the seven binary decimal bits. The values of the first two columns are rounded. Thus, these consist only of absolute pixel offset values without decimal places.

3.2 Hardware architecture

The proposed lens undistortion and image rectification module was realized with the System Generator Tool. The generated ip core is integrated into a vivado project (with image acquisition via LVDS and image output via HDMI and GigE). The passive stereo system used is based on a Xilinx Zynq-7020 SoC. The two cameras are equipped with e2V EV76C570 CMOS sensors. The lens undistortion and image rectification ip core is implemented in the programmable logic cells (Zynq PL).

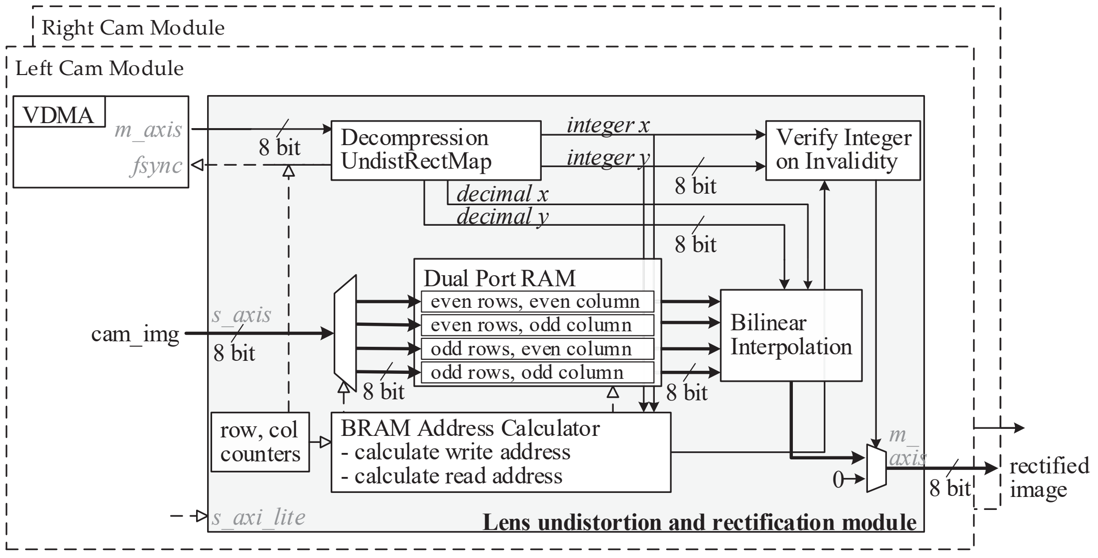

Figure 2. Architecture of the lens undistrectification and image rectification module.

Algorithm 1 Compress UndistRectMaps map_{k_*} for camera k

```
for ($k = 0; k < 2; ++k$) do
    for ($j = 0; j < M; ++j$) do                                        ▷ subtract pixel index
        for ($i = 0; i < N; ++i$) do
            $map_{k\_x}[j][i] = UndistRectMap_{k\_x}[j][i] - i$
            $map_{k\_y}[j][i] = UndistRectMap_{k\_y}[j][i] - j$
        end for
    end for

    for ($j = 0; j < M; ++j$) do                                        ▷ subsample and merge maps
        for ($i = 0; i < N; i = i + 2$) do
            if $j\%2 == 0$ then
                $map_{k\_c}[j][i] = map_{k\_x}[j][i]$
                $map_{k\_c}[j][i+1] = map_{k\_y}[j][i+1]$
            else
                $map_{k\_c}[j][i+1] = map_{k\_x}[j][i+1]$
                $map_{k\_c}[j][i] = map_{k\_y}[j][i]$
            end if
        end for
    end for

    for ($j = 0; j < M; ++j$) do                                        ▷ reinterpret values
        for ($i = N - 2; i > 1; i = i - 2$) do
            $map_{k\_c}[j][i] =$
              $\text{FIX}(map_{k\_c}[j][i]) - \text{FIX}(map_{k\_c}[j][i-2])$
            $map_{k\_c}[j][i+1] =$
              $\text{FIX}(map_{k\_c}[j][i+1]) - \text{FIX}(map_{k\_c}[j][i-1])$
        end for
        $map_{k\_c}[j][0] = round(map_{k\_c}[j][0])$                     ▷ round integer values for first two cols
        $map_{k\_c}[j][1] = round(map_{k\_c}[j][1])$
    end for
end for

function FIX($value, precision$)
    $round(value \cdot 2^{precision})/2^{precision}$
end function
```

The module consists essentially of four main parts (see Figure 1) - dual port RAM with BRAM address calculator, decompression, bilinear interpolation and verify on invalidity. Figure 2 shows the modified hardware architecture of the lens undistortion and image rectification module for one camera. The bold arrows show the main path of the image. In comparison to the presented hardware architecture in paper 8 a total of four dual port $36\,k$ RAMs are used for intermediate buffering of 50 rows of the raw image. The number of rows to be saved depends on the possible maximum vertical displacement, which depends on the base distance and the angle between both cameras. The maximum vertical displacement value is obtained from the transformation map $UndistRecrMap_{k\_y}$. The compressed rectification map is loaded synchronized by VDMA from the external memory. Figure 3 shows a block diagram of the decompression subsystem. The compressed transformation map map_{k_c} is compressed in reverse order to the compression algorithm 1 (see subsection 3.1.2). A line buffer is required to interpolate the missing values. After decompression, the two integer and two decimal values are available, which include the vertical and horizontal displacement.[5] The BRAM address calculator uses the two integer values to determine the new position of the grey value $G0$ stored in the dual port RAM. The calculation of the bilinear interpolation is done with the two decimal places and the four neighbouring grey values $G0 - G3$ (see Figure 1). The four determined grey values $G0 - G3$ are the grid points for the calculation of the bilinear

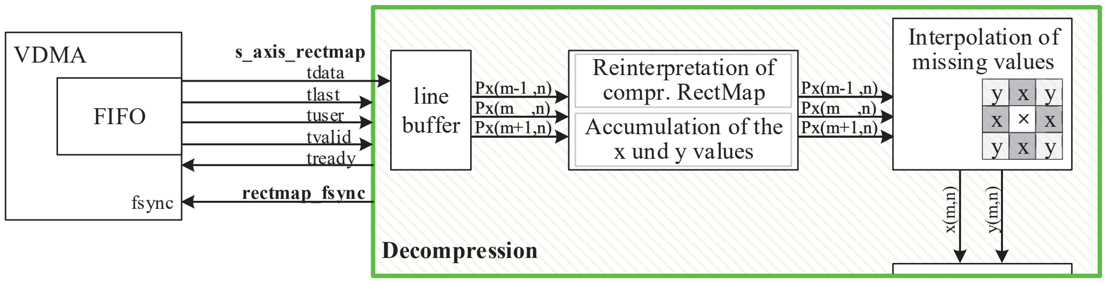

Figure 3. Subsystem *Decompression* from figures 1 and 2: Recalculation of the horizontal $x(m,n)$ and vertical $y(m,n)$ displacement values from compressed rectification map ($M \times N$).

interpolation. The Verify of Invalidity subsystem checks whether the pixel position of the calculated value is within the image area.

3.3 Interfaces and processing pipeline

Figure 2 shows the architecture and interfaces of this module. The module calculates a rectified image pixel by pixel. To do this, both data streams, which are streamed over an AXI4-Stream interface [12, pp. 5], must be synchronous, but still offset accordingly. The data management is very complex. Figure 4 shows the rough and simplified course of time. The two input streams and the output stream are visible - buffered values from raw image stream, decompressed values from the compressed transformation map map_{k_c} and rectified values from image stream. The both input signals are provided via Slave AXI4-Stream interface. Initially, some rows are buffered in four ring buffers (see range A). The number of the buffered rows depends on baseline between the both cameras and camera orientation. In our case 50 rows are sufficient. This is absolutely necessary so that grey values are also available at the corresponding calculated coordinates during the reverse transformation (see section 2). After calculating a rectified value, the next pixel of the raw image stream is buffered in one of the ring buffers (see range B). The pixels of the stored compressed transformation map are requested at a time offset to this image data stream. The map is requested from the VDMA via the signal *fsync*. A rectified pixel is calculated as soon as the ring buffer is filled accordingly and the two corresponding decompressed shift

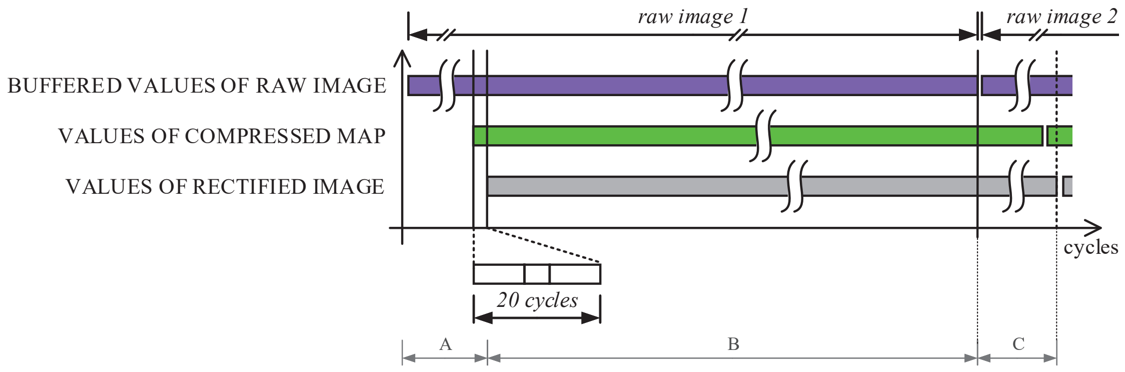

Figure 4. Simplified timeline of the processing pipeline. Filling buffer and start decompressing the transformation map (range A); starting pixelwise rectification (range B); rectify the values of the last 50 rows and start saving the values of the next raw image (range C).

values (horizontal and vertical from the transformation map) are available. With the beginning of range B, this is fulfilled for the first time. The rectification takes place a further 50 rows, although the entire raw image has already been written to the ring buffers (see range C). During this time it is possible to write the 50 rows of the next raw image into the ring buffers. The provision of the calculated rectified image is provided via the master AXI4 stream interface *m_axis*.

"This module is also configured via the Slave AXI4-Lite interface *s_axi_lite*. This interface inform about possible error messages such as synchronization error or an overflow when the fill level of AXI FIFO ImgRect is exceeded."[5]

4. PERFORMANCE OF THE SYSTEM

4.1 Use of resources

The undistortion and rectification ip core needs the following resources (see Table 1) with a maximum image size of 1280 Px × 960 Px and a maximum and minimum permissible vertical shift value of ± 24 Px. The values can be changed if desired. The IP core needs a BRAM load of 12 % when using a Xilinx Zynq 7020 SoC. This is made up of four ring buffers (bufferd raw image: $2 \cdot (24\,\text{Px} + 1) \cdot img_{with\_max} \cdot 8\,\text{bit}$) and one line buffer (buffered compressed rectification map: $img_{width\_max} \cdot 8\,\text{bit}$).

4.2 Deviations from different calculated rectified images

The lens distortion correction and image rectification was calculated without and with a compressed transformation map. In comparison to the use of uncompressed transformation maps the rectified image shows a deviation of one till five grey value at the most (along strong gradients) using a binary precision of seven decimal places for the offset values.

5. CONCLUSION AND FUTURE WORK

This paper shows an efficient FPGA-based lens distortion correction and image rectification module. To reduce the bandwidth utilization, it is necessary to use compressed lens undistortion and rectification transformation maps. Especially due to the effective lossy compression of the lens undistortion and rectification transformation maps, pixel by pixel rectification of an image is possible. Table 2 shows the bandwidth utilization of the different compressed undistortion and rectification transformation maps. With the new extension of the compression algorithm (see subsection 3.1.2), a reduction of the bandwidth utilization by a factor of eight is achieved. The extension of the compression algorithm in Ref. 8 is the additional subsample of the vertical and horizontal maps (see algorithm 1). Compared to the generated rectified images without and with compressed transformation maps, minor deviations of grey value are achieved (see subsection 4.2).

There are several applications of this FPGA-based lens undistortion and image rectification module, such as its use in a stereo-based phase measuring Profilometry system.[5] The presented lens undistortion and rectification ip core is used in a passive stereo system (using a Xilinx Zynq-7020 SoC) without structured-light illumination. Within the research group DIADEM, the stereo system will be integrated into a sensor arrangement with free-form projection in future work. The planned stereo sensor setup has a working distance of 500 mm, a base line with about 100 mm and a triangulation angle of 11°.

Table 1. Resource utilization of the *UndistRect IP-Core* in Xilinx Zynq 7020 SoC; at a set image size of 1280 Px × 960 Px and a set vertical maximum of 24 rows.

Ressource	LUT	LUTRAM	FF	BRAM	DSP	IO	BUFG
in %	3	1	2	12	7	33	3

Table 2. Difference in memory utilization between OpenCV and different compressed UndistRectMaps (abbr. maps). [a] *Zynq-7000 32-bit DDR3 memory controller: maximal theoretical bandwidth* 4267 MB/s *[13, p. 13]*

Maps per camera	**Map type**	**Data type**	**Size** (byte)	**Memory load** (MB/s) 2 Mpix, 60 fps	**Bandwidth utilization**[a] (%)
OpenCV	x	float	$(M \times N) \cdot 4$	960	22.5
	y	float	$(M \times N) \cdot 4$		
Compressed	merged[8]	unsigned short	$(M \times N) \cdot 2$	240	5.6
Compressed	subsample & merged	unsigned short	$(M \times N) \cdot 1$	120	2.8

ACKNOWLEDGMENTS

This research is supported by the Free State of Thuringia, the European Social Fund (ESF) of the European Union and the Thüringer Aufbaubank (TAB) within the research group DIADEM (2016 FGR 0044). Furthermore, this is funded by the Federal Ministry of Education and Research within the project FASTER (BMBF, FKZ: 03ZZZZ0442E).

REFERENCES

[1] Winkler, S., Rosenberger, M., Höhne, D., Munkelt, C., Liu, C., and Notni, G., "3d image acquisition and processing with high continuous data throughput for human-machine-interaction and adaptive manufacturing," *Engineering for a Changing World: Proceedings; 59th IWK, Ilmenau Scientific Colloquium, Technische Universität Ilmenau, September 11-15, 2017* **59, 2017** (Nov 2017).

[2] Munkelt, C., Heinze, M., Zimmermann, T., Kühmstedt, P., and Notni, G., "3d-sensornetzwerk mit geringer latenz für die echtzeit objektrekonstruktion," in [*119. Jahrestagung der DGaO, Aalen*], (2018).

[3] Won Nam, K., Park, J., Kim, I., and Kim, K., "Application of stereo-imaging technology to medical field," *Healthcare informatics research* **18**, 158–63 (09 2012).

[4] Zhan, G., Tang, H., Zhong, K., Li, Z., Shi, Y., and Wang, C., "High-speed fpga-based phase measuring profilometry architecture," *Opt. Express* **25**, 10553–10564 (May 2017).

[5] Hess, A., Junger, C., Rosenberger, M., and Notni, G., "FPGA-based phase measuring profilometry system," in [*Three-Dimensional Imaging, Visualization, and Display 2019*], Javidi, B., Son, J.-Y., and Matoba, O., eds., **10997**, 114 – 121, International Society for Optics and Photonics, SPIE (2019).

[6] Hartley, R. and Zisserman, A., [*Multiple View Geometry in Computer Vision*], Cambridge University Press, New York, NY, USA, 2 ed. (2003).

[7] Junger, C., Heß, A., Rosenberger, M., and Notni, G., "FPGA-based lens undistortion and image rectification for stereo vision applications," in [*Photonics and Education in Measurement Science 2019*], Rosenberger, M., Dittrich, P.-G., and Zagar, B., eds., **11144**, 284 – 291, International Society for Optics and Photonics, SPIE (2019).

[8] Junger, C., Heß, A., Rosenberger, M., and Notni, G., "FPGA-accelerated phase rectification for a stereo-based phase measuring profilometry system," *Journal of Physics: Conference Series* **1065**, 032017 (aug 2018).

[9] Akin, A., Gaemperle, L. M., Najibi, H., Schmid, A., and Leblebici, Y., "Enhanced compressed look-up-table based real-time rectification hardware," in [*VLSI-SoC: At the Crossroads of Emerging Trends - 21st IFIP WG 10.5/IEEE International Conference on Very Large Scale Integration, VLSI-SoC 2013, Istanbul, Turkey, October 6-9, 2013, Revised and Extended Selected Papers*], 227–248 (2013).

[10] Bradski, G. and Kaehler, A., [*Learning OpenCV: Computer Vision with the OpenCV Library*], O'Reilly Media, Sebastopol (CA) (2008 (first edition)).

[11] openCV dev team, "Camera calibration and 3D reconstruction." https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html. (v2.4.13.7, 03 May 2019).

[12] XILINX, “AXI4-Stream Video IP and System Design Guide.” https://www.xilinx.com/support/documentation/ip_documentation/axi_videoip/v1_0/ug934_axi_videoIP.pdf. (Doc Id: UG934, v2018.3, december 2018).

[13] Lucero, J. and Slous, B., “Designing high-performance video systems with the Zynq-7000 all programmable SoC using ip integrator XAPP1205.” https://www.xilinx.com/support/documentation/application_notes/xapp1205-high-performance-video-zynq.pdf. (v1.0, marche 2014).