A fisheye distortion correction algorithm optimized for hardware implementations

Conference Paper · June 2014

DOI: 10.1109/MIXDES.2014.6872232

4 authors, including:

Jakub Piotr Cłapa
Lodz University of Technology
3 PUBLICATIONS **15** CITATIONS

SEE PROFILE

Kamil Grabowski
FastLogic Sp. z .o.o.
29 PUBLICATIONS **431** CITATIONS

SEE PROFILE

Przemyslaw Sekalski
Lodz University of Technology
32 PUBLICATIONS **150** CITATIONS

SEE PROFILE

A Fisheye Distortion Correction Algorithm Optimized for Hardware Implementations

Jakub Cłapa, Henryk Błasiński, Kamil Grabowski, Przemysław Sękalski
Department of Microelectronics and Computer Science
Łódź University of Technology, Poland
{jclapa,hblasinski,kgrabowski,sekalski}@dmcs.pl

***Abstract*—This paper presents a universal algorithm for correcting distortion in images obtained with fisheye lenses. We analyze and optimize the algorithm for implementation on hardware platforms such as Field Programmable Gate Arrays (FPGAs). A fully pipelined architecture, implemented in an FPGA, is capable of processing full HD images at 30 frames per second. Furthermore the proposed algorithmic modifications introduce very little error in fixed point arithmetic pixel coordinate computations. When compared to double precision results, most coordinates have an error of less than 0.5 pixel.**

***Keywords*—fisheye lens, barrel distortion, distortion correction, real-time video processing, CORDIC, FPGA**

I. Introduction

Vast majority of camera systems used today have a field of view (FOV) of about 60 degrees, much less than the human visual field. However, larger FOVs can be achieved with specially designed wide angle or fisheye lenses that project much bigger part of the scene onto an image sensor. This increase in the field of view size is extremely useful in a number of applications ranging from laparoscopic surgery, through rear-view cameras in cars to closed circuit (CCTV) systems [1], [2]. Unfortunately, the increase in the FOV comes at a price of more pronounced geometric distortion causing straight lines from the scene to appear as curves in the image plane. Such a transformation produces less realistic images and affects objects' relative sizes, depending on their position in the image.

The problem of geometric distortion correction has been extensively studied in literature. For example [3] propose a polnominal based method for barrel distortion correction of images captured with a wide angle lens. Rectangular image pixel coordinates are transformed into polar coordinates. Next, taking advantage of the rotational symmetry around the optical axis, only pixel distances from the optical center are mapped according to some polynominal function. This algorithm was implemented a Field Programmable Gate Array (FPGA) by [4] who used CORDIC algorithm [5] to perform polar to rectangular coordinate conversion. The system was capable of processing 1024×1024 pixel images at 30 frames per second. Unfortunately, despite its simplicity, the algorithm is less applicable to images acquired with a fisheye lens, where only the central portion of the image can be well corrected [6], [7].

(a) Fisheye (b) ROI

Fig. 1. A stereographic fisheye image example. The ROI represents a synthesized rectilinear view of a region of interest (ROI) defined by the red dots. White line represents the image circle boundary.

Many studies focused on algorithms more suitable for fisheye lens distortion removal, achieving very good results [8]–[10]. Most of these algorithms avoid the hemisphere to plane projection problem by correcting a smaller fragment of the scene within a predefined region of interest (ROI). All algorithms, however, are computationally expensive and therefore real-time video processing may not be achievable on general computational platforms. For example [11] compare algorithm speed on three different architectures: a heterogenous multicore processor, a homogenous multicore processor and an FPGA. The latter platform achieved the highest frame rate of 22 fps, about four times faster than a software implementation.

Benefitting from hardware acceleration and design flexibility of FPGAs several hardware architectures for fisheye distortion correction have been proposed. It seems however that most designs do not attempt to create a system highly tailored to the target application. For example [12] use a special software tool to automate generation of hardware acceleration blocks, or [13] propose a memory intensive solution. Even the systems proposed by the industry follow a similar pattern. A fisheye distortion correction ASIC by [14] is composed of a general purpose processing unit used to pre-compute the image transformation and additional logic to copy and interpolate pixel values.

In this paper we are analyzing a fisheye distortion correction algorithm proposed by [15] and adapting this algorithm for implementation on an FPGA platform. Furthermore we generalize the method to a variety of fisheye lens types creating a parametrized intellectual property (IP) core applicable to any fisheye lens. Our solution performs all the computations as needed and does not cache any results, limiting external memory requirements. This simple and cost-effective architecture can be the key building block of a digital pan and tilt system [16], in which a fully stationary camera can mimic the behavior of an imaging device moved by external motors. The digital

solution, however, has the advantage of almost instantaneous viewpoint changes.

This paper is organised as follows. Section II briefly compares different types of fisheye lenses. The mathematics of the correction algorithm are presented in section III which is followed by the proposed adaptations to hardware platforms. Experimental results comparing the original and modified algorithms are presented in section V which are followed by conclusions and future research directions in section VI.

II. Fisheye Lens Models

One of the simplest camera models is the rectilinear camera also known as the pinhole camera. In this framework the image formation is modeled as a geometric projection about the pinhole. Figure 2 illustrates this principle with a 1D example. The distance between the point's projection and the sensor principal axis is given by $r = f \tan\beta$, where f denotes the focal length, and β is the incidence angle. For small and moderate values of β the distance r is also small. However, when a ray's incidence angle becomes less and less acute, then the corresponding points are projected further and further from the optical center, as illustrated by the incidence angle β_2 and the corresponding projection r_2. This model does not capture the behavior of fisheye lenses, which project rays incoming at angles close to $80°$ or $90°$ onto points on the sensor that are close to the optical axis. In addition the pinhole projection would severely distort the dimensions of objects positioned at periphery of the visual field.

One of the fundamental characteristics of a fisheye lens is the *lens function* which, similarly to the pinhole projection, relates the incidence angle of a ray with the distance from the optical axis of the point this ray represents. Assuming rotational symmetry about the optical axis this function is sufficient to relate any point in the scene with its projection in the image plane. Figure 4 presents schematic incidence ray projection patterns of lens functions most commonly used in fisheye design. All of these functions approach a finite value, when the incidence angle β reaches $90°$, and possibly even more.

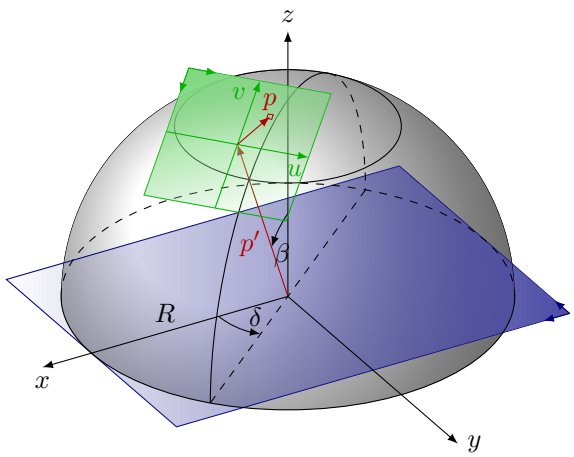


Fig. 3. Fisheye image formation model proposed by [15]. The xy plane represents the sensor, and the uv plane is the specified region of interest.

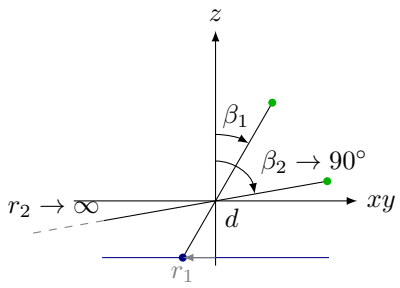


Fig. 2. The pinhole camera in 1D. When the ray incidence angle approaches $90°$, the required size of the planar image approaches infinity.

Fisheye lenses capture rays incoming from a much larger solid angle, than a traditional pinhole camera would. Therefore, a fisheye image contains enough information to recreate an image as seen by a rotated rectilinear camera, which would acquire a much smaller subset of rays. This phenomenon is illustrated in Fig. 1 which shows a typical image acquired with a stereographic fisheye lens (Samyang 8mm f/3.5) and a synthesized pinhole camera image obtained purely by transforming a region of interest from that image.

III. Correction Algorithm

An efficient algorithm for removing fisheye distortion from an ROI was proposed by [15]. Their method projects the acquired image onto a virtual hemisphere placed on top of the image sensor. The region of interest is a plane orthogonal to a specific ray intersecting the center of the hemisphere and located mR away from its center, where R is the hemisphere radius and m denotes the magnification. If $m = 1$ then the ROI plane is tangential to the hemisphere. Figure 3 illustrates the geometry of this model, the xy plane represents the sensor, and the uv plane is the selected ROI. The position of the ROI is defined in terms of the ray zenith β and azimuth δ angles. If the camera is not pointing directly up or down it is useful to incorporate an additional parameter, the uv plane rotation angle ϕ, to maintain a proper horizon by rotating the resulting image around its center. Finally, the radius of the fisheye image R must be measured during system calibration.

The distortion correction algorithm consists in expressing the 2D coordinates of a point in the ROI in terms of the 3D coordinate system and then using the lens function to compute the location of this point in the image plane. Let p represent an arbitrary point on the uv plane, this point can be expressed in terms of the uv coordinates as $p^{(uv)} = (p_u, p_v)$ or using the xyz reference frame, $p^{(xyz)} = (p_x, p_y, p_z)$. One approach to compute the conversion between the frames involves transforming unit vectors in the uv plane, and then using the linear relationship

$$p^{(xyz)} = p'^{(xyz)} + \left(p^{(xyz)} - p'^{(xyz)}\right) \quad (1)$$

$$= p'^{(xyz)} + p_u \hat{u}^{(xyz)} + p_v \hat{v}^{(xyz)}, \quad (2)$$

where $\hat{u}, \hat{v}$ are unit vectors in the uv plane and p' is a vector pointing at the center of the ROI with length equal to mR. These unit vectors can be calculated by a sequence of 2D vector rotations. Let $\hat{u}^{(uv)} = (1, 0)$ be one of the unit vectors,

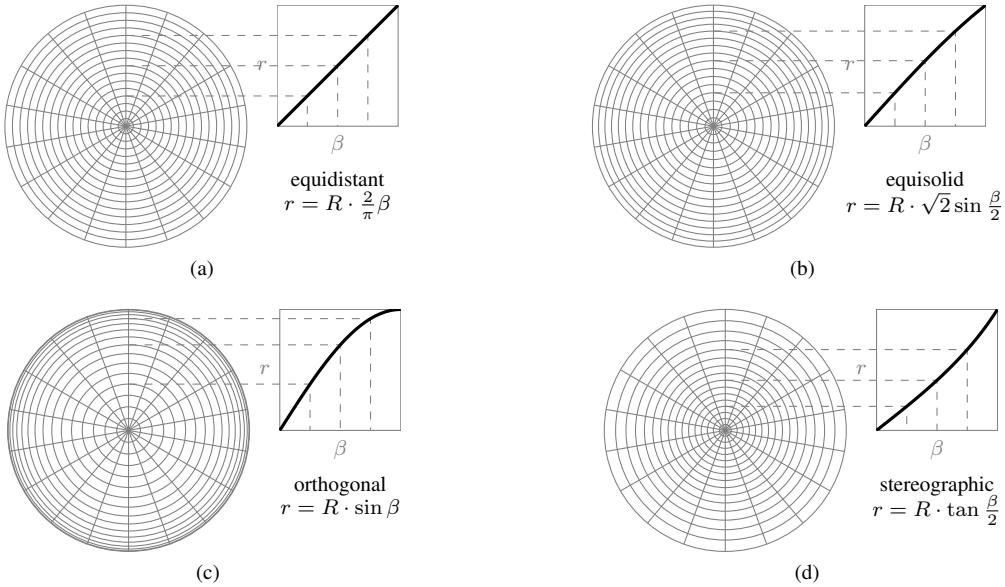


Fig. 4. Common fisheye lens functions, the equidistant projection (a) is often used in practice.

and $(k)_{\measuredangle\,\alpha}$ denote a rotation of a 2D vector k by an angle α, then its representation in the xyz frame is

$$a = (\hat{u}^{(uv)})_{\measuredangle\,\phi}, \tag{3a}$$
$$b = (a_1, 0)_{\measuredangle\,-\beta}, \tag{3b}$$
$$c = (b_1, a_2)_{\measuredangle\,\delta}, \tag{3c}$$
$$\hat{u}^{(xyz)} = (c_1, c_2, b_2), \tag{3d}$$

where a, b, c are vectors resulting from intermediate computations, and the ith coordinate of a vector q is denoted q_i. The same procedure can be repeated to obtain the transformation of the second unit vector $\hat{v}^{(uv)}$.

The next step involves a rectangular to polar conversion of $p^{(xyz)}$. The angles δ_p and β_p define the location of the ray intersection with the hemisphere,

$$\delta_p = \arctan\left(\frac{p_y}{p_x}\right), \tag{4a}$$
$$\beta_p = \arctan\left(\frac{p_{xy}}{p_z}\right), \tag{4b}$$

where $p_{xy} = \sqrt{(p_x)^2 + (p_y)^2}$. The angle β_p is the sensor incidence angle. If an orthogonal fisheye lens was used then (p_x, p_y) would give the exact location of the pixel in the image acquired with a fisheye lens. In a more general case, however, the lens function needs to be included in the analysis. For an arbitrary lens function F_{lens} these coordinates will be mapped to a point (p_{Fx}, p_{Fy}) given by

$$r_p = R\,F_{\text{lens}}(\beta_p), \tag{5a}$$
$$p_{Fx} = r_p \cos\delta_p, \tag{5b}$$
$$p_{Fy} = r_p \sin\delta_p. \tag{5c}$$

Note that the transformed unit vectors given by (3) are fixed for a given set of zenith, azimuth and rotation angles β, δ and ϕ as well as the magnification m. Furthermore, the distortion correction algorithm has been divided into a sequence of simple operations involving rectangular to polar conversions and function evaluation. These building blocks can now be efficiently implemented in hardware.

IV. Correction Block Design

For a given selection of the ROI the p' vector and the unit vectors of the uv plane remain constant, and therefore need to be computed only once. In contrast transformations given by equations (4) and (5) need to be executed for every output image pixel. Once the coordinates of a point $p^{(xyz)}$ are found by means of unit vector superposition, three major steps remain: rectangular to polar coordinate transformation, lens function mapping and finally polar to rectangular conversion.

The signal flow, and high level system block diagram are presented in Fig. 5. Modules *2polar* and *2rect* perform the polar to rectangular coordinate system conversion. These conversions can be efficiently performed in hardware using CORDIC algorithm [5]. The CORDIC engine iteratively rotates a two dimensional vector and keeps track of the accumulated angle. In most architectures in order to convert the angle estimate to radians a small look-up table needs to be implemented. This additional logic is not needed whenever the computed angle estimate is fed into another CORDIC module, as is the case

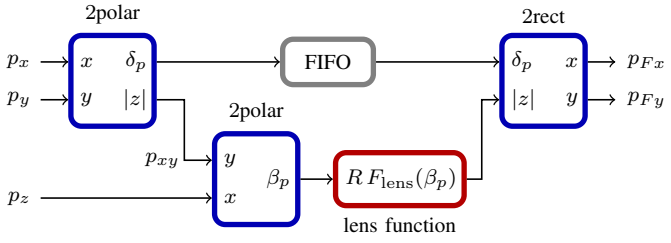
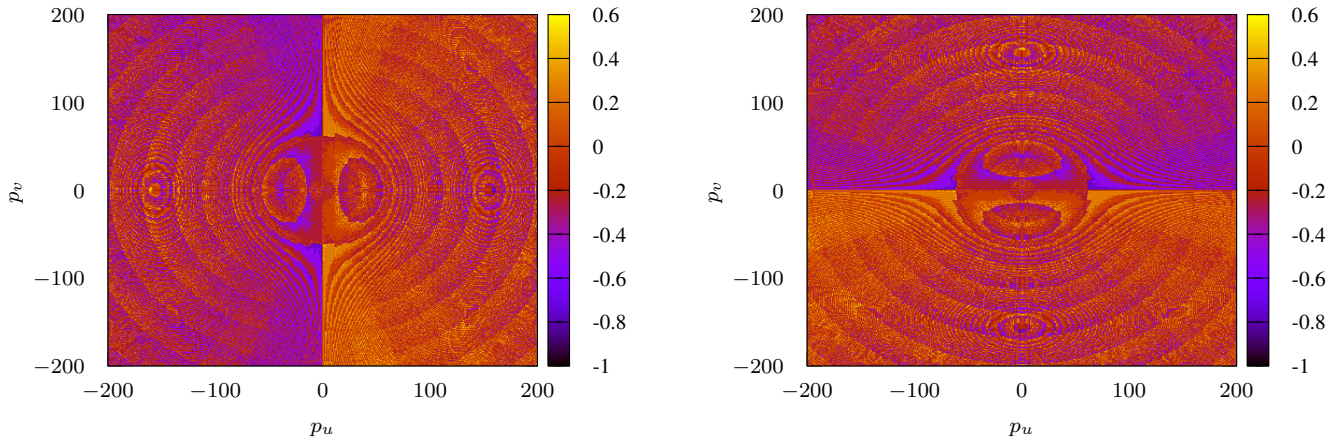


Fig. 5. Block diagram of the correction algorithm implementation. Blue boxes contain CORDIC engines. The red block performs the mapping according to a lens function.



(a) Coordinate p_{Fx} error

(b) Coordinate p_{Fy} error

Fig. 6. Spatial distributions of fixed point arithmetics coordinate approximation errors (in pixels).

of δ_p in our architecture. Similarly, the vector length estimate is scaled by a parameter k that is a function of the number of iterations. However, explicit division by this factor can be avoided if the increase is accounted for in subsequent logic blocks.

The second element in our architecture is the lens function block which performs the incidence angle β_p to image radius r_p mapping. The implementation details of this block depend on the fisheye lens type and the specifics of the distortion curve. Possible architectures can range from a simple look-up table through multiplier based implementations similar to [6], [7], to CORDIC based trigonometric computations. Finally a small FIFO is required to account for pipeline delays in the *2polar* and *lens function* blocks.

V. Results

The hardware system was described using VHDL and implemented in a XC6SLX45CSG324-2 Spartan 6 FPGA available on a Starter Kit. All coordinate conversion blocks contained CORDIC IP cores provided by Xilinx. The distortion removal core was managed by a Microblaze microcontroller. The final solution was fully pipelined and thus pixel coordinates were available at every clock cycle. The pipeline length was 76 cycles. With the FPGA running at 100Mhz and a target of 30 frames per second, the coordinate conversion block is capable of processing 3 megapixel images. In comparison a standard full HD image is about 2 megapixels. Table I summarizes the resource utilization by block. Overall less than 16% of logic slices are used, and most of the resources are consumed by the three CORDIC modules. Also this implementation assumed a simple equidistant lens function model, equation of which is a straight line. For this reason the compact size of the *Lens function* block is by no means typical.

Coordinate maps generated by our block were compared to ones computed using double precision floating point algebra. Figures 6a and 6b present error spatial maps for p_{Fx} and p_{Fy} coordinates respectively. More than 98% of pixels in these images have errors smaller than 0.5 pixel demonstrating the high accuracy of the proposed hardware implementation.

TABLE I. Spartan 6 FPGA resource usage.

Module	Slices	FF	LUT	LUT-RAM	DSP48A1
2polar (p_x, p_y)	346	1179	1053	26	4
2polar (p_{xy},p_z)	300	1021	1007	10	0
lens function	24	16	82	0	1
2rect	372	1252	1193	18	8
FIFO	18	32	32	16	0
Total	1060	3500	3367	70	13
	15,54%	6,41%	12,34%	1,09%	22,41%

VI. Conclusions

This paper presents an analysis of a fisheye distortion correction algorithm and its modifications for hardware implementations. We have separated the ROI specific computations from those needed for every output image pixel. We then efficiently implemented pixel coordinate transformations using CORDIC engines. Finally, the architecture we proposed can easily be adapted to any type of fisheye lens. The system we described was implemented and tested using an FPGA and the hardware was capable of processing full HD images at 30 frames per second. Numerical inaccuracies in coordinate computations, resulting from fixed point arithmetics, in most cases did not exceed 0.5 pixel.

The distortion correction block described in this paper is the main element of a digital pan and tilt system. Future work will include implementing the entire image processing pipeline in an FPGA and optimization of the CORDIC architecture as described in section IV. Other improvements could include the addition of a gyroscope for system orientation and automatic horizon selection, or a pixel bilinear interpolation block.

Acknowledgments

The authors would like to thank Mateusz Michalak, Dariusz Makowski, PhD and Bartosz Sakowicz, PhD for their helpful comments and continued support. This project was financially supported by the Polish National Center for Research and Development (grant number LIDER/30/110/L-3/11/NCBR/2012).

REFERENCES

[1] C. Hughes, M. Glavin, E. Jones, and P. Denny, "Wide-angle camera technology for automotive applications: a review," *IET Intelligent Transport Systems*, vol. 3, no. 1, pp. 19–31, 2009.

[2] L. Meinel, M. Findeisen, M. Hes, A. Apitzsch, and G. Hirtz, "Automated real-time surveillance for ambient assisted living using an omnidirectional camera," in *IEEE International Conference on Consumer Electronics, ICCE*, 2014, pp. 396–399.

[3] S. Shah and J. Aggarwal, "A simple calibration procedure for fish-eye (high distortion) lens camera," in *IEEE International Conference on Robotics and Automation*, May 1994, pp. 3422–3427.

[4] H. Ngo and V. Asari, "A pipelined architecture for real-time correction of barrel distortion in wide-angle camera images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 3, pp. 436–444, March 2005.

[5] R. Andraka, "A survey of cordic algorithms for fpga based computers," in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. ACM, 1998, pp. 191–200.

[6] H. Blasinski, W. Hai, and F. Lohier, "Fpga architecture for real-time barrel distortion correction of colour images," in *IEEE International Conference on Multimedia and Expo, ICME*, July 2011, pp. 1–6.

[7] ——, "Real-time, color image barrel distortion removal," in *IEEE International Symposium on Circuits and Systems, ISCAS*, May 2012, pp. 1911–1914.

[8] F. Devernay and O. Faugeras, "Straight lines have to be straight," *Machine Vision and Applications*, vol. 13, no. 1, pp. 14–24, 2001.

[9] J. Courbon, Y. Mezouar, L. Eck, and P. Martinet, "A generic fisheye camera model for robotic applications," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2007, pp. 1683–1688.

[10] C. Hughes, P. Denny, M. Glavin, and E. Jones, "Equidistant fish-eye calibration and rectification by vanishing point extraction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2289–2296, 2010.

[11] K. Daloukas, C. Antonopoulos, N. Bellas, and S. Chai, "Fisheye lens distortion correction on multicore and hardware accelerator platforms," in *IEEE International Symposium on Parallel Distributed Processing, IPDPS*, April 2010, pp. 1–10.

[12] N. Bellas, S. Chai, M. Dwyer, and D. Linzmeier, "Real-time fisheye lens distortion correction using automatically generated streaming accelerators," in *IEEE Symposium on Field Programmable Custom Computing Machines, FCCM*, April 2009, pp. 149–156.

[13] B. Zhang, Z. Qi, J. Zhu, and Z. Cao, "Omnidirection image restoration based on spherical perspective projection," in *IEEE Asia Pacific Conference on Circuits and Systems, APCCAS*, Nov 2008, pp. 922–925.

[14] Intersil/Techwell, *Fisheye Distortion Correction LSI with Built-in Image Signal Processor, Motion Detector and DDR2 Controller*, 2014.

[15] D. Kuban, H. Martin, and S. Zimmermann, "Omniview motionless camera endoscopy system," May 17 1994, US Patent 5,313,306.

[16] M.-S. Lee, M. Nicolescu, and G. Medioni, "Fast digital pan tilt zoom video," Aug. 17 2004, US Patent 6,778,207.