Reconfigurable Forward Homography Estimation System for Real-Time Applications

Vladan Popovic and Yusuf Leblebici
Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.
vladan.popovic@epfl.ch, yusuf.leblebici@epfl.ch

***Abstract*—Image processing and computer vision algorithms extensively use projections, such as homography, as one of the processing steps. Systems for homography calculation usually observe homography as an inverse problem and provide an exact solution. However, the systems processing larger resolution images cannot meet inherently tight real-time constraints. Look-up table based systems provide an option for forward homography solutions, but they require large memory availability. Recent compressed look-up table methods reduce the memory requirements at the expense of lower peak signal-to-noise-ratio. In this work, we present a forward homography estimation algorithm which provides higher image quality than compressed look-up table methods. The algorithm is based on bounding the homography error, and neglecting the pixels out of the determined bound. The presented FPGA implementation of the estimation system requires a small amount of hardware, and no memory storage. The prototype system project an image frame onto a spherical surface at 295 Mpixels/s rate which is, up to our knowledge, currently the fastest homography system.**

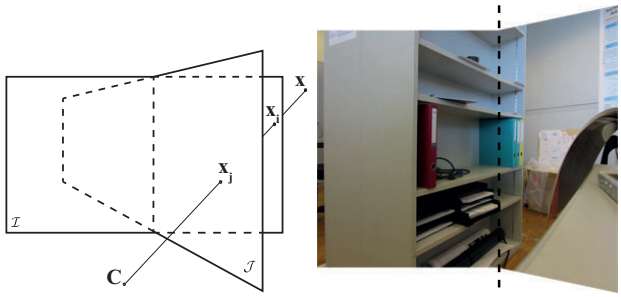

Fig. 1. a) Illustration of 2D homography on the example of two different views of the same scene point $\mathbf{x}$. Points $\mathbf{x_i}$ and $\mathbf{x_j}$ are related by homography; b) Homography example on a real photograph. The same photograph is projected on two different view planes. Intersection line of the two planes is marked by a dashed line. Change of perspective is noticed in the right part of the image.

I. Introduction

A homography, or projective transformation, is a mapping of vectors belonging to the same projective space. It is often used in image processing and computer vision to change the perspective view of the image [1]–[3]. An illustration of a projective transformation is shown in Fig. 1(a). The planar view $\mathcal{I}$ of the scene is transformed into an alternate view $\mathcal{J}$. The point $\mathbf{x}$ in the scene is observed at different locations in two given views, which is illustrated in Fig. 1(b) on a real image example.

A homography can be defined for any N-dimensional projective space. However, the most common ones are two-dimensional (planar) and three-dimensional (spherical and cylindrical) projections. Besides changing the perspective view, planar homography is used for camera calibration [4], stereo camera rectification [5]–[8], and object tracking and focusing [9], [10]. Applications of 3D homography vary from estimation of 3D scene models [11] and volume [12] to generating cylindrical and spherical panoramas [13], [14].

In this work, we consider applications such as real-time spherical panorama construction. Real-time homography is usually perceived as an inverse problem thanks to rather simple reconstruction pipeline. Reconstructing panoramic views requires multiple shots that are stitched into one single image. Inverse homography is suitable for this application, since the raw images can be stored into memory before performing the actual reconstruction. For each desired pixel in the reconstruction, the most appropriate pixel can be found in the original images. The mapping function is either determined by using runtime calculations [14] or pre-calculated and stored in lookup tables (LUTs) [13]. Thus, inverse homography is determined by straightforward implementation of mathematical functions [14].

However, timing constraints become very tight when the desired output resolution is high. The image processing systems can hardly meet the real-time constraints of 25-30 frames per second (fps) when reconstructing high resolution images. Forward homography is suggested as a possible solution to this problem in stereo image rectification systems [6]–[8], where the same real-time constraints apply.

The forward homography solves the issue of system constraints, such as memory bandwidth, since the correct destination is calculated for each input pixel. Hence, only the necessary pixels for the final reconstruction are stored in memory, thus reducing the required bandwidth. The destination coordinates are pre-calculated offline and stored in registers of the processing system. Another advantage of the LUT-based approach is that LUT size is independent of the final image resolution.

However, LUT size linearly increases with respect to the raw image resolution. Modern cameras have more than 10 Mpixels, and LUTs become too large for on-chip storage. Compressed LUT methods [8] may partially solve this problem, but the peak-signal-to-noise ratio (PSNR) drops signifi-

978-1-4799-6016-3/14/$31.00

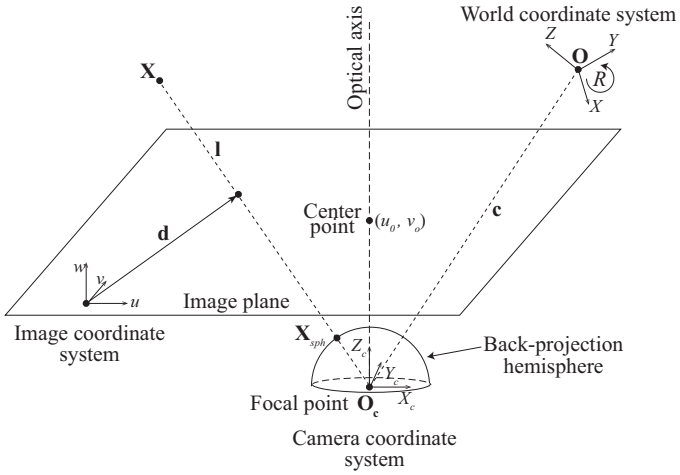

Fig. 2. Geometric transformations during image formation process. The world, camera, and image coordinate systems are shown with relations between them. A light ray passing through the world scene point $\mathbf{X}$ and the sensor's focal point intersects the image plane in point $\mathbf{d}$, which represents a pixel in the acquired image. The back-projection procedure reconstructs the original light ray $\mathbf{l}$ and locates the intersection point with the back-projection hemisphere, $\mathbf{X}_{sph}$.

cantly in the presence of large differences between input and output image resolution. These differences are observed in the majority of modern cameras, whose high resolution images are usually displayed on 2 Mpixels monitors. Analysis of this problem is given in Section III.

We propose a novel method for real-time forward homography based on direct calculation of destination pixels. The system operates on a pixel stream coming from a camera, and it is appropriate even for large ratios of input and output resolutions. We determine the optimal projection error bound based on camera calibration, and disregard all projections with the error higher than the determined bound. The chosen error bound provides the highest PSNR of the projected image. The algorithm is prototyped on an FPGA development system for the spherical panorama application. The prototype provides input/output pixel ratio of more than 200, *i.e.* more than 200 pixels are mapped into a single one.

II. Theoretical Background

The image formation can be approximated by the pinhole camera model for many computer vision applications [1]. The pinhole camera projects a 3D world scene into a 2D image plane. In order to perform this projection, three coordinate systems are considered, as depicted in Fig. 2. The coordinates of point $\mathbf{X}$ are expressed in the world coordinate system with its origin at the point $\mathbf{O}$. The same point can also be expressed in the camera coordinate system by coordinates $\mathbf{X_c}$. Origin of the camera coordinate system coincides with the camera's focal point $\mathbf{O_c} = \mathbf{C}$. The relation between the world and camera coordinates is unique and consists of a single translation and a single rotation:

$$\mathbf{X_c} = R(\mathbf{X} - \mathbf{c}) \tag{1}$$

The vector $\mathbf{c}$ denotes the distance between origins of the world and the camera coordinate systems, whereas the rotation matrix R expresses three Euler rotations in order to align the mentioned systems' axes. Parameters R and $\mathbf{c}$ are called extrinsic camera calibration parameters.

The image coordinate system, has aligned axes with the camera coordinate system. Position of the projected point in the image coordinate system is expressed by:

$$\mathbf{d} = M(\mathbf{X} - \mathbf{c}) = K\mathbf{X_c} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{X_c} \tag{2}$$

where $\mathbf{d}$ is the pixel position in the image coordinate system, f is the focal length, M is the projection matrix, and K is the intrinsic calibration matrix. The origin of the system is coincident with one of the corners of the image sensor. Thus, an additional shift of the sensor's central point to the sensor's corner is needed to obtain the final pixel position.

III. Estimation Algorithm

Estimating a forward homography in real-time is not a trivial problem. The system should determine the final pixel position, *e.g.* position in a panoramic image, based only on the pixel coordinates in the original frame. The problem arises due to non-integer values of the mapped pixel coordinates (see Fig. 4). When observing homography as an inverse problem, it is easy to scan through the desired pixel grid and choose the closest pixel from the original frame. Forward homography processes a pixel stream, and the system can determine the closest position on the destination pixel grid. However, it cannot determine if the current pixel in the stream is the best option. Thus, pixels that are mapped to the same position are overwritten and the last pixel that appears in the stream will be considered as the correct one. Hence, the PSNR is significantly decreased. This problem is even more emphasized in modern high-resolution cameras when hundreds of pixels are mapped into a single one, which is shown in the results section.

We developed a new homography estimation algorithm to overcome this issue. Equation (2) expresses the projection of a point in the 3D space onto the image plane. The first step of the algorithm is to back-project the pixels from the image frame. Each pixel $\mathbf{d}$ is back-projected into a line $\mathbf{l}$ in a 3D world that includes the focal point (projection center) $\mathbf{O_c}$. The line is illustrated in Fig. 2 and expressed by the inverse of Equation (2):

$$\mathbf{l} = M^{+} \begin{bmatrix} \mathbf{d} & 1 \end{bmatrix}^{\top} \tag{3}$$

where M^{+} denotes a Moore-Penrose pseudoinverse of the projection matrix. Thus, back-projection results in a set of lines (light rays), where each of them contains the focal point of the lens. In order to obtain 3D world coordinates, we should define a back-projection surface. We choose a unit sphere, $|r| = 1$, for the purpose of panorama construction. Other options include a cylindrical projection, or a planar projection for image rectification or disparity estimation applications. Back-projection onto the unit sphere is performed by normalizing the line $\mathbf{l}$ by its L^2 norm, and transforming Cartesian (x, y, z) coordinates into spherical (θ, ϕ, r), where θ is the polar angle, ϕ is the azimuth, and r is the radius:

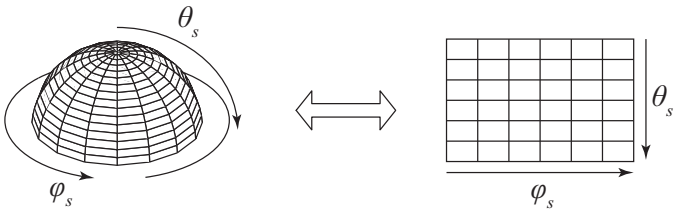

Fig. 3. The pixelized hemisphere on which the pixels are back-projected is shown on the left. The hemisphere can be unwrapped into an equivalent planar image shown on the right. The back-projection problem is regarded as homography between the original image coordinate system (source frame) and the unwrapped hemisphere (destination frame).

$$\begin{aligned} \mathbf{X}_{sph} &= \mathbf{l} \,/\, ||\mathbf{l}||_2 \\ \theta &= \arccos\left(\mathbf{X}_{sph}(z)\right) \\ \phi &= \arctan\left(\mathbf{X}_{sph}(y)/\mathbf{X}_{sph}(x)\right) \\ |r| &= 1 \end{aligned} \tag{4}$$

The spherical pixel grid is defined by the equidistant angles $(\theta_s, \phi_s, 1)$, where each pixel corresponds to a single angle pair. Fig. 3 shows a pixelized sphere and its unwrapped, planar representation.

The camera projection matrix M is obtained by camera calibration and the back-projection pixel grid (θ_s, ϕ_s) is defined *a priori*. If $\mathbf{X}_{sph}$ is the back-projected pixel, and $\mathbf{X}_s$ is the desired pixel on the hemispherical pixel grid, we define a projections error as:

$$e = ||\mathbf{X}_{sph} - \mathbf{X}_s||_2 \tag{5}$$

Afterwards, we find a threshold value ϵ, such that at least one distinct back-projected pixel $\mathbf{X}_{sph}$ exists for each grid pixel $\mathbf{X}_s$ with the error $e \leq \epsilon \leq \frac{\Delta}{\sqrt{2}}$, where Δ is the distance between two pixels on the hemisphere.

Five outcomes are possible in a 2D homography depending on the source and destination pixel positions. The simplest one is a 1-to-1 mapping when each pixel from the source frame maps to one in the destination frame. Furthermore, 1-to-0 and 0-to-1 are also possible, when the source pixel does not have a corresponding pixel in the destination frame, and vice versa. These three mappings are trivial cases and they will not be considered in the analysis. Complex 1-to-N and N-to-1 mappings occur when destination and source frames are oversampled, respectively. A possible real-time solution for 1-to-N mapping is suggested in [8]. However, a real-time solution with high PSNR has not yet been presented for N-to-1 mapping.

We resolve the 1-to-N mapping in the estimation algorithm by choosing the optimal ϵ. The optimal ϵ value ensures a distinct source pixel for each ϵ-neighborhood in the destination frame. Hence, a 1-to-N mapping is replaced by N 1-to-1 mappings.

Oppositely, ϵ value should be kept as low as possible in order to efficiently resolve N-to-1 mappings. The problem that arises in forward homography is that the N^{th} pixel in the stream is considered as the correct, unless a full mapping is stored in internal LUT or external memory. Thus, the optimal ϵ is the lowest value that ensures 1-N resolving. Reduction of the ϵ value in the proposed estimation, also reduces the number of pixel candidates to $M < N$. The benefit of this reduction is two-folded: 1) increased chance of choosing the optimal pixel, and 2) lower error and higher PSNR when non-optimal pixel is chosen.

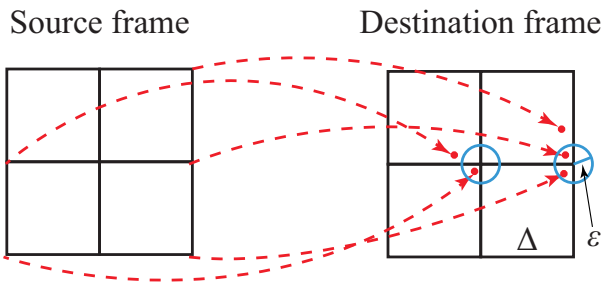

Fig. 4. Possible results of the proposed forward homography estimator. Intersections of black lines represent source and destination grid pixels, whereas red lines and dots are projections and projected pixel locations. The blue circles of radius ϵ mark the area in which the projected pixels are considered correct. When more than one pixel is within the blue circle, value of the last pixel in the incoming pixel stream is assigned to the pixel in the circle's center.

The homography between the image frame and unwrapped hemispherical surface is shown in Fig. 4. Intersections of black lines represent pixel positions on the respective grids. Red dashed lines illustrate homography between two frames, and red dots are projected pixel positions in the destination frame. Distances between red points and the closest intersection of black lines is the corresponding error e. The blue circles mark the ϵ-neighborhood in which projected pixels are considered as potential candidates for the final pixel value.

Fig. 4 illustrates two different cases of N-to-1 homography. Two source pixels on the left are mapped to the vicinity of a single destination pixel. The ϵ-neighborhood around the central destination pixel is set such that only one of the mapped pixels is inside the circle. Hence, the central destination pixel is given the value of the pixel inside the circle, which is indeed the closest projected pixel. In another situation, three pixels on the right side of the source frame are projected around one destination pixel. Two projections are inside the ϵ-neighborhood and one of them will be chosen as the destination pixel, *i.e.* the last one read out from the sensor.

IV. FPGA Implementation

An FPGA-based system is developed for the purpose of prototyping the real-time forward homography estimation. The full system architecture is shown in Fig. 5. The camera provides pixel values p in the raster scan order, *i.e.* line-by-line. Image is acquired using a single CMV20000 sensor, which outputs 20 Mpixels frames at maximum 30 frames per second (fps) rate. Pixels are streamed out of the camera as serialized 12-bit raw Bayer data. The Camera Interface block in Fig. 5 deserializes the pixel values and performs image pre-processing operations, such as white balancing and demosaicing. Additionally, an internal counter generates horizontal and vertical coordinates (u, v) of the processed pixel in the image frame (Fig. 2).

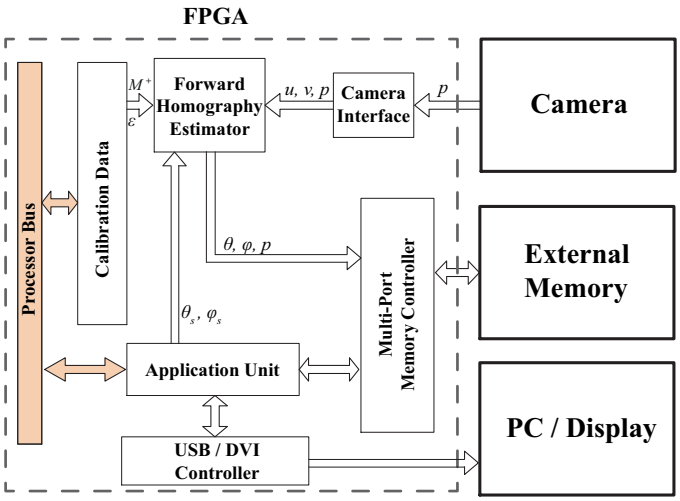

Fig. 5. Top-level FPGA architecture of the implemented system with the external peripherals. Dataflow is denoted by arrow directions.

Calibration Data block contains a bank of software accessible registers, and it is accessed by a MicroBlaze softcore microprocessor. The data stored in registers is obtained through the processes of intrinsic and extrinsic camera calibration and projection matrix calculation. The calibration data comprises the lens focal length f, sensor's central point position (u_0, v_0), the rotation matrix R, and the translation vector $\mathbf{c}$. These four parameters are used to calculate the projection matrix M and its pseudo-inverse. Only the elements of the pseudo-inverse matrix M^+ are stored in the registers of the Calibration Data block, since they are required for forward homography estimation.

The real-time image processing is realized inside the Application Unit. We implement the spherical panorama projection, which provides pixel positions on the spherical grid (θ_s, ϕ_s) to the Forward Homography Estimator (FHE). Since only one camera is connected in the current implementation, the final panorama results in a reduced field-of-view.

Internal architecture of the Forward Homography Estimator is shown in Fig. 6. Subtraction of the camera center point position translates the image frame origin to the frame center. Different row vectors of the matrix M^+ are provided to the dot product blocks, that evaluate the matrix multiplication in (3). A single dot product block in Fig. 6 is implemented as a pipelined multiply-accumulate unit in order to increase the speed performance of the system.

Equation (4) expresses the hemispherical back-projection and coordinate system change from Cartesian to spherical. The L^2 norm sub-block in Fig. 6 consists of two consecutive square root calculations. The square root module implements a CORDIC algorithm in the vectoring mode, which calculates the L^2 norm of its two inputs, *i.e.* $\sqrt{a^2 + b^2}$. The dividers for coordinate normalization are implemented using the iterative fast Anderson algorithm [15]. The transformation of coordinate system requires evaluation of inverse trigonometrical functions $\arctan$ and $\arccos$. The spherical angles (θ, ϕ) are calculated by applying the CORDIC algorithm to the Cartesian coordinates, as illustrated in Fig. 6.

The Application Unit provides information on the desired pixel grid . The error e from (5) is evaluated by the identical square root module used for the previous L^2 norm calculations. The error is compared to the pre-calculated ϵ, which is provided by MicroBlaze through a software accessible register. Output of the comparator serves as the output enable signal for a set of output registers, and as a write enable signal for the Multi-Port Memory Controller.

The FHE is a fully pipelined block. Each computation is followed by a register to shorten the critical path and increase the maximum frequency. Furthermore, each sub-block, *e.g.* dot product, square root, or trigonometric functions, is also pipelined providing a very fast operation of the system. The pipeline registers are not shown in Fig. 6.

A. Reconfigurability

The presented system is highly reconfigurable and has five degrees of freedom: camera resolution and frame rate, display frame resolution, projection surface, and the number of cameras. Fully pipelined processing allows change of camera resolution and frame rate even during the operation. The counter in the Camera Interface block uses horizontal and vertical synchronization signals from the camera to properly reset u and v values, hence it does not require resolution and frame rate information neither during synthesis, nor in run time. Modifying the display resolution or projection surface is performed in MicroBlaze and it does not influence the internal architecture of the FHE. MicroBlaze recalculates inverse M^+ of the projection matrix and the optimal ϵ, whereas operation of the FHE remains unchanged.

Additionally, the system can be reconfigured to support more than one camera. Multiple Camera Interface and FHE blocks can be instantiated and operated in parallel and independently. Thanks to the parallel operation, the maximum number of cameras is not constrained by system's performance, but by number of I/O pins and external memory bandwidth.

Furthermore, the proposed forward homography estimation is not dedicated to the developed prototype, and it can be used in any real-time system requiring image projections.

V. Experimental Results

The proposed FPGA design is implemented on a Xilinx Virtex-5 XUPV5-LX110T Evaluation Platform. A CMOSIS CMV20000 camera is mounted on a custom designed PCB, which is connected to the evaluation board. A photograph of the built setup is shown in Fig. 7. FPGA utilization summary of the FHE block and comparison with the designs that can perform the same function is given in Table I. The utilization reports were taken from the original publications. Even though these designs present their results for the image rectification applications, they are essentially implementations of forward homography. All of the compared works implement LUT-based methods, and hence require a lot of on-chip or off-chip memory. The proposed design uses less than 2% of the available LUTs and flip-flops on the used FPGA, and no internal nor external memory. Slightly higher utilization in terms of FPGA logic is due to real-time calculation of homography. The advantage of this design over the LUT-based

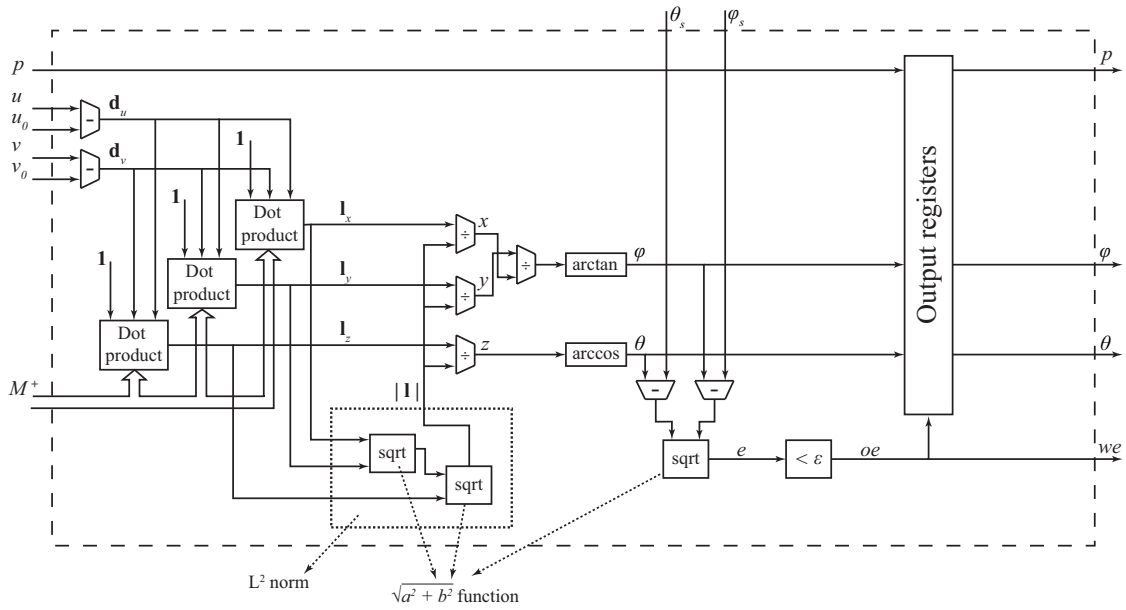

Fig. 6. Internal architecture of Forward Homography Estimator. The presented hardware evaluates expressions (3) – (5) using pipelined architecture. Pipeline registers are not shown for better visibility. The sub-blocks for square root and trigonometric functions evaluation utilize the CORDIC algorithm, whereas the fast Anderson algorithm [15] is used for implementation of the dividers.

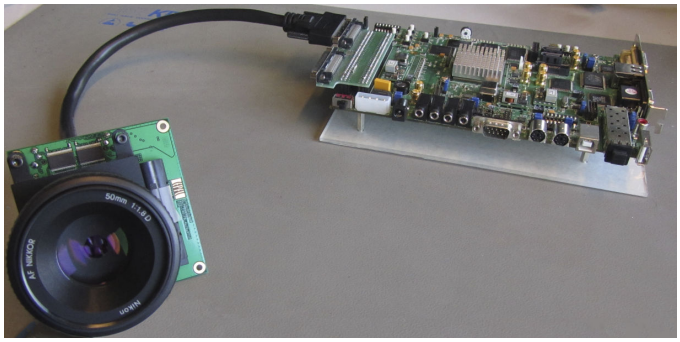

Fig. 7. Setup of the prototype system. A custom-made PCB with the installed image sensor is connected to the XUPV5 board with a VHDCI cable. A 50 mm Nikon F-mount lens is installed on top of the sensor.

TABLE I. FPGA DEVICE UTILIZATION SUMMARY AND COMPARISON

	FHE	[6]	[7]	[8]
Platform	Virtex-5	Virtex-5	Virtex-E	Virtex-5
Resolution	5120×3840	1280×720	640×512	1024×768
LUTs	1848	N/A	2459	784
Registers	1422	N/A	2075	427
BRAM [kB]	0	1300	99	104
DSP48Es	32	N/A	N/A	N/A

methods is that it can support very high image resolutions without any addition to the inferred hardware.

The measured maximum operating frequency of the implemented system is 308.16 MHz, which allows the system to process 15 fps of the full-resolution 20 Mpixels frames, *i.e.* 295 Mpixels/s. Furthermore, the input image resolution is reduced to 1024 × 768 for the purpose of comparison with

TABLE II. PSNR COMPARISON OF HOMOGRAPHY ESTIMATION

PSNR [dB]	Downsampled *Indoor*	*Indoor*	*Outdoor*
Resolution	1024×768	5120×3840	5120×3840
FHE	42.20	42.03	38.44
[8]	39.58	36.72	33.41

[8]. The measured frame rate is 391 fps, compared to 347 fps in [8].

The place and route tool reports a critical path through the large adders in the dot product evaluation sub-block. These adders utilize the FPGA logical elements, unlike multipliers that infer DSP48 blocks. Thus, it is possible to further increase the bandwidth by manually replacing the large adders with faster DSP48 blocks.

Results of the proposed forward homography estimation and its FPGA implementation are shown in Fig. 8(a) - 8(f). Fig. 8(a) and Fig. 8(d) represent an indoor and an outdoor scene at the input of the FHE system. Fig. 8(b) and Fig. 8(e) show the reduced field-of-view spherical panorama, at the output of the FHE system. Fig. 8(c) and Fig. 8(f) illustrate the unwrapped sphere suitable for display on 2 Mpixels screens.

The image quality loss is also evaluated for FHE and compared to the loss of LUT-based implementations. PSNR of the image in Fig. 8(c) and Fig. 8(f) are calculated, considering the inverse homography results as the ground truth images. Resolution of the reconstructed frame is 1920 × 1080, and region where the source frames are projected contains 87823 pixels.

Table II shows that the PSNR is increased compared to the LUT-based methods even in low-resolution projections, such as the downsampled *Indoor* image. Effects of the proposed ho-

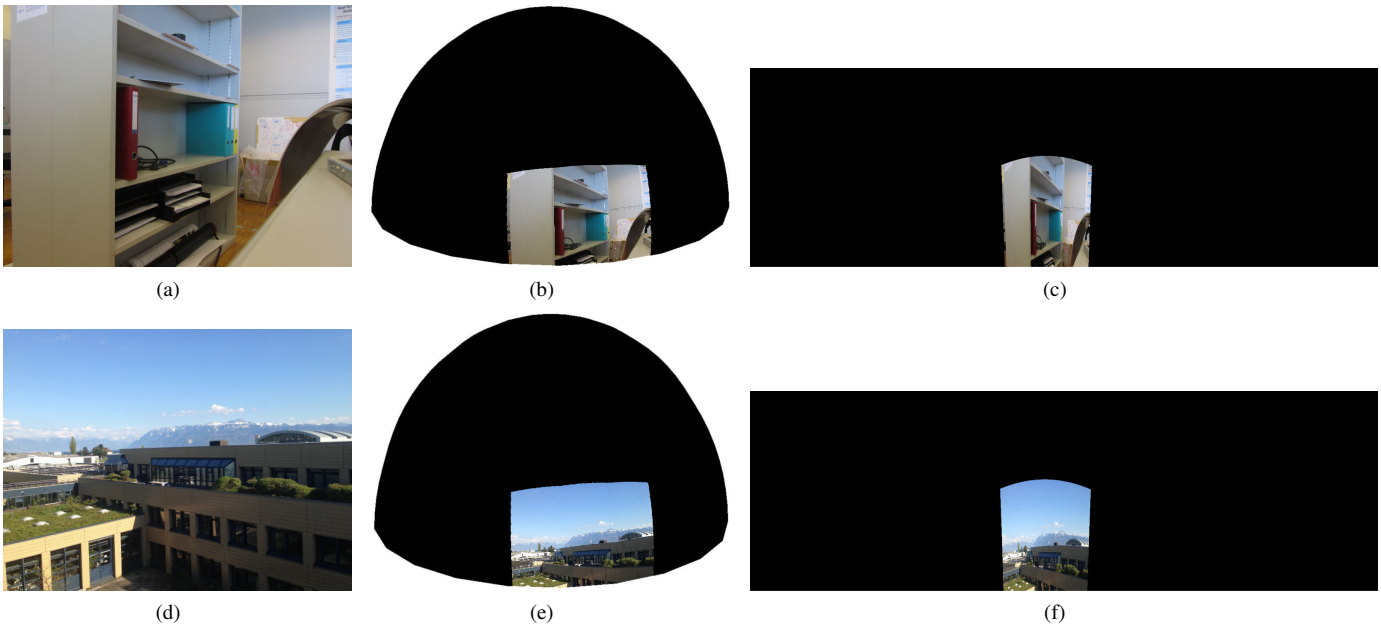

Fig. 8. Results of the proposed forward homography estimation: (a) and (d) The original 20 Mpixels *Indoor* and *Outdoor* images; (b) and (e) Outputs of the forward homography estimator displayed as true spherical projections; (c) and (f) unwrapped spherical projections displayed as 1920 $\times$ 1080 pixels panoramas.

mography estimation are emphasized with large input images. The PSNR of the full resolution *Indoor* image homography suffers a negligible drop, while the PSNR of the LUT-based method drops by 3 dB. The input image resolution does not influence the PSNR thanks to the flexibility of the ϵ parameter. The optimal error bound for the full resolution image is $\epsilon = 0.0061$, whereas $\epsilon = 0.11$ for the low resolution projection.

VI. Conclusion

In this manuscript, we presented a real-time forward homography estimation system. The proposed estimation is based on bounding the projection error, and reducing the number of incorrect pixels. The proposed FPGA implementation does not require internal LUTs or external memory for large projection data storage. The inferred hardware consumes a small amount of resources and provides real-time output, without significant loss in image quality. The presented estimation algorithm and its implementation outperform the state-of-the-art systems in terms of both system bandwidth and PSNR of the resulting image.

Acknowledgment

The authors gratefully acknowledge the support of Xilinx, Inc., through the Xilinx University Program. This work has been partially funded by the Science and Technology Division of the Swiss Federal Competence Center Armasuisse.

References

[1] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, 3rd ed. Cengage Learning, 2008.

[2] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[3] M. Sapienza, M. Hansard, and R. Horaud, "Real-time 3d reconstruction and fixation with an active binocular head," INRIA Grenoble, Tech. Rep. 7682, July 2011.

[4] Z. Chuan, T. D. Long, Z. Feng, and D. Z. Li, "A planar homography estimation method for camera calibration," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, July 2003.

[5] R. Yang, M. Pollefeys, and S. Li, "Improved Real-Time Stereo on Commodity Graphics Hardware," in *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, June 2004.

[6] D. H. Park, H. S. Ko, J. G. Kim, and J. D. Cho, "Real Time Rectification Using Differentially Encoded Lookup Table," in *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*. New York, NY, USA: ACM, 2011, pp. 47:1–47:4.

[7] C. Vancea and S. Nedevschi, "LUT-based Image Rectification Module Implemented in FPGA," in *IEEE International Conference on Intelligent Computer Communication and Processing*, 2007, pp. 147–154.

[8] A. Akin, I. Baz, L. Gaemperle, A. Schmid, and Y. Leblebici, "Compressed Look-Up-Table Based Real-Time Rectification Hardware," in *IFIP/IEEE 21st Int. Conf. on Very Large Scale Integration (VLSI-SoC)*, Oct 2013, pp. 272–277.

[9] V. Vaish, G. Garg, E. Talvala, E. Antunez, B. Wilburn, M. Horowitz, and M. Levoy, "Synthetic Aperture Focusing using a Shear-Warp Factorization of the Viewing Transform," in *IEEE Conference on Computer Vision and Pattern Recognition - Workshops*, June 2005.

[10] Y. Lu, C. Guo, J. Qiu, P. Liu, and T. Ikenaga, "Low Complexity Homography Matrix Based SIFT for Real-Time 2D Rigid Object Tracking," in *International Conference on Wireless Communications Networking and Mobile Computing*, Sept 2010.

[11] Y. Cheng, "Real-time Surface Slope Estimation by Homography Alignment for Spacecraft Safe Landing," in *IEEE International Conference on Robotics and Automation*, May 2010, pp. 2280–2286.

[12] T. Wada, X. Wu, S. Tokai, and T. Matsuyama, "Homography Based Parallel Volume Intersection: Toward Real-Time Volume Reconstruction Using Active Cameras," in *IEEE International Workshop on Computer Architectures for Machine Perception*, 2000, pp. 331–339.

[13] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, New York, NY, USA, 2011.

[14] V. Popovic, H. Afshari, A. Schmid, and Y. Leblebici, "Real-time Implementation of Gaussian Image Blending in a Spherical Light Field Camera," in *Proceedings of IEEE International Conference on Industrial Technology*, February 2013, pp. 1173–1178.

[15] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, 3rd ed. Berlin, Germany: Springer-Verlag, 2007.