A Full-Featured FPGA-Based Pipelined Architecture for SIFT Extraction

PHILIPP KREOWSKY1 AND BENNO STABERNACK1,2
1 Fraunhofer Institute for Telecommunications—Heinrich Hertz Institute, 10587 Berlin, Germany
2 Embedded Systems Architectures for Signal Processing, University of Potsdam, 14469 Potsdam, Germany
Corresponding author: Benno Stabernack (benno.stabernack@hhi.fraunhofer.de)

ABSTRACT Image feature detection is a key task in computer vision. Scale Invariant Feature Transform (SIFT) is a prevalent and well known algorithm for robust feature detection. However, it is computationally demanding and software implementations are not applicable for real-time performance. In this paper, a versatile and pipelined hardware implementation is proposed, that is capable of computing keypoints and rotation invariant descriptors on-chip. All computations are performed in single precision floating-point format which makes it possible to implement the original algorithm with little alteration. Various rotation resolutions and filter kernel sizes are supported for images of any resolution up to ultra-high definition. For full high definition images, 84 fps can be processed. Ultra high definition images can be processed at 21 fps.

INDEX TERMS Scale-invariant feature transform (SIFT), field-programmable gate array (FPGA), image processing, computer vision, parallel processing architecture, real-time, hardware architecture.

I. INTRODUCTION
Image matching is a typical problem in computer vision. It is applied, amongst other things, in the fields of object recognition, motion tracking and to generate 3D objects from multiple images. All of these rely on robust and efficient feature extraction. Several robust features have been proposed in the past, such as the Scale Invariant Feature Transform (SIFT) [1] in 2004. These features are invariant to translation, illumination, rotation and scale. Its major drawback is the time-consuming computation and large memory usage which imposes the need of acceleration for real-time applications. To cope with that issue, several approaches have been proposed, such as the Speeded Up Robust Features (SURF) algorithm [2]. Nevertheless, SIFT has shown to be the most accurate algorithm [3]. Over the years, many FPGA implementations have been proposed to suite various purposes. All of the implementations neglected parts of the algorithm which were irrelevant for their particular use case such as robotic vision for a mooving vehicle, where the rotation of the SIFT descriptor can be neglected [4]. This results in a degeneration of the accuracy and versatility of the generated descriptors. In order to compute general purpose SIFT features on an FPGA that are suitable for multi-view 3D reconstruction, the rotation of the images must be taken into account [5], [6]. To the best of our knowledge, this is the first published FPGA implementation, which features the full functionality of the original SIFT algorithm.

In this paper, we propose a versatile architecture that is capable of processing images up to UHD (3840 × 2160) resolution. FHD images (1920 × 1080) can be processed at 84 frames per second (fps) and UHD images at 21 fps. All calculations are performed using single precision (SP) floating-point Digital Signal Processing Blocks (DSP-Bs) due to the fact that SIFT achieves the best accuracy using SP format [7]. The descriptors are computed in a rotation invariant manner, where the window of interest is rotated according to the assigned orientation of the particular keypoint.

The structure of this paper is as follows: An overview of other FPGA implementations and their experimental results is given in Section II. The SIFT algorithm is presented briefly in Section III. The FPGA implementation on an Intel Arria 10 FPGA is discussed in Section IV and evaluated in Section V. Finally, a conclusion is drawn in Section VI.

II. RELATED WORK
In this Section, related state-of-the-art FPGA implementations are discussed. There are hybrid implementations such as in [8] and [9], where only the Gaussian scale-space and the SIFT detection is implemented in hardware while the descriptor computation is realized in software on a processor.
Here, the focus is on the implementation of a stand-alone design which computes keypoint locations, the orientation assignment and the corresponding SIFT descriptors from a given image on-chip.

Bonato et al. [10] presented one of the first stand-alone FPGA implementations for the SIFT algorithm. The detector was able to process images of size 320 × 240 at 30 fps. The computation of the descriptor was implemented in a Nios-II soft processor and took 11.7 ms to compute one feature. This is a significant bottleneck in the descriptor computation. Also, only local maxima where taken into account as possible keypoints.

Yao et al. [11] modified the SIFT algorithm and reduced the dimension of the descriptor vector from 128 to 72. Their design takes 31 ms to detect features on a VGA image.

Chiu et al. [12] presented an implementation where they used integral images in order to reduce the complexity of the computation of the scale-space. The trigonometric functions for the descriptor calculation used custom multi-cycle components implemented on an ASIC. The implementation can process FHD images at 30 fps with up to 2000 features per frame. This results in approximately 16 μs per feature. For VGA images, they were able to compute 6000 features per frame at 30 fps.

Zhong et al. [13] presented an FPGA implementation of the SIFT keypoint detection for images of size 320 × 256 which can process images with up to 100 fps. A Digital Signal Processor (DSP)-based computation of the descriptor requires 80 μs per feature. However, only low-resolution images with at most 125 descriptors per image can be processed at the specified framerate.

Vourvoulakis et al. [4] proposed an implementation for robotic vision where they discarded the rotation of the descriptor window and therefore the rotation invariance. They were able to achieve 70 fps at VGA resolution and can compute one descriptor at every clock cycle.

Li et al. [14] presented an FPGA implementation that runs at 50 MHz and can process VGA images at 150 fps. They are able to compute one feature on every clock cycle. It is not described how, or if they cope with the rotation of the descriptor window.

The implementations discussed above can be divided into two groups: The first one in which the calculation of the descriptor is a significant bottleneck, so that the achieved frame rates are only valid for images that have less than the given number of keypoints per image. The second group, namely the implementations of Vourvoulakis et al. [4], Li et al. [14] in which the throughput is independent of the number of features of an image. However, the rotation of the descriptors was not implemented in any of the implementations with high throughput, so that the matching of the resulting descriptors leads to satisfactory results only for small rotations.

III. OVERVIEW OF THE SIFT ALGORITHM

An in-depth description of the SIFT algorithm can be found in [1]. Here, only a brief outline of the algorithm is given. As shown in Figure 1, the SIFT algorithm can be subdivided into four major computation stages to generate a set of features from a given image. These are, the scale-space, the SIFT detection, the orientation assignment and the SIFT descriptor which are described hereinafter. The whole procedure can be repeated for various octaves. That is: For every subsequent octave the input image is subsampled by taking every second pixel in each row and column.

A. SCALE-SPACE

The scale-space \( L(x, y, \sigma) \) is constructed by the convolution of an image \( I(x, y) \) with a 2D Gaussian kernel (Eq. 2) of variable scale \( G_2(x, y, \sigma) \), as in (Eq. 1). For the detection of stable keypoints in the scale-space the Difference of Gaussian (DoG) function (Eq. 3) is evaluated.

\[
L(x, y, \sigma) = G_2(x, y, \sigma) * I(x, y) 
\]

\[
G_2(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} 
\]

\[
D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) 
\]

B. SIFT DETECTION

Keypoint candidates are pixels in the DoG scale-space that are local extrema. Therefore, at least three successive DoGs must be evaluated. This requires at least four scales, which are used here. Two further checks are performed in order to verify the stability: Candidates at \((\hat{x}, \hat{y})\) with low contrast (Eq. 4) along edges (Eq. 5) are discarded. The derivatives...
are estimated by taking differences of neighboring pixels, \( H \) denotes the Hessian matrix.

\[
|D(\hat{x}, \hat{y})| < \text{tsh}
\]
\[
\text{Tr}^2(H(\hat{x}, \hat{y})) = \frac{(D_{xx} + D_{yy})^2}{D_{xx}D_{yy} - D_{xy}^2} \geq \frac{(r + 1)^2}{r}
\]

C. ORIENTATION ASSIGNMENT

To achieve rotation invariance, an orientation is assigned to each keypoint. For each pixel, gradients are calculated and converted from cartesian- to polar coordinates, giving (6) and (7), as shown at the bottom of the page, for the magnitude \( m(x, y) \) and the orientation \( \varphi(x, y) \). In a window around each keypoint at \((\hat{x}, \hat{y})\), the orientation assignment works as follows:

1) The orientations are weighted with their magnitude and a Gaussian weighting function (Eq. 2) whose input is the distance from the keypoint \( g(x - \hat{x}, y - \hat{y}) \), to have a weighted magnitude \( m_H \). The standard deviation is 1.5 times the standard deviation of the closest scale \( \hat{\sigma} \):

\[
m_H(x, y) = m(x, y) * g(x - \hat{x}, y - \hat{y}, 1.5 \cdot \hat{\sigma})
\]

2) An orientation histogram with \( N \) bins covering 360° is formed.

3) The highest peak in the histogram, corresponding to the dominant direction of local gradients, is detected. It is now the assigned orientation \( \hat{\varphi} \).

D. SIFT DESCRIPTOR

In order to compute the descriptor, a 16 × 16 window around the keypoint is rotated according to the assigned orientation \( \hat{\varphi} \). This window is divided into sixteen 4 × 4 subwindows, which is shown in Figure 2b. Within each subwindow, the magnitudes are put into an eight bin histogram according to their orientations. Finally, these 4 × 4 × 8 = 128 values are normalized and ordered to form the resulting descriptor.

IV. PROPOSED ARCHITECTURE

In this Section, an in-depth description of the proposed architecture is given. As the division of the algorithm suggests, it consists of four independent components:

1) scale-space: In order to save resources, the separability of the Gaussian kernel is exploited. The subsequent scales are substracted, yielding the DoGs for further processing.

2) SIFT detection: The SIFT detection, which computes flags at each point of interest (PoI) by performing the aforementioned operations.

3) orientation assignment: The orientation assignment, which computes a histogram of weighted image gradients in an area around each PoI.

4) SIFT descriptor: The SIFT descriptor, which consists of the window rotation and the actual descriptor calculation.

An overview is given in Figure 3, where each individual component is shaded for better recognition.

A. SCALE-SPACE

The 2D Gaussian convolution can be performed as two 1D convolutions as done in [13]. Here, the corresponding kernel (Eq. 2) is a product of two 1D Gaussian functions (Eq. 9) which are orthogonal. Therefore, the convolution is split into a column- and a row-wise convolution in any order.

The dataflow of the separated 2D Gaussian filter is shown in Figure 4 for a grayscale image with an eight bit resolution.

\[
G_1(x, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2/2\sigma^2}
\]

This drastically reduces the number of arithmetic operations required, especially for larger kernels. Only odd kernel sizes \( S \) are supported, requiring \( 2 \cdot \lceil \frac{S}{2} \rceil \) multiplications and \( 2S - 2 \) additions per pixel. Image data is provided as unsigned integers, therefore half of the additions are performed before the conversion to SP format, further reducing the number of required DSP-Bs by \( \lceil \frac{S}{2} \rceil \). Therefore, only \( 2S - \lceil \frac{S}{2} \rceil - 2 \) SP additions are required. Also the column buffer shown in Figure 4 requires less memory because the data is converted.
B. SIFT DETECTION

The three consecutive DoGs are buffered in RAM-based shift registers (SRs) in order to buffer each 3 × 3 neighborhood in a plane buffer (PB). Each pixel is compared to its 26 neighbors and the minimum magnitude (Eq. 4), yielding the local extremum flag. From the central DoG, the horizontal and vertical gradients are computed in parallel using a prewitt mask, shown in Figure 5, as in [4]. The second order mixed partial derivatives are approximated as multiplications of the orthogonal gradients, which are used to check if (Eq. 5) holds.

C. ORIENTATION ASSIGNMENT

The assigned orientation calculator computes an N bin histogram of weighted magnitudes in a quadratic window around a keypoint. The size of the window is restricted to odd sizes for such that the keypoint is located at its center. Here, only windows of size O = {3, 5, 7} were evaluated. A block diagram of the pipelined histogram architecture is given in Figure 6. The input data is a window around a PoI containing the magnitude and angle for each pixel. It is stored in a RAM and provided sequentially. In order to weight each value, the coefficients of a 2D Gaussian kernel are precomputed and stored in an array and each magnitude is multiplied with the corresponding coefficient in the SP multiply accumulate DSP-B (SPMA).

The straightforward way to compute a histogram would be to add each value to the matching bin sequentially. Therefore, each value passes a multiplexer (MUX), is added to the array which then passes a demultiplexer (DEMUX) and is fed back into the adder if subsequent bins are the same. This must be
realized within a single clock cycle, which limits the clock frequency drastically.

To achieve a better performance, the in- and output of the SP addition DSP-B (SPADD) and the histogram array is registered which leads to three different cases for the histogram creation.

- If subsequent bins are the same, enableAccumulate is set and they are accumulated using the SPMA.
- If the next bin but one is the same, bypassBins is set and the result of the SPADD is fed right back as the second summand.
- For every other case, the old value from the histogram array is the other summand and the result of the SPADD is stored in the array.

**D. SIFT DESCRIPTOR**

In order to compute a descriptor, the central 16 \times 16 pixels of a 23 \times 23 window around a PoI are rotated. The rotation is implemented as a MUX, which can resolve a resolution of N steps. It is implemented as a backward mapping to avoid holes in the rotated window. In contrast to forward mapping, backward mapping iterates over the positions of the output image to determine the position in the input image. A visualization thereof is given in Figure 7 for three different rotations. A rotation resolution of \( N = 16 \) results in steps of 22.5°. Due to the fact that at maximum every other pixel can be a keypoint, a multi cycle path constraint can be applied for the rotation to achieve a better timing. The rotated window of size 16 \times 16 is now divided into sixteen subwindows, each one having its own histogram computation as in Figure 6. Additionally the sum of the squared magnitudes, required for the normalization, is computed using an SPMA. For the normalization one division intellectual property core (IP) per subwindow and one square-root IP in total is utilized. One descriptor can be processed in 28 clock cycles.

The number of generated features depends on the image content, though Lowe [1] states that a typical image of size 500 \times 500 pixels will create about 2000 feature points. That is approximately one feature every 125 clock cycles. With PoIs occurring at most on every other pixel and the requirement of the whole window to be within the image, this does not lead to a degradation of the throughput if a FIFO is used to buffer some potential clusters of descriptors.
V. EVALUATION AND VERIFICATION

In this Section, the proposed architecture is evaluated in terms of resource utilization, throughput and accuracy. For better comparability, the resources and performances of the other state-of-the-art implementations are provided as well. The focus is on the accuracy, in particular for matching rotated images with various rotation resolutions \( N \) and window sizes of the orientation assignment.

A. RESOURCE UTILIZATION

The resource utilization, for FHD and UHD images, is presented in Table 2. In each Table, the SIFT descriptor and
FIGURE 10. Matching rates for various rotation resolutions for $N = \{1, 2, 4, 8, 16, 32, 64\}$ and window sizes for the orientation assignment for $O = \{3, 5, 7\}$.
the orientation assignment (for $\Theta = 5$) are lumped together. For the Gaussian filter, only the required memory changes depending on the image resolution. In fact, the number of required LUTs, registers and DSP-Bs of the whole SIFT detection module is constant for all cases. The resources for the descriptor are given for two rotation resolutions of $N = \{8, 16\}$ steps. Different rotation resolutions only affect the number of LUTs and registers within the SIFT descriptor module. More precisely, required LUTs and registers of the orientation assignment and subsequent window rotation module are affected by the rotation resolution. For different image resolutions only the required memory changes. Table 3 shows the resource utilization for the state-of-the-art implementations discussed in Section II.

**TABLE 3.** Overview of the hardware resource usages of state-of-the-art implementations.

| Image resolution | LUTs | Registers | DSPs | RAM [Mbits] |
|------------------|------|-----------|------|-------------|
| Bonato [10]      | 320 x 240 | 43,366    | 19,100 | 64          | 2.35 |
| Zhong [13]       | 320 x 256 | 18,195    | 11,821 | 56          | 2.8  |
| Yao [11]         | 640 x 480 | 35,889    | 19,529 | 87          | 3.2  |
| Chiu [12]        | 640 x 480 | 57,598    | 24,558 | 87          | 1.2  |
| Vourvoulakis [4] | 640 x 480 | 125,644   | 8,372  | 77          | 0.406|
| Li [14]          | 640 x 480 | 65,560    | 39,482 | 642         | 0.32 |
| This             | 1920 x 1080 | 108,298   | 253,588 | 302        | 16.925|

**TABLE 4.** Overview of the performance of state-of-the-art implementations.

| Image resolution | detection time [fps] | descriptor time [μs] | Full Alg supported |
|------------------|----------------------|----------------------|--------------------|
| Bonato [10]      | 320 x 240 | 30 | 11.7 ms | $\cong 3$ | - |
| Zhong [13]       | 320 x 256 | 100 | 80 μs | 125 | - |
| Yao [11]         | 640 x 480 | $\cong 32$ | - | - | - |
| Chiu [12]        | 640 x 480 | 30 | 5.5 μs | 6000 | - |
| Vourvoulakis [4] | 640 x 480 | $\cong 70$ | $\cong 46$ ns | 307200 | - |
| Li [14]          | 640 x 480 | 150 | 20 ns | 333312 | - |
| Chiu [12]        | 640 x 480 | 150 | 20 ns | 333312 | - |
| This             | 1920 x 1080 | 30 | 16 μs | 2000 | - |

**B. THROUGHPUT**

The architecture is integrated in the framework presented in [15]. Here, the image data is provided with sufficient bandwidth via PCIeDMA to the DDR3 on-card memory. Therefore the throughput is limited by the clock frequency of the core. Hosted on an Nallatech 385A accelerator card equipped with an Intel Arria 10 GX 1150 FPGA at 175 Mhz, this results in just over 84 fps and 21 fps for FHD and UHD images, respectively. In Table 4, the performances of the designs discussed in Section II are presented. For better comparison: This corresponds to VGA images that can be processed with approximately 560 fps, which is a significant speed-up.

**C. ACCURACY**

The accuracy is evaluated in terms of the matching ability of the generated features with focus on rotated images. Therefore, a variety of test images including various landscapes and objects where evaluated. In Figure 8 and 9 two matched images are shown with a rotation of 45 and 90 degrees, respectively. Valid matches are connected with a blue line, whereas wrong matches are connected with a red line. Only a random subset of twenty matches are shown. Hence their distribution does not correlate with the achieved matching rate.

The proposed architecture was evaluated using various rotation resolutions from one up to 64 steps. Plots of the achieved matching rates for images rotated by one degree increments are given in Figure 10. The tests were conducted with a low contrast threshold $tsh = 0.1$ (Eq. 4) and edge threshold $r = 10$ (Eq. 5) as in [1]. If the architecture is implemented without rotation ($N = 1$), the generated features provide satisfactory matching rates of well beyond 90 percent for images that are rotated by less than $\pm 20^\circ$. For gradually finer rotation resolutions of two, four and eight steps, matchingrates peak at additional rotations of $\pm 180^\circ$, $\pm 90^\circ$ and $\pm 45^\circ$, respectively.

An interesting observation is that with a minimum of eight steps, the matching rate is above zero percent for arbitrary rotations. The worst matching rates occur at multiples of $45^\circ$ with an offset of $22.5^\circ$ (e.g. $22.5^\circ$, $67.5^\circ$, ...). For a further increased rotation resolution of $N = 16$ steps, the minimum matching rate is about 50% and the effect of different window sizes for the assigned orientation emerges. The orientation assignment windows are quadratic with a width of three, five and seven pixels while the standard deviation is $\sigma = 2$. The matching was performed using the vl_ubcmatch function with the default threshold $\tau = 1.5$ provided by the VLFeat implementation [16] of the algorithm suggested by Lowe [1].

Another metric for the matching ability of the generated features is the recall-precision metric [17]. Matches are labeled positive if their euclidean distance is below the
threshold $\tau$. Among those positive labeled matches, there can be actual positive- as well as actual negative matches resulting in the confusion matrix given in Table 5.

The recall (Eq. 10) and the precision (Eq. 11) are calculated as follows:

$$\text{recall} = \frac{TP}{TP + FN}$$  \hspace{1cm} (10)

$$1 - \text{precision} = \frac{FP}{TP + FP}$$  \hspace{1cm} (11)

The following plots are obtained for increasing thresholds $\tau$. Figure 11 shows the recall-precision plots for nine different rotations between zero and 90 degrees for the implementation with a rotation resolution of $N = 1$, which in this case corresponds to an implementation where the rotation of the descriptor is neglected. In this case the matching of an image rotated by five degrees achieves recall values better than 0.8 for a precision- or false rate less than 0.1 which is comparable to the implementation in [4]. When neglecting the rotation of the descriptors, the matching rate falls to zero for larger rotations, as shown also in Figure 10.

For finer resolutions, such as $N = 8$ which is shown in Figure 12a, the recall rate for images rotated by 90° is above 0.95. The matching of an image rotated by five degrees dropped, resulting in a recall rate just over 0.6 for a false rate less than 0.1. In return, images rotated by 85° and 45° achieve...
a similar performance. This effect extends to rotations that are slightly off 45° with 40° and 50° degrees achieving recall rates just below 0.4 for a false rate less than 0.1. Figures 12a to 12d show that gradually finer rotation resolutions result in an increase in the performance for the worst performing rotations at the expense of the best performing ones. As a simple measure for the overall performance of each rotation resolution, the mean- and minimal matching rates over all 360 degrees, as shown in Figure 10, are evaluated. The results are given in Table 6. The best overall performance is achieved for the rotation resolution of N = 16.

### VI. CONCLUSION

In this paper, a versatile and pipelined architecture for SIFT feature detection is presented. The efficient design of the Gaussian scale-space offers a wide-ranging support for various scales. It is implemented on an Intel Arria 10 FPGA, computing SIFT keypoints and descriptors on-chip while performing computations in SP DSP-Bs with up to 84 fps for FHD images and 21 fps for UHD images. Normalized descriptors can be computed in 161 ns. To the best of our knowledge, this is the first implementation that is able to compute rotation invariant SIFT descriptors on-chip.

### REFERENCES

1. D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

2. H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-up robust features (SURF),” *Comput. Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, 2008.

3. S. A. K. Tareen and Z. Saleem, “A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK,” in *Proc. IEEE Int. Conf. Comput., Math. Eng. Technol.*, Sukkur, Pakistan, Mar. 2018, pp. 1–10.

4. J. Pourvoulakis, J. Kalomiris, and J. Lygouras, “Fully pipelined FPGA-based architecture for real-time SIFT extraction,” *Microprocessors Microsyst.*, vol. 40, pp. 53–73, Feb. 2016.

5. M. Imanullah, E. M. Umiyanto, and S. Sumpeno, “SIFT and ICP in multi-view based point clouds registration for indoor and outdoor scene reconstruction,” in *Proc. Int. Seminar Intell. Technol. Appl. (ISITA)*, Aug. 2019, pp. 288–293.

6. D. C. Blumenthal-Barby and P. Eisert, “Technical section: High-resolution depth for binocular image-based modeling,” *Comput. Graph.*, vol. 39, pp. 89–100, Apr. 2014, doi: 10.1016/j.cag.2013.12.001.

7. T. Ko, Z. M. Charibiwala, S. Ahmadian, M. Rahimi, M. B. Srivastava, S. Soatto, and D. Estrin, “Exploring tradeoffs in accuracy, energy, and latency of scale invariant feature transform in wireless camera networks,” in *Proc. 1st ACM/IEEE Int. Conf. Distrib. Smart Cameras*, Sep. 2007, pp. 313–320.

8. H. D. Chati, F. Muhlbauer, T. Braun, C. Bobda, and K. Berns, “Hardware/software co-design of a key point detector on FPGA,” in *Proc. 15th Annul. IEEE Symp. Field-Program. Custom Comput. Mach.* (FCCM), Apr. 2007, pp. 355–356.

9. L. Chang and J. Hernández-Palancar, “A hardware architecture for sift candidate keypoint detection,” in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, E. Bayro-Corrochano and J.-O. Eklundh, Eds. Berlin, Germany: Springer, 2009, pp. 95–102.

10. V. Bonato, E. Marques, and G. A. Constantinides, “A parallel hardware architecture for scale and rotation invariant feature detection,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 12, pp. 1703–1712, Dec. 2008.

11. L. Yao, H. Feng, Y. Zhu, Z. Jiang, D. Zhao, and W. Feng, “An architecture of optimised SIFT feature detection for an FPGA implementation of an image matcher,” in *Proc. Int. Conf. Field-Program. Technol.*, Dec. 2009, pp. 30–37.

12. L.-C. Chiu, T.-S. Chang, J.-Y. Chen, and N. Y.-C. Chang, “Fast SIFT design for real-time visual feature extraction,” *IEEE Trans. Image Process.*, vol. 22, no. 8, pp. 3158–3167, Aug. 2013.

13. S. Zhong, J. Wang, L. Yan, L. Kang, and Z. Cao, “A real-time embedded architecture for SIFT,” *J. Syst. Archit.*, vol. 59, no. 1, pp. 16–29, Jan. 2013.

14. S.-A. Li, W.-Y. Wang, W.-Z. Pan, C.-C. J. Hsu, and C.-K. Lu, “FPGA-based hardware design for scale-invariant feature transform,” *IEEE Access*, vol. 6, pp. 43850–43864, 2018.

15. F. Steinert, P. Kreowsky, E. L. Wisotzky, C. Unger, and B. Stabernack, “A ‘hardware/software framework for the integration of FPGA-based accelerators into cloud computing infrastructures,” in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, Nov. 2020, pp. 23–28.

16. A. Vedaldi and B. Fulkerson. (2008). **VLFeat**: An Open and Portable Library of Computer Vision Algorithms. [Online]. Available: http://www.vlfeat.org/

17. J. Davis and M. Goadrich, “The relationship between precision-recall and ROC curves,” in *Proc. 23rd Int. Conf. Mach. Learn.* (ICML), 2006, pp. 233–240.

### PHILIPP KREOWSKY

PHILIPP KREOWSKY received the B.Sc. degree in engineering science and the M.Sc. degree in electrical engineering from the Technical University of Berlin, Germany, in 1996 and 2004, respectively. In 1996, he joined Fraunhofer Institute for Telecommunications—Heinrich-Hertz-Institute (HHI), Berlin, Germany, where he is currently a Research Assistant with the Image Processing Department. His current interests include VLSI architectures for computer-vision and machine-learning applications.

### BENNO STABERNACK

BENNO STABERNACK received the Diploma and Dr.-Ing. degrees in electrical engineering from the Technical University of Berlin, Germany, in 1996 and 2004, respectively. In 1996, he joined Fraunhofer Institute for Telecommunications—Heinrich-Hertz-Institute (HHI), Berlin, Germany, where he is the Head of the Embedded Systems Group, Image Processing Department, he is currently responsible for research projects focused on hardware and software architectures for image and video processing algorithms. Since summer 2005, he has been giving lectures on the design of application-specific processors with the Technical University of Berlin. Since October 2016, he holds the Chair of “Embedded Systems Architectures for Signal Processing” with the University of Potsdam as a Joint Appointment with Fraunhofer Institute for Telecommunications—Heinrich-Hertz-Institute (HHI). His current research interests include VLSI architectures for video signal processing, machine learning, application-specific processor architectures for embedded media signal processing, and system-on-chip (SOC) design. He has been engaged in several national, international, and European research projects as well as in the standardization process of ITU.H.266 video coding standards.