Real-Time FPGA-based Detection of Speeded Up Robust Features using Separable Convolution

Petr Čížek Jan Faigl

Abstract—**In this paper, we propose a novel architecture for efficient detection of Speeded Up Robust Features (SURF) for Field-Programmable Gate Array (FPGA). The main benefits of the proposed architecture are in real-time low-latency performance and scalability. The proposed solution provides a significant acceleration of salient points extraction which is fundamental image processing technique for vision-based methods including the simultaneous localization and mapping (SLAM). Based on the presented practical results, the proposed architecture is capable of processing streaming image data at the rate of 140 Megapixels per second which roughly scales from the 640×480@420fps up to 1920×1080@60fps video streams on a low-end, low-cost FPGA solution (Cyclone V). Moreover, the proposed feature detection utilizes only about 20% of logic elements of the FPGA which supports further parallel processing of multiple inputs.**

Index Terms—**Field Programmable Gait Array, Speeded Up Robust Features, Separable convolution, Image processing, Stream Data Processing**

I. Introduction

MACHINE vision becomes a part of industrial applications, and it constitutes a source of broad information which resembles the natural way how humans perceive their environment. Images provide a lot of data, and thus a high-level processing is necessary to extract semantic information specific for a particular domain and task. In industrial applications, Field-Programmable Gate Arrays (FPGAs) play an important role because of high computational capabilities and low power requirements [1], which make them especially suitable for embedded applications [2] like vision based localization of robotic vehicles [3]. Therefore, FPGA architectures are investigated to speed up image processing of large data and provide a real-time response to high frame rates [4], [5].

One of the essential image processing techniques is an extraction of salient points [6] which are the image distinctive patterns that are reliably and repeatedly detectable by a feature extraction method. The existing methods include the Scale Invariant Feature Transform (SIFT) [7], Speeded-up Robust Features (SURF) [8] or Features from Accelerated Segment Test (FAST) [9], etc. From these, the SURF method plays a prominent role in localization [10] and object recognition tasks [11]. Since it is still considered as computationally demanding, its FPGA implementation is worth of investigation.

SURF extraction has been firstly implemented on FPGA in 2009 by the authors of [12] and since that, several improvements have been proposed, e.g., [13]–[16]. In addition to simplifications such as block processing [15], the performance improvements are achieved by new computational architectures that exploit capabilities of FPGAs. There are three fundamental principles for designing computationally efficient image processing architectures: 1) streaming approach [17]; 2) avoiding redundant computations [18]; and 3) avoiding complex memory access patterns, which are typical for image processing with a convolution computation. A streaming architecture without redundant computations results in an efficient processing pipeline with the minimal processing latency [19]. The third principle has been addressed by a separable convolution for the SIFT detection [20] which enables processing 640×480 images at 160 frames per second (fps).

In this work, we consider the principles employed in [19], [20] and propose a novel architecture for a separable convolution of the SURF detection, which significantly reduces the computational requirements. Regarding the existing work, the main contributions of the paper are considered as follows.

- Separable convolution employed in the SURF detection.
- Improved utilization of FPGA by avoiding redundant computations and abundant buffering of image data.
- Image processing pipeline with unified interfaces based on the efficient implementation of the **r-line buffer** with simultaneous access to individual pixels.
- Efficient implementation of the non-maximum suppression that avoids a high utilization of memory resources.
- Experimental validation of the developed architecture that provides processing of 640×480 images at 420 fps and power consumption about 4.76 Watts for processing 1920×1080 images at 60 fps with about 40% utilization of the logic modules of the low-cost Cyclone V FPGA.
- Detailed description of the proposed architecture allowing reimplementation of the whole image processing pipeline using another FPGA systems.

The rest of the paper is organized as follows. Section II overviews the most relevant approaches for FPGA-based SURF extraction. The proposed FPGA-based architecture for the SURF detection is described in Section III. Evaluation results and comparison to existing FPGA-based solutions and CPU-based implementation are reported in Section IV. Concluding remarks are in Section V.

II. Related work

Probably the first FPGA implementation of SURF extraction [8] has been proposed by Šváb *et al.* in [12] which has been followed by several further deployments, e.g., [13], [21]. The main computational improvements are achieved by dedicated units to calculate the Hessian responses for individual

The presented work has been supported by the Czech Science Foundation (GAČR) under research project No. 15-09600Y.

Authors are with Dept. of Computer Science, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Republic `{cizekpe6|faiglj}@fel.cvut.cz`

pixels, but descriptors are computed at the embedded PowerPC CPU of the utilized board with Virtex 5. The reported achieved processing is of 1024×768 at 10 fps with about 0.7 ms for computing a single descriptor. The utilization of the FPGA fabric is relatively high (above 60%). Even though it provides better performance than a pure CPU-based approach, it is far below a GPGPU implementation [22]. It is mainly due to a direct implementation of parallel computations without exploiting streaming capabilities of FPGAs.

Improved stream-based SURF extractor is presented in [14] with the achieved processing speed of the complete features detection and description at 18 fps for 640×480, while only the features detection runs at 232 fps for the same resolution. A simplified version of SURF extractor is proposed in [15] to achieve processing speed of 640×480 images at 324 fps using Virtex 5 at the cost of reduced smaller box-filter, which provides different results than the reference CPU implementation [8]. The method also avoids using integral image in expense of higher FPGA fabric utilization. A recent implementation of SURF detector on a XC6SLX150T (i.e., Spartan 6) running at 66.7 Mhz has been presented in [16]. The reported detection speed is about 50 fps on 640×480 images; however, due to unspecified simplifications, the detector also performs differently than the CPU-based one, which is not further elaborated in the article.

Stream-based FPGA architectures [17] also improve the performance of the SIFT [7], which is computationally more demanding than SURF. Authors of [23] report on SIFT detection combined with BRIEF descriptors fully implemented on FPGA with the achieved processing speed of 1280×720 images at 120 fps (for the feature detection) and 60 fps for computing feature descriptors, which provides the overall throughput 60 fps. In [20], the authors speed up their SIFT-based image classification pipeline by using separable convolution in the SIFT detection phase. They report that the separable convolution allows to detect SIFT features in 640 × 480 images at 160 fps. A descriptor of single keypoint is computed in $25.9\,\mu\mathrm{s}$ and the authors consider only keypoints that could be extracted within $33\,\mathrm{ms}$ time window because their real-time requirements are 30 fps.

It can be observed from the presented overview of the existing FPGA-based SURF/SIFT extractors that computing feature descriptors directly on FPGA decreases the performance significantly, e.g., from 232 fps to 18 fps for the approach [14], from 120 fps to 60 fps in [23], and a decrease from 160 fps to 30 fps is reported in [20]. On the other hand, a combination of SURF detector with BRIEF descriptors [24], which is computationally inexpensive, has been proposed by several authors [25], [26] not only to reduce the memory and computational requirements, but it may also improve the feature matching in certain scenarios [27]. Therefore, we propose to take advantage of System-on-Chip (SoC) solution and compute feature descriptors using CPU with access to the memory shared with the FPGA similarly as in [19].

The high-level structure of the proposed architecture is most similar to the solution presented in [19] except the non-maxima suppression which is calculated directly as the part of the feature detection. Moreover, the proposed image processing pipeline is designed to exploit streaming capabilities of FPGA [17]. The proposed design is directly related to the unified interfaces of particular processing blocks to realize a direct pipeline with the minimal latency [19]. Besides, we follow the principles of avoiding redundant computations [18], which saves logic cells and thus allow their further utilization in parallel processing. However, the most important part in convolution-based feature detectors is an organization of the memory access for computing the response by box-filters. A separable convolution for SIFT has been utilized in [20] and in this paper, we propose a separable convolution for the SURF detection. By combining all these principles and ideas together, we propose a novel SURF detection architecture which provides processing speed of 420 fps for 640×480 images, and thus outperforms all the above-mentioned implementations not only in processing speed but also in resource utilization. The proposed framework is presented in the next section and experimental results together with discussion are reported in Section IV. A brief overview of SURF is provided in the next subsection to make the paper more self-contained.

A. Speeded-up Robust Features (SURF)

The SURF extractor [8] is a simplified version of the SIFT extractor [7] which uses several approximations that allow acceleration of the feature detection process. The algorithm starts with computation of the integral image I_Σ calculated as

$$I_\Sigma(x,y) = \sum_{i=0}^{x}\sum_{j=0}^{y} p(i,j), \qquad (1)$$

where p is the image intensity. The integral image allows to quickly compute any rectangular surface integral using only four values of the integral image, see Fig. 1a.

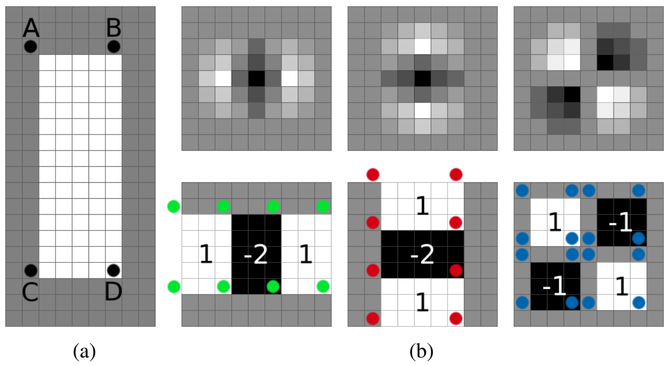

Fig. 1. SURF principle (a) The image integral over the white area Σ is computed as $\Sigma = I_\Sigma(D) - I_\Sigma(B) - I_\Sigma(C) + I_\Sigma(A)$ using the values of the integral image (b) SURF detection based on approximation of the Gaussian kernels [8] by the response of the box-filters. The response is computed using the integral image with the access only to the cells marked by the color disks.

Then, the feature detection is performed by finding a local maxima of image Hessian determinants approximated by

$$H(x,y,\sigma) = \begin{vmatrix} D_{xx}(x,y,\sigma) & D_{xy}(x,y,\sigma) \\ D_{xy}(x,y,\sigma) & D_{yy}(x,y,\sigma) \end{vmatrix}, \qquad (2)$$

where $D_{xx}(x,y,\sigma)$ represents a convolution of the image with second-order derivative of the Gaussian of the variance σ

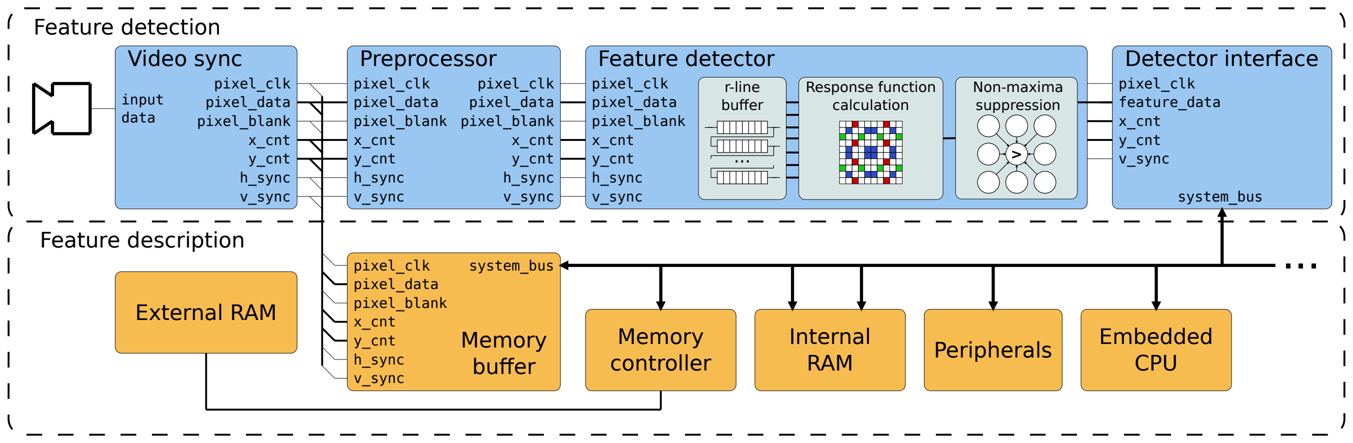

Fig. 2. Proposed FPGA-based feature extraction architecture with the FPGA-based feature detection image processing pipeline (the upper blue row) and computation of feature descriptors on the embedded CPU (the bottom yellow row)

approximated by box-filters. Fig. 1b shows Gaussian kernels together with their box-filter approximations used in SURF [8].

A scale-space is constructed using multiple size filters arranged in octaves and intervals to achieve scale invariance. Typically used filter sizes are listed in Table I.

TABLE I
FILTER SIZES

	Interval 1	Interval 2	Interval 3	Interval 4
Octave 1	9	15	21	27
Octave 2	15	27	39	51
Octave 3	27	51	75	99

Once the Hessian approximation in the scale-space is calculated, local maxima are selected as feature points through the process of non-maxima suppression done within the 26-neighborhood of candidate point in the scale-space.

Next, the feature descriptor is assembled from the square shaped neighborhood around the feature point oriented according to the dominant direction based on responses of Haar wavelet filters centered around the feature point. This neighborhood forms a grid of the size 4×4 regularly sampled by Haar wavelet filters which assigns each square a tuple d_x and d_y responses. Vectors consisting of $\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|$ form the description of each subsquare which are then chained into 64-dimensional SURF descriptor of floating point numbers. Nevertheless, the results of the feature detection can be combined with any other feature descriptor like, e.g., BRIEF [24] which is utilized in this work.

III. Proposed FPGA-based SURF detection

The overall schema of the proposed stream-based FPGA architecture for the SURF detection is shown in Fig. 2. The proposed architecture consists of **Video sync** core, which provides the stream of visual data, **Preprocessor** core that calculates the integral image necessary for the feature detection. In parallel to the feature detection, the data are buffered into the frame buffer in the memory of the processor subsystem because the proposed FPGA architecture concerns only the feature detection and leaves the feature description on CPU.

There are two main reasons for separating the computation of feature descriptors. First, each point in the image has to be checked for the presence of the feature point; however, only a limited number of points are selected for feature description which can be effectively handled by the CPU. Second, a feature description is usually assembled from a larger area around the feature point which requires irregular memory access patterns. Regarding the image processing, the main limitations of nowadays FPGAs are insufficient memory resources. Therefore, it seems more suitable to dedicate calculation of feature descriptors on the CPU, especially for the low-cost System-on-Chip (SoC) solution with shared memory.

A. Stream-based architecture for feature detection

The main idea of the proposed stream-based image feature detection is to decide whether the currently processed image point is a feature or not with each tick of the architecture. The architecture is optimized to process the digital video signal, which comprises a series of successive images where every video frame consists of active pixels and blanking display area. Individual lines are discriminated by the horizontal synchronization `h_sync` pulses and individual frames are separated by the `v_sync` pulses. The image data stream is accompanied by `x_cnt` and `y_cnt` signals determining coordinates of currently processed pixel. All data throughout the feature detection pipeline are synchronized with the `pixel_clk` and are valid on its rising edge. No other clock signals are used.

B. Image window operations

Feature detection belongs to the group of window image processing operations [14]; hence, to identify image features it is necessary to assess each point in the image with a response function value. Access to the rectangle area $r \times c$ around the candidate point is necessary for the calculation of the feature response function. The true parallelism of FPGA is usually exploited to buffer the image data and form a window to access all of them simultaneously. However, such an approach becomes soon intractable as the resource utilization grows quadratically with an increasing size of the image window.

This issue is addressed by the decomposition of the calculation of the SURF box-filter responses into the vertical and horizontal components. Henceforth, the FPGA utilization is only $O(n)$ and not $O(n^2)$ as for an ordinary direct parallel implementation of the full window buffering. The proposed separable convolution described in Section III-D depends on efficient accessing the vertically aligned pixels. The access is realized by the proposed implementation of the **r-line buffer**.

C. r-line buffer

The **r-line buffer** core is utilized for accessing to r vertically aligned pixels which requires $r-1$ lines of the image to be buffered into the internal memory resources of the FPGA, i.e., BRAM. The core comprises of $r-1$ chained dual-port single-clock memories with buffered output with write address given by the `x_cnt` signal provided by the **Video sync** core and read address determined by asynchronously calculated `x_cnt`+2 (see Fig. 3). The aggregated outputs of all BRAMs and the input pixel provide simultaneous access to the r vertically aligned pixels `d_out 0..r-1`. Hence, only the respective number of the individual BRAMs are used for the image width w and bit width of the image data.

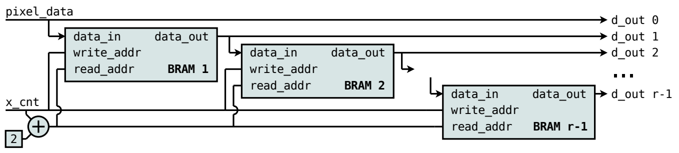

Fig. 3. Architecture of the **r-line buffer** core

The **r-line buffer** is utilized for accessing the integral image values. Prior the buffering, the integral image calculation requires simultaneous access to the given pixel $p(i,j)$, the value of $I_\Sigma(i,j-1)$, and the horizontal partial sum of all pixels of the row i. The implementation of the calculation uses one FIFO memory to buffer the resulting integral image values for the duration of one line to provide access to $I_\Sigma(i,j-1)$. The horizontal buffer is incremented with the value of $p(i,j)$ with each tick and resets on the `h_sync` signal. The resulting integral image value is given as the sum of the given pixel, the horizontal buffer, and the buffered one-line-delayed integral image value.

D. Separable Convolution of box-filters

The main source of the performance improvements in the SURF detection is the proposed implementation of the separable convolution of box-filters implemented with the proposed **r-line buffer** cores. The idea of the proposed architecture is visualized in Fig. 4 and it works as follows.

Let W be the window of the size $r \times c$ necessary for simultaneous access to all the pixels required for calculation of the response function according to the particular filter size (see Table I). Then, the individual box-filter responses are given as

$$\begin{aligned}D_{xx} &= (W_{97} - W_{92} - W_{07} + W_{02}) - 3\cdot(W_{67} - W_{62} - W_{37} + W_{32}),\\ D_{yy} &= (W_{79} - W_{29} - W_{70} + W_{20}) - 3\cdot(W_{76} - W_{26} - W_{73} + W_{23}),\\ D_{xy} &= (W_{44} - W_{41} - W_{14} + W_{11}) + (W_{88} - W_{85} - W_{58} + W_{55})\\ &\quad - (W_{84} - W_{81} - W_{54} + W_{51}) - (W_{48} - W_{45} - W_{18} + W_{15}),\end{aligned}$$

where W_{ij} denotes the integral image value at the offset i,j in the box-filter. As it can be seen in the equations and also

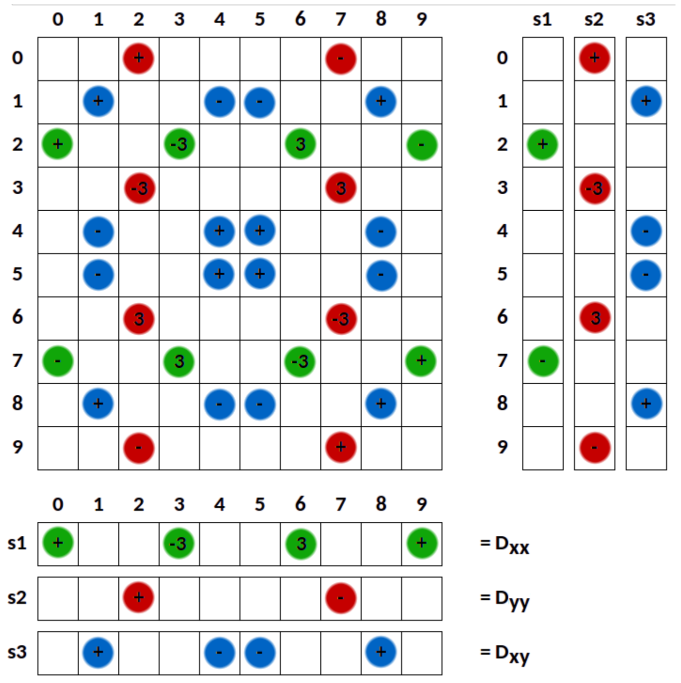

Fig. 4. Proposed assembly of the Hessian determinant calculation. The sampled points for a 9×9 box-filter response calculation are separated into vertical ($s1, s2, s3$) and horizontal (D_{xx}, D_{yy}, D_{xy}) components.

in Fig. 4, it is possible to separate the sampled points into three partial sums

$$\begin{aligned}s1 &= W_{02} - W_{07},\\ s2 &= W_{00} - 3\cdot W_{03} + 3\cdot W_{06} - W_{09},\\ s3 &= W_{01} - W_{04} - W_{05} + W_{08},\end{aligned}$$

and utilize these in the calculation of the box-filter responses:

$$\begin{aligned}D_{xx} &= s1(0) - 3\cdot s1(3) + 3\cdot s1(6) + s1(9),\\ D_{yy} &= s2(2) - s2(7),\\ D_{xy} &= s3(1) - s3(4) - s3(5) + s3(8),\end{aligned}$$

where the numbers in the parentheses (j) denotes the j-th sampling coordinate. Such a scheme can be generalized for a box kernel of arbitrary size.

For the feature score calculation given by (2), it is necessary to get simultaneous access to r vertically aligned integral image data using **r-line buffer** core and then only 3 shift registers of the length c are necessary for buffering the $s1, s2, s3$ partial sums for feature response function calculation of an arbitrarily large SURF window. Thus, the Hessian value of $r \times c$ window is obtained with the latency of $r\cdot w/2 + s_{calc} + c + d_{calc} + H_{calc}$ where r is the height of the window, s_{calc} is the depth of the pipeline computing the 3 partial sums $s1, s2, s3$, c is the width of the window (which stands for the depth of the shift register necessary for buffering the s-values), d_{calc} is the depth of the pipeline used for calculation of the box-filter responses, and H_{calc} is the latency of the Hessian value calculation.

In the proposed implementation, the s-values and box-filter responses are computed in two pipelined steps, and the Hessian response value [8] given as $H = D_{xx}D_{yy} - (D_{xy}^2 - D_{xy}^2/8)$ is calculated in 3 pipelined steps as visualized in Fig. 5. We pipeline each step to allow higher frequencies, and thus

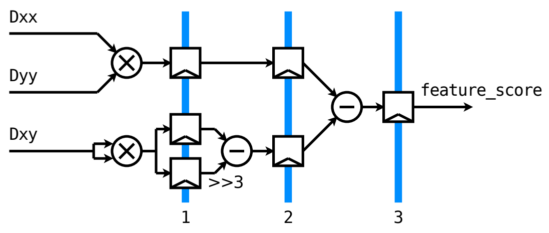

Fig. 5. Architecture of the pipelined Hessian calculation

improve the throughput of the whole architecture. Note, the whole feature detection architecture uses integer numbers with appropriate bit-width of each signal. The bit-width of individual signals for processing of 1920×1080 images does not exceed the range of 32-bit signed integer.

E. Scale-space construction and non-maxima suppression

The major limitation of the stream-based SURF detection on FPGAs is a lack of sufficient internal memory resources for storing integral image and Hessian values. For non-maxima suppression on feature detectors, a typical solution is to buffer feature response values and their comparison in an 8-neighborhood [19]. In SURF detection, the Hessian values are compared to the 26-neighborhood in $3 \times 3 \times 3$ box because of the scale-space [8]. The values are 32-bit integers and further memory resources are necessary to buffer the values and compare them. This is intractable for low-cost FPGAs as the utilization of the integral image itself is resource greedy. Therefore, we propose a novel scheme to immediately select the maxima during the feature detection process, see Fig. 6.

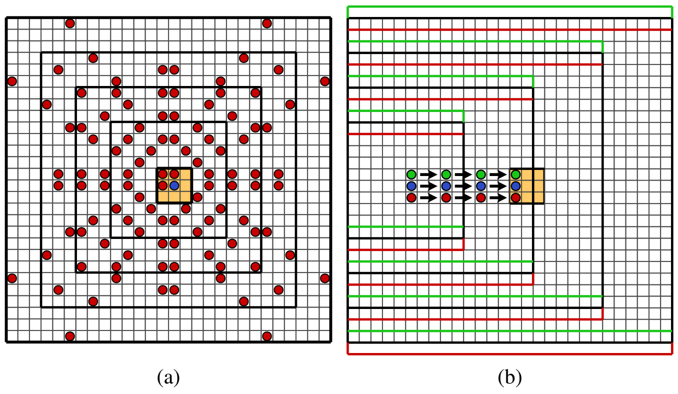

Fig. 6. Principle of the proposed non-maxima suppression: (a) sampling points for the Hessian calculation on all four scales (red) for one particular examined pixel (blue); and (b) separable windows stacking and calculation reuse for non-maxima suppression calculation. The yellow area is the neighborhood for the non-maxima suppression to select if the examined pixel is a feature point.

The proposed separable convolution architecture allows to utilize the retained FPGA resources for calculation of response triplets which are then buffered for the appropriate number of steps to gain access to all the values necessary for calculation the non-maxima suppression simultaneously. Each triplet is formed by the same scale box kernels displaced vertically by one pixel up and down. Therefore, only two additional FIFOs for buffering the integral image in **r-line buffer** are necessary in contrast to 8 FIFOs that would be necessary to buffer the Hessian values on the four scales. If a feature is detected, its x, y coordinates are assigned in the **Detector interface** core. The coordinates are derived from the current `x_cnt` and `y_cnt` by subtracting the induced Δx and Δy pixel latencies which are, in our case, deterministic.

The **Detector interface** core act as a memory mapped slave connected directly to the system bus. Hence, it allows to read out the position of individual features by CPU in either busy wait or it can generate an interrupt signal. The system bus is application specific. In our implementation, the AXI-4 system bus is used with the hard CPU system and Avalon system bus with the soft CPU system, see Section IV.

F. Feature description

We propose to take an advantage of the System-on-Chip (SoC) solution and compute feature descriptors using CPU with the access to the memory shared with the FPGA. A large portion of the image needed for the feature description is buffered in the off-chip memory using the **Memory buffer** core with the direct memory access, and thus the system bus is effectively shared between the CPU and **Memory buffer**.

The feature description is done as soon as the feature is detected and there is a sufficient portion of the image inside the frame buffer. Such a simultaneous processing yields low-latency which is crucial in mobile robotic applications [19]; however, depending on the number of detected features and their distribution in the image, the latency of the feature description is not deterministic. Therefore, it might be necessary to cut the calculation after the specified period as in [20].

Note, in our work, we focus on acceleration of SURF detection whereas we do not implement the SURF description. However, we exploit the combination of SURF detector and BRIEF [24] feature descriptor to provide a complete solution of the feature extraction under real-time constraints. Moreover, in certain scenarios, the combination of SURF detector and BRIEF descriptor outperforms the SURF extraction [27].

IV. Results

The proposed architecture has been primarily deployed on the FPGA development board Terasic DE0-Nano-SoC that comprises of Cyclone V FPGA and ARM Cortex A9 CPU with shared memory and features enough resources for processing 1920×1080 images by the proposed solution. Besides, we also consider even lower-cost FPGA solution with Soft-CPU. The main features of the utilized boards are summarized in Table II. Note, the logic utilization of Cyclone V is referred to the number of utilized Adaptive Logic Modules (ALMs).[1]

The resource utilization of the proposed SURF detection pipeline (i.e., the upper row in Fig. 2) with the settings of 1 Octave and 4 Intervals and also the whole features extraction together with the CPU-based computation of the descriptors is listed in Table III. The numbers were attained after the place and route process. The required resources are only slightly influenced by increasing the resolution as only the memory limit for the used BRAMs need to be doubled from

[1]Each ALM can support up to eight inputs and eight outputs, contains two or four register logic cells and two combinational logic cells, two dedicated full adders, a carry chain, and a register chain.

TABLE II
PLATFORMS USED FOR DEPLOYMENT OF THE PROPOSED ARCHITECTURE

Hard-CPU	**FPGA**:	Cyclone V, Type CSEMA4U23C6N 15880 ALMs; 2460 kBit BRAMs; 84 DSPs *Approximately 40000 Logic Elements (LEs)*
	CPU:	Dual-core ARM Cortex A9 925 MHz, 1 GB DDR3 SDRAM
Soft-CPU	**FPGA**:	Cyclone IV, Type EP4CE22F17C6N 22320 LEs; 594 kBit BRAMs
	CPU:	Soft-CPU, NIOS, 150 MHz, 32 MB SDRAM

TABLE III
FPGA UTILIZATION OF THE PROPOSED ARCHITECTURE

Image resoluion	640 × 480	1920 × 1080
SURF detection only		
ALMs	3 413 (21%)	3 479 (22%)
Total registers	7 783	7 791
BRAM [bits]	985 566 (36%)	1 968 606 (71%)
DSPs	24 (of 84 available)	24 (of 84 available)
Whole feature extraction including description		
ALMs	6 542 (41%)	6 591 (42%)
Total registers	12 732	12 811
BRAM [bits]	1 544 102 (55%)	2 527 142 (91%)
DSPs	24 (of 84 available)	24 (of 84 available)

The utilization of all available resources are percentage points in parentheses.

1024 entries for 640 × 480 to 2048 entries for 1920 × 1080. The rest of the architecture remains completely intact. This also shows the effectiveness of the proposed **r-line buffer** implementation as only the BRAM components are influenced by the resolution.

In the case of the complete feature extraction, additional resources of the FPGA are needed for implementation of the memory controller for transferring data from the input source to the memory. The number of utilized ALMs is almost doubled and about additional 20% BRAM (of available BRAM) is needed; however, the implementation of the memory controller is independent on the resolution. Moreover, computation of feature descriptors completely resides in the CPU, and thus it is independent on FPGA. Therefore, the proposed architecture scales well with the resolution and the only limitation of the current FPGAs is available BRAM.

Processing speed and latency

The maximum frequency of 136.3 MHz has been reported for the proposed SURF detector post-mapping the design into the FPGA by the TimeQuest timing analyzer. This is also the maximum throughput of the proposed architecture, and thus it can process 136 Megapixels per second which roughly scales from 640×480 at 420 fps up to 1920×1080 at 60 fps. The processing latency is 197.4 µs which is also the time to know coordinates of all features in the image after the readout of the last pixel of the camera sensor.

The feature description is done in parallel with the feature detection, and thus whenever a feature is available together with the respective portion of the image in the frame buffer, the feature descriptor is calculated. The utilized 32-Byte long version of the BRIEF descriptor comprises a set of 256 pairwise comparisons and the estimated calculation time per single descriptor is 15.9 µs, which is 6.36 ms for the average 400 features per image.

The proposed architecture is optimized for real-time processing of streamed visual data and the blanking area during the time of vertical synchronization usually provides enough time to calculate descriptors of detected features before the next image is started. In the case that features are not uniformly distributed and/or there are more than average features in the images, the description phase is terminated prematurely. Such a behavior maximizes the throughput through the pipeline which is beneficial, e.g., for rapid camera pose tracking in robotics. However, it is application specific and if such a behavior is not desirable, the image processing can be interrupted or the next frame can be skipped to finish the calculation of the descriptors. Notice, for real-time processing, a suitable trade-off is to design the computational resources for a typical situation rather than for the worst-case, and therefore, the proposed solution is sufficient for 400 features per image.

Performance comparison

The proposed architecture has been compared with other published FPGA-based implementations of the SURF detection and extraction briefly introduced in Section II. In particular, the reported performance indicators on implementations for Virtex 5 [12], Spartan 6 [14], [16], and Virtex 6 [15] have been included in the comparison. The achieved processing speeds and utilization of FPGAs are reported in Table IV. Note, the individual devices differ in the organization of the FPGA fabric and amount of resources. Whereas Xilinx Spartan and Virtex architectures features Logic Elements (LE) with 6-input LUTs, the Altera Cyclone V features Adaptive logic Modules (ALM) with 8 input LUT each. Thus one ALM is approximately equivalent of 1.25 LE. Regarding the results, the proposed SURF detection approach outperforms all other approaches in both the processing speed and FPGA utilization.

Precision of the FPGA-based SURF detection

The correctness of the results provided by the proposed SURF detection pipeline has been verified against the CPU-based OpenCV [28] implementation with same parametrization (1 Octave, 4 Intervals) using a set of digital image patterns of the dataset [29]. The proposed FPGA implementation provides the same results as its CPU-based counterpart which is not surprising as there are no simplifications introduced in the process of feature detection. A demonstration setup using online detection by a camera module is depicted in Fig. 7.

Contrarily, most of other FPGA-based implementations introduce simplifications into the SURF detection which results in different detections in comparison to the original approach [8]. In [12], a fixed point arithmetic is used which slightly lowers the repeatability and distinctiveness. Fewer bits than the required are used for the integral image in [14] to save the memory resources. In [16], unspecified simplifications are made in the detection process and the provided results are different in comparison to the CPU-based approach.

Deployment of the Proposed Architecture with Soft CPU

In addition to the Cyclone V FPGA, we further deployed the proposed architecture on a less powerful platform Terasic

TABLE IV
PERFORMANCE COMPARISON OF SURF DETECTION AND DESCRIPTION

Method	Num. scales	Resolution	Det. [fps]	Det.+Desc. [fps]	Device	Slices LEs (ALMs)	BRAM [kBit]	Freq. [MHz]
Šváb *et al.* (2009) [12]	8	1024×768	10	10	Virtex 5	16 548	2 016	100.0
Pohl *et al.* (2014) [14]	4	640×480	232	18	Spartan 6	15 945	4 680	79.4
Pohl *et al.* (2014) [14]	4	1920×1080	24	—	Spartan 6	12 032	4 284	79.4
Shene *et al.* (2016) [15]	1	640×320	324	$324^{*\flat}$	Virtex 6	39 276	$\leq$540	100.0
Chen *et al.* (2016) [16]	4	640×480	50	—	Spartan 6	37 662	1 872	66.7
Proposed *Hard-CPU*	4	640×480	420	$60^{\ddagger}$	Cyclone V	8 177 (6 542)	1 544	136.3
Proposed *Hard-CPU*	4	1920×1080	60	$60^{\ddagger}$	Cyclone V	8 238 (6 591)	2 527	136.3
Proposed *Soft-CPU*	1	640×480	443	$30^{\flat\ddagger}$	Cyclone IV	5 062	411	142.0
Proposed *Soft-CPU*	1	1920×1080	60	$30^{\flat\ddagger\natural}$	Cyclone IV	5 104	$655^{\natural}$	142.0

*A simplified 3×3 version of the SURF descriptor with fixed orientation
$^{\flat}$Only one scale is used for the SURF detection.
‡BRIEF32 used as a feature descriptor and computed on the CPU of the SoC architecture.
$^{\natural}$Only 24 bits per pixel and one scale is utilized to fit the design into the architecture.

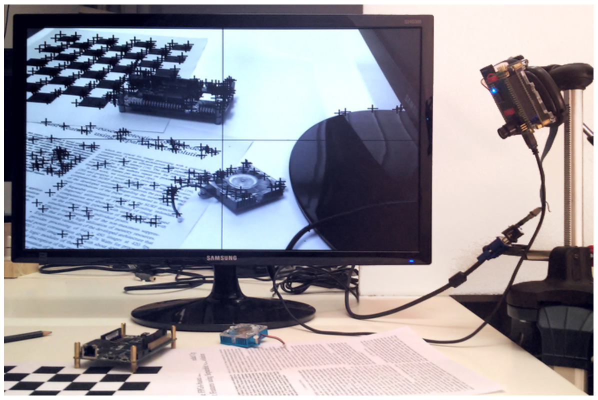
Fig. 7. Online feature detection. On the right there is a camera module with Cyclone IV FPGA and MT9V034 camera. The detected points visualized as black crosses are directly injected into the 640×480 video stream at 60 fps.

DE0-Nano with the Cyclone IV FPGA and without the hard CPU to demonstrate advantages of the proposed architecture. In this case, the FPGA has two times less logic elements and four time less memory resources which are essential for the SURF detection and the CPU is implemented as a software core within the FPGA and it runs at 150 MHz, see Table II.

The limited BRAM resources allow deployment for only one scale as in [15]. The results in Table IV show that the architecture outperforms all the other approaches in the feature detection as the timing analysis reports the maximum frequency of 142 MHz. The soft-CPU is clocked only at 150 MHz, and thus the feature description is much slower than in [15] with FPGA-based computation of the descriptors.

In the case of 1920×1080 images, the limited BRAM resources allow us to buffer only 24 bits per pixel for having the persistent integral image for 12 lines necessary for the calculation of 9×9 window response function value. With this simplification, even the limited Cyclone IV with soft-CPU is capable to provide the detection speed of 60 fps for 1920×1080 resolution using the proposed architecture.

A. Discussion

The main limitation of the current SURF extraction deployment on FPGAs is a lack of sufficient internal memory resources given the integral image values occupy roughly 4 times more memory than the source image. E.g., even for small image of 640×480 the integral image values can reach up to value 78336000 which representation needs 27 bits. This forms a major limitation on low-cost FPGAs as they are capable of storing only a few lines of integral image in BRAM.

On the other hand, the adder tree structure presented in [15], which computes the integral image values on demand, together with the herein proposed separable convolution approach may significantly reduce the number of necessary resources. This is also tightly related to the problem of the descriptors calculation as it can be done either directly in hardware [14], [15] or using CPU as in the proposed solution.

The achieved performance allows to employ the proposed solution in real-time processing of frames from multiple image sensors using the same FPGA fabric with similar power requirements. For the lower resolution, the proposed SURF detection can easily process frames from two or four image sensors that are sequentionaly fed into the **Video sync** core. Moreover, the images can even have different resolution as the whole streamed architecture is controlled only by the `h_sync`, `v_sync`, and `pixel_clk` signals. The only processing limitations is the calculation of descriptors that is made on the relatively slow CPU. However, two cores are available on the utilized DE0-Nano-Soc board, and thus two image sensors with 60 fps or four sensors at 30 fps can be processed by the proposed solution and particular hardware.

Besides, more powerful FPGA fabrics such as the recent Spartan 7 can be easily utilized with the proposed architecture, as only the input signals and the BRAM component in the **r-line buffer** need to be adjusted for a new hardware. The resource friendly architecture allows deployment of several processing pipelines and images from multiple sensors can be processed in parallel at the high frame rate.

V. CONCLUSION

An efficient FPGA architecture for SURF detection is presented in this paper. The main improvements are in the proposed separable convolution combined with the efficient implementation of the **r-line buffer** together with the non-maxima suppression scheme to avoid redundant computations and abundant buffering of image data during computing the responses of box-filters. Moreover, the proposed interfaces of the processing blocks in pipeline support streamed processing with almost zero latency. Overall, the proposed architecture provides real-time performance in the feature detection.

The feature descriptors are computed on the CPU part of the processor-centric SoC design, which provides flexibility to easily use different descriptors for further applications, as an efficient implementation of the proper FPGA architecture is time-consuming. Besides, using computationally cheap BRIEF descriptors on the CPU does not slow down the image extraction significantly. In the case of processing 1920×1080 images, the same processing speed of 60 fps is achieved. The proposed architecture requires only a small portion of FPGA resources, and therefore, it is suitable for processing multiple inputs. On the other hand, it also allows to further deploy efficient architecture for computing SURF descriptors directly on FPGA fabric, which is a subject of our future work.

References

[1] E. Monmasson and M. Cirstea, "Guest Editorial Special Section on Industrial Control Applications of FPGAs," *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp. 1250–1252, 2013.

[2] J. J. Rodriguez-Andina, M. D. Valdes-Pena, and M. J. Moure, "Advanced Features and Industrial Applications of FPGAs: A Review," *IEEE Trans. Ind. Informat.*, vol. 11, no. 4, pp. 853–864, 2015.

[3] J. Rodriguez-Araujo, J. J. Rodriguez-Andina, J. Farina, and Mo-Yuen Chow, "Field-Programmable System-on-Chip for Localization of UGVs in an Indoor iSpace," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1033–1043, 2014.

[4] A. Desjardins, B. Vakoc, M. Suter, Seok-Hyun Yun, G. Tearney, and B. Bouma, "Real-Time FPGA Processing for High-Speed Optical Frequency Domain Imaging," *IEEE Trans. Med. Imag.*, vol. 28, no. 9, pp. 1468–1472, 2009.

[5] S. Dhote, P. Charjan, A. Phansekar, A. Hegde, S. Joshi, and J. Joshi, "Using FPGA-SoC interface for low cost IoT based image processing," in *Int. Conf. Advances in Computing, Communications and Informatics (ICACCI'16)*. IEEE, 2016, pp. 1963–1968.

[6] M. Hassaballah, A. A. Abdelmgeid, and H. A. Alshazly, "Image Features Detection, Description and Matching," in *Image Feature Detectors and Descriptors*. Springer, 2016, vol. 630, pp. 11–45.

[7] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Int. Conf. on Computer Vision (ICCV'99)*, vol. 2, 1999, pp. 1150–1157.

[8] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[9] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Europ. Conf. Comp. Vis. (ECCV'06)*. Springer, 2006, pp. 430–443.

[10] C. Häne, L. Heng, G. H. Lee, F. Fraundorfer, P. Furgale, T. Sattler, and M. Pollefeys, "3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection," *Image and Vision Computing*, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885617301117

[11] T. Kistel, O. Wendlandt, and R. Vandenhouten, "Using distributed feature detection for an assistive work system," in *IEEE Int. Conf. Systems, Man, and Cybernetics (SMC'14)*, 2014, pp. 1801–1802.

[12] J. Šváb, T. Krajník, J. Faigl, and L. Přeučil, "FPGA based Speeded Up Robust Features," in *IEEE Int. Conf. Technologies for Practical Robot Applications (TePRA'09)*, 2009, pp. 35–41.

[13] M. Schaeferling and G. Kiefer, "Flex-SURF: A Flexible Architecture for FPGA-Based Robust Feature Extraction for Optical Tracking Systems," in *Int. Conf. Reconfigurable Computing and FPGAs*, 2010, pp. 458–463.

[14] M. Pohl, M. Schaeferling, and G. Kiefer, "An efficient FPGA-based hardware framework for natural feature extraction and related Computer Vision tasks," in *24th Int. Conf. on Field Programmable Logic and Applications (FPL'14)*, 2014, pp. 1–8.

[15] T. N. Shene, K. Sridharan, and N. Sudha, "Real-Time SURF-Based Video Stabilization System for an FPGA-Driven Mobile Robot," *IEEE Trans. Ind Electron.*, vol. 63, no. 8, pp. 5012–5021, 2016.

[16] W. Chen, S. Ding, Z. Chai, D. He, W. Zhang, G. Zhang, Q. Peng, and W. Luo, "FPGA-Based Parallel Implementation of SURF Algorithm," in *IEEE 22nd Int. Conf. Parallel and Distributed Systems (ICPADS'16)*, 2016, pp. 308–315.

[17] P. Wang and J. McAllister, "Streaming Elements for FPGA Signal and Image Processing Accelerators," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, vol. 24, no. 6, pp. 2262–2274, 2016.

[18] B. Axelrod and M. Laverne, "Reducing FPGA algorithm area by avoiding redundant computation," in *IEEE Int. Conf. Robot. Autom. (ICRA'15)*, 2015, pp. 503–508.

[19] P. Čížek, J. Faigl, and D. Masri, "Low-latency image processing for vision-based navigation systems," in *IEEE Int. Conf. Robot. Autom. (ICRA'16)*, 2016, pp. 781–786.

[20] M. Qasaimeh, A. Sagahyroon, and T. Shanableh, "FPGA-Based Parallel Hardware Architecture for Real-Time Image Classification," *IEEE Trans. on Comp. Imag.*, vol. 1, no. 1, pp. 56–70, 2015.

[21] T. Krajník, J. Šváb, S. Pedre, P. Čížek, and L. Přeučil, "FPGA-based module for SURF extraction," *Machine vision and applications*, vol. 25, no. 3, pp. 787–800, 2014.

[22] W. Yan, X. Shi, X. Yan, and L. Wang, "Computing OpenSURF on OpenCL and General Purpose GPU," *Int. J. Advanced Robotic Systems*, vol. 10, no. 10, p. 375, 2013.

[23] J. Wang, S. Zhong, L. Yan, and Z. Cao, "An Embedded System-on-Chip Architecture for Real-time Visual Detection and Matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 3, pp. 525–538, 2014.

[24] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: binary robust independent elementary features," in *Europ. Conf. Comp. Vis. (ECCV'10)*. Springer, 2010, pp. 778–792.

[25] J. Straub, S. Hilsenbeck, G. Schroth, R. Huitl, A. Moller, and E. Steinbach, "Fast relocalization for visual odometry using binary features," in *Int. Conf. Image Processing*, 2013, pp. 2548–2552.

[26] J. Heinly, E. Dunn, and J.-M. Frahm, "Comparative evaluation of binary features," in *Europ. Conf. Comp. Vis. (ECCV'12)*. Springer, 2012, pp. 759–773.

[27] T. Krajník, P. Cristóforis, K. Kusumam, P. Neubert, and T. Duckett, "Image features for visual teach-and-repeat navigation in changing environments," *Rob. Auton. Syst.*, vol. 88, pp. 127–141, 2017.

[28] G. Bradski and A. Kaebler, *Computer vision with the OpenCV library*. O'Reilly Media, 2008.

[29] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *Int. J. of Comp. Vis.*, vol. 60, no. 1, pp. 63–86, 2004.

Petr Čížek received the Bc. and Ing. degree in robotics and cybernetics from the Czech Technical University in Prague (CTU), Prague, Czech Republic, in 2013 and 2015 respectively. He is currently pursuing his Ph.D. degree in computer science with the Computational Robotics Laboratory, Artificial Intelligence Center, FEE, CTU. His research interests are FPGA architectures, computer vision and mobile robotics.

Jan Faigl (M'13) received the Ph.D. degree in artificial intelligence and biocybernetics and the Ing. degree in cybernetics from Czech Technical University in Prague (CTU), Prague, Czech Republic, in 2010 and 2003, respectively. In 2013 and 2014, he was a Fulbright Visit Scholar with the University of Southern California, Los Angeles, CA, USA. He is currently an Associate Professor of Computer Science with the Faculty of Electrical Engineering (FEE), Czech Technical University in Prague. Since 2013, he has been the Head of the Computational Robotics Laboratory with the Artificial Intelligence Center, FEE, CTU. His current research interests include unsupervised learning, self-organizing systems, navigation of mobile robots, and path and motion planning techniques. Dr. Faigl has been awarded the Antonin Svoboda Award from the Czech Society for Cybernetics and Informatics in 2011.