A Real-Time 720p Feature Extraction Core Based on Semantic Kernels Binarized

Michael Schaffner*†, Pascal Hager*, Lukas Cavigelli*, Pierre Greisen*†, Frank K. Gürkaynak*, Hubert Kaeslin*
*ETH Zurich, 8092 Zurich, Switzerland
†Disney Research Zurich, Switzerland

***Abstract*—Several image processing applications rely on a sparse set of correspondence points between stereo images to discern a sparse but robust depth structure of the scene. There exist several methods to extract and match correspondences, but they are all computationally extensive and require significant memory bandwidths. In this paper, we describe an efficient ASIC core that is able to detect up to 25 k interest points in real time on a 720p video stream using the recently proposed *Semantic Kernels Binarized* (SKB) algorithm. To keep the memory bandwidth low, an optimized method to calculate the filter responses in the interest point detection stage has been devised. Instead of the 2D integral image we use a local 1D integral image combined with an incremental updating scheme to calculate the box filters. The ASIC core is manufactured in 180 nm technology and has a complexity of 254 kGE. It runs at 100 MHz, has a power dissipation of 184 mW and is the central processing block for a larger FPGA based stereo vision system that calculates a sparse depth map by locating corresponding interest points between left and right images in real time.**

I. Introduction

Image features are an important tool in many computer vision applications. They have been intensively studied during the past decade, and a variety of different algorithms and variations thereof have been proposed [1]. SIFT [2] and SURF [3] are two well-known methods which provide high quality feature descriptors with a high degree of invariance. However, the descriptors are costly to compute and consist of many floating point entries that require a lot of memory. This renders them less attractive for embedded devices or hardware implementations, and has spurred the development of more efficient descriptors such as BRIEF [4], BRISK [5], FREAK [6] and *Semantic Kernels Binarized* (SKB) [7]. The key concept these algorithms leverage is the direct computation of a *binary* descriptor. In BRIEF and SKB this is done by means of intensity comparisons or thresholding. The use of binary descriptors has the additional benefit that they occupy much less memory and the matching can be performed very efficiently using e.g. the Hamming distance as the cost function.

In this work we consider the calculation of a sparse disparity map from stereo video, as this is a crucial ingredient for certain video processing methods such as automatic multiview conversion [8]. We are currently developing a system which is able to extract and match descriptors from 720p stereo video at 30 frames per second and with up to 25 k features per frame.

Our system implements the SKB algorithm [7], as it provides competitive results in the restricted setting of stereo vision and is amenable to efficient hardware implementations. In this paper we present a hardware architecture of the core part of a system which performs the calculation of the descriptors. The developed architecture has been implemented and fabricated in 180 nm CMOS technology.

Related work: There exist many FPGA and ASIC implementations of SIFT and SURF, such as [9]–[14]. But to our knowledge there is only one other ASIC implementation [15] of an algorithm that makes use of binary descriptors. Their ASIC is based on a variant of BRIEF and can achieve a throughput of 94.3 full-HD frames per second with 512 extracted feature points per frame.

Summary of contributions: We have developed a hardware architecture for real-time SKB feature extraction from 720p video at 30 fps for stereo vision applications. Instead of using a two-dimensional integral image[1] to compute the filter responses, we use a local, one-dimensional integral image in order to overcome the large memory entries and the associated bandwidth that a two-dimensional integral image entails.

Paper Organization: The system and our version of the SKB algorithm is explained in Section 2, and a performance simulation of the implemented configuration is shown at the end of this section. The hardware architecture is explained in Section 3, the results and conclusions are given in Section 4.

II. System Overview

When searching for sparse point correspondences in a stereo pair using image features, the three main steps that have to be performed are the *interest point detection*, the *descriptor calculation* and *descriptor matching*. The system we are currently developing is shown in Figure 1. It is partitioned into an ASIC and FPGA part, where the ASIC performs the interest point detection and descriptor calculation, which are computationally intensive. The FPGA manages the interfaces, buffers the input images and performs the matching of the descriptors. In the remainder of this paper we concentrate on the parts implemented on the ASIC (blue in Figure 1).

A. Algorithm Details

In the following we summarize our version of the SKB algorithm and point out where it differs from the original [7].

[1]The 2D integral image is defined as $ii(x,y) = \sum_{x' \leq x} \sum_{y' \leq y} i(x',y')$ [16].

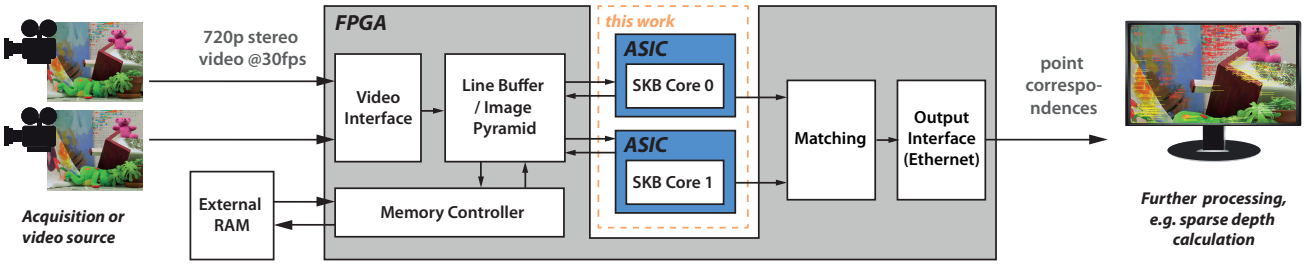

Fig. 1. Overview of the stereo video feature detection system. The FPGA (in gray) provides the IO interfaces, external memory access, and the matching operation. The ASIC (blue) performs the computationally intensive SKB feature extraction (there are two identical ASICs, one for each stereo video channel). The 'Teddy' image shown here is from the dataset provided by [17].

1) Interest Point Detection: Similar to the original SKB implementation, we use a variant of the simple *difference of boxes* (DOB) filter to detect interest points in the image. However, our system builds upon the original DOB version of CenSurE [18] instead of the modified SUSurE DOB [19] that is used in the SKB paper. The original CenSurE detector basically performs a pixel-dense scan on all scales, whereas the SUSure detector uses a scan line sparsification to leave out pixel positions which are not likely to lead to an extremal filter response. The SUSurE detector is about $3\times$ faster than the CenSurE in software [19], but it exhibits a data dependent, irregular flow which inhibits parallel processing of different filter scales. The DOB filter is a simplified *Laplacian of Gaussian* (LoG) filter and its response is given by the subtraction of the pixel sums within two quadratic boxes with side length $2n+1$ and $4n+1$. The advantage of this simplified filter is that it can be efficiently calculated using an integral image [16] as shown in [7]. The image is filtered using different sizes of this filter, thereby forming a volume of filter responses which is also denoted as *scale-space*. As was noted in [3] and [13], the largest number of interest points is found on the few first scales, and in our application we do not require a large scale invariance of the features. Thus the scale-space is limited to the first 8 scales (i.e. $n \in [1, 2, ..., N_{max} = 8]$).

The DOB filter responses in the scale-space are checked for extremal points in a local $3 \times 3 \times 3$ neighborhood using *non-maximum suppression* (NMS), and a weak response threshold is applied in order to filter out non-robust interest points. Maxima can therefore only occur on 6 scales if a total of 8 scales is employed. Note that the integrated box areas must be properly normalized [18] such that a comparison among different scales is possible. The DOB filter responses are already pixel-dense, and thus no further interpolation of the coordinates is performed[2]. The output of the interest point detection is a list of $m \in [0, 1, ..., M]$ containing (x_m, y_m, s_m) tuples, where x_m and y_m are the integer image coordinates and s_m is the index of the scale of a particular interest point m.

2) SKB Descriptor Calculation: The SKB descriptor makes use of a set of sixteen 4×4 filter kernels (also called *semantic kernels*, shown in Figure 4e) which are evaluated at 16 positions within a normalized support region around an interest point. This leads to 256 values which are binarized using a certain thresholding scheme. In [7] they propose three different binarization variants *A*, *B* and *C* where *A* leads to a 256 bit descriptor and *B* and *C* lead to a 512 bit descriptor. Here we use the fast variant *A* where the 256 values are binarized by comparing them against 0, i.e. only the sign bit is kept. In [7] they also define two different support regions (type *A* and *B*) out of which we use the larger 16×16 region (*B*), as our experiments show that it performs slightly better (Figure 2).

The (x_m, y_m, s_m) tuples from the interest point detector are used to calculate the coordinates of the support region of that interest point. Bilinear interpolation is then used to resample the support region such that it fits into the normalized frame of 16×16 pixels (the scale factors are given by the ratio of the outer DOB box size of the actual scale and the smallest scale i.e. $(4 \cdot s_m + 1)/5$). Note that we do not perform any rotational alignment as this is not necessary in the case of stereo matching. In order to facilitate the bilinear resampling, we precompute an image pyramid by successive downsampling of the input image by a factor of two. Depending on the scale factor of an interest point m, the nearest pyramid level is then selected as the pixel source (this concept is also known as *mipmapping* [20]). This has the advantage that aliasing artifacts are reduced in the resampled patches and that the accessed image patch is always contiguous.

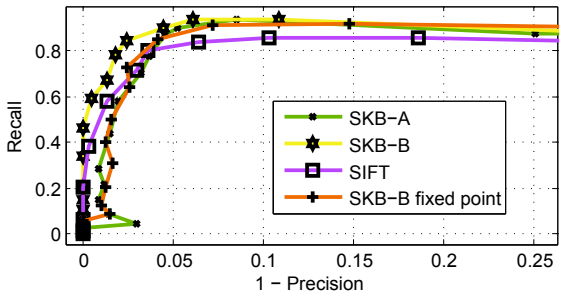

Fig. 2. Matching performance simulation with the stereo test set 'Baby1' from [21]. The *Recall* is the ratio of correct matches and existing correspondences between left and right image, and *1-Precision* is the percentage of false matches [1]. The SKB type *B* performs better than type *A*. The slightly lower performance of our ASIC implementation is due to the fixed point arithmetic. The performance of the SIFT descriptor (evaluated on DOB interest points) is shown as a reference (SIFT code from [1]).

[2]In our implementation, no Harris corner test is performed as this operation is very costly and often not necessary in this application [7].

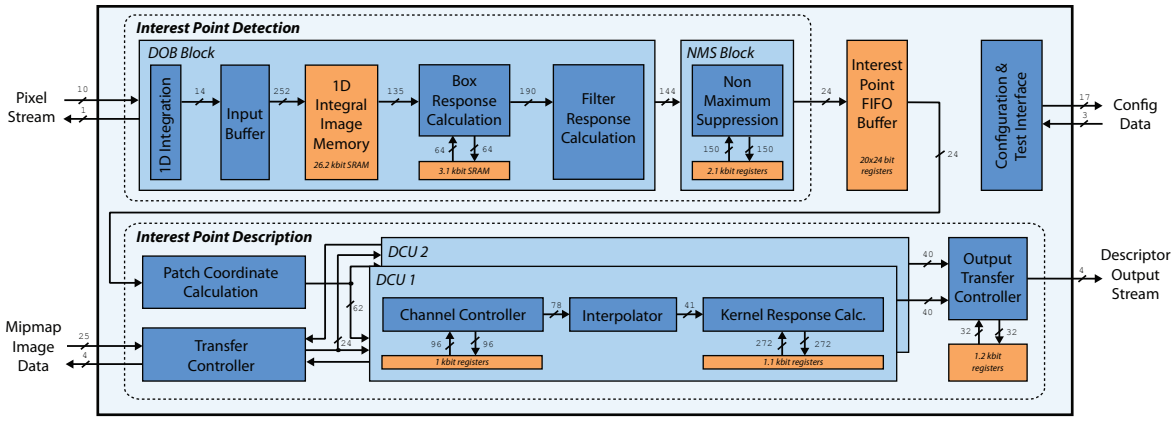

Fig. 3. ASIC Top-Level Diagram with two Descriptor Calculation Units (DCUs): The ASIC is supplied with raw 8 bit gray-scale image pixels. It first searches interest points within the image, whose surroundings are described in a second step. For the description an image patch around the interest point is transferred to the ASIC as well. The final descriptors are sent back to the FPGA for matching.

III. ASIC Architecture

A top-level diagram of the ASIC architecture is shown in Figure 3. It is composed of two main blocks which perform the interest point detection and the descriptor calculation. The interest point detection performs a dense scan over the whole image and is constantly supplied with image data by the FPGA. The detected interest points are temporarily stored in a FIFO, before they are fetched by the descriptor calculation units (DCUs). Note that the interest points are distributed sparsely over the whole image. The DCU has been designed to handle up to 12.5 k descriptors. Depending on the desired descriptor throughput, several instances can be operated in parallel. The FIFO serves to compensate local variations in throughput. Based on the position and scale of a certain interest point, the DCUs request the corresponding image patch from the FPGA. The resulting descriptors, their position, and the scale are then transferred back to the FPGA for further processing. For all throughput calculations we use a clock frequency of 100 MHz which is reasonable for the target technology (180 nm) and allows for convenient interfacing with the FPGA.

A. Interest Point Detection

1) Evaluation of DOB Filters using a 1D Integral Image: A total of eight DOB responses have to be evaluated for each pixel - one for each scale in the scale-space. Each DOB filter response can be decomposed into a linear combination of an inner and an outer box filter response. Those box filter responses are usually calculated with the aid of a 2D integral image, which allows to compute the sum over an arbitrary rectangular area by accessing only four values in the integral image [16]. However, this integral image requires a lot of memory: while a conventional 720p gray scale image composed of 8 bit values requires almost 7.4 Mbit, the corresponding integral image requires large 28 bit entries which results in 25.8 Mbit. Moreover, a large bandwidth from the memory is required as 8 values have to be accessed per DOB filter response. For $N_{max} = 8$ scales and an effective image size of $x_{eff} \times y_{eff} = 1248 \times 688$ pixel this results in a bandwidth of $x_{eff} \times y_{eff} \times N_{max} \times 8 \times 28\,\text{bit} \approx 1.54\,\text{Gbit}$ per frame. The effective image dimensions are given by $x_{eff} = x_{res} - 4 \times N_{max}$ and $y_{eff} = y_{res} - 4 \times N_{max}$, respectively. One option to reduce this bandwidth is to transfer and locally store whole blocks of the integral image in order to leverage the spatial overlap among subsequent filters [12]. In our implementation we use only a one-dimensional local integral image. Our approach builds on the observation in [19]

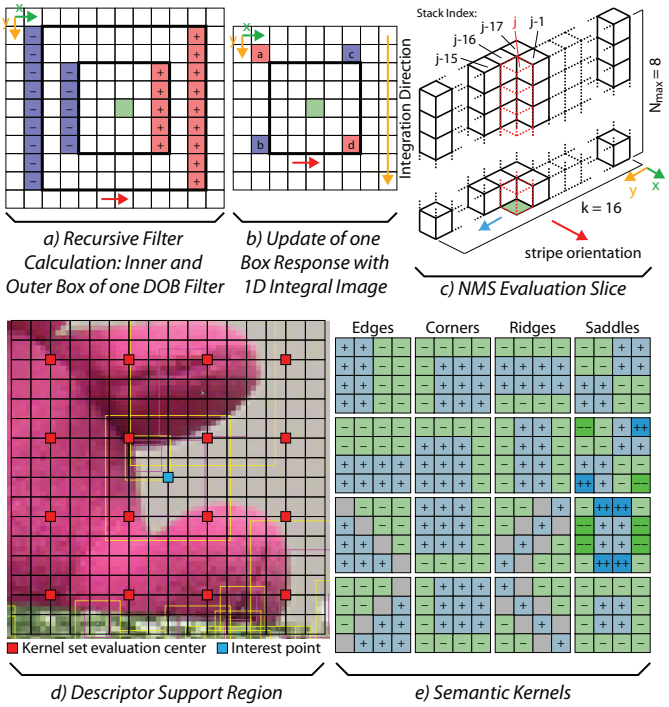

Fig. 4. Algorithmic Details: a) Recursive DOB filter calculation for the green pixel position. b) 1D integral image box response update. c) NMS evaluation: the most recently added stack is colored in red. d) The descriptor support region is overlayed over an image patch showing the centers around which the set of semantic kernels depicted in e) is evaluated.

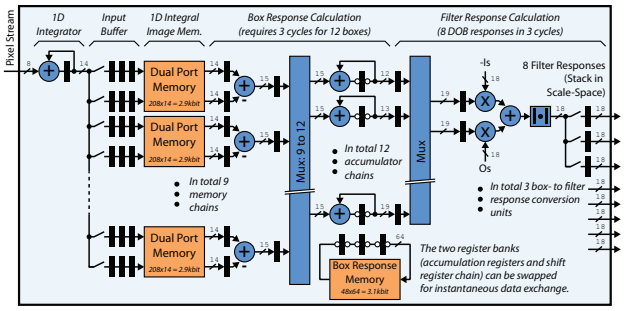

Fig. 5. The DOB filter block receives a dense pixel stream and calculates the DOB responses, which are output as stacks of eight values in scale-space (highlighted in red in Figure 4c). The constants I_n and O_n are the appropriate normalization factors for the inner and outer boxes of scale n.

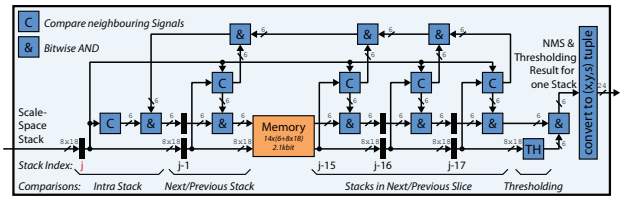

Fig. 6. The NMS block receives a stack of 8 DOB responses per pixel position (red stack in Figure 4c), performs the NMS, and applies the weak response threshold. The 8 DOB responses are stored in 8×18 bit wide registers and the intermediate comparison results in 6 bit wide registers.

where they show how a box filter response can be calculated recursively, provided that a dense scan is performed on the whole image. When scanning from left ro right, a box filter response can be updated by adding the new pixels the box covers on the right side, and subtracting the pixels that are no longer covered by the box on the left side (Figure 4a). However, the number of additions is still linearly dependent on the filter size. In our architecture, we additionally make use of the observation that the pixel groups that have to be added or subtracted are always continuous pixel columns. It is therefore possible to use a *one-dimensional integral* image which enables the calculation of 1D-sums along the columns in constant time (two memory accesses and one subtraction). This allows us to update a box response by accessing only four values (see Figure 4b):

$$B_i = B_{i-1} - (b-a) + (d-c) = B_{i-1} + (a-c) + (d-b). \quad (1)$$

The terms can be reordered such that only differences between two values in the same row need to be added. Note that the one-dimensional integral image can be easily constructed locally as the integration direction is orthogonal to the scanning direction - i.e. if the image is processed in stripes of a certain height h, the integration amounts to the addition of h values, and is completely independent of the *width* of an image. This enables a hardware architecture that only needs to store a sliding window of the original image in an external memory, and the memory bandwidth can be reduced considerably compared to a naive implementation using a two dimensional integral image.

2) DOB Block Details: The image is scanned on all scales in parallel, as otherwise several scanning passes through the image would be required. The FPGA contains the sliding window buffer of the input image and supplies the DOB block on the ASIC with a constant stream of raw image data. The image is processed in overlapping stripes with height $h = 4 \times N_{max} + k$ where $4 \times N_{max} = 32$ is the minimum neighborhood required for 8 scales, and k is the number of effectively calculated box filter responses within one column of the stripe. In order to enable non-maximum suppression in a $3 \times 3 \times 3$ neighborhood, the inner part of evaluated responses of a stripe need to be overlapped by another 2 pixels - i.e. subsequent stripes have a relative offset of $k-2$ rows. A larger value of k reduces the overhead due to the overlap among subsequent stripes, but it also increases the size of the local integral image buffer. In our implementation we use a value of $k = 16$. This results in a total bandwidth of $(4 \times N_{max} + k) \times x_{res} \times \lceil 1 + \frac{y_{eff}-k}{k-2} \rceil \times 8\,\text{bit} \approx 24.1\,\text{Mbit}$ per frame, and the local integral image buffer has to hold at least $(4 \times N_{max} + k) \times (4 \times N_{max} + 1) = 1584$ entries. A detailed block diagram of the DOB unit is shown in Figure 5. The input is supplied in column major format, which enables simple 1D integration along the columns. Note that with $k = 16$, it is sufficient to transfer one pixel to the ASIC per cycle as this translates into a frame rate of roughly 33.2 frames per second at a clock frequency of 100 MHz. The 1D-integration of a column takes 48 cycles in this case, and the downstream circuitry is designed to calculate all $N_{max} \times k = 128$ DOB responses during this time. Some of the inner and outer boxes of the DOB-filter across different scales coincide, such that only 12 out of 16 boxes have to be effectively evaluated in the case of 8 scales. The 1D integral image memory is organized as a ring buffer an can hold 48 rows with a width of 34 pixels. The rows of this memory are segmented into 9 dual port memories in a special pattern to guarantee a collision free, parallel access for the box filter response calculation. Note that the values are accessed in such a way that the difference of two values in one row (see previous section) can be immediately calculated at the memory output, which reduces the multiplexing overhead of the subsequent logic. The previous box filter responses for the recursive calculation are stored in another memory. In one computation cycle, the $k = 16$ sets of 12 box responses are sequentially loaded, updated and stored again. For faster loading and storing, as illustrated in Figure 5, two interchangeable register banks are used: while one bank is in accumulator bank configuration, the other bank is organized as a shift register such that the intermediate values can be shifted to and from the memory. In three cycles all 12 accumulators update their box response according to (1). Finally, the weighted sums among the intermediate box responses are formed to get the DOB response. For this purpose, three units are used which are able to calculate 8 DOB responses in 3 cycles.

3) NMS Block Details: The detailed block diagram of the NMS unit is given in Figure 6. It receives the filter responses from the DOB unit and performs a slice-wise NMS on the local

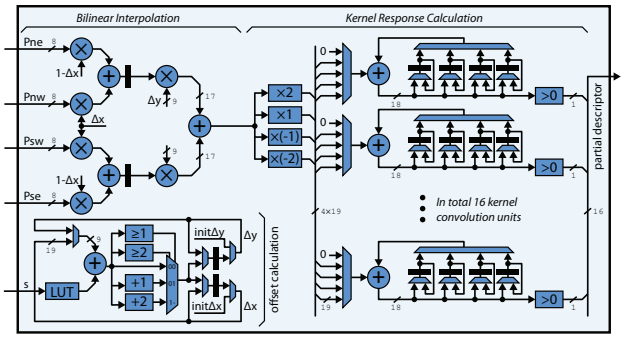

Fig. 7. The descriptor calculation unit (shown without the channel controller) contains a bilinear interpolator and 16 kernel response units. Note that the kernel weights are implemented without multipliers.

scale-space volume with dimensions $k \times 3 \times N_{max}$. Note that since the DOB block outputs one entire scale-space stack in parallel (shown in red in Figure 4c) it is sufficient to only keep track of the last 16 scale-space stacks and the 6 intermediate comparison results within one stack. Further observe that the intermediate results need not be calculated for the scales at the border of the volume.

After suppressing the non-maximum responses, the weak response threshold is applied to the remaining interest point candidates. Finally, the coordinates of the points that pass this test are written to the interest point FIFO.

B. Interest Point Description

In contrast to the interest point detection, the descriptor calculation operates on sparse data. It has to operate fast enough such that - on average - it is able to process all interest points in an image. Evaluations have shown, that several thousand interest points per frame are detected when setting the weak response threshold in a reasonable range. The interest point description block is designed to be scalable and consists of parallel descriptor calculation units (DCUs) which can process up to 12.5 k descriptors per frame each. If the application requires a high throughput, this can be easily achieved by instantiating the appropriate amount of DCUs and adjusting the bandwidth of the image memory accordingly. The current implementation uses two DCUs which results in an aggregated throughput of 25 k descriptors per frame.

The interest point description block takes interest points from the FIFO buffer and assigns each one of them to a DCU, which acquires the required data from the nearest mipmap level from the FPGA through the transfer controller. Then this data is interpolated to a normalized 16×16 image patch which is convolved with several filter kernels and the responses are binarized using a threshold before the result is written to the output buffer.

1) DCU Assignment and Data Transfer: The interest points are read from the FIFO and the patch coordinate calculation block determines additional parameters such as the corresponding mipmap level, the coordinates, the dimensions of the patch to be fetched from that mipmap level, the initial offset, and the step sizes of the bilinear interpolator. The patch coordinate calculation unit then dispatches this information together with the interest point to an available DCU.

The mipmap is selected according to the scale parameter s of the interest points. The employed strategy is to always choose the nearest mipmap (in scale). This reduces the required amount of data by a factor of approximately 2 on average compared to always selecting the lower mipmap without significant changes to the matching performance. Moreover, since the first scale where feature points can occur is $(4 \times 2 + 1)/(4 \times 1 + 1) = 1.8$, it is not necessary to access the lowest mipmap (which is the original image). This allows to reduce the sliding window size for the original image to the minimum of around $4 \times N_{max} + k + k - 2$ on the FPGA side.

Each DCU has a local pixel buffer that temporarily stores the data used by the interpolator. This reduces the required throughput of the interface significantly - in some cases certain pixels are used up to 9 times within a few cycles. Whenever there is enough free space in the buffer, the DCU requests another two lines of the mipmap image patch from the transfer controller. The transfer controller acknowledges them whenever it has the capacity, and passes the request on to the FPGA while still receiving the response to an earlier request.

2) Descriptor Calculation: A detailed block diagram of the datapath of one DCU is shown in Figure 7. First, the acquired patch from the nearest mipmap is resampled with a scaling factor in $[0.75, 1.5)$ using bilinear interpolation to complete the normalization of the support region. In a second step, the resulting row-wise stream of single pixels of the resulting 16×16 support region (Figure 4d) is convolved with the 16 semantic filter kernels shown in Figure 4e. One DCU is able to process one interpolated pixel per cycle, which involves the evaluation of 16 kernel updates. Since the normalized image patch is processed in scanline order, it is necessary to keep track of four sets of 16 temporary kernel responses. Whenever the convolution of a set of filters is completed, it is binarized using a threshold, and the resulting part of the descriptor is written to the output buffer.

3) Throughput: The raw descriptor calculation throughput of one DCU is one descriptor in 256 cycles which translates into around 12.5 k descriptors per frame (slightly less than 13 k due to control overhead). However this assumes that image data is always present for the bilinear interpolation. This is not always the case since the size of the image patches that are requested from the FPGA vary up to a factor of 2.44 in size (the smallest image patch size is 16^2 and the largest 25^2 pixels). The effective throughput may thus be dependent on the speed of this interface. In our implementation, the interface can deliver 24×100 Mbit/s of pixel data, which depending on the patch size corresponds to between 15 k and 38 k image patches in the worst and best case including control overhead. This rate is sufficient to supply one DCU continuously. However, to cope with feature point clusters, it is important to have a higher throughput than what the average case suggests. This is necessary to keep the interest point FIFO size within reasonable limits. If the FIFO is too small, some of the interest

TABLE I
MEASUREMENT RESULTS OF THE SANDSTORM CHIP.

Physical Characteristics	
Technology	UMC 180 nm, 6 Metal Layers
Core Voltage	1.8 V
Package	CQFP 120
# pads	82 (I:40, O: 26, PWR: 16)
Core Area	3.08 mm^2
Circuit Complexity (with SRAM)	254 kGE (2.4 mm^2)
Logic (std. cells)	193 kGE (1.8 mm^2)
On-chip SRAM	29 kbit
Maximum Clock Frequency	100 MHz
Power Dissipation @ 100 MHz,1.8 V	146 mW (core) + 38 mW (pads)
Performance	
Throughput (with 720p frames)	30 fps
Max. Desc./Frame	15 k-25 k

points can be dropped when the density of interest points gets too large. In our design we use two DCUs which together are able to process at least 15 k descriptors in the IO-limited case, and up to 25 k in the computationally limited case.

C. Output Interface

The output interface transmits each interest point together with its descriptor, i.e., the 24 bit triplet (x, y, s) and the 256 bit descriptor as a data packet over a 4 bit wide bus requiring 71 cycles. A DCU requires at least 256 cycles to calculate a descriptor. The output interface guarantees that all results can be transmitted within these 256 cycles. Each DCU contains an output buffer that can store two finished descriptors in order to bridge time periods where the output transfer controller is busy transmitting a descriptor from a different DCU.

IV. RESULTS AND CONCLUSIONS

The ASIC has been named SANDSTORM and was fabricated in 180 nm CMOS technology. Table I shows the key figures. At 100 MHz, it is able to process 720p video at 30 fps with 15 k-25 k descriptors per frame (depending on the distribution of the descriptor scales). The same design has also been synthesized for a Stratix IV EP4S100GSF45I FPGA. On this FPGA, it runs with up to 102 MHz and consumes 15'715 logic cells (4%), 15'242 registers (4%), 27 k memory bits (<1%) and 34 18 bit DSP elements (3%). We showed that it is possible to implement the SKB feature matching algorithm efficiently in hardware even for demanding video applications.

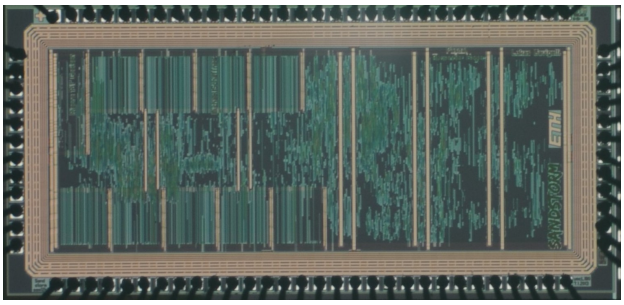

Fig. 8. Microphotograph of the SANDSTORM chip.

REFERENCES

[1] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE transactions on pattern analysis and machine intelligence*, pp. 1615–1630, 2005.

[2] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer Vision ECCV 2006*, pp. 404–417, 2006.

[4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision ECCV 2010*, ser. Lecture Notes in Computer Science, 2010, vol. 6314, pp. 778–792.

[5] S. Leutenegger, M. Chli, and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011, pp. 2548–2555.

[6] A. Alahi, R. Ortiz, and P. Vandergheynst, "Freak: Fast retina keypoint," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2012, pp. 510–517.

[7] F. Zilly, C. Riechert, P. Eisert, and P. Kauff, "Semantic kernels binarized - a feature descriptor for fast and robust matching," in *Visual Media Production (CVMP), 2011 Conference for*, nov. 2011, pp. 39 –48.

[8] M. Farre, O. Wang, M. Lang, N. Stefanoski, A. Hornung, and A. Smolic, "Automatic content creation for multiview autostereoscopic displays using image domain warping," in *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, 2011, pp. 1–6.

[9] V. Bonato, E. Marques, and G. Constantinides, "A parallel hardware architecture for scale and rotation invariant feature detection," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 12, pp. 1703–1712, 2008.

[10] J. Svab, T. Krajnik, J. Faigl, and L. Preucil, "FPGA based Speeded Up Robust Features," in *Technologies for Practical Robot Applications (TePRA), 2009. IEEE International Conference on*, 2009, pp. 35–41.

[11] D. Bouris, A. Nikitakis, and J. Walters, "Fast and efficient fpga-based feature detection employing the surf algorithm," in *Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on*, 2010, pp. 3–10.

[12] M. Schaeferling and G. Kiefer, "Object recognition on a chip: A complete surf-based system on a single fpga," in *Reconfigurable Computing and FPGAs, 2011 International Conference on*, 2011, pp. 49–54.

[13] T. Sledevic and A. Serackis, "Surf algorithm implementation on fpga," in *Baltic Electronics Conference (BEC), 13th Biennial*, 2012, pp. 291–294.

[14] D. Jeon, Y. Kim, I. Lee, Z. Zhang, D. Blaauw, and D. Sylvester, "A 470mv 2.7mw feature extraction-accelerator for micro-autonomous vehicle navigation in 28nm cmos," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2013 IEEE International*, 2013, pp. 166–167.

[15] J.-S. Park, H.-E. Kim, and L.-S. Kim, "A 182 mw 94.3 f/s in full hd pattern-matching based image recognition accelerator for an embedded vision system in 0.13-cmos technology," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 23, no. 5, pp. 832–845, 2013.

[16] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition (CVPR). 2001 IEEE Conference on*, vol. 1, 2001, pp. I–511–I–518 vol.1.

[17] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Conference on*, vol. 1, 2003, pp. I–195.

[18] M. Agrawal, K. Konolige, and M. Blas, "Censure: Center surround extremas for realtime feature detection and matching," in *Computer Vision ECCV 2008*, ser. Lecture Notes in Computer Science, 2008, vol. 5305, pp. 102–115.

[19] M. Ebrahimi and W. Mayol-Cuevas, "Susure: Speeded up surround extrema feature detector and descriptor for realtime applications," in *"Workshop on Feature Detectors and Descriptors: The State Of The Art and Beyond" as part of IEEE Conference CVPR 2009*, June 2009.

[20] T. Akenine-Möller, E. Haines, and N. Hoffman, *Real-time rendering*. AK Peters, 2008.

[21] H. Hirschmüller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Computer Vision and Pattern Recognition (CVPR), 2007 IEEE Conference on*, 2007, pp. 1–8.