Real-Time Traffic Sign Detection Using SURF Features on FPGA

Jin Zhao, Sichao Zhu, and Xinming Huang
Department of Electrical and Computer Engineering
Worcester Polytechnic Institute

***Abstract*—Drivers' failure to observe traffic signs, especially the stop signs, has led to many serious traffic accidents. Video-based traffic sign detection is an important component of driver-assistance systems. In earlier systems, simple color and shape-based detection methods have been broadly applied. Recently, feature-based traffic sign detection algorithms are proposed to obtain more accurate results, especially when combined with the previous two. The Speeded Up Robust Features (SURF) algorithm is an outstanding feature detector and descriptor with rotation and illumination invariance. Unfortunately, due to its computational complexity, the application of SURF algorithm remains limited in real-time systems. In this paper, we present a real-time SURF-based traffic sign detection system by exploiting parallelism and rich resources in FPGAs. The proposed hardware design is able to accurately process video streams of 800×600 resolution at 60 frame per second.**

***Index Terms*—Driver-assistance system, traffic sign detection, SURF, FPGA**

I. Introduction

Video-based driver-assistance system is becoming an indispensable part of smart vehicles. It monitors and interprets the surrounding traffic situation, which greatly improves driving safety. Related computer vision algorithms have been studied extensively and producing inspiring results in this area. A variety of methods have been applied to driver-assistance applications such as obstacle detection [1], lane detection [2], parking assistance [3], etc. In addition, researchers have been paying the same, if not more, attention to the realization of real-time algorithms in embedded driver-assistance systems with limited resources and power [4][5].

In driver-assistance systems, traffic sign detection is among the most important and helpful functionalities. Many traffic accidents happen due to drivers' failure to observe the road traffic signs, such as stop signs, do-not-enter signs, speed limit signs, and etc. Three kinds of methods are often applied to perform road traffic sign detection – color-based, shape-and-partition-based, and feature-based algorithms [6]. Traffic signs are designed with unnatural color and shape, making them conspicuous and well-marked. Color-based and shape-based method are the initial and straightforward ones to be used for sign detection [7][8]. However, these methods are sensitive to the environment. Illumination change and partial occlusion of traffic signs seriously degrade the effectiveness of these methods. The third type of methods is based on feature extraction and description. These algorithms detect and describe salient blobs as features. The features usually stay unaffected under illumination, position and partial occlusion variance. Reported feature-based algorithms contain Scale Invariant Feature Transformation (SIFT) [9], Histogram of Oriented Gradient (HoG) [10], Haar-like feature algorithms [11], etc.

Speeded Up Robust Features (SURF) [12] is one of the best feature-based algorithms and has been widely used in computer vision applications [13][14][15]. It extracts Hessian matrix-based interest points and generates a distribution-based descriptor, and is a scale- and rotation-invariant algorithm. These features make it perfect for object matching and detection. A matching result between standard stop sign and natural image is shown in Fig. 1. The key advantage of SURF is to use integral image for feature detection and description, which greatly boosts the process efficiency. Even so, like other feature-based algorithms, SURF is computationally expensive and often results in very low frame rate. In order to employ SURF for real-time applications in portable driver-assistance systems, parallel processing architectures and platforms need to be considered. By analyzing the parallelism during each step of the SURF algorithm, we choose Field Programmable Gate Array (FPGA) as the embedded platform because of its rich resources for computations and convenient access to local memories. Finally, we implement a SURF-based real-time road traffic sign detection system on a Xilinx Kintex-7 FPGA with SVGA video input and output.

Figure 1. SURF matching example

This paper is organized as follows. Section 2 presents SURF algorithm and an overview of its main steps and functionalities. Section 3 introduces the hardware architecture to implement the SURF algorithm on FPGA. In Section 4, we present the experimental platform and results. Section 5 concludes our work.

978-1-4799-1365-7/13/$31.00 ©2013 IEEE

II. SURF ALGORITHM

This section overviews the SURF algorithm [12]. The algorithm consists of three main steps: integral image generation, interest point localization, and interest point description.

$$I_{\sum}(P) = \sum_{i=0}^{x}\sum_{j=0}^{y} I(x,y) \tag{1}$$

Integral image is one of the main advantages of SURF. Integral image $I_{\sum}(P)$ in $P = (x, y)$ represents the sum of all the pixels on the left and top of P as in (1). Integral image is used in both the subsequent interest point detection and description to obtain higher efficiency. Once integral image is computed, it takes only 3 additions/subtractions to get the sum of the pixels intensities over an upright rectangular region (see Fig. 2, $\sum = I_{\sum}(D) - I_{\sum}(C) - I_{\sum}(B) + I_{\sum}(A)$). Another benefit is that the calculation time is independent of the box size.

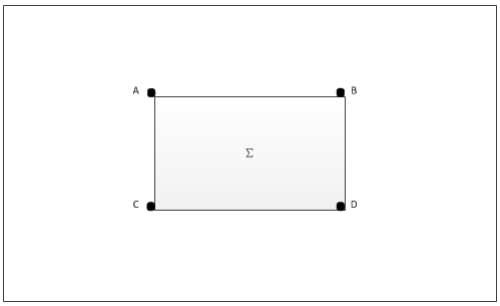

Figure 2. Functionality of integral image

SURF detector locates features based on the Hessian matrix. The original definition of Hessian matrix is shown in (2),

$$\mathcal{H}(P,s) = \begin{bmatrix} L_{xx}(P,s) & L_{xy}(P,s) \\ L_{xy}(P,s) & L_{yy}(P,s) \end{bmatrix} \tag{2}$$

where $L_{xx}(P,s)$ denotes the convolution of Gaussian second-order derivative in $x-$direction with input image in point P at scale s, and similarly for $L_{xy}(P,s)$ and $L_{yy}(P,s)$. Simple box filters using the integral image are used to approximate the second-order Gaussian partial derivation, and yielding less computation burden (see Figure 3). The problem thus reduces from calculating Gaussian second-order derivative responses to the box filter responses. Denoting the blob responses by D_{xx}, D_{yy} and D_{xy}, then the determinant of the original Hessian matrix in SURF is approximated as follows:

$$\det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \tag{3}$$

where 0.9 is used to balance the Hessian determinant. In order to achieve scale invariance, SURF applies box filters of different sizes on the original image to search and compare interest points. Box filters of different sizes construct the scale space, which is divided into octaves. Table 1 gives box filter edge sizes of the first two octaves and corresponding Gaussian kernels scales. The local maxima of box filter responses larger than a predefined threshold in image and scale space are selected as interest point candidates. A non-maximum suppression in a $3 \times 3 \times 3$ neighborhood is applied to screen out "false" candidates with position correction elements above 0.5 and localize interest points.

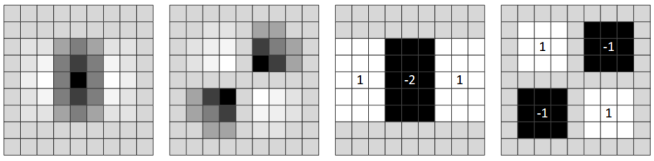

Figure 3. The left half shows Gaussian second order partial derivative in x- and xy- direction; the approximation for them - box filters, are presented in the right half, respectively. The grey regions are equal to 0.

Table I
BOX FILTER EDGE SIZES OF THE FIRST TWO OCTAVES AND CORRESPONDING GAUSSIAN KERNELS SCALES

Octave	1				2			
Box filter size l	9	15	21	27	15	27	39	51
Gaussian kernel scale $s = 1.2\frac{l}{9}$	1.2	2.0	2.8	3.6	2.0	3.6	5.2	6.8

SURF builds an descriptor around the neighborhood of each interest point. First, a square region of 20s-by-20s centered on the interest pointis constructed along the dominant direction. In order to keep it simple, the dominant directions of interest points are set to be upright. The region is then divided into 4×4 smaller sub-regions with each window size 5s-by-5s (sampling step s). For each of these sub-regions, Haar wavelet responses (filter size 2s) are computed at 5×5 regularly distributed sample points. These responses are then weighted by a Gaussian function ($\sigma = 3.3s$) centred at the interest point. We use d_x and d_y to denote weighted Haar wavelet response in horizontal direction and vertical direction. Each sub-region generates a 4-D vector $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. All sub-regions are then concatenated into a vector, resulting in a 64-dimensional descriptor vector.

III. FPGA IMPLEMENTATION OF SURF

The SURF algorithm introduced in Section 2 is considered as a high-speed, robust and accurate method to extract and describe interest points of an image or a video frame. Descriptors of these key points could be used as pivotal features of images and videos, and applied in many applications, such as traffic sign detection, machine stereo vision, object tracking, etc. However, the complexity of the algorithm itself leads to frequent memory access and long computing latency, which make the SURF algorithm difficult to be used in real-time systems. On a dual-core 2.5 GHz processor, simulations of the SURF Matlab code configured with interest point detection at the first two octaves takes 1.38 seconds to detect and describe all 284 feature points from an image of 800×600 resolution. More recently, an FPGA SoC architecture was proposed and it achieved about 10 fps processing rate on a Xilinx XC5VFX70 FPGA incorporating PowerPC-440 CPU

with floating-point processing units [16]. It was an approach of hardware/software co-design with integral image generator and fast-Hessian generator implemented by dedicated hardware, and the rest of the algorithm was implemented using software of PowerPC. But the frame rate is still unsatisfactory for real-time systems such as traffic sign detection.

This section presents an FPGA-only architecture to implement the real-time SURF algorithm toward videos of 800×600 resolution at 60 fps. We will first present the overall system architecture using data flow and state the limitation and approximation of the FPGA implementation. Then each function module is described in detail.

A. Overall system architecture

We make every effort to follow the original SURF algorithm as closely as possible. However, we still have to apply several approximations for the hardware design. Firstly, traffic sign usually occupies only a small part of a frame, which makes most of interest points detected at the first two octaves with small filter sizes. Based on this, the hardware architecture computes the fast-Hessian determinants of image and detects the interest points among them only at the first two octaves. The current design is oriented towards videos of 800×600 resolution at 60 fps and it could be easily adapted to other sizes. Secondly, as mentioned in Section 2, the dominant direction of each interest point is set upright to simplify calculation. Thirdly, we implement a modified descriptor using larger, overlapping sub-regions to ensure better matching performance. Finally, in order to make SURF suitable for FPGA-only architecture, floating-point to fixed-point data conversion is carefully conducted with small loss of precision. The proposed FPGA-based SURF is implemented entirely in fixed-point domain.

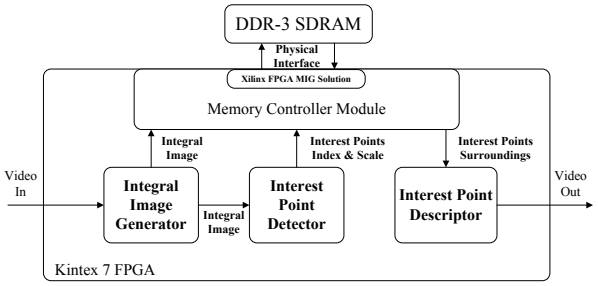

Figure 4. Overall system block diagram

The overall system diagram is summarized in Fig. 4. The FPGA-based SURF architecture is composed of 4 sub-modules - Integral Image Generator, Interest Point Detector, Interest Point Descriptor and Memory Controller Module. The input is RGB format video with 40 MHz pixel rate. Integral Image Generator converts the original data to gray integral image and transfers it to Interest Point Detector and Memory Controller Module. Interest Point Detector is aimed to find all the local Fast-Hessian determinant maxima as interest points and to obtain their index and scales. Memory Controller Module is designed for writing and reading data through the SDRAM interface. It stores the integral image and provides the integral image of interest points surroundings to the Interest Point Descriptor, which gets interest points descriptors from their surroundings. In addition, a comparator module (not shown because it is independent of SURF) matches the interest points descriptors of the current frame with those of the traffic sign images in the library to determine if a traffic sign is detected. In the video output, the traffic sign is highlighted before sending to the display. The details of each module are explained below.

B. Integral image generation

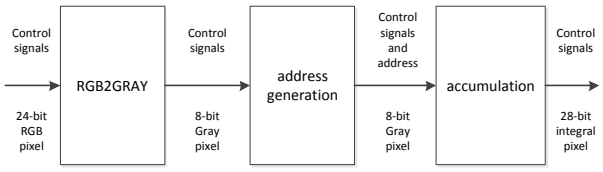

Figure 5. Integral image generation block diagram

Integral image generation module is the entry point of the entire architecture. Its block diagram is shown in Fig. 5. It consists of 3 lower-level modules. The input video data is in RGB format while all the SURF calculation is based on gray scale. The RGB2GRAY unit is responsible for this transformation. After that, row and column information is produced according to control signals (vsync, hsync and de) in Address Generation module. Finally, integral image is calculated in the Accumulation module. Suppose the current video position is D, its integral image can be quickly obtained from $I_D = I_A + i_D - I_B - I_C$, where I denotes integral image and i the gray value. A,B,C are the upper-left, upper and left pixel of D, respectively. To reduce register uses, the integral image of upper line is stored in Block RAM.

It is also noted that the video of integral image has the same format with the original one. The only difference is that the 24-bit RGB information is replaced by a 28-bit gray integral value.

C. Interest points detector

This module is designed to calculate fast-Hessian determinants in all sample points at multi-scale and then find local maxima at x, y and scale space as interest point candidates. A non-maximum suppression is applied to these candidates to screen out "false" interest points. Interest points detector is divided into 3 blocks: Build_Hessian_Response, Is_Extreme and Interpolate_Extremum. They are explained as follows.

1) Build_Hessian_Response: Build_Hessian_Response block diagram is shown in Fig. 6. The largest filter size among the first 2 octaves is 51. According to SURF algorithm, integral image located in the rectangle of relative location between (-26,-26) and (25,25) around sampling points are needed to calculate responses of all box filter size of the first 2 octaves. In our design, the initial sampling step is

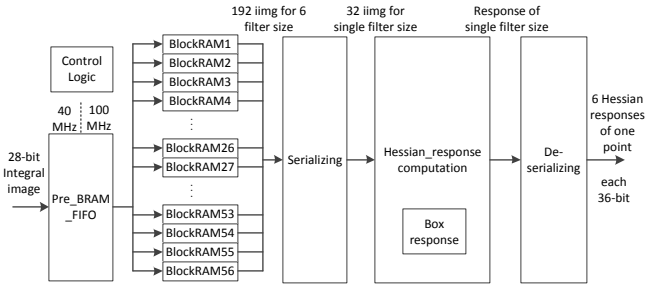

Figure 6. Build_Hessian_Response block diagram

defined as 2 and octave 2 doubles that. Therefore, iterative cycle is 4 lines. Calculation of filter response of octave 1 is carried out in line 1, 3, 5, 7, etc, while that of octave 2 in line 1, 5, 9, 13, etc. Based upon these analysis, this block utilizes 56 Block RAMs of depth of 800 to buffer 56 integral image lines. Each Block RAM could be accessed independently. 600 lines of integral image are stored into the 56 Block RAMs iteratively. This architecture makes the writing of incoming pixels and calculation of Hessian response happen concurrently. For example, the incoming integral image pixel of line 53 is stored into Block RAM 53 while the first 52 image lines are all available for Hessian response calculation of octave 1 and octave 2. 100 MHz clock frequency ensures the pace of 192 pixels (due to overlap, it is actually 164 pixels) access for the response computation of 6 different filter size. Pre_BRAM_FIFO buffers incoming integral image at 40 MHz pixel rate and delivers the data at 100 MHz system clock. Hessian_Response_Computation is the pipelined calculation unit of Hessian-matrix determinants. To save DSP resources of FPGA, 192 pixels are serialized into 6 groups of 32 pixels. Each group are necessary for Hessian determinant calculation of one scale. Afterwards, 6 Hessian determinants are de-serialized and transferred to Is-Extreme block in parallel.

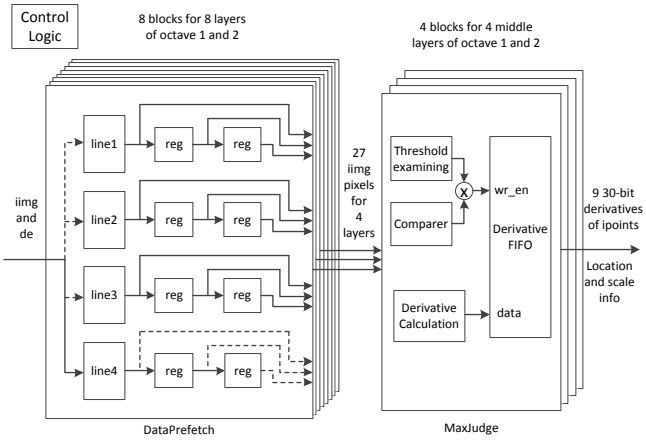

Figure 7. Is_Extreme block diagram

2) Is_Extreme: Is_Extreme unit (in Fig. 7) is aimed to find local maxima of Hessian-matrix determinants as interest points candidates. With the incoming Hessian-matrix responses of first two octaves, local maxima are detected on layers of filter size of 15, 21, 27 and 39. The block consists of DataPrefetch, MaxJudge and control logic sub-blocks. For each scale, DataPrefetch block uses 4 Block RAMs of depth 400 or 200 (400 for octave 1 and 200 for octave 2) to store current Hessian-matrix determinant line and buffer last 3 lines. When a determinant is stored in one Block RAM, the three determinants on top of it are read. Two registers to buffer the read data are refreshed concurrently. With 8 such blocks, each determinant is sent to the downstream MaxJudge block together with its 26 neighbors at x, y and scale space. Each determinant is compared with its 26 neighbors as well as the threshold to determine whether it is an interest point candidate in MaxJudge block. To save clock cycles, this block also performs the first and second order derivatives of Hessian-matrix response calculation on x, y and scale simultaneously. If it passes the local maximum and threshold check, their derivatives are then stored into Derivative FIFO with their locations and scales information. Control logic block generates the address information and controls data flow of Is_Extreme module.

3) Interpolate_Extremum: Non-maximum suppression is carried out in this block. At the beginning, a task queue block chooses entering data from one of the derivative FIFOs of filter size 15, 21, 27 and 39 in Is_Extreme block. Then, the main computation is to perform the matrix operation as in (4).

$$\begin{aligned} O &= -H \backslash D \\ &= -\begin{bmatrix} H_{xx} & H_{xy} & H_{xs} \\ H_{xy} & H_{yy} & H_{ys} \\ H_{xs} & H_{ys} & H_{ss} \end{bmatrix} \backslash \begin{bmatrix} D_x \\ D_y \\ D_s \end{bmatrix} \\ &= -\frac{1}{\det(H)} \cdot \begin{bmatrix} H^*_{xx} & H^*_{xy} & H^*_{xs} \\ H^*_{xy} & H^*_{yy} & H^*_{ys} \\ H^*_{xs} & H^*_{ys} & H^*_{ss} \end{bmatrix} \cdot \begin{bmatrix} D_x \\ D_y \\ D_s \end{bmatrix} \end{aligned} \tag{4}$$

where D and H are first and second order derivatives of Hessian-matrix determinant of current candidate.

Considering enormous resources demanding of division, only fraction $\frac{1}{\det(H)}$ is computed to transform other arduous division to multiplication. The absolute value of matrix O elements should be less than 0.5 so that the local maximum is accepted as interest point. The x, y and scale of interest points are then adjusted accordingly.

D. Memory management unit

A frame of 800×600 resolution is too large for on-chip Block RAM and can only be stored in on-board SDRAM. Xilinx provides designers the 7 Series FPGA Memory Interface Generator (MIG) to communicate with off-chip DDR3 SDRAM. It wraps the complex physical signals of DRAM and provides an easily-access user interface. The user interface runs at 200 MHz clock with 512-bit data width. In our design, this unit manages all memory access, and plays the role of

a bridge between interest point detector and interest point descriptor. It performs the following functions:

1) Grouping 8 successive integral image points and writing them into the MIG and SDRAM. A clock domain crossing FIFO is deployed to interface between 40 MHz pixel rate to 200 MHz MIG data rate.
2) Generating the addresses of interest point surroundings in the rectangle between (-13s, -13s) and (12s, 12s) with interval *s*, where *s* is the scale. Then read command is applied to MIG with desired pixels addresses. Due to the single port property, we grant write command with higher priority than read command.
3) When all the surroundings of an interest point, 26×26 integral image pixels, are available, transferring them to Interest Point Descriptor module serially with index and scale information of current interest point. The clock domain crossing between 200 MHz MIG UI clock and 100 MHz system clock is also implemented by independent clock FIFO.

E. Interest point descriptor

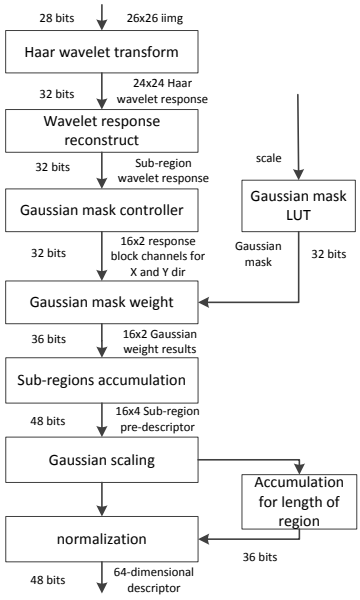

Figure 8. Interest point descriptor block diagram

Unlike original SURF implementation, modified SURF calculates descriptor from a square region with edge size of $24s$. Equally, this rectangle is split up to 16 sub-region. Each sub-region is $9s \times 9s$ (sampling step s). Adjacent sub-regions have overlap of 9×4 points. To perform modified SURF descriptor computation, a 26×26 matrix around interest point needs to be extracted from the integral image to calculate the 2D Haar wavelet transform at x and y direction. The wavelet filter size is 2×2 with interval s. This block buffers 3 lines of serial-coming integral image pixels to calculate Haar wavelet response of the central lines. After that, the 24×24 responses are reconstructed as 16 9×9 sub-blocks. We use Gaussian mask ($\sigma = 3.3s$) to weight Haar wavelet responses of sampling points in each sub-block simultaneously. Gaussian mask of possible scales has been pre-calculated and loaded to look up table during the initial stage. After that, we sum d_x, d_y and $|d_x|$, $|d_y|$ of all 16 sub-regions respectively and get the pre-descriptor. The pre-descriptor is organized as a 64×1 vector. Finally, the pre-descriptor is Gaussian-scaled and normalized to get the final descriptor for a precise matching result. An accumulation unit is used to compute normalization factor. Fig. 8 shows key design details, such as data flow and bus width, that are important to achieve real-time performance.

F. Descriptor Comparator

After interest point descriptors calculation, the comparator computes the similarity of current frame and pre-built traffic sign libraries to determine whether a traffic sign is detected. As an initial experiment, the system stores only stop signs in the library using on-chip Block RAM. We choose 128 representative interest points in the standard stop sign image, then calculate their descriptors for the library.

When the descriptor of an interest points enters this module, the comparator reads descriptors of interest points in the library for matching. The entering descriptor needs to be compared with every descriptor in the library. The speed is unacceptable if the match is conducted completely serially. As a tradeoff between speed and resource use, the 128 descriptors are divided into 8 independent groups. The groups are matched with the current descriptor concurrently while 16 descriptors in a group enter in pipeline mode. This pipeline architecture reduces the processing time considerably.

The descriptor of each interest point is compared with 128 descriptors in the library, which produces a distance vector with 128 values. For the subsequent interest points, only the smallest distance to each descriptor in the library is retained. After all the interest points descriptors are processed, this module produce a vector with shortest distances to each of the 128 library descriptor. These 128 values are then sorted and the sum of the smallest 30 values is then compared with the threshold. If the sum is less than the threshold, then a stop sign is detected.

IV. Results

We build the system on a Kintex-7 FPGA board shown in Fig. 9. The video frames are sent to a Xilinx KC705 platform through the AVNET video I/O board on the FMC connector. FPGA detects traffic signs of every video frame and output the video for display using the same adaptor.

Table 2 presents the design summary on Kintex 7 FPGA. The maximum frequency is 125.92 MHz (the Xilinx memory interface IP core runs independently at a separate 200 MHz clock zone). All timing constraints are met. The test frame rate is 60 fps of 800×600 resolution. The FPGA-SURF implementation achieves real-time while producing the same results in video stream as original CPU-based SURF, thus very promising for future industry application.

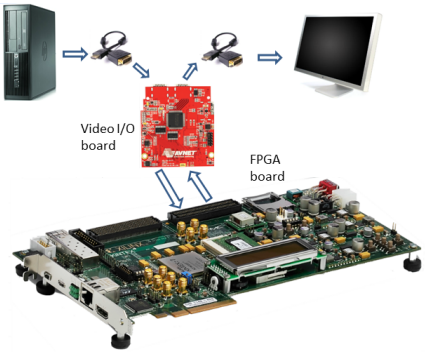

Figure 9. Traffic sign detection system

Table II
DEVICE UTILIZATION SUMMARY

Resources	Used	Available	Utilization
Number of Slices Registers:	108,581	407,600	26%
Number of Slice LUTs:	179,559	203,800	88%
Number of bonded IOBs:	173	500	34%
Number of RAM36s:	182	445	40%
Number of RAM18s:	80	890	8%
Number of DSP48s	244	840	29%

V. CONCLUSION

In this paper, we present a real-time implementation of the Speeded Up Robust Features (SURF) algorithm for a traffic sign detection on a Xilinx KC705 FPGA platform. Aiming to reduce the computational complexity, parallel architecture and data flow methods are applied in the system design. Furthermore, the pipeline design effectively produces high frame rate video processing. We have demonstrated traffic signed detection using SURF on an FPGA with real-time SVGA video at 60 fps.

VI. ACKNOWLEDGMENTS

The authors would particularly like to thank The MathWorks, Inc. for generously supporting this research. We also would like to thank D.Kroon for sharing his Matlab SURF code with us.

REFERENCES

[1] A. Broggi, C. Caraffi, R. Fedriga, and P. Grisleri, "Obstacle detection with stereo vision for off-road vehicle navigation," in *Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, 2005, pp. 65–65.

[2] K. Kluge and S. Lakshmanan, "A deformable-template approach to lane detection," in *Intelligent Vehicles '95 Symposium., Proceedings of the*, 1995, pp. 54–59.

[3] H. G. Jung, D. S. Kim, P. J. Yoon, and J. Kim, "Parking slot markings recognition for automatic parking assist system," in *Intelligent Vehicles Symposium, 2006 IEEE*. IEEE, 2006, pp. 106–113.

[4] R. Labayrade, D. Aubert, and J. P. Tarel, "Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation," in *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 2, 2002, pp. 646–651 vol.2.

[5] M. Bertozzi and A. Broggi, "Gold: A parallel real-time stereo vision system for generic obstacle and lane detection," *Image Processing, IEEE Transactions on*, vol. 7, no. 1, pp. 62–81, 1998.

[6] K. Brkic, "An overview of traffic sign detection methods," *Department of Electronics, Microelectronics, Computer and Intelligent Systems Faculty of Electrical Engineering and Computing Unska*, vol. 3, p. 10000.

[7] L.-W. Tsai, J.-W. Hsieh, C.-H. Chuang, Y.-J. Tseng, K.-C. Fan, and C.-C. Lee, "Road sign detection using eigen colour," *IET computer vision*, vol. 2, no. 3, pp. 164–177, 2008.

[8] A. De La Escalera, L. E. Moreno, M. A. Salichs, and J. M. Armingol, "Road traffic sign detection and classification," *Industrial Electronics, IEEE Transactions on*, vol. 44, no. 6, pp. 848–859, 1997.

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

[10] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2. IEEE, 2006, pp. 1491–1498.

[11] M. J. Jones and P. Viola, "Robust real-time object detection," in *Workshop on Statistical and Computational Theories of Vision*, 2001.

[12] 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.

[13] Z. Zhang, C. Cao, R. Zhang, and J. Zou, "Video copy detection based on speeded up robust features and locality sensitive hashing," in *Automation and Logistics (ICAL), 2010 IEEE International Conference on*. IEEE, 2010, pp. 13–18.

[14] M. Zhou and V. K. Asari, "Speeded-up robust features based moving object detection on shaky video," in *Computer Networks and Intelligent Computing*. Springer, 2011, pp. 677–682.

[15] A. C. Murillo, J. Guerrero, and C. Sagues, "Surf features for efficient robot localization with omnidirectional images," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 3901–3907.

[16] J. Svab, T. Krajník, J. Faigl, and L. Preucil, "Fpga based speeded up robust features," in *Technologies for Practical Robot Applications, 2009. TePRA 2009. IEEE International Conference on*. IEEE, 2009, pp. 35–41.