OpenCV Based Road Sign Recognition on Zynq

Matthew Russell and Scott Fischaber
Analytics Engines
Belfast, Northern Ireland
m.russell@analyticsengines.com, s.fischaber@analyticsengines.com

***Abstract*—Road sign recognition is a key component in autonomous vehicles and also has applications in driver assistance systems and road sign maintenance. Here an algorithm is presented using the Xilinx Zynq-7020 chip on a Zedboard to scan 1920x1080 images taken by an ON Semiconductor VITA-2000 sensor attached via the FMC slot. The PL section of the Zynq is used to perform essential image pre-processing functions and color based filtering of the image. Software classifies the shapes in the filtered image, and OpenCV's template matching function is used to identify the signs from a database of UK road signs. The system was designed in six weeks, and can process one frame in approximately 5 seconds. This is a promising start for a real-time System on Chip based approach to the problem of road sign recognition and also for using the Zynq platform for rapid deployment of these types of applications.**

I. Introduction

Detection and recognition of traffic signs have several uses in modern vehicles, one of which is driver assistance systems (DASs). Systems which can identify circular speed limit signs and overtaking restrictions are available on some recent cars (for example those produced by Mobileye [1]), but none detect all types of road signs in all countries. DASs do not aim to replace the driver, but to aid them so they can focus on driving safely. Another application is autonomous vehicles; if these vehicles are to work in a real world environment they will need to be able to recognize road signs and take appropriate actions for each sign. Maintenance and inventory of road signs are tedious tasks to perform without an automated system to check the presence and condition of the road signs. Due to age, vandalism or theft, road signs can become difficult to recognize, therefore keeping signs in good repair is important to keep drivers informed of potentially dangerous conditions.

The task of Road Sign Recognition (RSR) involves two main steps, detection and recognition/classification. Challenges in the detection stage involve finding a way to eliminate all the non-sign objects in the image whilst retaining enough of any signs in the image that their position and size is not corrupted. Within reasonable limits, this stage should also try to overcome problems of partial occlusion, rotation and perspective distortion, overlapping signs, and changing environmental conditions (e.g. poor lighting or fog). Recognition needs to take these detected signs and identify them in order to provide relevant information to the user.

Fortunately, road signs are designed to be distinctive against the background of their environment, and therefore certain properties of signs can be exploited to accurately detect them. Both shape and color are obvious choices for distinguishing traffic signs from a background image. With few exceptions traffic signs are composed of red, blue, white, yellow/orange, and black colors, and are either circles, rectangles, triangles or octagons; other colors are normally used for directional information, and is out of scope of this work. Using this, it is possible to create a detection algorithm which searches for some or all of these features to detect signs.

Deployment of this algorithm inside a vehicle should also be considered. ARM processor architectures lead the microcontroller/embedded microprocessor market, with one key end-use being the automotive industry [2]; however, these devices do not typically have enough processing power to perform image processing tasks on large images in real-time. Recently, Xilinx released the Zynq All Programmable System on Chip (SoC), an FPGA with an embedded dual core ARM processor. This allows an entire system to be developed very quickly, since existing software can easily be ported for the ARM processors with custom hardware accelerators in the FPGA section of the device. Zynq hardware uses ARM's Advanced Extensible Interface (AXI) for its connections to the processors, allowing the user to abstract the interface for fast implementation. Another advantage of this FPGA platform is low power consumption, ideal for embedded applications.

This paper aims to demonstrate the use of the Zynq chip on the Zedboard [3] as an embedded platform for RSR. It will also investigate the programmability of this platform and its suitability to rapid development of embedded applications. The system will use an ON Semiconductor VITA-2000 image sensor attached via FMC to provide HD video resolution images. Hardware cores in the Programmable Logic (PL) of the Zynq will perform some pre-processing operations on the image, and the color segmentation stage of the algorithm. Software running in the Processing System (PS) will be used to perform the remainder of the algorithm, making use of the OpenCV library. Whilst raw performance is not the main focus of this paper, the system should be able to process one frame within at least 10 seconds.

Section II gives a summary of some previous research which has been done on RSR, with emphasis on FPGA based designs. An overview of the RSR algorithm is presented in Section III, which is further split into hardware architecture in the PL and software running in the PS. Results from some test images are analyzed in Section IV, and in Section V improvements are suggested to overcome some of the problems experienced. Finally, the paper is concluded in Section VI.

978-1-4799-0752-6/13/$31.00 ©2013 IEEE

II. Related Work

Research into RSR systems tends to split the process into the two aforementioned steps of detection and recognition/classification. Detection algorithms usually employ color or shape as a method of determining regions of interest in the image for further processing. Møgelmose, Trivedi and Moeslund present an overview of the methods used by 41 different papers [4]; their work reveals that it is common to use color segmentation on the input to remove the background. Since RGB values are quite sensitive to lighting changes, these approaches usually convert the pixels into another color space (typically Hue Saturation Intensity or Hue Saturation Value (HSI/HSV), but others such as CIECAM97 [5] and Yuv [6] [7] have been used) or use relationships between the components for segmentation, as in [8]. After color segmentation, the remaining objects in the image are classified, usually by shape; this process is quite efficient since many background shapes will have been removed and small objects can be filtered out. Some algorithms use shape detection as the first step to find signs in the input image, as in [9]; this enables them to use grayscale images, which can be processed faster or used to reduce color sensitivity. These shape detection algorithms are generally optimized for a particular shape of sign, such as circles, or use another method to eliminate detected shapes which do not represent signs in the image. Once signs are detected, recognition takes place; this is often performed by Neural Networks (e.g. [8] and [10]) or Support Vector Machines (e.g. [11] and [12]) but other approaches, such as template matching [13], have been used.

Many different approaches are available at each stage of the RSR process; those which are most relevant to the proposed design, and those which could be used as a basis for improvement shall be summarized here. Color segmentation using the (HSI) color space is performed in [10], [11], [12] and others, while some papers use HSV (e.g. [14]). Some of these use a function of the color space to perform segmentation (e.g. [10]), and white pixels can be included using achromatic decomposition (e.g. [11]). The authors of [15] demonstrate an algorithm to join two halves of signs such as "No Through Road" which may be split during segmentation, they also use a function of hue and saturation to filter the image. Shape approximation using the Ramer-Douglas-Peucker algorithm is performed in [16] to determine the shapes in a pre-processed image.

Recently Field Programmable Gate Arrays (FPGAs) have been used as a platform for RSR; the authors of [17] have used a MicroBlaze processor on a Virtex-5 with hardware accelerators for color filtering and morphological operations. They point out that previous works on FPGA have been orientated towards embedded processors without much consideration for hardware acceleration of the algorithms. Their system currently uses small images of 320x240 pixels, they implement morphological operations in hardware and complete their algorithm in 777 ms. In [18] LEON3 CPUs are instantiated in a Virtex4 FPGA, however this leaves little room for custom accelerators; their design completes within 600 ms, but the size of the images used is not discussed. An Altera Cyclone II FPGA is used in [19], but only circular traffic signs are considered.

All these approaches have used 'soft' processors in the logic, i.e. the processor is created from programmable logic; Irmak uses a 'hard' PowerPC processor on a Virtex5 in [20]. The advantage of a hard processor is that higher frequencies can be obtained since the architecture is optimized for the processor. Utilization of the FPGA is quite low in the design and only 60x60 images are tested; these images would need to be output from a previous stage which had already detected the sign. Execution time is around 70 ms for one of these images.

III. System Overview

The Zynq Evaluation and Development Board (ZedBoard) produced by Avnet and Digilent uses a Xilinx Zynq-ZC702 part and 512 MB of off-chip memory (OCM). There are many external connectors on the ZedBoard including UART, HDMI, Ethernet and FPGA Mezzanine Card (FMC) among others. Avnet have developed the FMC-IMAGEON module, which is an FMC connector to interface the ON Semiconductor VITA-2000 CMOS image sensor. A reference design for this connector (available online [21]) has been adapted by Xilinx for XAPP794 [22] and an Xcell journal article [23], in which they create an image processing pipeline with a webserver to perform image correction on 1920x1080 images.

In the proposed system, input images are taken from the VITA-2000 sensor which is set up to capture HD video resolution (1920x1080 pixels) at 72 frames per second. The images are pre-processed in hardware and passed from the PL to the PS via a triple frame buffer in the 512 MB OCM. When the user wishes to begin processing the data, the PS reads the frame buffer and performs the remaining steps of the algorithm. Output can be obtained from a webserver or UART, both of which can initiate the RSR process. A system overview is given in Fig.1 showing the sensor and Zynq chip.

Development for the Zynq platform is not supported by version 2012.4 of Xilinx's Vivado Design Suite, so the design was developed in PlanAhead (which is similar in style to Vivado). PlanAhead itself is used mainly as a link between two other Xilinx tools, Xilinx Platform Studio (XPS) and Software Development Kit (SDK). XPS is used to develop the hardware portion of the design, including the interface to the processors from the PL; SDK is where the software is built to run on the ARM processor target. Together with a library of IP and software drivers these offer a way to rapidly implement an entire system on the Zynq. Fig. 2 details the development process for the system.

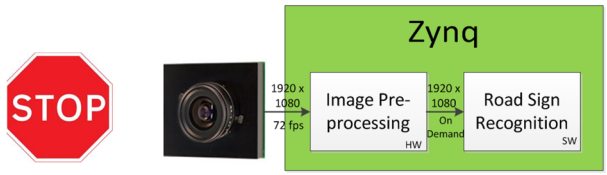


Fig 1. System Overview of the Design

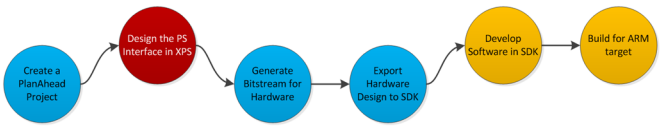


Fig 2. Development Process for Zynq in PlanAhead. Blue represents tasks performed in PlanAhead, red are performed in XPS, and yellow in SDK.

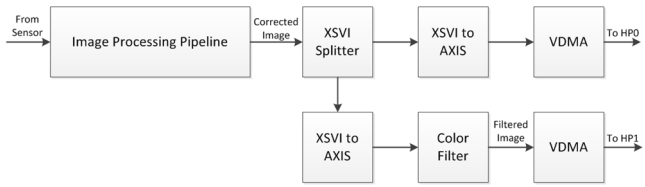


Fig 3. High Level Hardware Architecture in the Programmable Logic

A. Programmable Logic

The high level hardware architecture used in the design is shown in Fig. 3. An Image Processing Pipeline (IPP) is used to perform the pre-processing on the image; the Xilinx Streaming Video Interface (XSVI) is used in this pipeline to interface the cores. The ARM cores use AXI, specifically AXI-Stream (AXIS) for video applications. For this reason, the XSVI signal is split and converted into AXIS for the AXI Video Direct Memory Access (VDMA) core. A custom filter core applies color segmentation to one of the AXIS signals.

The VDMAs are connected to the PS via the High Performance (HP) ports; these have dedicated access to the memory controller in the PS, and can support high bandwidth transfers to/from the OCM. Each VDMA controls a triple frame buffer in the memory which is accessible from software. The video from the VITA is continuous, so the frame buffer is paused when it needs to be read; the software can then transfer the image to a bitmap file, which is stored on an SD card attached to the Zedboard, and the frame buffer is restarted.

The IPP (see Fig. 4 for details) contains hardware to pre-process the image and remove any defects in the sensor; with the exception of the VITA receiver core, which is designed by Avnet, all of these cores are available as part of Xilinx's Video and Image Processing Pack [24]. Defective pixel correction detects and corrects defective pixels in the image sensor. CFA Interpolation needs to be performed to reproduce a full image from the Bayer pattern used on the sensor. The Video Timing Controllers detect synchronization pulses in the input image and generate timing signals for the downstream core. CCM and Gamma Correction cores are used to perform automatic color correction on the input image. The CCM core performs automatic white balancing in different lighting environments or can be configured manually; the gamma correction core can also be configured for automatic or manual gamma correction.

The first stage in the RSR algorithm as shown in Fig. 5 is color filtering in the HSV color space on the corrected image to remove background colors. HSV is similar to HSI, but differs mainly in the way white is represented. HSI represents full intensity as white, no matter what the value of saturation is; whereas HSV represents white with full value and saturation of 0. The chief reason for the choice of HSV over HSI is the ease of implementation in hardware; calculating HSI requires more divisions and would therefore increase latency in hardware.

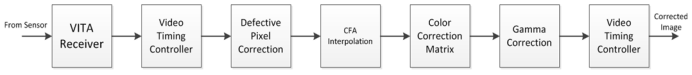


Fig 4. Image Processing Pipeline, arrows indicate the XSVI connections between the hardware cores

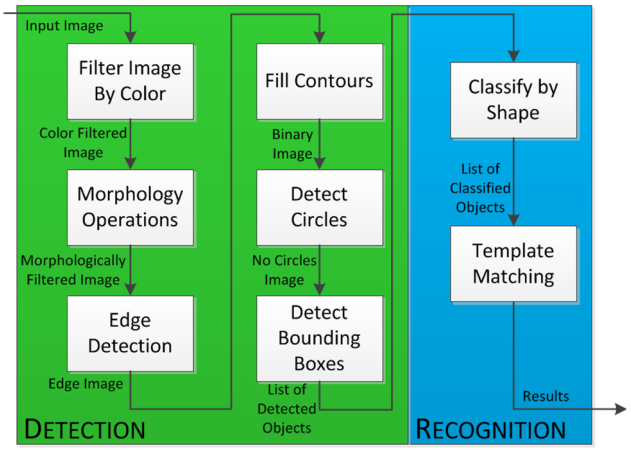


Fig 5. Flow Diagram for the Road Sign Recognition Algorithm.

The equations used to determine the components in HSV are shown in (1) (2) and (3).

$$Hue = \begin{cases} 0, & \text{if } \delta = 0 \\ \dfrac{g-b}{\delta}, & \text{if } \max(r,g,b) = r \\ \dfrac{b-r}{\delta} + 2, & \text{if } \max(r,g,b) = g \\ \dfrac{r-g}{\delta} + 4, & \text{if } \max(r,g,b) = b \end{cases} \quad (1)$$

$$Saturation = \frac{\delta}{\max(r,g,b)} \quad (2)$$

$$Value = \max(r,g,b) \quad (3)$$

Where r, g, and b are the components in the RGB color space, and $\delta = \max(r, g, b) - \min(r, g, b)$. Hue is in the range 0 to 6, which is scaled in hardware to the range 0 to 384; saturation and value are both represented in hardware as a number between 0 and 255. If the maximum value was the red component, and the blue component is greater than the green, then the hue will be negative; this is fixed by adding the maximum value of 6 (384 in hardware) to a negative hue.

Only traffic signs which are red or blue are considered; this covers the majority of signs which do not provide directional information in the UK. The chosen threshold values for Hue and Saturation are summarized in Table I. If a pixel's Hue and Saturation lie in one of these regions it is an intermediate pass.

TABLE I. SEGMENTATION THRESHOLDS IN HSV

Region[a]	Minimum Hue	Maximum Hue	Minimum Saturation
R1	373	8	128
R2	362	16	153
B1	234	260	153
B2	215	277	179

a. Region is an identifier, for example, R1 stands for Red 1. This indicates an area which is predominantly in the red part of the HSV color space, and bounded by the limits

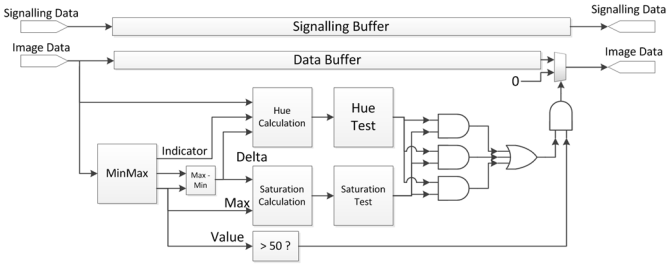


Fig 6. Hardware Colour Segmentation Core

In addition to these constraints, the Value component must be above 50 for the pixel to be passed. Each pixel is tested individually and if it is inside the thresholds it will be passed; otherwise the pixel will be replaced with a black pixel.

A block diagram of the hardware core used to perform the color segmentation is shown in Fig. 6; the signaling and input data are stored in buffers while the hardware determines whether the pixel should be passed. The MinMax module calculates the minimum and maximum values of the three components and also outputs a vector indicating which one (R, G, or B) is the maximum. Delta is the difference between the maximum and minimum values and is used in both the hue and saturation calculation modules. Once the hue and saturation have been calculated, they are tested as defined above along with the value component. Hue and saturation tests are combined to form an intermediate pass value; if this and the value test result indicate the pixel passed then it is output to the VDMA, otherwise it is replaced with zero (i.e. a black pixel).

B. Processor Algorithm

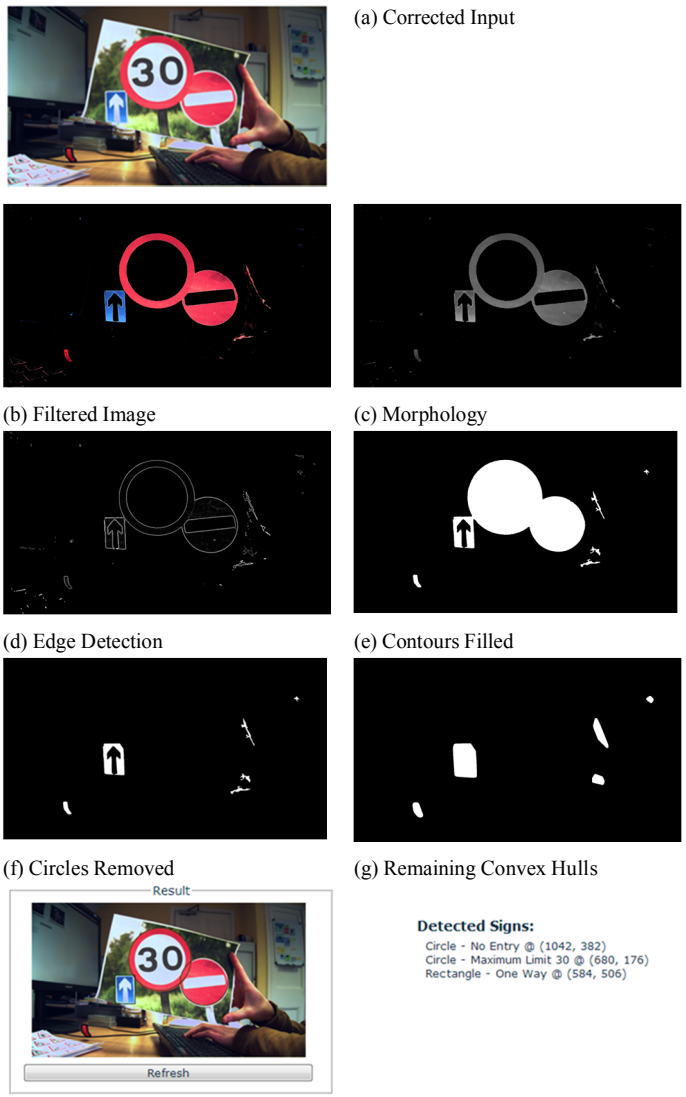
In the PS, both images are read from the two frame buffers described earlier; the unfiltered image is saved for use later in the matching stage. Fig. 7 gives an example of the outputs from each stage; Fig. 7(a) shows the corrected input from the IPP. As shown in Fig. 3 this is sent to the Colour Segmentation Core and the output from this hardware filter is displayed in Fig. 7(b). Morphology operations are applied to the filtered image to recover some broken signs and to remove very small objects. Firstly, the image is converted to grayscale, and the opening and closing operations are performed (see [25] for a detailled explanation). Opening (erosion followed by dilation) has the effect of removing small objects and closing (dilation followed by erosion) removes small black patches in the image. The output of this stage is shown in Fig. 7(c).

OpenCV provides a function to detect the contours in an image, but requires a grayscale image of the edges. To achieve this, Canny edge detection is used. An example output of the edge detection is shown in Fig. 7(d). An image containing the edges is passed into the contour detector, and any contours which pass tests for size are filled in white on a new image. To avoid using computation time on signs which are a long distance away, any objects less than 30 pixels wide or high are removed. The result is a binary image, such as in Fig. 7(e), where white indicates a correctly colored pixel group which could indicate a sign and black indicates the background.

The next stage uses the Hough Circle Transform, implemented in the OpenCV library, to detect and remove circular signs from the image. This stage helps in two ways: firstly it helps by identifying circles early on which would otherwise take up processing time, and secondly it helps to detect signs which overlap (as long as one sign is a circle). Multiple results for a single circle can be returned by this algorithm, so some filtering is done to ensure only the best match for each circle is kept. Any circles which are too close to the center of a previous circle are removed. All circles that pass filtering are filled in black on the input image to remove them from further processing steps, see Fig. 7(f).

The remaining contours are detected again using edge detection and the OpenCV function as before to find their bounding boxes. Those with an aspect ratio from 0.4 and 2.0 are removed from the list; this will incorrectly filter signs which have become distorted by perspective. Road signs are almost always convex, so the convex hull of the detected contour is drawn on a new image, see Fig. 7(g). This reduces the effect of occlusions, which normally creates an indent in the outline of the sign.

Each bound is used to select a region of interest (ROI) from



(a) Corrected Input

(b) Filtered Image

(c) Morphology

(d) Edge Detection

(e) Contours Filled

(f) Circles Removed

(g) Remaining Convex Hulls

(h) Resulting image with highlighted signs and textual list

Fig 7. Sample output from each stage.

the input image which is resized to 50x50 pixels; this ROI should ideally contain the convex hull of a single sign. OpenCV has a function that uses the Ramer-Douglas-Peucker algorithm to approximate a shape; this is applied to the hulls, and the number of corners is counted. The count is used to classify the shape as a triangle (3), rectangle (4), octagon (8) or unknown (others). This is repeated for all detected bounds and the detected shapes are joined with the list of circles to form a full list of detected sign candidates in the image.

OpenCV provides a template matching function which uses the squared difference method to match an image with a template; this is used to identify the sign. The UK's Department for Transport has a database of over 600 images of traffic signs [26]; 86 of these images were used as a test sample for the application. More templates, including ones from other countries, can easily be added using the SD card (up to 32 GB). The templates currently use 672 KB out of a total size of 36MB for all of the files, including boot files. In the first step of template matching, the original (unfiltered) input image is opened and the ROI indicated by the first shape's bound is selected. This is resized, as in the shape classification stage, to 50x50 pixels so both template and image are the same size. Converting the image and template to grayscale reduces the effect of color variation in different lighting conditions; since the template and sign candidate are now the same size, the squared difference operation is only applied at one location. Classification values from earlier are used to narrow the search for a match: circles and octagons will only be tested against all circular signs and the octagonal stop sign; rectangles and triangles are tested against rectangular and triangular signs; and unknown shapes are tested against all known templates. Once all selected templates have been matched with the input image, the best match for the sign candidate is chosen and stored.

If the best match for the sign candidate has a match level higher than a match threshold, it is declared confidently matched; a weak match threshold also exists to catch some signs which are badly colored due to the environment. During the design process, parts of the background in the image were sometimes matched to signs, mainly the "No Through Road" sign. To counteract this, the rectangular signs (and some circular signs) which were often incorrectly detected have been given a higher match threshold. Finally, any signs which were confidently matched are outlined in green on the unfiltered input image, and a message appears in the console and on the webserver to indicate the type of sign detected. Weak matches are outlined in yellow, and also output a message.

IV. RESULTS

Designing the system would normally take several months to develop the cores, connect them to the processor, and develop the software code. With the Xilinx tools, and the AXI standard interface, the entire system took only 6 weeks to design, implement and test. When implemented the hardware usage for the design is 80% of the available slices on the FGPA. There could be more room available for additional accelerators, but care should be taken to ensure latency and clock rate are not excessively affected. Along the image data path the clock rate is 142.86 MHz, which allows the 1920x1080 images to be streamed at 72 fps; the ARM processors have a fixed clock frequency of 800 MHz. The system was tested with several images during the development to ensure a robust performance under varying conditions. Color variation in the input image is handled well by the CCM core; however, conditions such as fog or snow will add noise to the input images, and may affect the color correction.

In an image with no signs to detect, the program will complete within 5 seconds (frame rate 0.2 fps), and with 20 signs in the image the program takes about 7 seconds (0.143 fps). This would indicate an increase of roughly 0.1 seconds per sign, although this number would vary depending on what shape was detected (e.g. an unknown shape is tested against all templates). Performance is within the 10 second target in both cases, although it is slower than the designs compared in Table III; as can be seen, they all use resolutions of VGA quality or less while this algorithm processes HD images, which are about four times as large as VGA. For further speed improvements the software code could be optimized. NEON acceleration (for Single Instruction Multiple Data operations) is available in the ARM cores, but the OpenCV library does not officially support this for most of its functions. Also the Morphology block could be implemented in hardware as it requires 34% of the total processing time (edge detection and circle detection both require 22% each).

V. FUTURE OPTIMIZATIONS

Segmentation in HSV rarely eliminates a sign from the input image, but it could be improved to more accurately approximate the histogram of sign colors. Some of the cited works (e.g. [9], [14]) use a function of the hue and saturation which would be more accurate. Separating the different color components in a sign and idealizing them by setting them equal to a "perfect" red/blue could improve the results of the template matching stage. Signs such as "No Entry" and "No Through Road" are occasionally split into two separate objects

TABLE II. COMPARISON OF FPGA BASED DESIGNS FOR ROAD SIGN RECOGNITION

FPGA Design	FPGA Family	Frame Rate	Processor		Image Size	Sign Types Detected
Proposed Design	Zynq	0.2	Hard	ARM Cortex A9	1920x1080	All red or blue
[17]	Virtex-5	1.29	Soft	Microblaze	320x240	All red
[18]	Virtex-4	2	Soft	LEON	Not Available	Circular
[19]	Cyclone-II	0.058	Soft	Nios II	320x240	Circular
[20]	Virtex-5	14.3	Hard	PowerPC	60x60	All red or blue

a. The design for [20] can recognize 32 different signs; under the current setup the proposed design can recognize 86 signs. [17], [18] and [19] do not state the number of signs which can be recognized by their systems.

by morphology or filtering. This causes the sign to be defined as two separate shapes, which are normally removed due to aspect ratio constraints. A solution to this problem is proposed in [15] by testing the area and center of mass of the two rectangular halves, which can then be successfully joined.

Overlapping signs are only dealt with during the circle removal stage, and only if one of the signs is a circle. This poses problems, even when two signs are not overlapping but their borders touch, since the collection of signs will be detected as one object. One possible method of improving this would be to implement a more robust shape detection stage after circle detection, which would identify overlapping shapes; this may affect the ability of the algorithm to deal with partial occlusion, since the method currently used relies on forming the convex hull of the detected shapes. Partial occlusions where the occlusion is at a corner should not affect the detection of the sign, but may cause shape and matching problems.

VI. CONCLUSIONS

A Road Sign Recognition algorithm has been implemented on the Zynq SoC with promising results. Hardware cores have been used to preprocess HD quality images and perform color based filtering, whilst software performs the remainder of the algorithm. The system successfully detects all red and blue signs in test images with few false positives. Future considerations include reducing the processing time for one frame, and improving the matching algorithm to cope with challenging conditions. Overall the algorithm performs well and demonstrates the potential which the Zynq possess as an embedded platform for RSR and other image processing applications; it also exhibits promising results for reducing design time and increasing flexibility using the inbuilt ARM processors and programming environment.

ACKNOWLEDGMENT

This work was funded with support from the UK Technology Strategy Board under grant number 710032.

REFERENCES

[1] Mobileye, "Traffic Sign Detection - Mobileye," [Online]. Available: http://www.mobileye.com/technology/applications/traffic-sign-detection/. [Accessed: Feb. 27, 2013].

[2] Electronic Specifier, "ARM and x86 Lead 32/64-bit Microcontroller & Embedded Microprocessor Market in 2011," [Online]. Available: http://www.electronicspecifier.com/Tech-News/ARM-and-x86-Lead-32-64-bit-Microcontroller--Embedded-Microprocessor-Market-in-2011.asp. [Accessed: March 28, 2013].

[3] ZedBoard.org, "ZedBoard," [Online]. Available: http://www.zedboard.org/. [Accessed: March 5, 2013].

[4] A. Møgelmose, M. M. Trivedi and T. B. Moeslund, "Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey," *IEEE Trans. Intell. Transp. Syst.,* vol. 13, no. 4, pp. 1484–1497, Dec. 2012.

[5] X. Gao, L. Podladchikova, D. Shaposhnikov, K. Hong and N. Shevtsova, "Recognition of traffic signs based on their color and shape features extracted using human vision models," *J. Vis. Commun. Image Represent.*, vol. 17, no. 4, pp. 6755–685, 2006.

[6] J. Miura, T. Kanda and Y. Shirai, "An Active Vision System for Real-Time Traffic Sign Recognition," in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, Dearborn, MI, Oct. 2000, pp. 52–57.

[7] A. Broggi, P. Cerri, P. Medici, P. P. Porta and G. Ghisio, "Real Time Road Signs Recognition," in *Proc. IEEE Intell. Veh. Symp.*, Istanbul, June 2007, pp. 981–986.

[8] A. d. l. Escalera, L. E. Moreno, M. A. Salichs and J. M. Armingol, "Road Traffic Sign Detection and Classification," *IEEE Trans. Ind. Electron.,* vol. 44, no. 6, pp. 848–859, 1997.

[9] F. Moutarde, A. Bargeton, A. Herbin and L. Chanussot, "Robust on-vehicle real-time visual detection of American and European speed limit signs, with a modular Traffic Signs Recognition system," in *Proc. IEEE Intell. Veh. Symp.*, Istanbul, June 2007, pp. 1122–1126.

[10] A. d. l. Escalera, J. Armingol and M. Mata, "Traffic sign recognition and analysis f or intelligent vehicles," *Image and Vision Computing*, vol. 21, no. 3, pp. 247–258, March 2003.

[11] S. Maldonado-Bascón, S. Lafuente-Arroyo, P. Gil-Jiménez, H. Gómez-Moreno and F. López-Ferreras, "Road-Sign Detection and Recognition Based on Support Vector Machines," *IEEE Trans. on Intell. Transp. Syst.,* vol. 8, no. 2, pp. 264–278, June 2007.

[12] S. Lafuente-Arroyo, S. Salcedo-Sanz, S. Maldonado-Bascón, J. A. Portilla-Figueras and R. J. López-Sastre, "A Decision Support System for the Automatic Management of Keep-Clear Signs Based on Support Vector Machines and Geographic Information Systems," *Expert Syst. with Applicat.: An Int. J.,* vol. 37, no. 1, pp. 767–773, Jan. 2010.

[13] D. M. Gavrila, "Traffic Sign Recognition Revisited," in *Mustererkennung (DAGM)*, Berlin, Germany: Springer Berlin Heidelberg, 1999, pp. 86-93

[14] F. Ren, J. Huang, R. Jiang and R. Klette, "General traffic sign recognition by feature matching," in *24th Int. Conf. Image and Vision Computing New Zealand*, Wellington, Nov. 2009, pp. 409–414.

[15] C. F. Paulo and P. L. Correia, "Automatic Detection and Classification of Traffic Signs," in *8th Int. Workshop on Image Anal. for Multimedia Interactive Services*, Santorini, June 2007, p. 11.

[16] D. Soendoro and I. Supriana, "Traffic sign recognition with Color-based Method, shape-arc estimation and SVM," in *Int. Conf. Elect. Eng. and Informatics*, Bandung, July 2010, pp. 1–6.

[17] S. Waite and E. Oruklu, "FPGA-Based Traffic Sign Recognition for Advanced Driver Assistance Systems," *J. Transp. Technologies,* vol. 3, no. 1, pp. 1–16, 2013.

[18] M. Muller, A. Braun, J. Gerlach, W. Rosenstiel, D. N.-. huser, J. M. Zollner and O. Bringmann, "Design of an Automotive Traffic Sign Recognition System Targeting a Multi-Core SoC Implementation," in *Design, Automation & Test in Europe Conf. and Exhibition*, Dresden, March 2010, pp. 532–537.

[19] M. A. Souki, L. Boussaid and M. Abid, "An Embedded System for Real-Time Traffic Sign Recognizing," in *3rd Int. Design and Test Workshop*, Monastir, Dec. 2008, pp. 273–276.

[20] H. Irmak, "Real Time Traffic Sign Recognition System on FPGAs," M.S. thesis, Middle East Technical Univ., Ankara, Turkey, 2010.

[21] Avnet Inc., "HDMI Input/Output FMC Module," [Online]. Available: http://www.em.avnet.com/en-us/design/drc/Pages/HDMI-Input-Output-FMC-module.aspx. [Accessed: March 5, 2013].

[22] M. Bergeron, S. Elzinga, G. Szedo, G. Jewett and T. Hill, "1080p60 Camera Image Processing Reference Design," Jan. 2, 2013. [Online]. Available: http://www.xilinx.com/support/documentation/application_notes/xapp794-1080p60-camera.pdf. [Accessed: March 5, 2013].

[23] G. Szedo, S. Elzinga and G. Jewett, "Image Sensor Color Calibration Using the Zynq-7000 All Programmable SoC," *Xcell Journal*, no. 81, pp. 14–21, 2012. [Online]. Available: http://www.origin.xilinx.com/publications/xcellonline/index.htm. [Accessed: March 5, 2013].

[24] Xilinx Inc., "Video and Image Processing Pack,". [Online]. Available: http://www.xilinx.com/products/intellectual-property/EF-DI-VID-IMG-IP-PACK.htm. [Accessed: 5 March 2013].

[25] OpenCV Dev Team, "More Morphology Transformations—OpenCV 2.4.4.0 Documentation,". [Online]. Available: http://docs.opencv.org/doc/tutorials/imgproc/opening_closing_hats/opening_closing_hats.html. [Accessed: 5 March 2013].

[26] UK Department for Transport, "Traffic signs image database," [Online]. Available: http://www.dft.gov.uk/trafficsignsimages/. [Accessed: 5 March 2013].