PROCEEDINGS OF SPIE

SPIEDigitalLibrary.org/conference-proceedings-of-spie

Real-time multi-camera video acquisition and processing platform for ADAS

Sergio Saponara

Sergio Saponara, "Real-time multi-camera video acquisition and processing platform for ADAS," Proc. SPIE 9897, Real-Time Image and Video Processing 2016, 98970E (29 April 2016); doi: 10.1117/12.2228235

SPIE.

Event: SPIE Photonics Europe, 2016, Brussels, Belgium

Real-time Multi-camera Video Acquisition and Processing Platform for ADAS

Sergio Saponara
[a]Dept. of Information Engineering, University of Pisa, via G.Caruso 16, 56122, Pisa, I,
sergio.saponara@iet.unipi.it, tel/fax: +39 050 217602/522

ABSTRACT

The paper presents the design of a real-time and low-cost embedded system for image acquisition and processing in Advanced Driver Assisted Systems (ADAS). The system adopts a multi-camera architecture to provide a panoramic view of the objects surrounding the vehicle. Fish-eye lenses are used to achieve a large Field of View (FOV). Since they introduce radial distortion of the images projected on the sensors, a real-time algorithm for their correction is also implemented in a pre-processor. An FPGA-based hardware implementation, re-using IP macrocells for several ADAS algorithms, allows for real-time processing of input streams from VGA automotive CMOS cameras.

Keywords: Real-time, Video/image processing, ADAS (Advanced Driving Assistance System), DSP (Digital Signal Processor), Traffic signs recognition, Automotive electronics, FPGA (Field Programmable Gate Array), CMOS camera

1. INTRODUCTION

The interest on ADAS (Advanced Driving Assistance System) is continuously increasing in the automotive field to improve vehicle safety and comfort [1-7]. Real-time video processing is a key technology for ADAS [3-14].
At the state-of-the-art several techniques have been proposed for ADAS using different technologies such as active sensors, RADAR (RAdio Detection And Ranging) and LIDAR (LIght Detection And Ranging), or passive ones such as CCD (Charge Coupled Devices) and CMOS (Complementary Metal Oxide Semiconductor) cameras.
In this work, after comparing in Section 2 CMOS-based imaging vs. other competing technologies, in Sections 3 and 4 we propose a real-time processing platform implementing multiple ADAS functions on the same device: traffic sign recognition, image enhancement in bad light and night conditions, all around view for parking assistance, image distortion correction. To ensure all around panoramic view, multiple CMOS cameras in the visible range are used on front, rear, left and right side of the vehicle plus an infrared (IR) camera on front side. Fish-eye lenses, extending the camera field of view, are also adopted. Conclusions are drawn in Section 5.

2. IMAGING SENSORS FOR ADAS

Active sensors such as RADAR [15] and LIDAR [16] have their own source of energy, which is properly modulated by the transmitter section of the radio frequency or optical transceiver: electromagnetic waves at 24 GHz or 77-79 GHz for automotive RADAR, and light for the LIDAR. The image they acquire from the environment surrounding the car is obtained by measuring the reflected energy. With respect to passive camera systems, RADAR and LIDAR allow for higher performance although at an increased complexity and cost. As example, a LIDAR leads to accurate distance and time-of-flight speed measurements. The HDL-64E LIDAR sensor from Velodyne [16] has 360° horizontal Field of View (FOV), 26.8° vertical FOV and allows for obstacle detection with an angular resolution better than 0.08° (azimuth) and a distance accuracy of 2 cm up to 120 m. It has an array of 64 channels (Lasers transmitting/receiving) working with a frame rate up to 20 Hz. The power consumption is 60 W in typical conditions, the weight is 15 kg and the size is about 20 cm (large) x 30 cm (height). To reduce the size at about 10 cm x 15 cm and the weight at about 1.3 kg, the HDL-32E LIDAR has 32 channels. It allows for a detection range from 1 m up to 100 m with an accuracy of 2 cm. The horizontal FOV is 360° and the vertical FOV is 20°. The frame rate is typically 10 Hz (user selectable, 5-20 Hz), and it is constructed to operate in the thermal range -10 °C to +60 °C and to ensure a protection grade IP67. The power cost is 12 W. By further reducing the array size to 16 channels, the VLP-16 covers a detection range up to 100 m, with an accuracy

of 3 cm. It has horizontal and vertical FOV of 30° and 360°, respectively. The horizontal and vertical FOV resolutions are 0.1^0 and 2^0 respectively at 5 Hz. VLP-16 operates in the thermal range -10 °C to +60 °C and its package has a protection grade IP67. The power cost is 8 W and the size is about 10 cm x 7 cm for a weight of 0.8 kg. Although the above performances fit well the requirement of an active sensor for vehicle ADAS, the cost is a limit. About 8000 USD are required for VLP-16 and more than 10000 USD for the HDL-32E and 64E.
With respect to LIDAR, using RADAR technology, e.g. a FMCW (Frequency Modulated Continuous Wave) operating at 77 GHz, the cost is reduced to about 1000 USD and developments are on going to make this technology affordable at hundreds of USD. As example, adopting the LRR3 (long Range RADAR 3rd generation) sensor technology from Bosch [15], when installed in the front of the vehicle, the Radar FMCW sensor has a range of up to 200 meters, with an accuracy of about 10 cm. The FOV is 30^0 since no rotating elements are used. The power consumption is 4 W in typical conditions, the weight is less than 300 g, the size is about 8 cm x 6 cm.
The cost of the above solutions is still too high for a widespread diffusion in low-/medium-end vehicles, particularly if multiple devices, e.g. multiple RADARs or mixed LIDAR-RADAR units have to be used to cover different views (e.g. front or rear) or different functions. As example, a long-range RADAR is needed for adaptive cruise control in high-speed scenarios (e.g. highway) while a short-range RADAR is needed when parking or for stop&go in urban scenario. Due to the stringent cost requirements of the automotive market, where high-end cars have just a small part of the market share (e.g. they are below 15% of the US automotive market in 2015 [17]) lower cost sensing solutions based on video cameras are needed. The use of cameras for automotive applications is growing rapidly, because among all the car safety technologies that are spreading, those that stimulate the sight are considered the most immediate and reliable from the drivers and are available at lower cost (tens of USD) than RADARS or LIDARS. Both CCD and CMOS image sensors are proposed in the market for automotive applications. CCD sensors typically ensure higher image quality, whereas CMOS cameras allow for single chip integration of the sensing matrix, plus the read-out and digitalization circuitry, and even of some signal processing tasks. The camera-on-chip approach saves space, power consumption and cost, which are key features for the widespread diffusion of vision-based ADAS in low-end and medium-end vehicles. Near Infrared (NIR) detection capability for vision in bad light conditions is also available in CMOS camera, since pixels implemented in silicon have an intrinsically panchromatic response to visible and NIR photons.

3. REAL-TIME MULTI-CAMERA VIDEO PLATFORM ARCHITECTURE FOR ADAS

Innovations for the hardware acquisition and processing platform, and new algorithms are required for several functions building up a complete ADAS solution, such as traffic sign recognition, image enhancement during night and in bad lighting conditions, all around view in parking assistance. Real-time processing is essential in driver warning system to keep low the driver reaction time. Moreover, low implementation cost and low-power consumption has to be guaranteed since the automotive field is a large volume market and energy efficiency is important.
Most of state-of-the-art works miss real-time processing thus involving too high driver reaction time. Those achieving real-time specification, realized using FPGA or specific DSP processors, usually are specific for a single function. Therefore, to build up a complete ADAS multiple hardware-software platforms have to be used increasing system cost.
The proposed system architecture, adopts 4 CMOS cameras operating in the visible range and placed on front, rear, left and right side of the vehicle to achieve a panoramic all around view. Fish-eye lenses, to extend the camera field of view, are also adopted [9]. This way, with only 4 cameras, the proposed platform can ensure all around view avoiding the use of complex panoptic cameras or multiple camera array or rotating devices with cumbersome mechanical and power-electrical parts. A NIR camera, placed on the front side, is used for improved night vision.
The system foresees a first pre-processing processor in charge of all the preparatory tasks needed to several ADAS functions such has:
- color domain transformation since some functions as traffic signs recognition rely on color-based decision algorithms;
- geometrical transformation to correct radial distortions created by the use of fish-eye lenses;
- denoising filter;
- Retinex-based filtering for edge–preserving image enhancement to enhance the contrast of the images (in images with sharp details the targets of interest are easy to be detected by the driver) while correcting the luminance background.

Such pre-processing blocks are placed one for each camera and they work in parallel. The outputs of the pre-processors are then passed to the main signal-processing core that implements the following ADAS functions. Some of these functions combine video from different cameras while other functions are specific to elaborate the video of a certain camera, typically the front-camera or the rear one during parking. The functions implemented are:
- Video mosaicking to fuse the images from all 4 different cameras to provide to the driver, on the dashboard LCD display, a panoramic view of all the obstacles around the car during parking.

- Blind spot elimination: on the dashboard display during a lane change the images from front, left and rear cameras are fused to avoid the blind zone due to the car chassis.
- Motion Detection to detect moving targets and to estimate their direction in cruise control applications.
- Traffic sign recognition, assisting drivers of cars, coaches or trucks, detecting signs they did not notice before passing them. Indeed, in high speed or bad road conditions, drivers sometime do not pay full attention on the traffic signs missing important information for traffic safety. The traffic sign recognition will also support the driver on reading the traffic sign because sometimes drivers, especially when they go abroad, may not know the meaning of the signs.

Such processing core is connected through CAN vehicle network to the vehicle body computer since some functions are automatically activated/deactivated depending on the driver actions: e.g. the all-round view function is activated during parking; the detection of moving targets is activated on the front camera.

The implementing platform integrates mixed reconfigurable hardware resources and a software programmable core. It has been realized using an FPGA Stratix from Altera where a NIOS II soft core is implemented to provide software programmable functionalities while dedicated HDL processors are implemented on the FPGA logic.

To reduce design development time, an IP macrocell reuse strategy has been adopted. We reused already developed algorithms and HDL macrocells we presented in [18-23] for the following tasks: real-time correction of fish-eye lens distortion, Retinex-based edge-preserving image enhancement, de-noising filter able to remove either Gaussian or salt and pepper noise type, Motion Detection to detect moving targets and estimate their direction in cruise control correction. Novel solutions have been proposed for the traffic sign recognition and image fusion in video mosaicking. The former is further detailed in Section 4, while the latter has been detailed by the author in 2016 in [24].

The following figures (Fig. 1) show an example of the results achieved in case of traffic sign detection using a large FOV fish-eye camera where two functions are integrated together:

- Image enhancement and correction vs. low cost large FOV camera, using the technique we presented in [9].
- Image extraction and target recognitions (traffic signs in this case), proposed in Section 4.

The implementing platform has to face the harsh operating requirements of automotive applications in terms of large temperature range, protection against ESD (Electro Static Discharge), over-voltage/current [25-27].

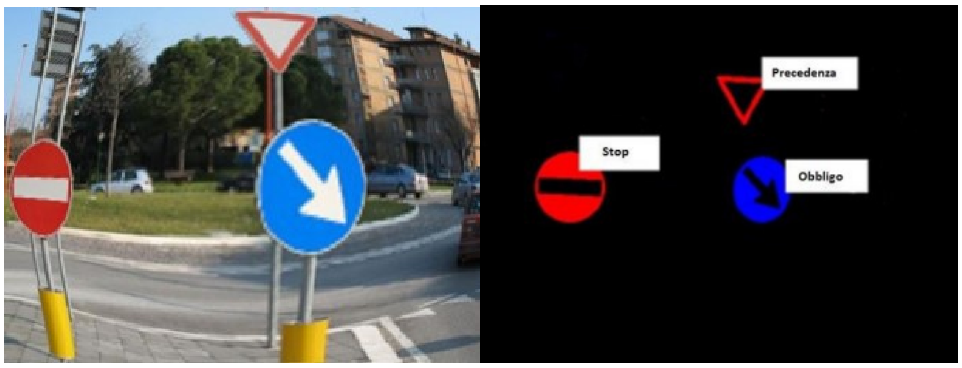

Fig. 1: Input image acquired with fish eye lenses (LEFT) and Output result (RIGHT) of the recognised traffic signs

4. IP MACROCELL FOR TRAFFIC SIGN RECOGNITION

By using the front CMOS camera, the system acquires in real-time the traffic signs that are passed and provide on the CAN vehicle network real-time info about their meaning and visual or acoustic feedbacks to the driver. For example, the sign recognition system can acquire and recognize in real time the speed limit sign and transmit through CAN bus this info to the dashboard controller that will display a warning only if the vehicle exceeds the maximum speed allowed. Even if road signs are designed to be easily readable, with high contrast and saturated colors, and are installed according to a strict regulation, environmental light, weather conditions, paint degradation, dirt, rotating invariance, shadows and occlusions make automatic traffic sign recognition a challenging task particularly if real-time is required.

After the image acquisition through on-board camera, traffic signs recognition is done through 2 main stages: detection and recognition. While some state of art techniques for driver assistance rely on target detection by edge analysis [3-7], often obtained from gray-scale images, in this work to increase the probability of detection and reducing false recognition a target detection exploiting also color analysis is addressed. Differently from other works in the state-of-the-art, using classic camera in RGB color space, the proposed system exploits HSV (Hue, Saturation, Value) color space that is less immune to lighting changes. The RGB to HSV conversion is done in the pre-processing block. To increase

the immunity to light changes in the pre-processing block the images, before being passed to the traffic sign recognition processing core, are filtered through the Retinex-algorithm for luminance correction. The Retinex algorithm implemented in this work is inherited from our previous work in [13], and hence is not further detailed in this paper.
After detection, for classification and recognition step a template-based correlation method is adopted in our algorithm to identify potential traffic signs, of different shapes, from acquired images. Starting from an edge-image, a matching with the template of those signs searched is carried out using a distance transform. These templates are organized hierarchically in order to reduce the number of operations and hence easing real-time processing.
Other works in literature propose the use of Hough Transform to detect circles within an image, but this method is able to detect only circular signs, and more specifically, it has been tested above all on speed signs [14]. Our approach overcomes this limit since the shaped-based template classification we propose ensures optimal performance for several types of traffic signs: circle, triangular, square and octagonal signs.
After traffic sign detection and shape classification for the recognition of the specific traffic sign (e.g. warning and priority sign have both triangular shape and dominant red colour but they have different meaning and can be distinguished by the orientation of the triangle) several techniques can be adopted. Among them, there are extraction of signs characteristics or pattern recognition with the use of neural networks. The latter needs a pre-given set of training images to have a good recognition.
Similarly, with extraction of sign characteristic the goodness of recognition is strictly linked to thresholds that are used to recognize the specific signs from shape, perimeter, area, specific ratios, etc. In our algorithm, this step is done after shapes and color information have been already used to detect and classify the traffic sign type and hence the recognition task is simpler and its performance (correct recognition vs. missed or wrong recognition) is less dependent on the fine-tuning of specific thresholds and hence is more robust.
Summarizing, by exploiting color based analysis our technique provides a faster focusing on the seeking-areas, then a robust traffic sign recognition is done relying first on shape classification and then on specific signs characteristics but limiting the search to those signs having the same shape and the same dominant color.
For the design and test of the algorithm proposed in this paper, and its real-time implementation, both a simulation environment and an emulation one have been adopted. The first has been used during algorithmic design and first system evaluation and is based on photos and images taken from Google® Earth software in Street View mode. The emulation environment has been used to test the implemented hardware-software system with real images acquired with commercial CMOS cameras from Aptina, suitable also for automotive systems, from city center and rural roads. During algorithm design, all the trade-offs between algorithm complexity and detection and recognition performance have been done targeting an acquisition and processing rate of one image each 150 ms. The processing frame-rate is 1 image every 2 meters or every 5 meters for a car traveling at 50 km/h in urban scenario or at 130 km/h in a highway scenario, respectively. Hereafter, we detail the procedure for traffic signs detection and recognition, which is based on five steps:
1. Algorithm initialization and configuration.
2. Color-based image decomposition that alters only red and blue, which are the main colors of traffic signs of interest.
3. Labeling of detected components in red and blue space and false positive reduction.
4. Extraction of key figure characteristics.
5. Traffic signs recognition based on specific rules linked to characteristics obtained in the previous points.
With reference to the system architecture discussed in Section 3, the first two steps are done in the pre-processor while the remaining three steps are implemented on custom HDL macrocells synthetized on the FPGA.

Step 1. Algorithm initialization and configuration.
Before starting the processing, a configuration step is needed to establish the communication between the camera and the unit (which needs loading the images that will be processed in intra mode) and to specify the camera format, frame-rate and resolution, the distance at which the system starts sign detection and to give info on the vehicle speed. Given a certain camera format and resolution, then a sign of a certain physical size will require a number of pixels depending on the distance. Similarly, when the vehicle is approaching a traffic or road sign, frame after frame, the same sign will be displayed with a different number of pixel depending on the frame-rate of the camera and the vehicle speed. The above configuration parameters are used to determine some particular ranges for filter colors and to perform noise reduction.

Step 2. Color-based image segmentation.
After system configuration, this step changes the color space of the loaded image from RGB to HSV and filter the obtained image, leaving only red and blue parts. The color space conversion is performed with a function that calculates for each pixel the values of hue, saturation and intensity in a range of [0, 179] for hue and [0, 255] for saturation and

value. After that, we perform a filtering of colors based on different ranges of hue and saturation. In RGB color space there can be many differences between similar images mainly due to light differences that are instead almost ignored at all with HSV filtering. If a pixel has a hue and saturation value that belongs to ranges specified by minimum and maximum hue, and saturation thresholds for red color, we label it as a red pixel and we build a new RGB image in which red component is put to 255 and green and blue components are put to 0. If a pixel respects blue hue and saturation thresholds, we put the blue component of the new image to 255 and we clear other components. This procedure produces images like those shown in Fig. 2, in which there is the initial image and the image with red and blue component filtered.

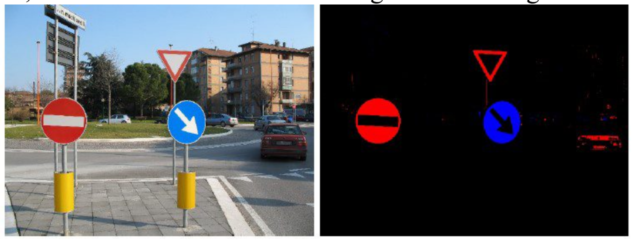

Fig. 2: Example application of the color based image segmentation of a real traffic scenario

Step 3. False positive reduction

There are different methods on how reduce the remaining noise (red or blue single points that can cause false positive) with different trade-offs between performance and complexity. Several approaches were tried: first technique used was a morphological operator like a Dilation operation, but this lead to some problems such as a welding between signs and background. Among morphological operators, also Erosion was tried but it can delete some little parts that can be essential for successive recognition phase. For this reason, a different approach has been adopted based on labeling of components detected from previous stage: given a RGB image extracted as described in previous steps, we scan the image to identify interesting regions that could represent a traffic sign. We analyze all pixels from bottom to top, from left to right, and for each pixel, we consider its 4 neighbors as shown in Fig. 3.

P3	P	
P2	P1	P0

Fig. 3: Pixels to be considered for regions labeling

For each pixel we follow these steps:

- If no one of the neighboring pixels (P0..P3) has a label, it means it's the first of a new region and we give it a new label in a progressive order.
- Else if two pixels have different labels, then P is a pixel that links two different regions: we write this equivalence between these two regions on a vector and we assign pixel P the greater label between the two.
- Else if only a pixel has a label or two or more of the pixels (P0..P3) have the same label, then P will have the same label.

In this way, a label is assigned to each pixel. After this step we have to perform the fusion of two near regions of the same color because they have to represent only one region. First, we scan the equivalence vector and for each equivalence we maintain the greater label. Then, we scan again the label matrix to replace labels consistently with what the equivalence vector reports. At this point, we are ready to delete noise. In fact, for each label we count how many pixels belong to that region and if this value is below a certain threshold called noise_threshold in the code, that region is deleted by putting 0 in all red, green and blue components of the pixels who belonged. In particular, noise threshold is directly proportional to the distance at which we want to perform traffic signs detection because if we are looking for big signs we can delete little regions, but we can not delete them if we are looking for far signs. If we have a high threshold, we can risk deleting signs on the background or at a medium distance. However, have a low threshold allows noise regions to pass to next step for the extraction of characteristics and this implies a big decrease in terms of performance.

Step 4. Key characteristic extraction

For all the regions remained after segmentation and noise removal steps, we extract the following characteristics: color, area, perimeter, object center, radius, distances from border to center, ratio between the area of the region and the area of

the rectangle in which it's inscribed, area under and above center. According to these characteristics, we are able to determine the shape of traffic sign and recognize the type. Given as input the image obtained in the previous steps we compute the following characteristics:
Area: is the number of pixels that belong to that figure.

3	2	1
4	P	0
5	6	7

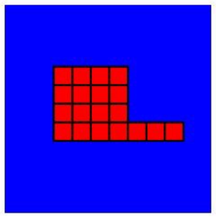

Fig. 4: Patterns useful for perimeter computation (LEFT) and particular case in perimeter computation (RIGHT)

Perimeter: is the number of pixels belonging to the border. To find border we begin with a scan of pixels until a pixel different from background is found; from it the perimeter is calculated this way: we proceed counterclockwise looking for next border pixel as the following table. When we examine a figure, pixels are considered border pixels only if they have not been visited, if they are of the same color of the considered figure and if they are near another border pixel, that is the pixel above, under, immediately at left or at right it's a border pixel. During border scanning a particular event can occur and this has been solved with a tailored procedure: this event is when a figure has an unusual shape, as for example shown in Fig. 4. In this case, we make the mistake to get stuck in the right appendix, because there would not be any other successor. The solution requires saving at each shift the position of previous border pixel and the position for the possible border pixel. If we arrive at this situation, we have to go back in border pixels until we found a new border pixel, because we have not found any background pixel among those adjacent to last border pixel and because we have not finished the perimeter calculus (the first border pixel is not adjacent to last). When the perimeter computation of a figure is completed, we start again scanning the image from where we had stopped until we found a pixel of a color not yet met.
Object center: it's computed as the difference between the minimum and the maximum coordinate on two axis. These four values are calculates during perimeter computation as reported in eq. (1):
Radius: in eq. (2) is the average value between the half-differences of minimum and maximum coordinates on two axis:

$$X_{center} = X_{min} + (X_{max} - X_{min})/2$$
$$Y_{center} = Y_{min} + (Y_{max} - Y_{min})/2 \quad (1),$$
$$Radius = \frac{(X_{max} - X_{min}) + (Y_{max} - Y_{min})}{4} \quad (2)$$

Distances from border to center: Using the algorithm for perimeter computation, we can consider only border pixel and choosing a sampling interval, we compute distances from the center at each interval.
Area below and above the center of the figure: useful to determine the verse of a triangle.
Ratio between the area of the figure and the area of the rectangle in which it's inscribed. This is a strong indicator of an object shape. The area of the rectangle is calculated as the product between the differences of minimum and maximum coordinate on two axis, so the ratio is evaluated according to eq. (3):

$$Ratio = \frac{area}{(X_{max} - X_{min}) * (Y_{max} - Y_{min})} \quad (3)$$

Now we are ready to classify traffic signs according to their shape and to recognize the specific sign according to particular rules and thresholds. First discriminant characteristic is, above all, the color. Second, we must consider the ratio between the square of the perimeter and its area; this ratio has different and specific values for the different classes of signs. Third, we must perform a roundness control and this can be done comparing distances from the center of the figure in the points we have sampled before with the radius of the sign and if all values are almost the same (we have to consider also a little tolerance), the shape it's a circle. On the contrary, if we have decided the sign is a triangle, we must know the orientation. If the area above the center it's greater than that below it means that the triangle basis is on the top and vice versa. Finally, the ratio between the area of the region and the area of the rectangle in which it's inscribed assumes specific values for the cases listed below. It has been used to recognize in particular triangles and octagons. This value is valid also for other signs. Thanks to the above rules the detected traffic sign is then recognized as: Speed Limit Sign; Stop Prohibition Sign; Access Prohibition; Stop Sign; Priority and Warning Sign; Obligation Sign; Traffic Roundabout Sign. In case of the classified speed limit sign, the specific limit is recognized from the analysis of the specific pattern (30, 50, 70, 80, 90, 100, 110, 130) recognized.

5. CONCLUSION

The paper has presented the design and the implementation of a real-time system for image acquisition and processing in Advanced Driver Assisted Systems (ADAS). A multi-camera architecture is adopted to achieve a panoramic view of the environment and objects surrounding the vehicle, avoiding the use of rotating devices and the relevant cumbersome mechanical and power-electrical parts. The acquired images are displayed on the LCD screen mounted in the dashboard. Fish-eye lenses are used to achieve a large Field of View (FOV). Since they introduce radial distortion of the images projected on the sensors, a real-time algorithm for their correction is also implemented in a pre-processor. An FPGA-based hardware implementation, re-using HDL macrocells for several ADAS functions (camera distortion correction, image fusion, traffic or road signs recognition, etc.), allows for real-time processing of VGA automotive camera inputs.

REFERENCES

[1] Okuda, R., et al., “A Survey of Technical Trend of ADAS and Autonomous Driving”, IEEE VLSI-DAT, pp. 1-4, (2014)

[2] Morignot, P., “Arbitration for balancing control between the driver and ADAS systems in an automated vehicle: Survey and approach”, IEEE Intelligent Vehicles Symposium, pp. 575-580, (2014)

[3] Estable, S., et al., "A Real-Time Traffic Sign Recognition System", IEEE Intelligent Vehicles Symp., pp. 24-26, (1994)

[4] Gavrila, D. M., "Traffic Sign Recognition Revisited," Proc. of the 21st DAGM Symposium, pp. 86-93, (1999)

[5] Zheng, Y. et al., "An adaptive system for traffic sign recognition", IEEE Intelligent Vehicles Symposium, Paris, (1994)

[6] Piccioli, G., et al., "Robust method for road sign detection and recognition", Image and Vision Computing, pp. 209-223, (1996)

[7] Gil-Jimenez, P., et al., "Traffic sign shape classification evaluation", IEEE Intelligent Vehicles Symposium, pp. 607-612, (2005)

[8] Hughes, C., et al. “Review of geometric distortion compensation in Fish-Eye cameras”, IEEE ISSC2008, pp. 162-167, (2008)

[9] Turturici, M., et al., “Low-power DSP system for real-time correction of fish-eye cameras in automotive driver assistance applications”, Journal of Real-Time Image Processing, vol. 9, n. 3, pp. 463-478, (2014)

[10] Hughes, C., et al., “Validation of Polynomial-based Equidistance Fish-Eye Models”, IET ISSC 2009. (2009)

[11] Ishii, C., et al., “An image conversion algorithm from fish eye image to perspective image for human eyes”, IEEE/ASME Int. Conf. on Advanced Intell. Mechatronics, pp. 1009–1014 (2003)

[12] Shimizu, S., et al. “Wraparound view system for motor vehicles”, Fujitsu Sci. Tech. J., vol.46, n.1, pp.95-102 (2010).

[13] Marsi, S., et al., “Integrated video motion estimator with Retinex-like pre-processing for robust motion analysis in automotive scenarios: algorithmic and real-time architecture design”, Journal of Real-Time Image Processing, vol. 5, n.4, pp. 275-289 (2010)

[14] Glavtchev, V. et al. “Feature-based speed limit sign detection using a graphics processing unit”, IEEE Intelligent Vehicles Symp., pp. 195-200, (2011)

[15] Bosch engineering, “Long Range Radar LRR3”, doc. N. 92000P0YL-C/CCA-201209-En, (2015)

[16] Velodyne Lidar HDL32, “High-definition real-time 3D LIDAR”, doc. 97-0038 rev. F, pp.1-2, (2015)

[17] Wall street journal, http://online.wsj.com/mdc/public/page/2_3022-autosales.html#autosalesD, accessed Feb. 2016

[18] Fanucci L., et al., “Power optimization of an 8051-compliant IP microcontroller”, IEICE Transactions on Electronics, vol. E88-C, n. 4, pp. 597-600 (2005)

[19] Fanucci, L., et al., “A parametric VLSI architecture for video motion estimation”, Integration the VLSI Journal, vol. 31, n. 1, pp. 79-100 (2001)

[20] Saponara, S., et al., “Motion estimation and CABAC VLSI co-processors for real-time high-quality H.264/AVC video coding”, Microprocessors and Microsystems, vol. 34, n. 7-8, pp. 316-328 (2010)

[21] Fanucci, L., et al., “Parametrized and reusable VLSI macro cells for the low-power realization of 2-D discrete-cosine-transform”, Microelectronics Journal, vol. 32, n, 12, pp. 1035-1045 (2001)

[22] Fanucci, L., et al., “Data driven VLSI computation for low power DCT-based video coding”, IEEE International Conference on Electronics, Circuits, and Systems, ICECS 2002, vol. 2, pp. 541-544, (2002)

[23] Chimienti, A., et al., “VLSI architecture for a low-power video codec system”, Microelectronics Journal, vol. 33, n. 5-6, pp. 417-427 (2002)

[24] Saponara, S. et al., “A real-time and low-cost embedded platform for car's surrounding vision system”, Proc. of the SPIE , vol. 9897, paper 9897_29, (2016)

[25] Costantino, N., et al., “Design and test of an HV-CMOS intelligent power switch with integrated protections and self-diagnostic for harsh automotive applications”, IEEE Transactions on Industrial Electronics, vol. 58, n. 7, pp. 2715-2727 (2011)

[26] Baronti, F., et al., “State-of-charge estimation enhancing of lithium batteries through a temperature-dependent cell model”, IEEE International Conference on Applied Electronics, pp. 29-34 (2011)

[27] Saponara, S., et al., “Sensor modeling, low-complexity fusion algorithms, and mixed-signal IC prototyping for gas measures in low-emission vehicles”, IEEE Transactions on Instrumentation and Measurement, vol. 60, n. 2, pp. 372-384 (2011)