A Hardware Architecture for Real-Time Object Detection Using Depth and Edge Information

CHRISTOS KYRKOU, CHRISTOS TTOFIS, and THEOCHARIS THEOCHARIDES

KIOS Research Center, Department of Electrical and Computer Engineering, University of Cyprus

Emerging embedded 3D vision systems for robotics and security applications utilize object detection to perform video analysis in order to intelligently interact with their host environment and take appropriate actions. Such systems have high performance and high detection accuracy demands, while requiring low energy consumption, especially when dealing with embedded mobile systems. However, there is a large image search space involved in object detection, primarily because of the different sizes in which an object may appear, which makes it difficult to meet these demands. Hence, it is possible to meet such constraints by reducing the search space involved in object detection. To this end this paper proposes a depth and edge accelerated search method, and a dedicated hardware architecture that implements it, to provide an efficient platform for generic real-time object detection. The hardware integration of depth and edge processing mechanisms, with a Support Vector Machine classification core onto an FPGA platform results in significant speed-ups and improved detection accuracy. The proposed architecture was evaluated using images of various sizes, with results indicating that the proposed architecture is capable of achieving real-time frame-rates for a variety of image sizes (271 fps for 320x240, 42 fps for 640x480, and 23 fps for 800x600) compared to existing works, while reducing the false positive rate by 52%.

Categories and Subject Descriptors: B.2.1 [**Arithmetic and Logic Structures**]: Design Styles—*pipeline, parallel*; B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Gate arrays, Algorithms implemented in hardware*; C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems
General Terms: Design, Performance
Additional Key Words and Phrases: FPGA, Parallel Architecture, Object Detection, 3D Vision, Disparity Computation, Edge Detection, Support vector Machines

1. INTRODUCTION

Object detection is an important integrated task in several embedded applications from the computer vision and artificial intelligence domains, and refers to the ability of a computer system to analyze an image and determine the presence of an object of interest. Such embedded applications are often associated with requirements for real-time performance, high detection accuracies and low energy consumption. Additionally, these constraints must often be met under limited available hardware resources. Software implementations of object detection for embedded applications, even if highly flexible, cannot satisfy the above constraints. Consequently, research has focused on the design of custom hardware architectures for object detection. Reconfigurable hardware platforms, such as FPGA, have emerged as a very attractive platform for implementing architectures for object detection applications. FPGAs offer high flexibility with regards to area, power, and performance and thus are able to meet application-specific constraints, which is difficult to achieve with other platforms such as CPUs and GPUs due to their fixed interconnect and high power demands.

Several FPGA-based object detection hardware architectures can be found in the literature. The majority of these architectures, operate on low resolution (320x240) images, and employ a traditional sliding search window approach to search for objects, and also downscale the image several times to find objects of different sizes. However, as the image resolution increases, the number of generated search windows increases as well. Depending on many factors, including the number of scales that need to be searched (from highest to lowest resolution), the overlap between successive windows, and the window size itself, the increase in search space can make it difficult to meet real-time constraints and maintain an acceptable detection accuracy. Software implementations of object detection use search reduction techniques such as motion detection and color processing to reduce the search space. However, only a few hardware implementations feature such search reduction methods that can potentially improve the efficiency of embedded object detection systems [Sadris et al 2004; He et al 2009; Ming et al 2010]. Additionally, some of these techniques, such as color processing, are application specific and thus cannot be used in a variety of object detection applications. Finally, search reduction techniques that have been used in hardware do not give information on the object size and thus exhaustive search must still take place even in a smaller image region.

Alternatively, with the emergence of 3D systems, it is possible to utilize depth information to accelerate object detection. Depth information has been successfully used in software implementations for intelligent object recognition systems [Wu et al, 2009; Darrell et al, 1998; Wang et al, 2004], however, many assumptions and simplifications are made to allow for software implementations to achieve near real-time performance. To the best of our knowledge our initial attempt towards a depth-accelerated object detection system, presented in [Kyrkou et al, 2011a], was the first to consider a fully custom hardware approach that utilizes depth information. That work demonstrated the performance speedups, detection improvements, and energy savings stemming from the use of depth information compared to a conventional sliding-window-based object detection system. Similarly [Kyrkou et al, 2011b] showed the hardware realization and subsequent benefits of an edge-based search reduction method, used both for window size estimation and fast window rejection, again compared to the traditional sliding window approach.

In this work we show how the above two methods can be merged together into a single algorithm and how that algorithm can be implemented in hardware utilizing a dedicated architecture in order to provide an efficient and generic framework for real-

time object detection. Compared to our previous works we perform experiments on larger images (640x480 and 800x600), rather than just 320x240, a common image size often used in object detection systems, to explore the potential and scalability of the proposed architecture. We also provide a more detailed comparison with other hardware implementations of object detection systems, some of which also feature search reduction methods. Finally, realizing that the classification performance is equally important towards an efficient object detection system, we employ the *reduced set method*, [Burges, 1996], to reduce the number of support vectors used by the Support Vector Machine (SVM) classifier, leading in increased classification performance and reduced memory demands. The proposed architecture is implemented on a Virtex 5 FPGA targeting a face detection application, and is able to handle various image sizes of 320x240, 640x480, and 800x600 pixels, achieving 271, 42, and 23 frames-per second (fps) respectively, while also providing a 52% reduction in the false positive rate, compared to the sliding window method.

The rest of the article is organized as follows: In Section 2 we present the basic steps that constitute the proposed depth and edge directed object detection algorithm. Following, is Section 3 which provides details on related object detection algorithms and hardware implementations. The hardware architecture of the proposed method is described in Section 4, while Section 5 details the experimental platform and evaluation methodology for the architecture, along with performance results and discussion, and also provides a discussion on how our framework can be used for other applications. Finally, Section 6 concludes the paper, while also provide directives for future research.

2. DEPTH AND EDGE DIRECTED SEARCH SPACE REDUCTION

Object detection is concerned with identifying the presence or not of an object of interest in an image. This is a tedious task which typically involves a sliding window scanning the input image, and various downscaled versions of it (a process called image pyramid generation), in order to find objects of interest in various sizes. This exhaustive search makes it difficult to meet real-time constraints, especially as the image resolution increases, since more scales will need to be searched for the object of interest and as a result the number of search windows also increases. It is possible to increase the window size as the image size increases in order to reduce the search space, however, in such a case the classifier demands in hardware resources, memory, and processing speed will also increase. As such it is preferable to keep the window size at a considerable small size and introduce search reduction techniques in the overall processes to increase

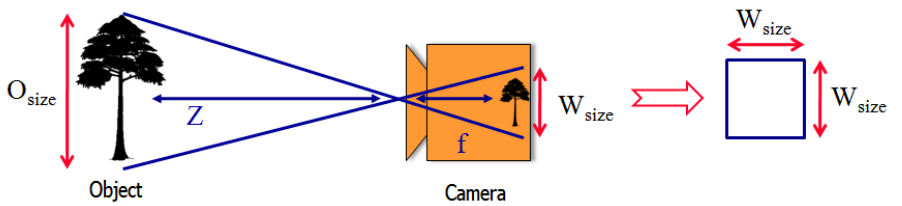

Fig. 1. Window Size Estimation Using Depth Information.

performance. In the remainder of this section we outline how depth and edge information can be utilized to reduce the search space and speedup the object detection process.

2.1 Depth Extraction and Object Size Estimation

Depth information (the distance of an object from the camera) can be extracted from the host environment of the embedded object detection system. There are a number of methods that can be used to extract depth information from the host environment, such as the Microsoft® Kinect™ sensor, or a stereo vision system that processes a stereo image (a pair of left and right images) [Trucco and Vierri, 1998]. In the context of stereo camera systems, information about depth *(Z)* evolves from the computation of the disparity map *d(x,y)*, using the formula $Z = f * (b / d(x,y))$ where *b* refers to the baseline distance between the stereo camera optical centers, and *f* refers to the focal length of the stereo camera system. As such, any stereo-based depth extraction method that can produce the disparity map can work in the context of our framework.

By using depth information extracted from a vision system it is possible to estimate the size of the object at a given location of the image, thus avoid downscaling the input image several times, and subsequently reduce the number of windows that need to be classified. The actual size of the object (O_{size}) as is represented in the real world and its projection in one of the stereo image frames (W_{size}) can be estimated using equation (1) [Trucco and Vierri, 1998]. The equation is derived from the setup shown in Fig. 1, which shows that the ration between the size of the object is the real world (O_{size}) and the distance from the camera *(Z)* equals the ratio between the size in the image (W_{size}) and the camera focal length *(f)*. Additionally, using the relationship between depth and the disparity map *(d)*, that applies for stereo vision systems, we can use the disparity value instead (2), thus, avoiding the need to compute the actual depth.

$$(W_{size} / f) = (O_{size} / Z) \Rightarrow W_{size} = f * (O_{size} / Z) \quad (1)$$

$$Z = f * (b / d(x,y)) \Rightarrow W_{size} = (O_{size} / b) * d(x,y) \quad (2)$$

2.2 Edge-Based Window Rejection Process

The performance of object detection systems also suffers from the necessity that all windows need to go through the classification process. Thus valuable computational time as well as power are wasted on potentially non-promising regions. An efficient way to eliminate windows prior to the classification process, in a manner that can be parallelized in hardware and does not require many hardware resources, is by utilizing edge information. Edges provide information about visual features in an image and thus the number of edge pixels in an image can give an indication of the useful information in a particular image region. An early approach to eliminate windows from the classification process, in addition to constructing the windows, was first proposed in [Anilla and Devarajan, 2010]. Overall, the edge-based window rejection process involves obtaining the edge image, using an edge detection algorithm such as Sobel, scan the image, and reject windows depending on the number of edge pixels. Specifically, after obtaining the edges of a particular image region (window), the edge pixels are counted and if they exceed a certain threshold, which is assigned according to the object of interest, the window is considered for classification; otherwise it is not classified and considered as a non-object. In [Anilla and Devarajan, 2010] a window is discarded if it contained no edge pixels. However, the zero edge pixels threshold is susceptible to noise and is only able to remove regions with almost uniform intensity, and so in our approach we use a larger threshold.

2.3 Depth and Edge Accelerated Object Detection Process

An overview of the proposed accelerated object detection approach is given in Fig. 2. The approach relies on the computation of the disparity map from a stereo image to extract depth information. Each value in the disparity map corresponds to the center of a candidate window. The candidate window size is determined by (2) and a candidate window is formed around each disparity value. The disparity map is sampled every few pixels (disparity search overlap) and a candidate window is extracted for the sampled disparity values only, rather than for every pixel in the disparity map. Furthermore, if the size of the candidate window exceeds the image boundaries, indicating the presence of a large object at the border of the image, that window can be discarded as the object may fully fit in another candidate window. Finally, any disparity values that map to window sizes which are smaller than the size of the classifier can also be discarded, in the premise that they wouldn't be considered for classification in the sliding window approach either. After the size of the candidate window has been validated, that window is extracted from

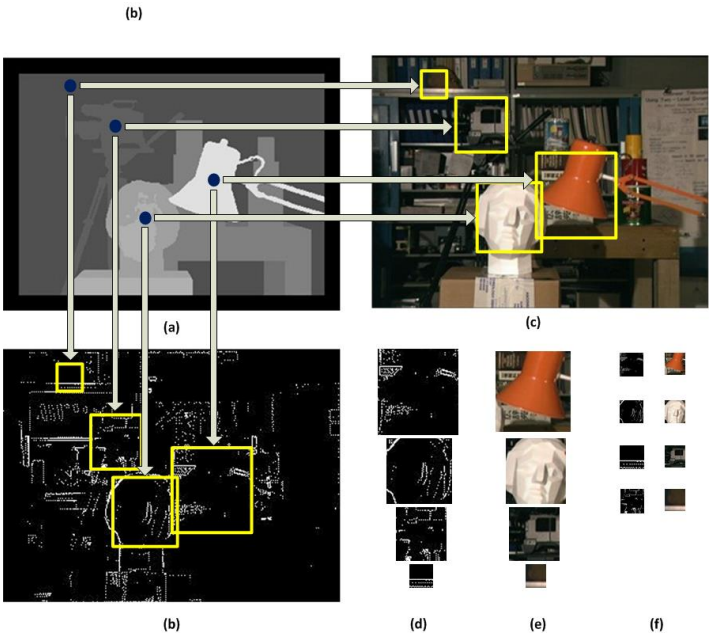

Fig. 2. Object Detection Using Edge and Depth Information: (a) The disparity map is sampled every few pixels. Each disparity value corresponds to the center of a candidate window. (b) If the window size is valid then the edges of that window are extracted from the edge image. (c) Corresponding window sizes in the original image. (d) Extracted edge windows. (e) Corresponding extracted windows from the original image. (f) Resized windows used for the calculation of the mean number of edge pixels and classification respectively.

the edge image, as show in Fig. 2, and the number of edge pixels contained in the window is determined. If it exceeds a predetermined threshold the candidate window is considered valid, and that window is extracted from the grayscale image, resized to the classifier size, and is then classified by the classification algorithm. Otherwise, it is discarded and the whole process is then repeated for the next candidate window.

3. RELATED WORK

In recent years, a fair amount of work has been done in development of algorithms and techniques used in 3D vision systems. In [Hetzel et al, 2001 for example, the authors use local feature histograms extracted from range images to recognize single object images. The work in [Browatzki et al, 2011] uses 2D and 3D features from color and depth images respectively, in order to recognize objects. Affine feature finders are combined with SIFT in [Moreels and Perona, 2005], in order to provide robust 3D recognition under viewpoint changes, and lighting and scale changes. More recently efficient 3D feature extraction algorithms have been proposed in the literature such as the

Fast Point Feature Histograms (FPFH) used for 3D image registration [Rusu et al, 2009], and Normal Aligned Radial Features (NARF) that are used to find and recognize 3D objects in range images [Steder et al, 2010]. Finally, other approaches such as in [Liebelt et al, 2008], use generated 3D object representations to recognize different objects. Common classification methods that have been used in 3D object recognition methods include nearest neighbor methods [Liebelt et al, 2008], multilayer perceptrons [Browatzki et al, 2011], and support vector machines [Roobaert and Van Hulle, 1999; Ruiz-Llata and Yebenes-Calvino, 2009]. The main focus of these works was on providing higher recognition accuracy that could be used in a generic framework for 3D object recognition. However, hardware implementations of such 3D object recognition systems have been rather limited to simple problems with low-resolution images [Ruiz-Llata and Yebenes-Calvino, 2009]. Nevertheless, the advancements in the research of 3D vision systems provide an opportunity to utilize additional available information, such as depth, in order to accelerate 2D object detection systems by reducing the search space.

A fair amount of work has been done in hardware implementations of 2D object detection algorithms, mostly utilizing FPGAs and targeting face detection. These works employ pattern recognition algorithms for the classification stage, such as neural networks, Support Vector Machines, and the Viola-Jones detection algorithm, and utilize the sliding window approach to search for objects of various sizes, while most works targeted 320x240 images The following paragraphs illustrate some of the most important works found in the literature while Table I provides a summary of the techniques used in each work and achieved performance.

One of the first attempts at implementing face detection in hardware was the work in [Rob McCready, 2000], and it was the first to demonstrate the benefits from a dedicated hardware solution. The authors designed a custom face detection algorithm based on a neural network classifier and optimized it for the TM-2 (Transmogriffer 2) configurable multiboard FPGA platform. It operated on a 320x240 image frame and achieved a frame rate of 30 fps while having a detection accuracy of 87%. However, the proposed implementation utilized nine boards on the TM-2 system.

The detection framework by Viola and Jones [Viola and Jones, 2001] is a popular approach used for 2D object detection. The main benefit of this algorithm in terms of hardware implementation is that it only requires additions and only a few multiplications making it attractive for resource-constrained systems. Hence, a few hardware implementations of this algorithm can be found in the literature. The work from [Cho et al, 2009] showed a parallel implementation of this algorithm on an FPGA, demonstrating

Table I. Summary of FPGA implementations of object detection systems

Work	Algorithms	Image Resolution	Classifier Window Size	Frames Per Second	Accuracy
He [2009]	Skin Detection, Motion Detection Haar-Features and Neural Network	320x240 downscaled to 80x60	11x11, 19x19, 27x27	600	n/a
Han [2011]	Modified Census Transform, Viola Jones	320x240	n/a	149	99.76%
Sadri [2004]	Neural Network, Skin Detector	800x600	18x22 or 36x44	9	n/a
				90	
Che [2010]	Skin Detection, Integral projections	640x480	n/a	10	n/a
Cho [2009]	Viola-Jones Detection Framework	320x240	20x20	23	n/a
Hiromoto [2009]	Viola-Jones Detection Framework	640x480	24x24	30	n/a
Kyrkou [2011a]	Viola-Jones Detection Framework	320x240	24x24	64	93%
Farrugia [2009][1]	Convolutional Neural Network	320x240	32x36	127	TP: 87%
				29	
		640x480		35	
McCready [2000]	Neural Network	320x240	30x32	30	87 %
Proposed Work	Depth Extraction, Edge Detection, Support Vector Machines	320x240	20x20	271	TP: 82% FP: < 1%
		640x480		42	TP: 82% FP: < 1%
		800x600		23	TP: 80% FP: < 1%

[1] 127 and 35 fps are achieved with a 25 PE ring on a Virtex 5 LX330
TP - True Positives, FP - False Positives
n/a - not available information

near real-time performance of 23 fps for 320x240 images. In [Hiromoto et al, 2009] a hybrid model is proposed that consisted of parallel and sequential modules. The most frequent parts of the algorithm are implemented by the parallel modules, while the least frequent by the sequential. Through this approach the authors manage to save hardware resources while achieving 30 fps. Finally, the work in [Kyrkou and Theocharides, 2011a] proposes a systolic architecture for the implementation of the Viola and Jones algorithm both for ASIC and FPGAs, with the FPGA system able to process images at 64 fps.

Han et al [2011] proposed a face detection system that utilizes the Modified Census Transform (MCT) and the AdaBoost learning algorithm with cascaded classifiers targeting mobile environments. They use only a single classification stage and their system can detect up to 32 faces. Their system achieves high frame rates and high detection accuracies, however, the FPGA architecture and FPGA utilization information is not presented making it difficult to assess their proposed system.

The hardware implementation of a Convolutional Face Finder [Garcia and Delakis, 2004] based on a convolutional neural network, was demonstrated in [Farrugia et al, 2009]. The proposed architecture consisted of a ring of processing elements (PEs) that implement the CFF algorithm and demonstrated how pipelined architectures can be used to speedup the detection process. They exploit parallelism by dividing the image in vertical strips with overlapping regions and each PE processes a block of that strip. However, a 25 ring PE is required to achieve a real-time performance on 640x480 images, and consequently a large FPGA is used to fit the architecture. Furthermore, not enough details are given on the I/O requirements of their architecture, and the memory and buffering requirements.

The aforementioned works have demonstrated how dedicated hardware solutions and classifier optimizations in hardware can provide high speedups and real-time performance. However, higher frame-rates can only be achieved by integrating hardware search space reduction mechanisms. The following works demonstrate different methods and approaches that have been implemented in hardware in order to reduce the search space.

Che et al [2010], use a hardware/software approach to design an FPGA-based face detection system with skin detection acceleration. A Nios embedded soft-processor handles the face identification task while a dedicated hardware accelerator handles the skin detection process. This specific work focuses on accelerating the search space reduction in hardware rather than the actual classification. The processor can then handle the classification process since the search space is reduced. However, this also depends on the input image size.

Sadri et al [2004] implemented a face detection neural network algorithm on a Virtex II Pro FPGA, which included a skin color filter to reduce the search space within the image, and an edge detection mechanism to produce a binary image on which the neural network operates on. The majority of the system was implemented on the FPGA custom logic fabric while higher level operations were left to the Power PC processor. Their system operated on 800x600 input images with a frame rate of 9 fps if the entire image was processed. The resulting frame rate could be improved up to 90 fps if only 25% of the image was searched, using skin detection, and only up frontal detection was considered. The approach of using binary image data demonstrates the potential performance benefits; however, the impact on detection accuracy is unclear.

Finally, the work by He et al [2009], demonstrated the hardware implementation of a massively parallel face detection system that achieved frame-rates of over 600 frames per

second. They utilize two search reduction techniques, motion detection and skin detection. They also make a few simplifications on the face detection procedure and adding two search reduction techniques. The input image is of 320x240 pixels, however, all subsequent operations including the detection process happen on a downscaled 80x60 image, greatly reducing the search space, while only considering three face sizes (11x11, 19x19, and 27x27). The classification for the three window sizes is done in parallel and thus the proposed system needs a lot of resources to provide these high frame rates. This approach may not be suitable for other object detection applications as downscaling the input image may result in loss of quality.

The above works have demonstrated the successful implementation of face detection systems on FPGAs. The majority of these implementations have either adopted the sliding window approach, or integrated face-detection-specific search reduction techniques such as skin detection. The majority of these works have targeted image sizes of 320x240 pixels for face detection, however other applications may require processing higher resolution images. Recognizing that there is a need for a more generic object detection framework, and with the advancements in the field of 3D vision systems we propose an object detection architectural framework that is based on depth information, as well as edge detection, and can provide a more generic platform for various detection applications, and can also process larger image sizes.

4. PROPOSED HARDWARE SYSTEM ARCHITECTURE

The architecture that implements the proposed object detection process consists of four major hardware units, each implementing the major tasks of the algorithm as outlined in Section 2. These units are the Disparity Computation Unit (DCU) which computes the depth information from a stereo image, the Edge Computation Core (ECC) that implements a Sobel edge detector, the Window Extraction Unit (WEU) that processes the depth and edge information, and the Classification Engine (CE) which implements a Support Vector Machine classifier. It is also possible to use different approaches and algorithms for each of these units, however, the overall architectural framework will remain the same. The system also consists of a memory controller and a system controller, that optimize accesses to the external memory, control I/O operation, and synchronize the other major units. The system uses three on-chip buffers to store the image data; there are dedicated buffers for the edge image, the disparity image, and the left image of the stereo pair, which is used as the reference image for the disparity computation. Fig. 3 shows an overview of the system architecture and the communication

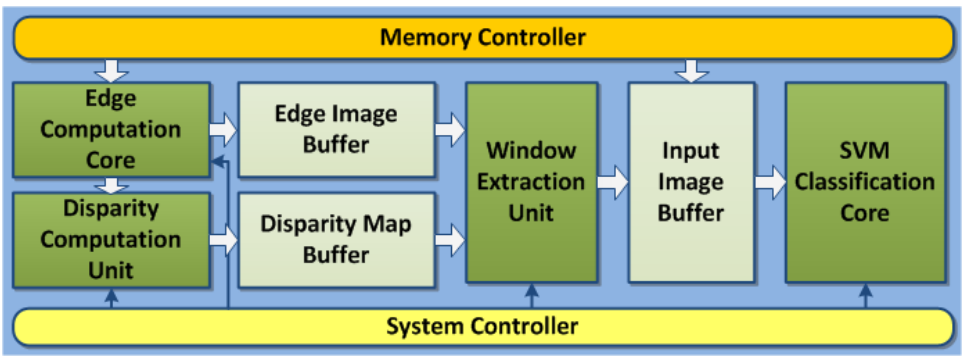

Fig. 3. System Architecture.

flow between units. Certain features of the architecture were optimized for FPGA implementation, however, the architecture can also be implemented using an ASIC design flow with only minor adjustments.

The detection operation begins when the memory controller fetches stereo image data from the external memory to the ECC in raster-scan fashion, and stores the incoming pixels of the left stereo image in the *input image buffer*, while the produced edge image is stored in the *edge image buffer*. The DCU reads the edge image pixels and performs the disparity map computation, and stores the disparity pixels in line buffers (*disparity image buffer*). The WEU reads pixels from the line buffers to validate the candidate window and extract window pixels from the edge image and the left stereo image. All valid windows are then fed to the CE for classification.

4.1 Edge Computation Core

The Edge Computation Core (ECC) integrated to the system implements a flexible and scalable Sobel edge detection architecture. It employs hardware features such as parallelism and pipelining in an effort to speedup the repetitive calculations involved in the Sobel operation, and uses optimized memory structures in order to reduce the memory reading redundancy. This is particularly important, as the time for edge detection must be small enough in order to obtain a speedup in the overall operation.

The architecture of the ECC is shown in Fig. 4 and consists of an I/O controller, a set of FIFO line buffers used for temporary pixel storage and parallel window processing, and a series of convolution units (CONV), as well as comparators. The I/O controller fetches pixel data from the input stereo image in a row-wise fashion, and forwards them to the input port of the scan-line buffers. The scan-line buffers produce four successive 3×3 windows per cycle, which are convoluted with the 3×3 Sobel kernels by the convolution units. The architecture utilizes two edge detection units, one for the left and

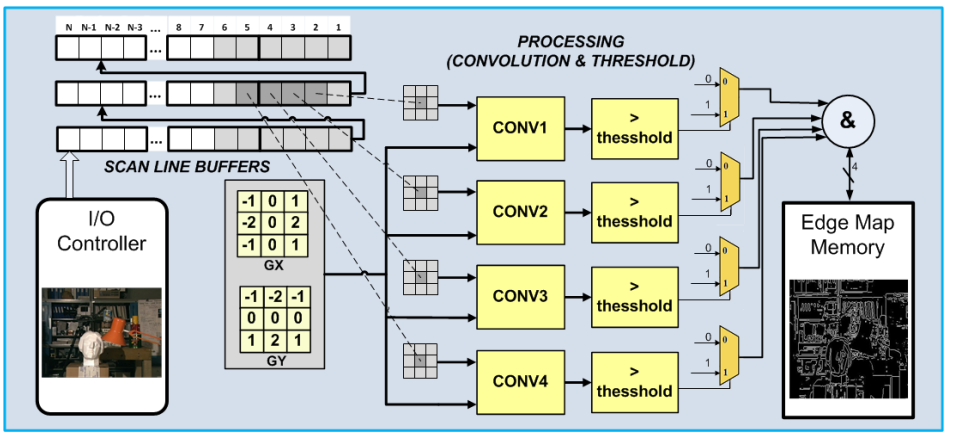

Fig. 4. Edge Computation Core

one for the right input image. The edge map of the left stereo image is stored in the *edge image buffer* and is used by the WEU, in the latter stages, to determine if a candidate window should be rejected prior to classification or not. Additionally, the produced edge detection images are also used for the fast computation of the disparity map as described in [Hadjitheophanous et al, 2010].

4.2 Disparity Computation Unit

The Disparity Computation Unit (DCU) used in the architecture extracts depth information from a stereo vision system, and its architecture is based on an improved version of the hardware architecture that was proposed in [Hadjitheophanous et al, 2010]. The architecture, which is shown in Fig. 5, combines the sum-of-absolute-difference (SAD) matching algorithm with edge features for faster and more efficient processing. Specifically the use of edge features reduces the hardware demands since processing happens with 1-bit pixels rather than 8-bit for grayscale images. Thus the area saved is used to increase parallelism and performance. This is significant for the efficient integration of this unit into an object detection system since the overhead introduced must not affect the performance of the classification process.

The DCU follows the ECC in the computation flow as it receives edge pixels from the ECC and uses them to perform correlation on the input stereo images to produce the disparity map. The DCU can process images with a disparity range of 80 pixels, and window sizes up to 11x11 pixels. The DCU architecture consists of scan-line buffers in order to receive edge pixels from the ECC in streaming fashion. The scan-line buffers temporarily store the pixels and allow for parallel processing of many windows. As such multiple parallel adders and subtractors are utilized which facilitate in the parallel

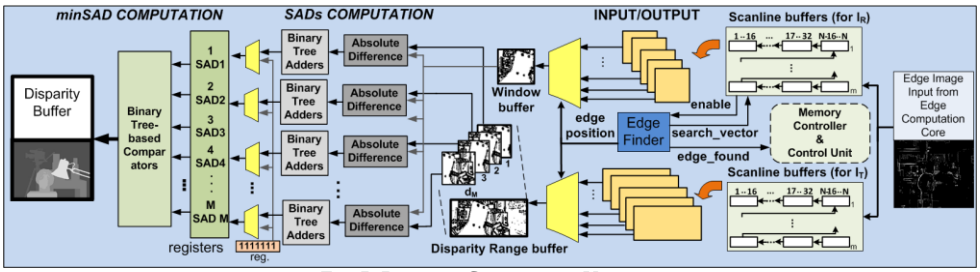

Fig. 5. Disparity Computation Unit

implementation of the SAD algorithm. Through this parallel structure the DCU is capable of producing disparity values every cycle. The disparity values are produced faster that they are consumed by the WEU and CE, thus it is only necessary to store a couple of lines of the disparity map in the *disparity image buffer*. More disparity map pixels can be produced very rapidly and so there will always be data available for the WEU which follows the DCU in the computation flow. There are additional advantages from activating the DCU only when disparity values are needed by the WEU. First energy is saved since the DCU is not active for a large amount of time, unless a lot of disparity values are not valid in which case the DCU will constantly produce disparity values. However, in that case the CE will not be active again resulting in energy savings. Second we reduce the on-chip memory requirements for storing the disparity map and use the remaining memory resources to store the input image. The reader is referred to [Hadjitheophanous et al, 2010; Ttofis et al, 2012] for a more detailed description of the concepts used in the design of the DCU.

4.3 Window Extraction Unit

The Window Extraction Unit (WEU), shown in Fig. 6, is the third unit in the detection system pipeline and performs the necessary operations for the validation of candidate windows, which are the window size estimation and the accumulation of edge pixels, additionally it also performs the reverse mapping process that rescales large windows to the appropriate window size for the classifier (*m x n*, can be either square or rectangular). The WEU is enabled once the DCU starts generating and storing disparity values in the *disparity image buffer*. The disparity map is sampled by the WEU every five pixels and thus not all values of the disparity map are processed. The WEU loads them from the buffer and proceeds to determine the corresponding window size of each disparity value using equation (2). To implement the equation the incoming disparity value is multiplied by the fixed point preloaded value (O_{size} / b), producing the window size in pixels. The calculated window size goes through some comparators that check if the size is equal or

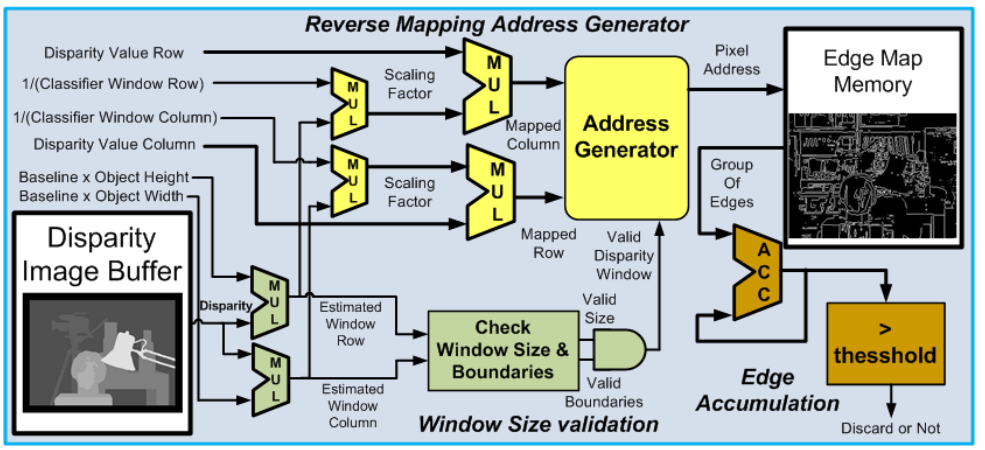

Fig. 6. Window Extraction Unit

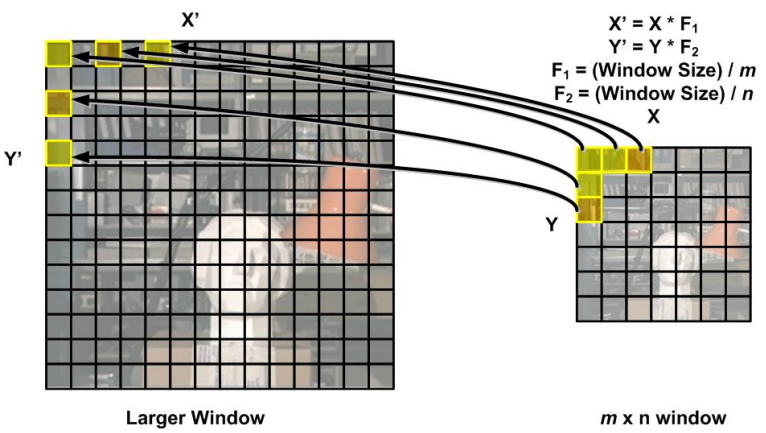

Fig. 7. The reverse mapping process involves reading only the pixels that correspond to an $m \times n$ window on a larger image. Each pixel address in the $m \times n$ window is transformed into an address corresponding to the pixel's location in the larger image using scaling factors F_1, F_2 which are computed by the larger window dimensions divided by the classifier window dimensions.

larger than the classifier window size, and if the window is within the image ranges. If the conditions are met then the corresponding window is extracted from the *edge image buffer*, using the reverse mapping process, and the number of edge pixels in the window are counted. The reverse mapping technique is implemented by multiplying the actual window coordinates with a predetermined scaling factor (*computed window size/ classifier window size*). This ensures that for every window, regardless of the estimated window size, only $m \times n$ pixel values will be read and processed by the subsequent stages. Details on the reverse mapping process are given in Fig. 7. The edge image pixel values are either 1 or 0, hence, a simple accumulation will suffice to give us the number of edge pixels in the window. Storing and processing of the edge image pixels happens in groups

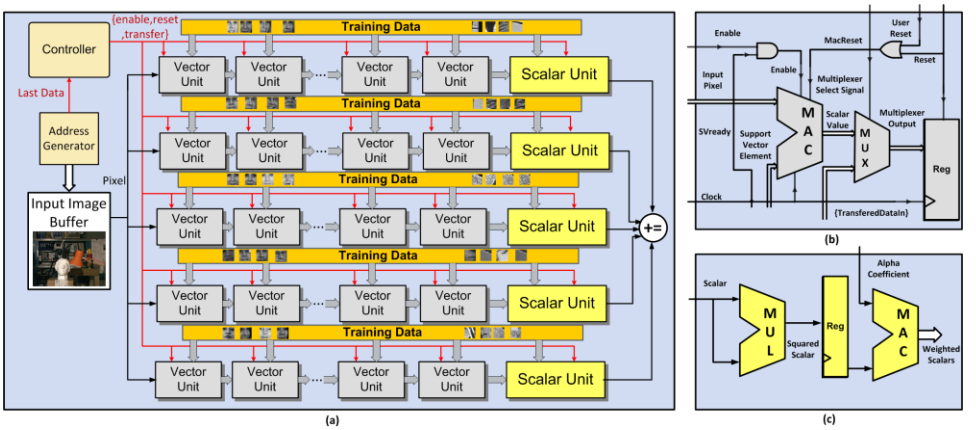

Fig. 8. (a) Support Vector Machine Classification Core (b) Vector Unit (c) Scalar Unit

of four bits to improve performance, and as such the accumulation is done for 4 edge image pixels as well. Once all the pixels are accumulated, the sum is compared to a threshold to determine if it should be discarded or not. If the threshold is exceeded the WEU starts fetching window pixels, this time from the *input image buffer* using the reverse mapping technique, to the classifier.

4.4 Classification Engine

The CE architecture is based on the SVM processing architecture proposed in [Kyrkou and Theocharides, 2011b]. It consists of an array of processing elements (PEs) and reflects the processing requirements of the SVM computation flow which requires the calculation of a kernel function that has both vector and scalar operations. The architecture consists of two types of PEs which perform the vector and scalar operations of the SVM computation flow (the vector and scalar units). The architecture proposed in [Kyrkou and Theocharides, 2011b] permits the SVM processing core to handle the processing of multiple windows as each row can perform classification of one window. Additionally, it allows trading-off the processing the training data in parallel, or processing input windows in parallel. The latter, however, depends on the memory I/O and parallel access capabilities.

We have optimized that architecture for the specific problem at hand. Specifically, in the case of the developed system where we use dual port on-chip memories available on the FPGA for storing the input image, with one port used for writing while the other is used for reading, only one window can be accessed at a time. As such we adapt the SVM classifier architecture in order to speed up the classification of a single window by processing as many training data as possible. The modified array architecture, which is

shown in Fig. 8, consists of 5 rows each processing the same window but with different reduced-set-vectors (training data) thus increasing performance. The results of the each row's vector units are accumulated by that row's scalar unit. In turn the result of each scalar unit is also accumulated to produce the final classification result. Using this approach a 20x20 window can be classified in 445 cycles. This makes the classification process the bottleneck in the overall computation since the ECC and DCU can produce results almost each cycle, based on their developed architectures. Hence, unless windows are discarded, there will almost always be data available to be processed by the CE.

5. EXPERIMENTAL METHODOLOGY AND RESULTS

5.1 Experimental FPGA Platform and Methodology

The proposed architecture was evaluated after implementation on a Xilinx ML505 board (Virtex 5-LX110T FPGA), targeting the application of face detection as benchmark. The evaluation image data set consisted of real world images that were taken from a custom built stereo vision system, as well as synthetic images. The stereo image capturing system was comprised of two video cameras separated by a baseline distance of 77mm, both with a focal length of 25mm. Stereo image pairs, were loaded to the FPGA board DRAM from the compact flash, through a Microblaze subsystem that was used for initialization and verification purposes. After the initialization phase, the object detection architecture was enabled and data was retrieved from the DRAM by a custom memory controller. A total of 20 images per size (320x240, 640x480, and 800x600) were used to evaluate performance metrics such as frame-rate and detection accuracy. Visual output was directed to a digital monitor, through the on-board DVI output. Classification was performed by a support vector machine which was able to classify 20x20 images. The kernel used for the SVM was a 2^{nd} degree polynomial ,which was found to perform well for image processing applications [Osuna et al 1997]. The SVM model was trained using MATLAB and the databse in [CBCL face database #1, 2000] using the methodology and strategies employed in [Osuna et al, 1997; Burges, 1998], producing a total of 80 reduced-set-vectors [Burges, 1996]. Table II summarizes the system algorithmic parameters.

5.2 FPGA Implementation Discussion

The system architecture was implemented on a Virtex 5 LX110T FPGA and was optimized for the specific FPGA hardware features and resources in order to be able to

Table II. Algorithmic parameters for face detection

SVM Kernel	2nd degree polynomial $((x \bullet y) + 1)^2$	Number of Reduced Set Vectors	80
Reduced Set vector Encoding	8-bits	Alpha Coefficients Encoding	10-bits
Disparity Window	11 x 11	Disparity Range	80 pixels
Minimum Number of Edge Pixels Threshold	50 pixels	Edge Detection Threshold	175 pixels
SVM Classification Window Size	20x20	Disparity Search Overlap	5 pixels

process images of up to 800x600 pixels. Specifically, the image buffers are implemented as dual port block rams, which are available on the FPGA, to facilitate the streaming nature of the operations. One port is used for writing of image data and the other for reading. A total memory space of 800x240 pixels was allocated for buffering the input grayscale image. As such a whole image of 320x240 can fit on the FPGA, for larger images (640x480 and 800x600) only a part of the image is stored on-chip. This is sufficient as in most cases the window will be available on the on-chip memory and as a result external memory access will not be needed. Furthermore, whenever the window pixels are not on-chip this will indicate the presence of a very large object, which covers most of the image, as such a large portion of the image will not need to be processed, and so the overhead from accessing the external memory is negligible. The Sobel convolution process of the ECC unit was implemented using shift registers rather than multiplications, to save hardware resources and increase frequency. The SVM array was comprised of a total of 80 vector units (5 rows and 16 columns), with each vector unit handling the processing of the input window with one reduced-set-vector. A total of 5 scalar units were required for the implementation of the SVM computation flow. All the units in the SVM processing core require multiplication and accumulation units, however, since the scalar units have a higher demand in bitwidth requirements, they were mapped on the DSP48E units of the FPGA for performance improvement, while the vector units where mapped on the LUT logic.

The FPGA resource utilization of the proposed system is illustrated in Table III, which also shows relevant results from related works. The FPGA technology used plays an important role in the efficiency and performance of a design. A feature-rich FPGA platform that provides more processing power, in the form of more reconfigurable logic, embedded multipliers and embedded Block RAM, to further exploit parallelism can potentially provide higher performance. However, the architecture design has to be scalable and utilize the FPGA resources efficiently in order to deliver the full

Table III. FPGA Related Work Synthesis Results

Work	FPGA Platform		Logic Elements		Embedded Multipliers / DSP48E	Block RAM / Embedded Memory	Operating Frequency (MHz)
			Slice Registers	Slice LUTs			
He [2009]	Xilinx FX130T Virtex 5 ML510 board		37,828 (46%)	67,704 (83%)	161 (50%)	276 (93%)	73
Han [2011]	Virtex 5 LX330		n/a	n/a	n/a	n/a	54
Sadri [2004]	XC2VP20		n/a	17,000 (74%)	n/a	n/a	200
Ming [2010]	Altera Cyclone II EPC2C70		9679		n/a	856605 bits	100
Cho [2009]	Virtex-5 LX155		21,902 (22%)	84,232 (86%)	7 (5%)	97 (50%)	N/A
Hiromoto [2009]	Virtex 5 XCVLX330		55,515 (26%)	63,443 (30%)	n/a	n/a	160.9
Kyrkou [2011a]	Xilinx Virtex II Pro		23,744 (86%)	25,818 (94%)	68 (50%)	24 (17%)	100
Farrugia [2009] [1]	Virtex 4 SX 35 and Virtex 5 LX330		2,466 (16%)		19 (9%)	8 (4%)	80
McCready [2000]	Transmogrifier 2A Altera 10K100		31500		n/a	n/a	12.5
Proposed Work [2]	Virtex 5 LX110T ML505 Board	SVM Core	7,128 (10%)	13,555 (19%)	45 (70%)	46 (31%)	100
		Edge Detection	20,868 (30%)	2,812 (4%)	---	3 (2%)	
		Disparity Unit	1,008 (1%)	31,504 (45%)	---	---	
		Other	8,438 (13%)	8,316 (12%)	3 (5%)	87 (58%)	
		Overall System	37,442 (54%)	56,020 (81%)	48 (75%)	136 (91%)	

[1] Synthesis results for one PE on a Virtex4 SX35
[2] Capable of processing images of up to 800x600. Less resources are needed for smaller images.
n/a - not available information

performance potential of a given FPGA platform. As such, comparing architectures implemented using different FPGA technologies, even if it may be indicative of potential performance and required hardware, may not be fair. Thus, we focus our comparison with works that have used the Virtex 5 FPGA technology, however, we also include other works in Table III for a more comprehensive view. The reported figures for our implementation are for a system capable of processing images of up to 800x600. Less resources will be needed if the system is only going to process lower resolution images. Our proposed architecture requires less LUT resources than other works, and only half of the available register resources of the targeted FPGA for the implementation of the architecture. The CE requires the majority of the DSP48E units which can be reduced by reducing the number of rows in the classifier, possibly reducing, however, the classification performance per window. Architectures that use the Viola-Jones detection algorithm have an additional advantage that they do not require many multiplier units

since the algorithm is mostly implemented with adders/subtractors and accumulators. On-chip storage is critical for reducing external memory I/O, and increasing performance, as also demonstrated in [He et al, 2009]. Hence our system utilizes the majority of available FPGA block rams to reduce external I/O. We targeted the frequency offered on the available FPGA board and as such we pipelined the critical paths of the design to achieve a frequency of 100 MHz. The frequency can be improved with further optimization for higher throughput. With this frequency the system manages to offer very good performance results.

5.3 Performance Results and Discussion

Performance in object detection systems is measured in terms of frame-rate and detection accuracy. A real-time performance of around 30 frames-per-second (fps) is sufficient for most video processing applications, however, applications with multiple data streams and high resolution video analysis may require higher frame rates. Through the speedup gained by the search reduction techniques the FPGA implementation of the proposed depth and edge directed object detection system is capable of high performance for 320x240 and 640x480 images (271 and 42 fps respectively) , while achieving near real-time performance of 23 fps for 800x600 images. The average frame-rate achievable by our system and other systems is shown in Table I. The bottleneck in the performance of our system is the CE, which takes a few hundred cycles to classify a window (depending on its size), whereas the ECC because of the parallel window operations, and the DCU because of the binary data processing produce, a result almost every cycle. Thus they are capable of achieving over 100 fps for all targeted image resolutions as demonstrated in [Ttofis et al, 2012]. The achieved performance is then limited by the number of windows that need to be classified, and the classification time per window. The work from He et al [2009] achieved 600 fps for 320x240 images, however, the actual size of the processed image size is 80x60 with only three object sizes considered. The FPGA system by Sadri et al [2004] can process 800x600 images in 90 fps, however, this is achieved if only 25% of the image is searched through skin detection, and the detection itself happens on binary images, and thus is unclear how detection accuracy is affected.

The depth of objects, the number of edge pixels in the image, and disparity map sampling step, are all factors that affect the frame-rate. Depending on the depth of objects and the number of edge pixels in the input stereo image the frame-rates may be a bit higher or lower. Coarser grain sampling of the disparity map can further improve performance as fewer windows will be generated. An increase of the disparity search

overlap for 800x600 images from 5 pixels to 10 pixels can potentially double the performance without any significant loss in detection accuracy. Increasing the disparity search overlap can enable the architecture to also process larger image sizes than 800x600 in real-time. The proposed system architecture was tailored to the available FPGA resources and as such there are a few limitations that can be overcome by investigating an ASIC design especially for higher resolution images. Some of the things that can be explored through an ASIC implementation are the design of on-chip memory structures that will allow higher pixel throughput and optimizations on the CE architecture to increase classification performance per window. Finally, it will be possible to instantiate parallel CEs to process multiple windows in parallel.

Detection accuracy is an equally important performance metric for object detection systems and it heavily depends on the classification algorithm used as well as the training data. The Viola-Jones detection algorithm has demonstrated very good results in terms of detection accuracies for 2D object detection, however, SVMs have also shown very good results and literature suggests that the detection results are comparable [Osuna et al, 1997], and furthermore, SVMs have also been used for 3D object recognition [Roobaert and Van Hulle, 1999; Ruiz-Llata and Yebenes-Calvino, 2009]. Detection results of our system and other related works is shown in Table I. A direct comparison of the detection accuracies, however, is not fair since the image datasets used in other works are comprised of a single image and as such we cannot use them in our stereo processing system. Additionally, the training data, and preprocessing methods also impact the classification performance. The proposed depth and edge accelerated system can achieve classification rates of ~80% which would suffice for most applications. The detection accuracy is a bit lower than the Viola and Jones face detection implementation in [Kyrkou and Theocharides, 2011a], however, higher classification rates are possible by improving the training data set. An inherit advantage of the proposed depth and edge search reduction approach comes from the fact that the number of windows that are processed is reduced, as a result the false alarm rate also decreases when compared to the traditional sliding window approach. This is true regardless of which classifier is used, granted that it has acceptable discrimination capabilities. The number of falsely classified faces (false positives) decreased in an average of 52%, compared to the sliding window approach, while the number of correctly classified faces (true positives) remained relatively the same. Some synthetic and real-world images used for the experiments and the resulting detections are illustrated in Fig. 9.

Fig. 9. (a) Detection using depth and edge information (b) Disparity maps of input stereo images (c) Edge detection results for the input images

The proposed hardware architecture has been optimized for face detection; however, with minor modifications it can be adjusted to perform detection of any object. First, we shall consider changes in order for the architecture to be used for other applications. In general only the classifier needs hardware changes; if an SVM produces adequate results, then a similar architecture with the one we propose in this work can also be used, however, any other classifier can be incorporated in the architecture if necessary (for example the Viola and Jones based classifier in [Kyrkou and Theocharides, 2011] can be used). The other changes concern specific parameters that need to be adjusted in each hardware module, and are necessary in order to facilitate the new object of interest. Specifically, the window size for each detection scenario is different and as a result the pre-computed WEU parameters are adjusted to the object size characteristics (window height and window width in Figure 6). The threshold for the minimum number of edge pixels in a window can also be adjusted since the window is now different. Additionally, it is also possible to introduce an upper threshold on the number of edge pixels depending on the object of interest to eliminate noisy regions or cluttered background regions. Finally, any feature extraction algorithm (SURF, SIFT, etc.) can be integrated by implementing an appropriate processing unit that will perform processing on the valid windows prior to the classification process.

To illustrate that the proposed framework can be used in other applications we performed simulations for car side view detection for 320x240 cropped and resized test images from [EISATS Set 1, 2007] and a window size of 100x40, and some detection results are shown in Fig. 10. We briefly describe the necessary changes done to facilitate the car side view detection. The disparity and edge detection tasks were carried out with the same parameters as face detection as they do not take any object specific parameters, while the window extraction process was adjusted for a 100x40 window. The classifier

Fig. 10. Early simulation results for car side view detection on 320x240 images. A polynomial SVM of 50 reduced set vectors is used for classification. (a) Detection using depth and edge information (b) Disparity maps of input stereo images (c) Edge detection results for the input images

architecture was adapted to the new training set. A polynomial SVM with 50 reduced-set-vectors was utilized. Because of the larger search window size it was possible to increase the disparity search overlap, which resulted in fewer generated windows compared to face detection. However, the time needed for classification and edge pixel accumulation increased, as both tasks are depended on the window size (vector dimensionality). The hardware demands for the SVM classifier also changed because of the increased window size. The memory needed was increased, since the reduced-set-vectors have higher dimensionality, however, the processing resources needed were reduced as there were fewer reduced set vectors. Using the Xilinx synthesis and simulation tools, it was estimated that the expected performance for this application can reach up to 70 fps. The figure is lower than face detection since the classification time per window, which is the bottleneck in the overall computation, was increased.

6. CONCLUDING REMARKS

This paper presented a search reduction approach that utilizes depth and edge information and its hardware realization on an FPGA, which can be integrated into existing 3D vision systems in order to build efficient embedded image analysis systems. The implemented object detection system offers improvements both in terms of frame-rate and detection accuracy. Specifically results indicate a real-time performance for a range of image sizes on a Virtex 5 FPGA. Furthermore, the detection accuracy is improved with respect to the

sliding window approach as a lot of windows are discarded prior to the classification process. The overall framework and proposed hardware architecture can be further improved either with application specific optimizations, or an ASIC implementation. Additionally, the edge and depth directed method can be used with any classification algorithm and potentially other search reduction techniques and feature extraction algorithms for a variety of object detection applications. As an immediate follow up to this work we also intent to explore how to incorporate 3D information into the classification process to provide more robust detection results. Finally, we will also explore different approaches and algorithms to implement the three main tasks of the algorithm namely edge detection, disparity computation, and classification, and the impact of different algorithms in terms of hardware implementation and performance.

REFERENCES

Anila S., Devarajan N., 2011, Simple and Fast Face Detection System Based on Edges, International Journal of Universal Computer Sciences, 1, 2 (March 2010).

Browatzki B., Fischer J., Graf B., Bulthoff H.H., Wallraven C., 2011, Going into depth: Evaluating 2D and 3D cues for object classification on a new, large-scale object dataset, IEEE International Conference on Computer Vision Workshops (ICCV Workshops), 6-13 Nov, pp.1189-1195.

Burges C. J. C., 1996 Simplified support vector decision rules. In 13th Int. Conf. on Machine Learning, pp. 71–77. San Francisco, CA: Morgan Kaufmann.

Burges, C. J. C., 1998. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Min. Knowl. Discov.* 2, 2 (June 1998), 121-167.

CBCL FACE DATABASE #1, 2000, Center for Biological and Computational Learning at MIT and MIT, http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html.

Che M. and Chang Y., 2010. A Hardware/Software Co-design of a Face Detection Algorithm Based on FPGA. *In Proceedings of the 2010 International Conference on Measuring Technology and Mechatronics Automation* - Volume 01 (ICMTMA '10), Vol. 1. IEEE Computer Society, Washington, DC, USA, 109-112.

Cho J., Mirzaei S., Oberg J., Kastner R., 2009. Fpga-based face detection system using Haar classifiers. In *Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays* (FPGA '09). ACM, New York, NY, USA, 103-112.

Darrell T., Gordon G., Harville M., Woodfill J., 1998, Integrated Person Tracking Using Stereo, Color, and Pattern Detection, *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp.601.

EISATS Set 1, 2007, Multimedia Imaging Technology portal, University of Auckland http://www.mi.auckland.ac.nz/.

Farrugia N., Mamalet F., Roux S., Yang F., and Paindavoine M., 2009. Fast and robust face detection on a parallel optimized architecture implemented on FPGA.*IEEE Trans. Cir. and Sys. for Video Technol.* 19, 4 (April 2009), 597-602.

Garcia C. and Delakis M., 2004. Convolutional Face Finder: A Neural Architecture for Fast and Robust Face Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* 26, 11 (November 2004), 1408-1423.

Han D., Choi J., Cho J, Kwak D., 2011, Design and VLSI implementation of high-performance face-detection engine for mobile applications, *IEEE International Conference on Consumer Electronics*, 705-706.

Hadjitheophanous, S., Ttofis, C., Georghiades, A.S., Theocharides, T., 2010 , Towards hardware stereoscopic 3D reconstruction a real-time FPGA computation of the disparity map, *Design, Automation & Test in Europe Conference & Exhibition (DATE),* (March 2010),1743-1748

He C., Papakonstantinou A., and Chen D., 2009. A novel SoC architecture on FPGA for ultra fast face detection. In Proceedings of the 2009 *IEEE international conference on Computer design (ICCD'09)*. IEEE Press, Piscataway, NJ, USA, 412-418.

Hetzel G., Leibe B., Levi, P., Schiele, B., 2001, 3D object recognition from range images using local feature histograms, *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 394-399.

Hiromoto M., Sugano H., and Miyamoto R., 2009. Partially parallel architecture for AdaBoost-based detection with Haar-like features. *IEEE Trans. Cir. and Sys. for Video Technol.* 19, 1 (January 2009), 41-52.

Kyrkou C. and Theocharides T., 2011. A Flexible Parallel Hardware Architecture for AdaBoost-Based Real-Time Object Detection, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19, 1 (June 2011), 1034-1047.

Kyrkou C. and Theocharides T., 2011. A Parallel Hardware Architecture for Real-Time Object Detection with Support Vector Machines, *to appear in IEEE Transactions on Computers*.

Kyrkou C., Ttofis C. and Theocharis T., 2011, Depth-Directed Hardware Object Detection, *Proceedings of the Conference on Design, Automation and Test in Europe*. 14-18 March, 1 – 6.

Kyrkou C., Ttofis C. and Theocharis T., 2011, FPGA-Accelelerated Object Detection using edge information, *21st International Conference on Field Programmable Logic and Applications*, Sept. 5-7, 2011.

Liebelt J., Schmid C., Schertler K., 2008, Viewpoint-independent object class detection using 3D Feature Maps, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, 23-28 June, pp.1-8.

McCready R., 2000. Real-Time Face Detection on a Configurable Hardware System. *In Proceedings of the The Roadmap to Reconfigurable Computing, 10th International Workshop on Field-Programmable Logic and Applications* (FPL '00), Reiner W. Hartenstein and Herbert (Eds.). Springer-Verlag, London, UK, 157-162.

Moreels P., Perona P., 2005, Evaluation of features detectors and descriptors based on 3D objects, *Tenth IEEE International Conference on Computer Vision*, 17-21 Oct., pp. 800-807.

Osuna E., Freund R., and Girosi F., 1997, Training Support Vector Machines: An Application to Face Detection, *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 130-136.

Raman M., Aggarwal H., 2009, Study and Comparison of Various Image Edge Detection Techniques, *International Journal of Image Processing*, 3, 1, (2009), 1 – 12.

Roobaert D., Van Hulle M.M., 1999, View-based 3D object recognition with support vector machines, *Proceedings of the 1999 IEEE Signal Processing Society Workshop Neural Networks for Signal Processing*, Aug 2009, pp.77-84.

Ruiz-Llata M. and Yebenes-Calvino M. 2009. FPGA Implementation of Support Vector Machines for 3D Object Identification. *In Proceedings of the 19th International Conference on Artificial Neural Networks: Part I (ICANN '09)*, Springer-Verlag, Berlin, Heidelberg, 467-474.

Rusu R. B, Blodow N., and Beetz M., 2009, Fast point feature histograms (FPFH) for 3D registration, *In Proceedings of the 2009 IEEE international conference on Robotics and Automation (ICRA'09)*, IEEE Press, Piscataway, NJ, USA, 1848-1853.

Sadri M.S., Shams N., Rahmaty M., Hosseini I., Changiz R., Mortazavian S., Kheradm S., Jafari R., 2004, Global Signal Processing Expo and Conference.

Steder B., Rusu R. B., Konolige K., and Burgard W., 2010, NARF: 3D range image features for object recognition, In *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Ttofis C., Hadjitheophanous S., Georghiades A.S., Theocharides T., 2012, Edge-Directed Hardware Architecture for Real-Time Disparity Map Computation, *accepted to appear IEEE Transactions on Computers*.

Trucco E. and Verri A., 1998. Introductory Techniques for 3-D Computer Vision. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Viola P. and Jones M.J., 2004. Robust Real-Time Face Detection. *Int. J. Comput. Vision*57, 2 (May 2004), 137-154.

Wang Y.G., Lim E.T., Venkateswarlu R., 2004, Stereo head/face detection and tracking, *International Conference on Image Processing*, pp. 605- 608.

Wu H., Suzuki K., Wada T., and Chen Q., 2009. Accelerating Face Detection by Using Depth Information. In *Proceedings of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology* (PSIVT '09), Toshikazu Wada, Fay Huang, and Stephen Lin (Eds.). Springer-Verlag, Berlin, Heidelberg, 657-667.