J. Imaging **2015**, *1*, 193-219; doi:10.3390/jimaging1010193

OPEN ACCESS

Journal of *Imaging*

ISSN 2313-433X

http://www.mdpi.com/journal/jimaging

Article

FPGA-Based Portable Ultrasound Scanning System with Automatic Kidney Detection

R. Bharath [1], Punit Kumar [1], Chandrashekar Dusa [1], Vivek Akkala [1], Suresh Puli [1], Harsha Ponduri [1], K. Divya Krishna [1], P. Rajalakshmi [1,*], S. N. Merchant [2], Mohammed Abdul Mateen [3] and U. B. Desai [1]

[1] Wireless Network Lab, Department of Electrical Engineering, Indian Institute of Technology Hyderabad, Hyderabad 502285, India; E-Mails: ee13p0007@iith.ac.in (R.B.); ee15mtech01002@iith.ac.in (P.K.); ee12m1014@iith.ac.in (C.D.); ee12m1041@iith.ac.in (V.A.); sureshpuli@iith.ac.in (S.P.); harshaponduri@iith.ac.in (H.P.); ee13m0003@iith.ac.in (K.D.K.); ubdesai@iith.ac.in (U.B.D.)

[2] Department of Electrical Engineering, Indian Institute of Technology Bombay, Mumbai 400076, India; E-Mail: merchant@ee.iitb.ac.in

[3] Asian Institute of Gastroenterology, Hyderabad 500082, India; E-Mail: drmateen@gmail.com

* Author to whom correspondence should be addressed; E-Mail: raji@iith.ac.in; Tel. +91-040-2301-6017.

Academic Editors: Philip Morrow and Gonzalo Pajares Martinsanz

Received: 24 September 2015 / Accepted: 17 November 2015 / Published: 4 December 2015

Abstract: Bedsides diagnosis using portable ultrasound scanning (PUS) offering comfortable diagnosis with various clinical advantages, in general, ultrasound scanners suffer from a poor signal-to-noise ratio, and physicians who operate the device at point-of-care may not be adequately trained to perform high level diagnosis. Such scenarios can be eradicated by incorporating ambient intelligence in PUS. In this paper, we propose an architecture for a PUS system, whose abilities include automated kidney detection in real time. Automated kidney detection is performed by training the Viola–Jones algorithm with a good set of kidney data consisting of diversified shapes and sizes. It is observed that the kidney detection algorithm delivers very good performance in terms of detection accuracy. The proposed PUS with kidney detection algorithm is implemented on a single Xilinx Kintex-7 FPGA, integrated with a Raspberry Pi ARM processor running at 900 MHz.

Keywords: portable ultrasound scanning; FPGA; organ detection; Viola–Jones

1. Introduction

Ultrasound scanning is a widely-used modality in healthcare, as it provides real-time, safer and noninvasive imaging of organs in the body, including heart, kidney, liver, fetus, *etc.* Conventional ultrasound machines cannot be used for point-of-care (POC) applications due to their high form factor. POC refers to treating the patients at their bedside, which has great potentiality in saving lives, especially in situations like ambulances, military, casualties, rural healthcare, *etc.*[1]. Ultrasound scanning needs limited setup for scanning, and advancements in computing platforms enable portable ultrasound scanning (PUS) systems' use in POC applications.

Commercially-available PUS with application-specific integrated circuit (ASIC) cannot be updated with new algorithms, while programmable ultrasound scanning machines provide the flexibility for rapid prototyping, updating and validating new algorithms [2]. Computing platforms, such as field-programmable gate array (FPGA), digital signal processor (DSP) and media processors, have capabilities in terms of hardware to realize complex ultrasound signal processing algorithms in real time. In [3], ultrasound signal processing is performed on a single media processor; due to the computational limitations of the processor, the beamformed data are acquired through an ultrasound research interface from a high-end system, and the real-time mid-end and back-end processing algorithms are implemented on a media processor. In [4], a complete standalone ultrasound scanning machine is realized using single FPGA and a proposed pseudo-dynamic received beamforming with an extended aperture technique targeting the PUS system. In [5], we proposed a low complexity programmable FPGA-based eight-channel ultrasound transmitter for research, which enables rapid prototyping of new algorithms. ASIC for ultrasound signal processing is beneficial for reducing the size of the ultrasound machine, and we proposed an ASIC design for integrated transmit and receive beamforming in [6]. In [7], the authors demonstrated the feasibility of implementing the ultrasound signal processing on a high-end android mobile platform. Low complexity algorithms have been proposed for portable ultrasound machines, such that they can be easily implemented on computational platforms that have less computational capability [8–11].

In this paper, we propose an FPGA-based standalone portable ultrasound scanning system with real-time kidney detection. We chose FPGA for implementing complete ultrasound signal processing due to its efficiency in handling high data rates. The PUS machine is realized using Kintex-7 [12], which is integrated with an ARM1176JZF-S Raspberry Pi-2 processor [13].

For displaying and recording the scanned ultrasound data, the H.264 codec is used. H.264 is a computationally-expensive procedure, and several optimized versions are proposed to implement in real time on different computing platforms, like FPGAs, DSPs, microprocessors, *etc.* Here, we list some of the optimizations proposed for the H.264 codec to implement on an FPFA platform. In [14], low complexity and low energy adaptive hardware for H.264 with multiple frame reference motion estimation is proposed and implemented on an FPGA platform. A bioinspired custom-made reconfigurable hardware architecture is proposed in [15,16] for efficient computation of motion between image sequences. The work in [17] proposed a flexible and scalable fast estimation processor targeting the computational requirements for high definition video using the H.264 video codec on an FPGA platform. Optimization for accelerating the block matching algorithms on a Nios II microprocessor, which is designed for FPGA, is proposed in [18,19]. A new early termination algorithm is proposed in [20] for effective computation of the correlation coefficient in template matching with fewer computations. Several optimizations are available for implementing the H.264 codec on an FPGA, but the FPGA resources in the proposed portable ultrasound system are preserved for implementing complex Doppler ultrasound. In the proposed portable ultrasound system, the H.264 codec is installed on a Raspberry Pi processor, which is downloaded from [21].

Even though ultrasound scanning is widely used, the ultrasound images have a low signal-to-noise ratio and contrast, and there will not be any significant distinction between the organ region and edges, making it difficult to identify the organ exactly. Portable ultrasound scanners in remote areas are mostly used by emergency physicians who may not be fully trained in sonography. Hence, there is a need for computer-assisted algorithms, which can assist physicians in making accurate decisions. These algorithms include speckle suppression, organ detection, computer-aided diagnosis (CAD), *etc.* Hence, automatic detection will be very beneficial for the development of applications related to CAD, image compression, segmentation, image-guided interventions, *etc.*

In the literature, portable ultrasound scanning machines are realized on FPGAs, DSPs and media processors [2–4], and the automatic detection of kidneys in CT images based on random forest is proposed. The algorithms proposed for CT images will not work for ultrasound images, as the characteristics of kidney change with imaging modality, and the kidney images acquired through ultrasound scanning depend on the person who scans. Therefore, manual cropping is employed for detecting the kidney in ultrasound images [22]. In this paper, we propose an automatic kidney detection algorithm for detecting the kidneys in a real-time ultrasound video and evaluate the algorithm on an FPGA-based PUS machine in real-time. To the best of our knowledge, this is the first working prototype of a portable ultrasound scanner with automatic detection of kidney. The prototype of the PUS is evaluated by scanning the tissue-mimicking gelatin phantom. The reconstructed images of the PUS are visually compared to the images of commercially available platforms. The proposed portable ultrasound scanner with the kidney detection algorithm is successfully tested at the lab level.

The rest of the paper is organized as follows. In Section 2, we describe the flow adhered to for automated kidney detection. The hardware implementation of the portable ultrasound system is described in Section 3. The experimental setup and performance of the system are discussed in Section 4. Section 5 concludes the paper with discussions on the future scope of work.

2. Automatic Kidney Detection

Automatic organ detection is very beneficial for semi-skilled persons who operate the device remotely. Organ detection in ultrasound is strongly influenced by the quality of data. The characteristic artifacts that make the organ detection complicated are speckle noise, acoustic shadows, attenuation, signal dropout and missing boundaries of the organ. In this paper, we focus on detecting kidney in ultrasound images, which is different from segmentation of kidney, *i.e.*, extracting the exact contour of kidney.

Kidney is made up of soft tissue, and it is nonrigid in nature. Moreover, the size of kidney in ultrasound images varies from patient to patient depending on the patient's age, anatomy, disease, orientation of acquisition, *etc.*, so, it becomes a challenging task to automatically detect kidney in ultrasound images.

In the literature, the problem of automatic detection of kidney with ultrasound images has never been addressed before. However, there are some algorithms proposed in the literature to segment the kidney in ultrasound. Automatic organ detection is addressed as one of the intermediate steps involved in the segmentation procedure. Organs have to be accurately segmented in designing applications, like 3D reconstruction, CAD, automatic measurements, *etc.* Organ detection and localization are the two major steps involved in segmentation. In the literature, not much interest is shown in automatic detection of organs, while major research is done on detecting the exact contour of an organ. The organs in the images are detected by manually cropping along the contours [22], placing the landmarks on the contour [23] and marking the bounding boxes around the kidney [24]. Automatic detection of kidney in CT slices using random forest has been used in [25] for reconstructing the 3D structure of kidney. A two-stage detection algorithm is employed for automatic detection of lymph nodes in CT data; Haar features with the Adaboost cascade classifier are used in the first stage, and in the second stage, self-assigning features with the Adaboost cascade classifier are used [26]. The algorithm proposed in [22] needs manual cropping and assumes that kidney is present in every slice. However, when we scan kidney using 2D probes, we cannot expect kidney to be present in each frame, and kidneys also have been occluded by surrounding organs, like liver and spleen. As kidney is not present in every slice, the correlation between two frames is not useful in tracking the kidney. Therefore, the algorithm has to detect the kidney in each incoming slice. The kidney detection algorithm has to be fast enough at automatically detecting the kidney in real time from incoming ultrasound data.

Active contours [27], also known as snakes [28], pose contour detection as an optimization problem that moves a parameterized curve towards the image region with strong edges; though this model is highly successful, it suffers from initialization, and the optimization function being nonlinear, as a result, the solution is trapped in many places. In level-set methods [29,30], the contour of kidney is represented by a level-set zero of a distance function; this approach is less sensitive to initialization and significantly suffers from imaging conditions. Both active contours and level-set approaches are highly successful in contour extraction; they present the drawback of incorporating prior knowledge about the shape and texture distribution in the optimization function; the prior knowledge may be in the form of a probability distribution formed from the manually-segmented contours and textures. Active shape models (ASM) [23] and active appearance models (AAM) [31] are supervised learning approaches, which estimate the parameters from a joint distribution representing the shape and appearance of an organ; these methods need large databases, and the initialization of the contour should be close to local optima. In [32], kidney segmentation based on texture and shape priors is proposed; the texture features are extracted from a bank of Gabor filters on test images, then an iterative segmentation is proposed to combine texture measures into a parametric shape model. In [22], a probabilistic Bayesian method is employed to detect the contours in a three-dimensional ultrasound.

The models discussed above are constraint based on the terms of shapes and the condition of the organ (normal and abnormal). For example, active contours will fail to extract the exact contour of kidney in the presence of cysts in kidney. However, the ASM and AAM can capture the contour of kidney even in the presence of cysts, but fail to capture the diversified variations of contour. In this paper, we are interested in detecting the region of interest for kidney, which is useful to emphasize only that particular region further. The method we adopted for detecting kidney in an ultrasound images is based on a Viola–Jones detector [33]. Viola–Jones is highly successful in detecting many object classes; it was primarily developed for face detection in images and later extended to detect faces in video sequences [34]. Recently, the Viola–Jones algorithm framework was used in medical image analysis to detect various organs, such as pelvis and proximal femur of a hip joint in 3D CT images [35]. In [36], a modified Viola–Jones algorithm is used to automatically locate the carotid artery in ultrasound images.

In this paper, we show the effectiveness of the Viola–Jones algorithm in detecting the region of interest for kidney in ultrasound images. The complete Viola–Jones algorithm is implemented on a Raspberry pi board, which is interfaced with the Xilinx Kintex-7 FPGA, where ultrasound signal processing algorithms are implemented.

The block diagram representation of the kidney detection algorithm is shown in Figure 1. The basic components corresponding to this algorithm include Haar-like features, the integral image, the Adaboost algorithm and the cascade of classifiers.

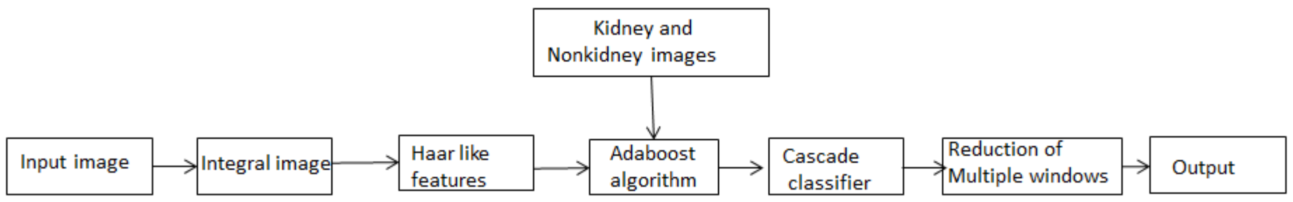


Figure 1. Block diagram of the Viola–Jones algorithm for detecting kidney.

2.1. Haar-Like Features Used in the Kidney Detection Algorithm

Haar-like features are derived from the kernels [37]; a few of these kernels are shown in Figure 2. The white region in the kernel corresponds to weight $w_0 = -1$, and the black region corresponds to $w_1 = +1$. The value of these features are then computed using the formula $f(x) = w_0 r_0 + w_1 r_1$, where $f(x)$ is the response of a given Haar-like feature to the input image x, w_0 is the weight of the area r_0 and w_1 is the weight of the black area r_1. The number of pixels in areas r_0 and r_1 varies because the features are generated for various possible combinations and positions in a given window. These dimensions start from a single pixel and extend up to the size of a given window.

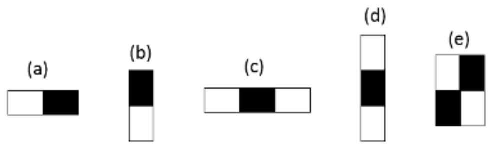


Figure 2. Kernels used to extract Haar-like features for the kidney detection algorithm. (**a**,**b**) Two-rectangle features. (**c**,**d**) Three-rectangle features and (**e**) Four-rectangle feature.

The features generated using these kernels are independent of image content. The process of feature generation is explained as follows: considering the kernel in Figure 2a, which is initially of a two-pixel column width (one pixel white and one pixel black), the feature value of $f(x)$ is computed. The kernel is shifted from the top left of the image by one pixel, and a new feature value is calculated. Similarly, the kernel is then moved across the complete image, until it reaches the right bottom of the image with all of the features computed. Hence, the features are evaluated hundreds of times as the kernel moves across all of the rows of the image, and every time, a new feature value is updated in the feature list. Later, the kernel is increased to a four-pixel width (two white pixels and two black pixels), and the process is repeated to get new feature values. Five different kernels are used for generating the rectangular features. The process is repeated for all of the kernels; considering all of the variations in size and position, a total of 586,992 features were computed [33] for a window of size 32 $\times$ 54 [38]. Figure 3 illustrates the flowchart for generating Haar-like features.

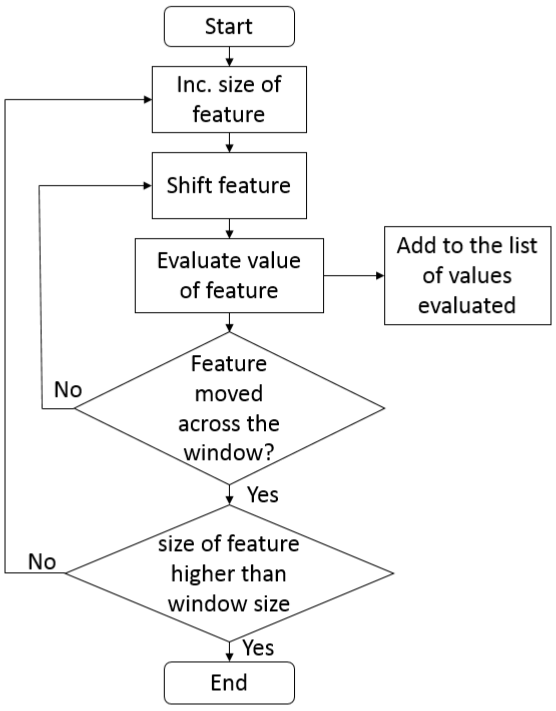

Start
Inc. size of feature
Shift feature
Evaluate value of feature
Add to the list of values evaluated
Feature moved across the window?
No
Yes
size of feature higher than window size
No
Yes
End

Figure 3. Flowchart to generate Haar-like features for the kidney detection algorithm.

2.2. Integral Image

Computing the sum of pixels in a given area is a computationally-expensive procedure; hence, intermediate representation of the image is used to compute the features rapidly and efficiently; this representation of the image is called an integral image [39]. The conversion of the image to an integral image is based on the following formulae:

$$s(u,v) = s(u,v-1) + i(u,v) \quad (1)$$

$$i_1(u,v) = i_1(u-1,v) + s(u,v) \quad (2)$$

where u, v are the indices of the pixel, $s(u,v)$ is the cumulative sum of pixel values in a row with initial conditions as $s(u,-1) = 0$ and $i_1(-1,v) = 0$, $i(u,v)$ is the pixel value of original image and $i_1(u,v)$ is the pixel value of the integral image.

In the integral image, the sum of all pixels under a rectangle can be evaluated using only the four-corner values of the image, which is similar to the summed area technique used in graphics [40]. In Figure 4, the value of the integral image at Location 1 is the sum of pixels in Rectangle A, at Location 2 A + B, at Location 3 A + C and at Location 4 A + B + C + D. The sum of pixels in D can be evaluated as (4 + 1) − (2 + 3).

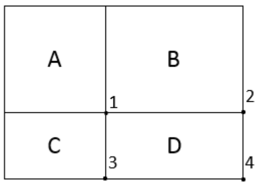


Figure 4. Sum of the pixels in Region D using the four-array reference.

2.3. Selection of Features for Automatic Kidney Detection Using the Adaboost Algorithm

Adaboost is a machine learning algorithm that helps with finding the best features among 586,000+ features [41,42] for detecting kidney. After evaluating the obtained features from the Adaboost algorithm, the weighted combination of these features is used to decide whether a given window has kidney or not. These selected features are also called weak classifiers. The output of a weak classifier is binary, either one or zero. One indicates that the feature is detected in the window, and zero indicates that there is no feature in the window. Adaboost constructs a strong classifier based on the linear combination of these weak classifiers.

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + ... \tag{3}$$

where $F(x)$ is a strong classifier, $f_i(x)$ is a weak classifier and α_i is the weight corresponding to the error evaluated using classifier $f_i(x)$.

The flowchart for computing the features using the Adaboost algorithm is shown in Figure 5 [36]. Adaboost starts with a uniform distribution of weights over training examples. The classifier with the lowest weighted error (a weak classifier) is selected. Later, the weights of the misclassified examples are increased, and the process is continued, till the required number of features is selected. Finally, a linear combination of all of these weak classifiers is evaluated, and a threshold is selected. If the linear combination of a new image in a given window is greater than this threshold, it is considered as kidney being present; if it is less than the threshold, it is classified as non-kidney. The Adaboost algorithm finds a single feature and threshold that best separate the positive (kidney) and negative (non-kidney) training examples in terms of weighted error. Firstly, the initial weights are set for positive (with kidney) and negative (without kidney) examples. Each classifier is used from 586,000+ features to determine the error, which in this case is to misclassify the presence of kidney in a window. If the error is high, the training process ends; else, the weights α_t are set to the selected linear classifier. The α_j are computed as:

$$\alpha_j = log\frac{1-\epsilon_j}{\epsilon_j} \tag{4}$$

where ϵ_j is the error occurring while classifying the images. Later, the weights of the positive and negative examples are adjusted, such that the weights of the misclassified examples are boosted, and the weights of the correctly-classified examples are not changed. Finally, a strong classifier is created, which is a combination of weak ones weighted according to the error that they had.

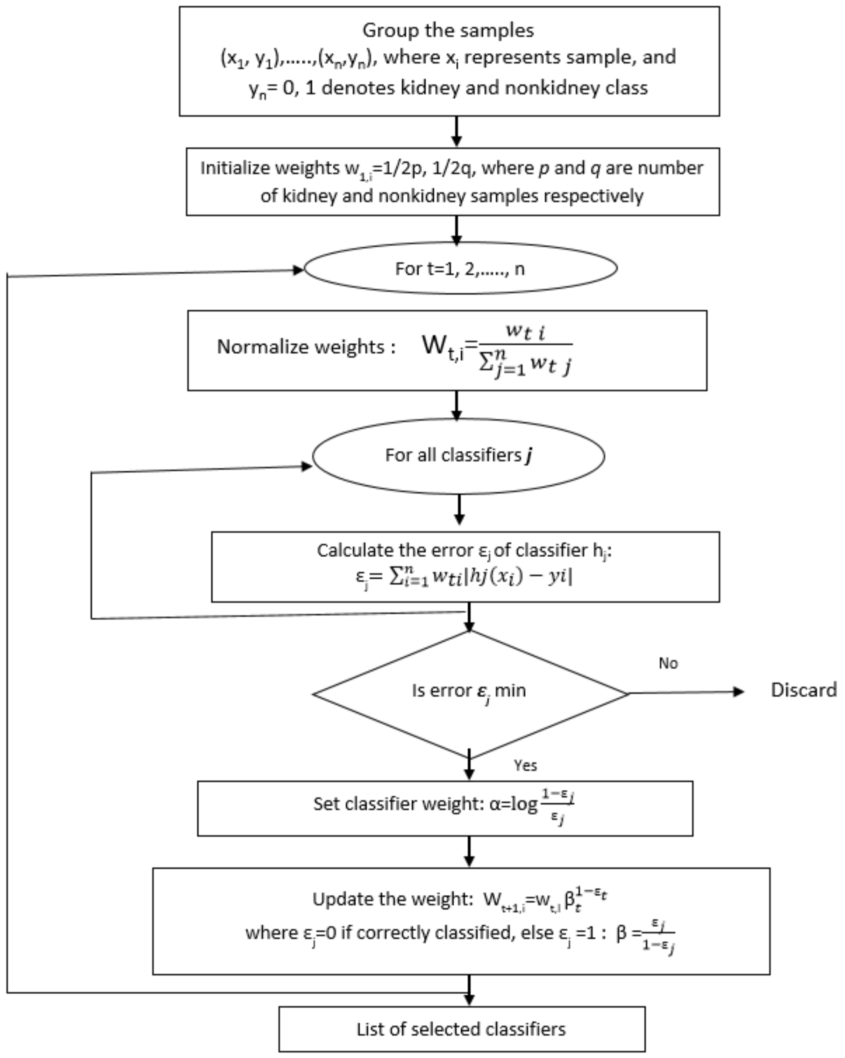

Group the samples $(x_1, y_1),\ldots,(x_n,y_n)$, where x_i represents sample, and $y_n = 0, 1$ denotes kidney and nonkidney class
Initialize weights $w_{1,i}=1/2p, 1/2q$, where p and q are number of kidney and nonkidney samples respectively
For t=1, 2,....., n
Normalize weights : $W_{t,i}=\frac{w_{t\,i}}{\sum_{j=1}^{n} w_{t\,j}}$
For all classifiers j
Calculate the error ε_j of classifier h_j: $\varepsilon_j = \sum_{i=1}^{n} w_{ti}|hj(x_i) - yi|$
Is error ε_j min
No
Discard
Yes
Set classifier weight: $\alpha=\log\frac{1-\varepsilon_j}{\varepsilon_j}$
Update the weight: $W_{t+1,i}=w_{t,i}\,\beta_t^{1-\varepsilon_t}$
where $\varepsilon_j=0$ if correctly classified, else $\varepsilon_j=1$: $\beta=\frac{\varepsilon_j}{1-\varepsilon_j}$
List of selected classifiers

Figure 5. Flowchart of the Adaboost algorithm for selecting the features for kidney detection.

2.4. Cascade Classifier for Automatic Kidney Detection

The strong classifier formed from the linear combination of these best features is a computationally-expensive procedure. Therefore, a cascade classifier is used, which consists of stages, and each stage has a strong classifier [43]. Therefore, all of the features are grouped into several stages, where each stage has a certain number of features along with a strong classifier. Thus, each stage is used to determine whether a kidney is present in a given sub-window. The block diagram of the cascade classifier is shown in Figure 6. A given sub-window is immediately discarded if kidney is not present and not considered for further stages. The cascade classifier works on the principle of rejection, as the majority of sub-windows will be negative. It rejects many negatives at the earliest stage possible. This reduces the computational cost, and hence, kidney in the image can be detected at a faster rate. The complete analysis and implementation of the Viola–Jones algorithm can be found in [44].

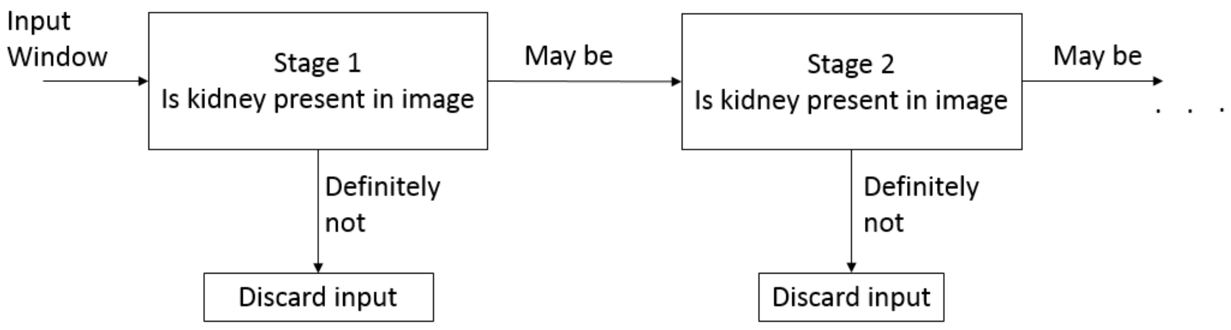


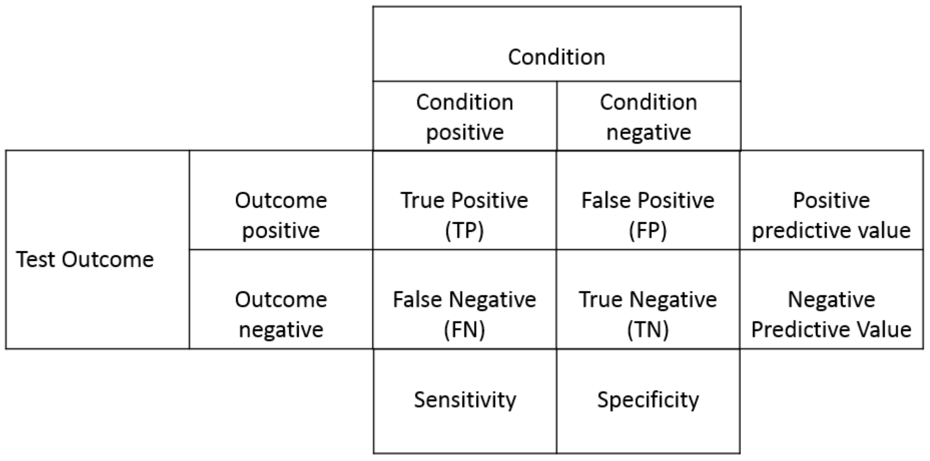
Figure 6. Block diagram of the cascade classifier for kidney detection.

2.5. Database

Kidney images and videos were acquired using a Siemens S1000 from 400 patients during the period May 2014 to February 2015. These patients were in the age group of 14 to 65 years, including both genders. The database consists of 790 images with 400 kidney (normal: 332; cyst: 40; stone: 28) and 390 non-kidney (liver: 100; heart: 95; carotid: 90; spleen: 105) images. The kidney and non-kidney images were collected from the same patients. The data were collected by acknowledging the patients. The data received from the doctor were annotated with the patient's age, gender and disease. The names of the patients were not revealed in the process.

2.6. Metrics Used for Evaluating the Automatic Kidney Detection Algorithm

The performance of the organ detection algorithm is evaluated by tabulating the confusion matrix. The metrics used in this paper, referring to Figure 7, are given below:



		Condition		
		Condition positive	Condition negative	
Test Outcome	Outcome positive	True Positive (TP)	False Positive (FP)	Positive predictive value
	Outcome negative	False Negative (FN)	True Negative (TN)	Negative Predictive Value
		Sensitivity	Specificity	

Figure 7. Confusion matrix.

$$Sensitivity = \frac{TP}{TP + FN} \tag{5}$$

$$Specificity = \frac{TN}{TN + FP} \tag{6}$$

$$Positive\ Predictive\ Value = \frac{TP}{TP + FP} \tag{7}$$

$$Negative\ Predictive\ Value = \frac{TN}{TN + FN} \tag{8}$$

$$Accuracy = \frac{TP + TN}{Total\ test\ images} \tag{9}$$

Sensitivity measures the proportion of actual positives, and specificity measures the proportion of actual negatives correctly identified as such, respectively. A positive predictive value measures the efficiency of an algorithm to correctly identify normal kidneys, and a negative predictive value measures the efficiency of an algorithm to correctly identify negative images. Accuracy gives the efficiency of an algorithm in correctly localizing kidney in the images.

For training the Viola–Jones algorithm, 640 images (positive images: 320; negative images: 320) are used. The positive and negative images used in training the cascade classifier are shown in Figures 8 and 9, respectively. The positive training dataset consists of kidneys with 270 normal, 30 cyst and 20 stone cases. The negative training dataset consists of 82 liver, 78 heart, 75 carotid and 85 spleen images. The region of interest for kidney differs from image to image. Therefore, in the training, all of the positive instances are rescaled to the expectation of width, the height of positive instances using nearest neighbor approach. The expectation for the region of interest is found to be 32 $\times$ 54 pixels.

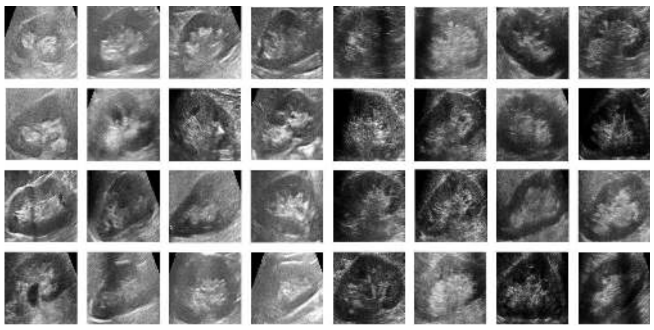


Figure 8. Some of the kidney images of the positive training dataset used in training the Viola–Jones algorithm.

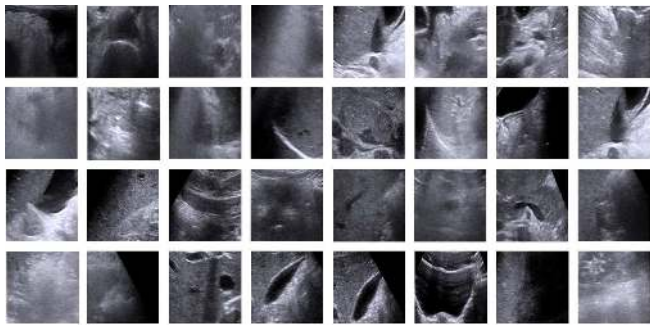


Figure 9. Some of the images of the negative training dataset used in training the Viola–Jones algorithm.

The kidney detection algorithm is trained using a 20-stage cascade classifier with a 0.2 false alarm rate. For training each stage in the cascade classifier, positive and negative examples are considered with a 1:2 ratio. In training the object model for kidney, each stage uses at most 320 positive instances and 640 negative instances. Negative instances are generated automatically from the negative images. The number of features used in first five stages of the cascade classifier are 3, 5, 4, 5 and 6, respectively. A total of 92 features are selected from 586,000+ features. The kidney detection algorithm is realized using functions of the Viola–Jones algorithm available in OpenCV. The inbuilt functions in OpenCV take the false alarm rate and the number of cascade classifiers as the input parameters and generate a model with feature selection and threshold parameters for cascade classifiers.

MATLAB 2015a running on a desktop with an i7 processor, 16 GB RAM and a 2.8-GHz clock takes 17.25 min to train the cascade model for kidney detection. The model generated from MATLAB is compatible with OpenCV and is used to detect the kidney in real time on an ARM processor.

The number of features and the threshold for each cascade classifier are automatically detected by the Adaboost algorithm. The number of features and the threshold are selected based on the given false alarm rate. The true positives of the detector increase with the increase in false positives, as shown in Table 1. The increase in false positives reduces the specificity and positive predictive value; ideally, these values should be equal to one. One hundred percent accuracy can be achieved with the cost of reducing the specificity and positive predictive value. An increase in parameters, like window enlargement in every step and displacement of the sliding window, slightly increases the detection speed at a cost of a slight decrease in accuracy. Detection accuracy heavily depends on false detections. A strong two-stage classifier with 100% detection accuracy can be constructed by keeping false alarms at more than 40%. The number of features selected for classification depends on the Adaboost algorithm. The detection time increased as the number of features used for classification increased. The displacement of the sliding window in detecting the kidney affected the accuracy of detection. The results presented here are for a one-pixel displacement. The detection time is improved with a two-pixel shift with slightly reduced accuracy.

Table 1. Number of false detections *vs.* accuracy.

Number of False Positives (%)	Accuracy (%)
13	83
20	91
25	93.5
30	98
35	100
40	100

The kidney detection algorithm is initialized with the following parameters: the window enlargement in every step is set to 1.1, and every time, the sliding window is shifted by one pixel. A 10-fold cross-validation scheme is employed to test the variability in the database. The 10-fold cross-validation is applied on the test set ten times with different partitions. Ten random seeds are generated, each of them separating the test set into 10 disjoint sets. Each subset is used for testing, while the other remaining nine sets are used in training. Each 10-fold cross-validation produces classification results for 320 kidney and 320 non-kidney images. Thus, a total instance of 3200 and 3200 kidney and non-kidney images is tested. The overall classification accuracy is averaged over ten rounds. The Viola–Jones algorithm performed with an overall accuracy of 92.6% in detecting kidney.

3. Portable Ultrasound Overview

3.1. System Description

Figure 10 shows an overall block level representation of the portable ultrasound imaging system using a single FPGA. The architecture consists of a 64-element linear transducer array, an eight-channel data acquisition module, a Kintex-7 FPGA and a Raspberry Pi-2 ARM processor. The data acquisition module consists of a beamformer and analog front end circuits, which include a high voltage (HV) pulser, analog receivers and 14-bit eight-channel analog to digital converters operating at 40 MHz.

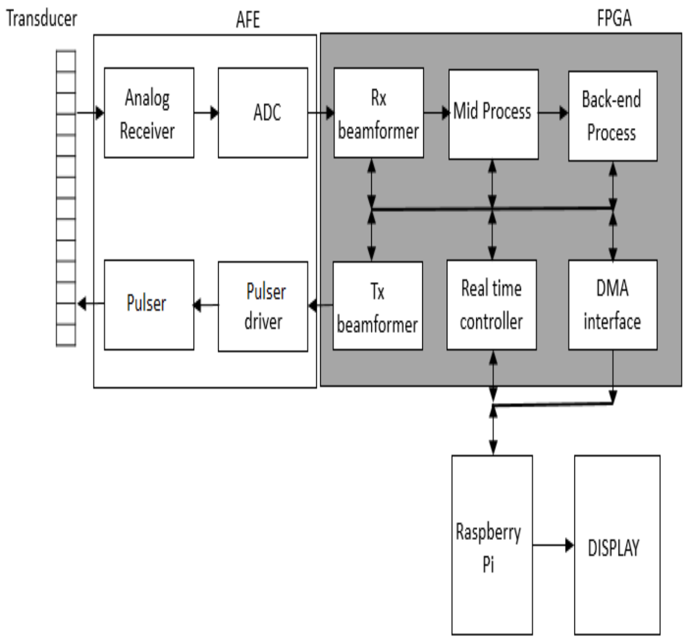


Figure 10. Block diagram representation of the portable ultrasound imaging system.

The signal processing algorithms, including the front-end, mid-end and back-end processing algorithms, are implemented on the Kintex-7 FPGA. The Raspberry Pi-2 processor running weezy-raspbian, a Debian-based Linux operating system with 1 GB RAM and a 900-MHz clock frequency, coordinates between all of the processing modules in ultrasound processing and also is used for displaying and communication purposes. The circuit-level organization and implementation of the portable ultrasound scanning system is shown in Figure 11. The Beamformer IC (LM96570 [45]) is programmed through Kintex-7 for generating eight P and N channels with 3.3-V logic pulses. The HV pulser acts like an amplifier, which converts 3.3-V logic pulses into transducer-compatible HV pulses.

The excitation pulses and received echoes traverse through the HV pulser. Therefore, when we apply HV pulses to the transducer, the transmit path will be shorted and the receive path will be opened to avoid damage to the analog circuitry. The analog front-end (AFE) receives reflected signals from the HV pulser and digitizes them. These digitized 14-bit ADC data are fed to the Kintex-7 for implementing mid-end and back-end algorithms. The Raspberry Pi-2 processor displays the data received from the Kintex-7 board in the form of an image.

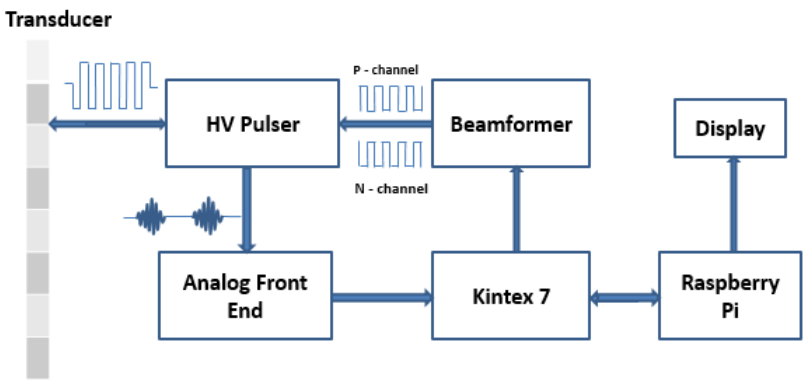


Figure 11. Circuit-level implementation of the portable ultrasound imaging system.

Based on the functionality, the signal processing algorithms in ultrasound scanning are categorized into frond-end, mid-end and back-end processing modules. The functionality and implementation of each module is discussed below.

3.2. Front-End Processing

The front-end processing module consists of transmit (Tx) and receive (Rx) beamforming, low noise amplification and time gain compensation algorithms. Tx and Rx beamforming algorithms are essential to obtain good spatial and temporal resolution. In Tx beamforming, more than one element from the transducer is excited in such a way that all of the sound waves will converge at a particular point. In Rx beamforming, the echoes reach the different transducer elements with different delays. The echoes received from the transducer elements are delayed in such a way that all of the echoes are added coherently, as shown in Figure 12. The delay values for the transducer elements, referring to Figure 12, are computed in the following way:

$$R_{bf} = \sum_{e=0}^{E-1} a_e r_e[n - t_e(n)], \quad n = 1, 2, 3.....N \tag{10}$$

where R_{bf} is the receive beamformed signal, *E* represents the number of active transducer elements used in acquiring the echo, *n* represents the number of focal zones, a_e is the apodization coefficient and t_e is the receive focusing delay of the e-th channel. The delay t_e is computed as:

$$t_e(n) = \frac{1}{v}[\sqrt{(x_e - x_f)^2 + (y_e - y_f)^2} - y_{fc}(n)] \tag{11}$$

here, (x_e, y_e) is the location of the e-th channel, (x_f, y_f) is the location of the focal point, y_{fc} is the distance from the focal point to the central channel and v is the velocity of sound. In electronic beam steering, the delay value for the channels, referring to Figure 13, is computed in the following way:

$$t_i = \frac{\sqrt{R^2_{fp(\alpha)} + x_i^2 - 2x_i R_{fp(\alpha)} sin(\alpha)}}{c} \tag{12}$$

where $R_{fp(\alpha)} = \frac{R_{fp(0)}}{cos(\alpha)}$

$$T_i = t_{max} - t_i \tag{13}$$

where P represents the focal point, as shown in Figure 13, $R_{fp(0)}$ denotes the distance from the center element to point P, t_i is the time required for the wavefront to reach point P, x_i is the coordinate of the i-th element, t_{max} is the maximum time required for the wavefront to reach point P, T_i is the delay for the i-th element and α is the steering angle.

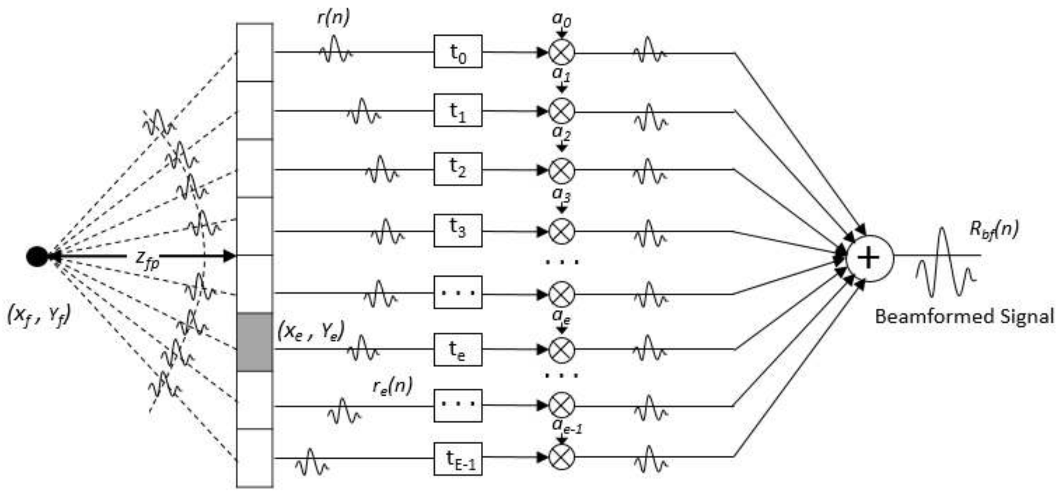


Figure 12. Receive beamforming technique employed in ultrasound scanning.

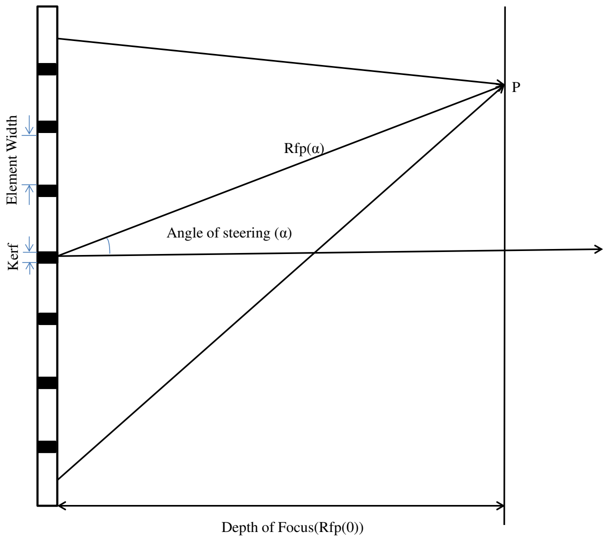


Figure 13. Calculations of delay patterns for each transducer element.

The delay values are pre-computed, and lookup table (LUT)-based processing is employed to reduce the hardware complexity. The Rx beamforming is designed for 1024 focal points. Two bytes of memory are allocated for each delay value, constituting of a total of 2 KB of memory for each channel. LUT occupies 16 KB of memory to store the delay values of eight channels. Further processing of the beamformed signal, including low noise amplification, low pass filtering and analog to digital conversion, are implemented using AFE 5808 IC [46].

3.3. Mid-End Processing

Envelope detection and log compression operations are performed in mid-end processing. The significance of each operation is discussed below.

3.3.1. Envelope Detection

Echoes received from the interfaces have several cycles of oscillations with the same frequency of transmitted pulse. The high frequency nature of the signal cannot be viewed as an image, as the human eye cannot perceive the distinction between the samples. In Figure 14, the curves passing from above and below the baseline represent the envelope of the signal, thus eliminating the high frequency components present in the signal. The amplitude of the envelope signal corresponds to the pixel intensity of the B-mode image. The envelope of the signal is computed by taking the square root for the summation of the squared in phase and quadrature phase signals. The Hilbert transform is used to obtain in phase and quadrature phase components of the signal; it provides a +90 degrees phase shift to the positive frequency and a −90 degrees phase shift to the negative frequencies [47,48]. The impulse response of an M length Hilbert filter is given by [49].

$$h[m] = \begin{cases} \frac{2}{\pi}\frac{sin^2(\pi(m-\alpha)/2)}{m-\alpha} & m \neq \alpha \\ 0 & m = \alpha \end{cases} \tag{14}$$

where $\alpha = (M-1)/2$

The Hilbert transform for echo signals is generated using a 32-tap FIR Hilbert filter; *m* is chosen based on the normalized root mean square error (RMSE) between the ideal Hilbert filter and the designed m-tap FIR Hilbert filter. Table 2 gives the RMSE of FIR Hilbert filter with respect to order m [50]. The 32-tap FIR Hilbert filter is chosen as the RMSE is very low. Figure 14 shows the envelope detected data of the RF signal obtained from the AFE with the 32-tap FIR Hilbert filter.

Table 2. Normalized RMSE of the FIR Hilbert filter with respect to order m.

FIR Hilbert Filter Order	Normalized RMSE Value
16	0.0109
20	0.0096
24	0.0092
28	0.0091
32	0.0090

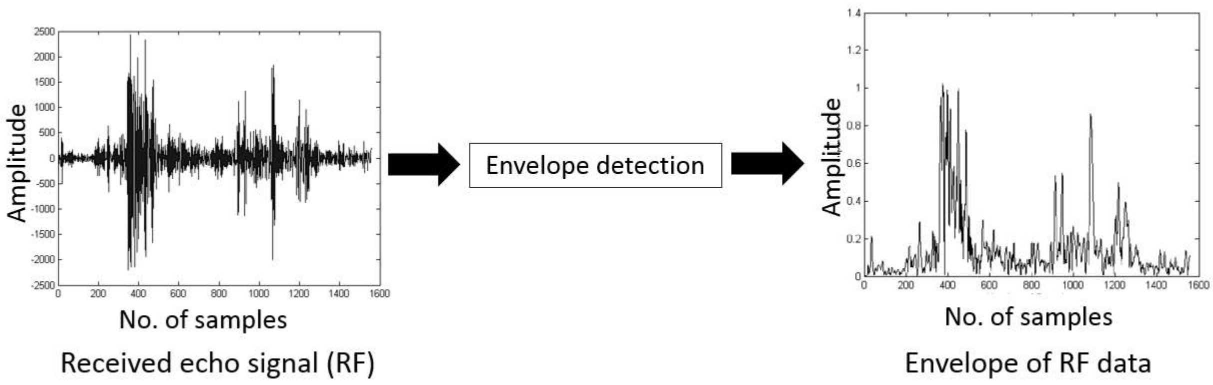


Figure 14. Envelope detected RF signal.

3.3.2. Range Compression

The envelope detected RF signal has a high dynamic range over 80 dB and, hence, does not fit the resolution of the monitors. A dynamic range compression technique is used to compress the RF signal to display on devices having a low dynamic range, which is around seven or eight bits. A nonlinear compression technique, like log compression, is used to selectively compress the large input signals. Figure 15 shows the dynamic range compressed data of the envelope signal (shown in Figure 14). To reduce the computational complexity, the LUT approach is employed, and 64 KB of memory are utilized for computing the log compression.

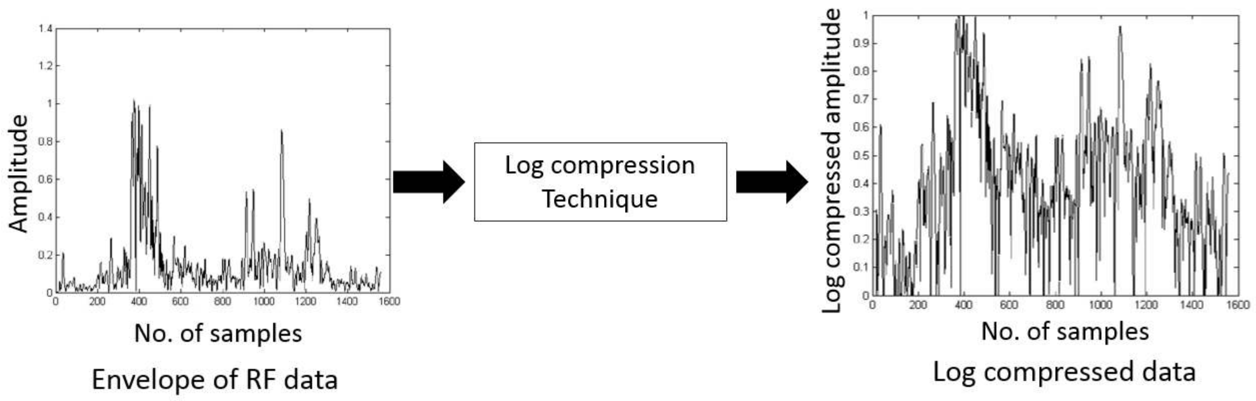


Figure 15. Dynamic range compression.

3.4. Back-End Processing

In back-end processing, the scan conversion and interpolation algorithms are performed on dynamic range compressed data. In phased array excitation, the dynamically compressed RF data are in the polar coordinate system. To display data on the monitor, the data in the polar coordinate system should be transformed to the rectangular coordinate system. The coordinate rotational digital computer (CORDIC) algorithm is employed for computing the square root and arctangent operations. The scanned data are inadequate to fit the resolution of the display monitor. A 2 $\times$ 2 linear interpolation method is used to increase the number of data points, such that it fits the resolution of the screen.

The processed data in the Kintex board are transmitted to Raspberry Pi through the RS-232 protocol with speeds up to 2,000,000 baud and transmit/receive 100 bytes in one transaction. The RS-232 protocol uses only Tx and Rx lines. A simple flow control between two boards is implemented through a transmit and wait protocol. The RS-232 controller of Kintex sends a frame triggered by the Tx signal from Raspberry Pi and becomes idle. A buffer matrix is created in the SDRAM of Raspberry Pi to store the frame. Raspberry Pi running a GUI application is programmed using the GTK+2.0 libraries [51]. The GTK toolkit allows the stored frame buffer to be displayed on a portion of the display window as a gray scale image, and an interrupt is generated for every one millisecond, which forces the processor to fetch a new frame from the Kintex, update the buffer and display the updated buffer in the window. The typical processing time for fetching the image, updating the buffer and displaying takes about 45 ms, and a streaming rate of approximately 22 fps is achieved through the RS-232 protocol. If a new frame arrives, it overwrites the existing frame in the buffer. Before overwriting, the existing frame is dumped into a data file.

4. Experimental Setup and Results

The prototype of the FPGA-based portable ultrasound scanning system is shown in Figure 16. The transducer elements are excited with a voltage of ± 50 V with a 1-A current. A custom-made power supply is designed to generate the required voltage, as shown in Figure 17. A 230-V, 50-Hz supply is given as the input to the power supply. A step down transformer is used to convert AC supply to ±50-V DC voltage with a 1-A current. These voltages are connected to the HV pulser for generating HV pulses to excite the transducer.

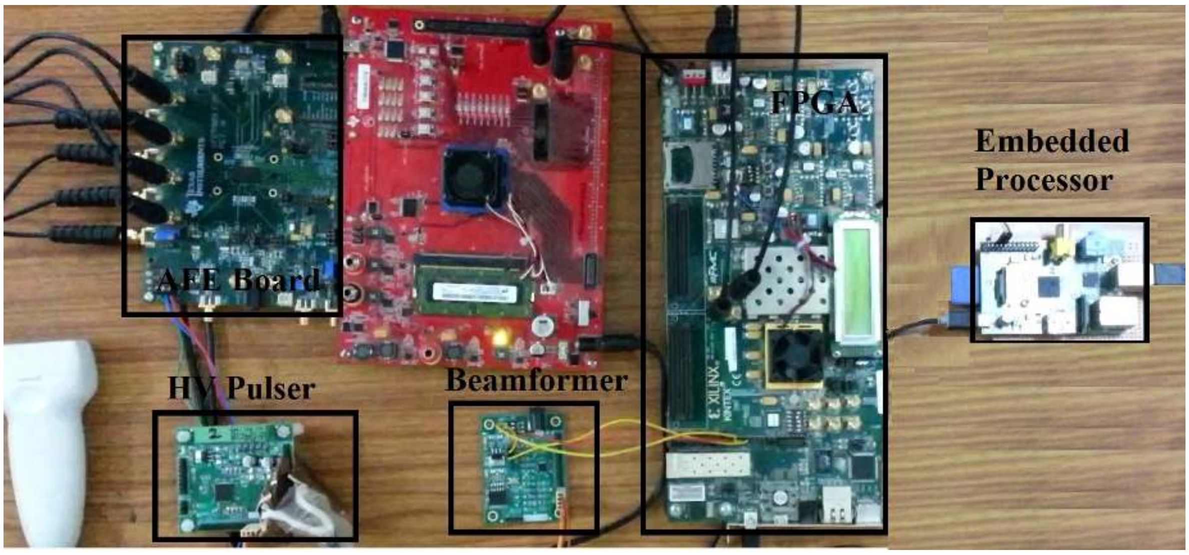


Figure 16. Prototype of the proposed FPGA-based portable ultrasound scanning system.

The performance of the proposed PUS system is evaluated by scanning a tissue-mimicking gelatin phantom. The specifications of the scanner used for scanning the phantom are shown in Table 3. Out of 64 elements, eight transducer elements are active all of the time in transmitting ultrasound waves and receiving echoes from the tissue. The delay values of the channel are automatically updated depending on the receive focal zones.

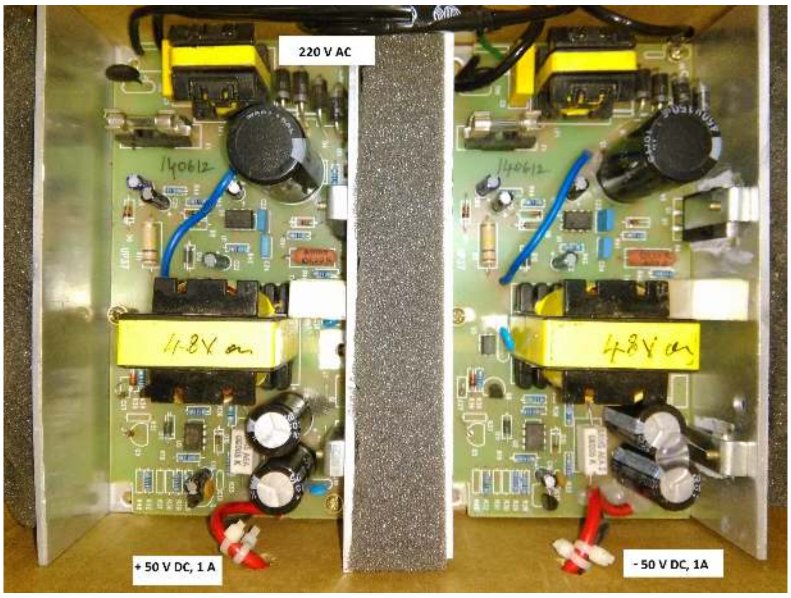


Figure 17. Power module designed for generating ±50 V DC.

Table 3. Parameters used for scanning the gelatin phantom.

Specifications	**Value**
Transmit frequency	5 MHz
Number of channels	8
Kerf of transducer	0.025 mm
Element width	0.154 mm
Depth of scan	50 mm
Number of scanlines	192

The performance of the proposed portable ultrasound system is evaluated by comparing to the commercially available platforms, like the PCI Extensions for Instrumentation (PXI) system and Biosono.

4.1. Ultrasound Scanning System Based on PCI Extensions for Instrumentation

National Instrument's (NI) PXIe-1078 [52] is a rugged PC-based platform for measurement and automation systems. It has two eight-channel high speed analog input modules. We have interfaced our data acquisition board to the PXI analog channels. Spartan-3 FPGA [53] is used to establish synchronization between PXI and our data acquisition module. Figure 18 shows the PXI platform hardware setup. The portable ultrasound board is provided with control pins in the FPGA to program the transmission parameters for the logic pulse driver. The FPGA generates a transmit control signal, which is used for synchronization of the PXI with the ultrasound front-end board. The PXI system uses LABVIEW 2013 software to acquire high speed ultrasound data and to reconstruct the ultrasound image. A linear array transducer with a center frequency of 5 MHz is used to scan the gelatin phantom.

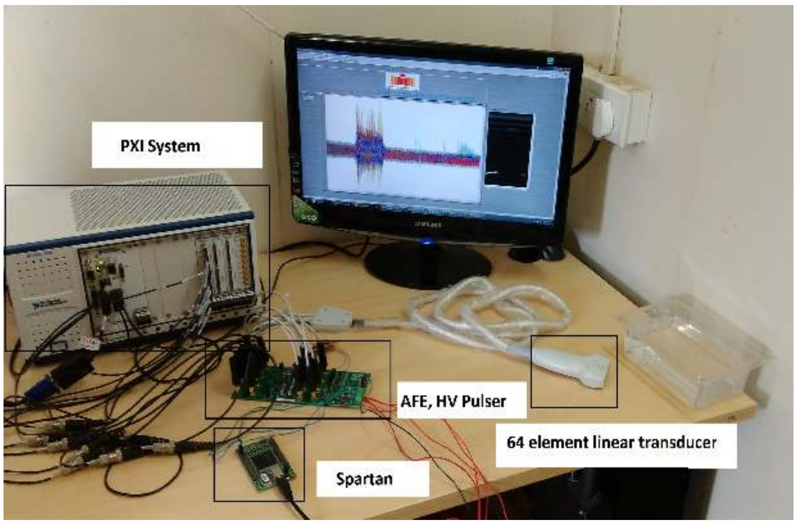
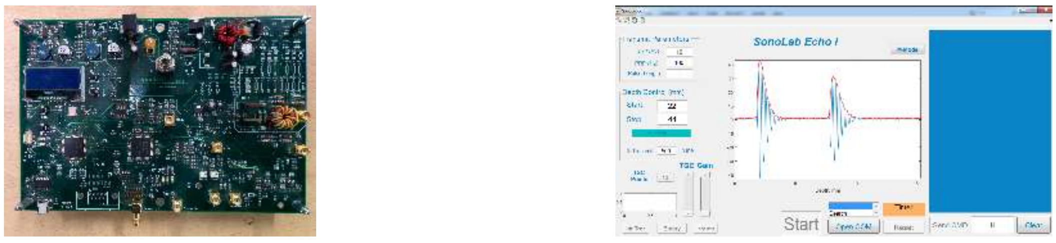


Figure 18. Ultrasound system based on the PCI Extensions for Instrumentation (PXI) platform.

4.2. Biosono Ultrasound Platform

The proposed portable ultrasound data acquisition module is compared to the Biosono ultrasound platform. Figure 19a shows the Biosono ultrasound board designed for A-mode and M-mode imaging applications [54]. The echos are sampled and digitized at a rate of 100 MHz and streamed out through the serial port at 115,200 bps. The MATLAB GUI (Figure 19b) is provided for viewing the real-time echo.



(a) **(b)**

Figure 19. Biosono ultrasound board. (**a**) Biosono board; (**b**) GUI for the Biosono board.

The visual comparison of the gelatin phantom image reconstructed from Biosono, NI and the proposed FPGA-based portable ultrasound scanning system is shown in Figure 20. The geometrical shapes look similar in all of the images, validating the performance of the proposed FPGA-based portable ultrasound scanning system.

The resources utilized in the Kintex-7 FPGA for implementing the beamforming, mid-end and back-end algorithms are analyzed using the Xilinx ISE 14.4 synthesis tool, and this is summarized in Table 4. The ultrasound signal processing algorithms use only 53% of the available LUT flip flops, so the remaining resources can be utilized for implementing other complex beamforming and image-processing algorithms.

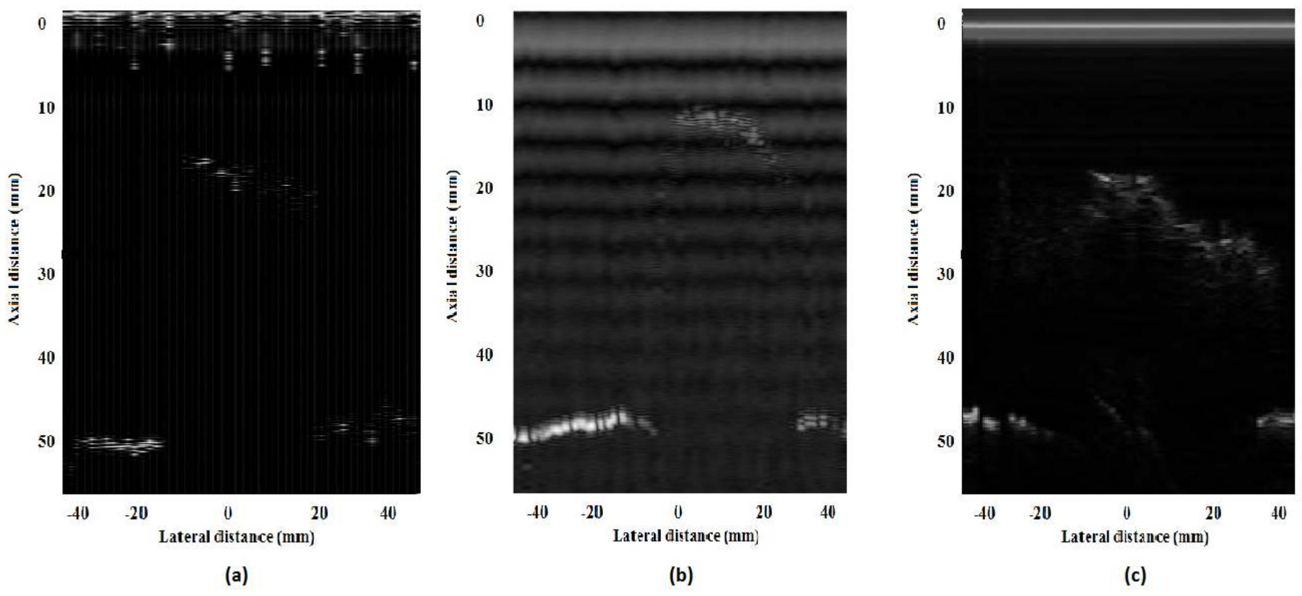
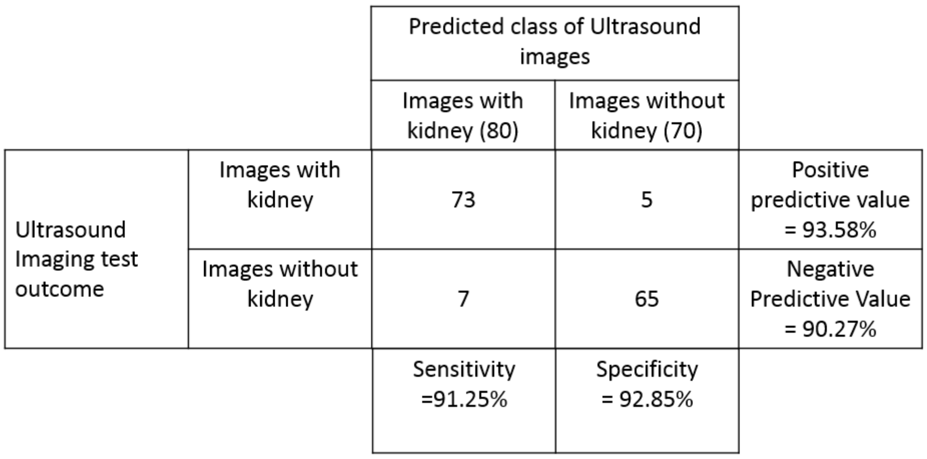


Figure 20. Image acquired and reconstructed from: (**a**) the Biosono platform; (**b**) the NIPXI platform; (**c**) the proposed FPGA-based portable ultrasound scanning system.

Table 4. Resources utilized in the Kintex-7 board for implementing the beamforming, mid-end and back-end processing algorithms.

Resources Available	Resource Used	Percentage
Slice Logic Utilization		
Number of slice registers (437,200)	7351	1%
Number of slice LUTs (218,600)	6086	2%
Number used as logic (218,600)	5329	2%
Slice Logic Distribution		
Number of LUT flip flop pairs used	7,960	
Number with an unused flip flop (7,960)	1,867	23%
Number with an unused LUT (7,960)	1874	23%
Number of fully-used LUT-FF pairs (7,960)	4219	53%
Number of unique control sets	382	
IO Utilization		
Number of IOs (250)	60	24%
Specific Feature Utilization		
Number of block RAM/FIFO (545)	64	11%
Number of global clock buffers BUFG/BUFGCTRLs (32)	16	18%

The performance of the automatic kidney detection algorithm is evaluated using the confusion matrix discussed in Section 2.2. The confusion matrix of the proposed automatic kidney detection algorithm is shown in Figure 21.



		Predicted class of Ultrasound images		
		Images with kidney (80)	Images without kidney (70)	
Ultrasound Imaging test outcome	Images with kidney	73	5	Positive predictive value = 93.58%
	Images without kidney	7	65	Negative Predictive Value = 90.27%
		Sensitivity =91.25%	Specificity = 92.85%	

Figure 21. Confusion matrix for the proposed automatic kidney detection algorithm.

The kidney detection algorithm was tested on 80 kidney images (normal: 62; cyst: 10; stone: 8) and 70 non-kidney images (liver: 18; heart: 17; carotid: 15; spleen: 20). The kidney detection algorithm performed with an accuracy of 92% (138 out of 150), a sensitivity of 91.25%, a specificity of 92.85%, a positive predictive value of 93.58% and a negative predictive value of 90.27% in detecting the kidney in ultrasound images. False negatives (seven images) include two kidney images with cysts and one kidney image with stones. Multiple detections of kidney are an inherent property of the Viola–Jones algorithm, which is a result of detecting the organs at multiple scales, as shown in Figure 22. Multiple windows with overlapping regions of interest are merged into a single window by averaging the coordinates of the window. The merging is allowed only if the kidneys have the minimum number of detections. Having a low threshold for multiple detections give rise to high false positives, and a high threshold leads to high false negatives. From observations, it is found that a threshold of 50 gives the maximum accuracy for detecting the kidney. The detection of kidney using the Viola–Jones algorithm is shown in Figure 23. Some of the false positives that are detected in localizing kidney in the images are shown in Figure 24.

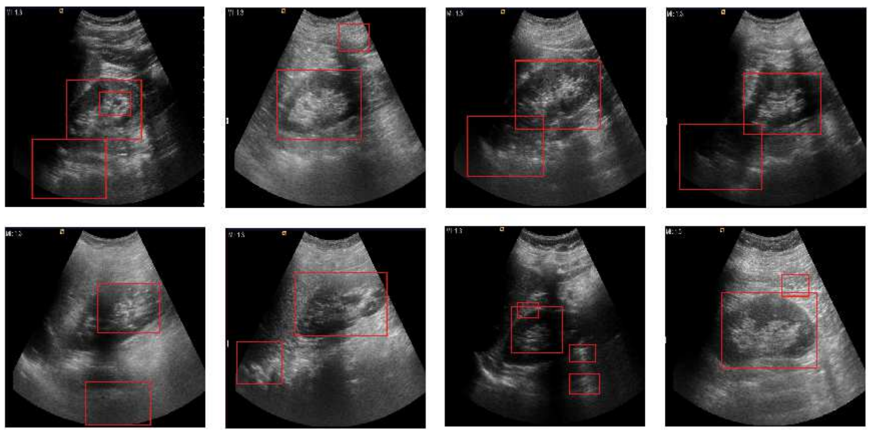


Figure 22. Multiple detections of kidney in ultrasound images.

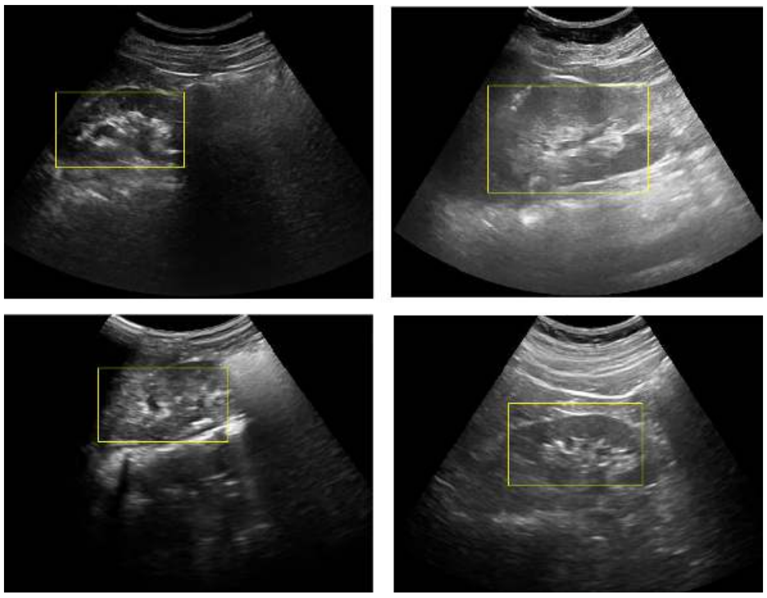


Figure 23. Automatic kidney detection.

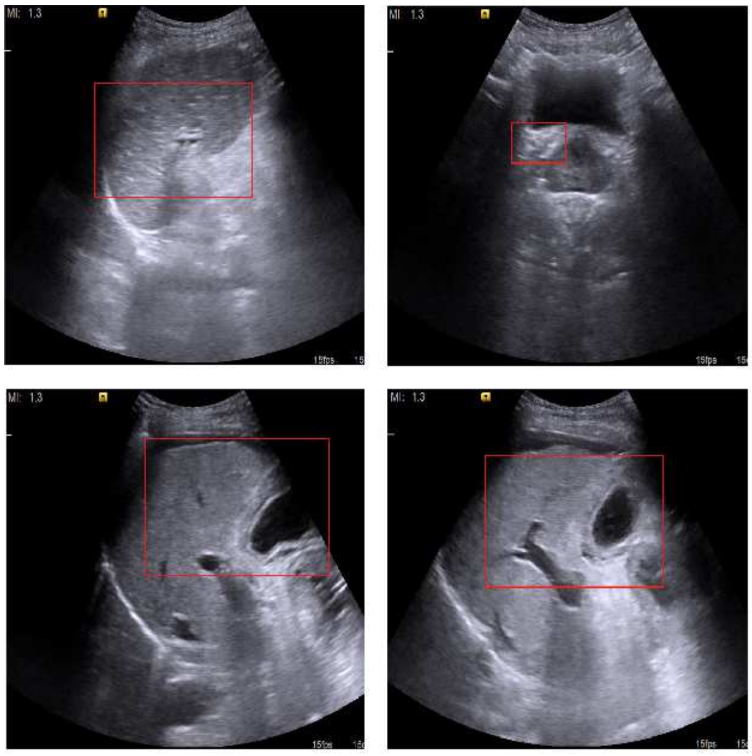


Figure 24. False positives detected in localizing the kidney.

The automatic kidney detection algorithm on Raspberry Pi-2 (900 MHz, 1 GB RAM) takes 60 ms to detect the presence of kidney in a 480 × 640 resolution frame. The kidney detection algorithm is used as an add-on feature in the Raspberry Pi board. By enabling the kidney detection option, the system directly applies the algorithm on the incoming frame, which is stored in the SDRAM.

For evaluating the kidney detection algorithm in real time, an ultrasound video is acquired from 10 patients using the Siemens S1000 through Ultrasonix 500RP [55]. The scanned kidney videos have a frame rate of 15 fps, and this is transferred to the SDRAM of the Raspberry Pi processor. The kidney detection algorithm installed on the Raspberry Pi processor is successfully able to detect the kidney in live streaming video.

Implementing the Viola–Jones algorithm on an FPGA requires 11,505 slice registers, 7,872 slice flip flops, 20,901 four-input LUTs, 44 block RAM (BRAM) and one global clock (GCLK) buffer [56]. Optimized algorithms have been proposed for fast real-time implementation of the Viola–Jones algorithm to detect faces in video having frame rates greater than 100. The Viola–Jones algorithm is effectively implemented on various computing platforms, like FPGAs, DSPs, GPUs, mobile platforms, *etc.* [56–59]. The same computing platforms are also used for realizing PUS systems, so the Viola–Jones algorithm can be implemented on these platforms for real-time detection of kidney. From Table 4, the Kintex-7 has enough resources to implement the Viola–Jones algorithm. Implementing the kidney detection algorithm on an FPGA platform will reduce the load on the ARM processor, which can be used for other tasks, like real-time recording, compression, data transferring, *etc.*

5. Discussions and Conclusions

In this paper, we implemented entire B-mode ultrasound signal processing algorithms on a single FPGA-based Kintex-7 board with real-time automatic kidney detection in ultrasound video. The Viola–Jones algorithm proposed for face detection proved effective at detecting kidney in ultrasound images. The Viola–Jones algorithm proves very effective at capturing the diversified shapes of kidney, where other geometrical models fail to capture them. The Viola–Jones localizes the kidney by placing the bounding box around the kidney. The bounding box may not capture the entire organ inside it and can intersect the organ in the image. The bounding box can be used as the initialization to other algorithms, like ASM, active contours, *etc.*, to capture the contour of a kidney. The automatic organ detection, which is also termed region of interest detection, can be very beneficial for performing high signal processing tasks, like segmentation and computer-aided diagnosis. Automatic organ detection is also beneficial for image compression, where non-organ parts are lossy compressed and loss-less compression is employed for organ parts. These sorts of data-dependent compression algorithms are very useful for tele-radiology applications. The hardware architecture proposed in this paper provides flexibility for installing new image processing algorithms in the system, this is done by installing the algorithms on an ARM processor. As the future extension of this work, we would like to develop signal processing algorithms to detect more organs in the image and CAD algorithms to automatically detect diseases in kidney without manual intervention.

Acknowledgments

This work is funded by the Department of Science and Technology (DST), India, under the IUATC-IoT eHealth project with Sanction Number SR/RCU-DST/IUATC Phase2/2012-iith(G).

Author Contributions

P. Rajalakshmi and U. B. Desai conceived of and designed the experiments. Punit Kumar, Chandrasekhar Dusa, Vivek Akkala, Suresh Puli, Harsha Ponduri, K. Divya Krishna and R. Bharath performed the experiments. Vivek Akkala and R. Bharath analyzed the data. P. Rajalakshmi, U. B. Desai

and S. N. Merchant provided resources for building the system. Mohammed Abdul Mateen provided the ultrasound database and helped in the analysis of the data.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Lapostolle, F.; Petrovic, T.; Lenoir, G.; Catineau, J.; Galinski, M.; Metzger, J.; Chanzy, E.; Adnet, F. Usefulness of hand-held ultrasound devices in out-of-hospital diagnosis performed by emergency physicians. *Am. J. Emerg. Med.* **2006**, *24*, 237–242.
2. Kim, Y.; Kim, J.H.; Basoglu, C.; Winter, T.C. Programmable ultrasound imaging using multimedia technologies: A next-generation ultrasound machine. *IEEE Trans. Inf. Technol. Biomed.* **1997**, *1*, 19–29.
3. Sikdar, S.; Managuli, R.; Gong, L.; Shamdasani, V.; Mitake, T.; Hayashi, T.; Kim, Y. A single mediaprocessor-based programmable ultrasound system. *IEEE Trans. Inf. Technol. Biomed.* **2003**, *7*, 64–70.
4. Kim, G.-D.; Yoon, C.; Kye, S.; Lee, Y.; Kang, J.; Yoo, Y.; Song, T.-K. A single FPGA-based portable ultrasound imaging system for point-of-care applications. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control.* **2012**, *59*, 1386–1394.
5. Dusa, C.; Rajalakshmi, P.; Puli, S.; Desai, U.B.; Merchant, S.N. Low complex, programmable FPGA based 8-channel ultrasound transmitter for medical imaging researches. In Proceedings of the 16th IEEE International Conference on e-Health Networking, Applications and Services (Healthcom), Natal, Brazil, 15–18 October 2014; pp. 252–256.
6. Dusa, C.; Kalalii, S.; Rajalakshmi, P.; Rao, O. Integrated 16-channel transmit and receive beamforming ASIC for ultrasound imaging. In Proceedings of the 2015 28th International Conference on VLSI Design (VLSID), Bangalore, India, 3–7 January 2015; pp. 215–220.
7. Kim, K.C.; Kim, M.J.; Joo, H.S.; Lee, W.; Yoon, C.; Song, T.; Yoo, Y. Smartphone-based portable ultrasound imaging system: A primary result. In Proceedings of the 2013 IEEE International Ultrasonics Symposium (IUS), Prague, Czech, 21–25 July 2013; pp. 2061–2063.
8. Tomov, B. G.; Jensen, J.A. Compact FPGA-based beamformer using oversampled 1-bit A/D converters. *Trans. Ultrason. Ferroelectr. Freq. Control.* **2005**, *52*, 870–880.
9. Ranganathan, K.; Santy, M.K.; Blalock, T.N.; Hossack, J.; Walker, W.F. Direct sampled I/Q beamforming for compact and very low-cost ultrasound imaging. *Trans. Ultrason. Ferroelectr. Freq. Control.* **2004**, *51*, 1082–1094.
10. Karaman, M.; Li, P.-C.; O'Donnell, M. Synthetic aperture imaging for small scale systems. *Trans. Ultrason. Ferroelectr. Freq. Control.* **1995**, *42*, 429–442.
11. Synnevag, J.-F.; Austeng, A.; Holm, S. A low-complexity data-dependent beamformer. *Trans. Ultrason. Ferroelectr. Freq. Control.* **2011**, *58*, 281–289.
12. Kintex-7. Available online: http://www.xilinx.com productssilicondevices/fpga/kintex-7. (accessed on 4 February 2014).

13. Raspberry Pi hardware. Available online: http://www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM- Peripherals.pdf (accessed on 5 March 2015).
14. Aysu, A.; Sayilar, G.; Hamzaoglu, I. A low energy adaptive hardware for H. 264 multiple reference frame motion estimation. *IEEE Trans. Consum. Electron.* **2011**, *57*, 1377–1383.
15. Botella, G.; Garcia, A.; Rodriguez-Alvarez, M.; Ros, E.; Meyer-Baese, U.; Molina, M.C. Robust bioinspired architecture for optical-flow computation. *IEEE Trans. Very Large Scale Integr. Syst.* **2010**, *18*, 616–629.
16. Botella, G.; Martin H.J.A.; Santos, M.; Meyer-Baese, U. FPGA-based multimodal embedded sensor system integrating low-and mid-level vision. *Sensors* **2011**, *11*, 8164–8179.
17. Nunez-Yanez, J.L.; Nabina, A.; Hung, E.; Vafiadis, G. Cogeneration of fast motion estimation processors and algorithms for advanced video coding. *IEEE Trans. Very Large Scale Integr. Syst.* **2012**, *20*, 437–448.
18. Gonzalez, D.; Botella, G.; Garcia, C.; Prieto, M.; Tirado, F. Acceleration of block-matching algorithms using a custom instruction-based paradigm on a Nios II microprocessor. *EURASIP J. Adv. Signal Process.* **2013**, *1*, 1–20.
19. Gonzalez, D.; Botella, G.; Meyer-Baese, U.; Garcia, C.; Sanz, C.; Prieto-Matias, M.; Tirado, F. A Low cost matching motion estimation sensor based on the NIOS II microprocessor. *Sensors* **2012**, *12*, 13126–13149.
20. Bilal, M.; Masud, S. Efficient computation of correlation coefficient using negative reference in template matching applications. *IET Image Process.* **2012**, *6*, 197–204.
21. Git. Available online: https://wiki.videolan.org/Git (accessed on 7 August 2014).
22. Marcos, M.-F.; Alberola-Lopez, C. An approach for contour detection of human kidneys from ultrasound images using Markov random fields and active contours. *Med. Image Anal.* **2005**, *9*, 1–23.
23. Cootes, T.F.; Taylor, C.J.; Cooper, D.H.; Graham, J. Active shape models-their training and application. *Comput. Vis. Image Underst.* **1995**, *61*, 38–59.
24. Zheng, Y.; Barbu, A.; Georgescu, B.; Scheuering, M.; Comaniciu, D. Four-chamber heart modeling and automatic segmentation for 3-D cardiac CT volumes using marginal space learning and steerable features. *IEEE Trans. Med. Imaging.* **2008**, *27*, 1668–1681.
25. Cuingnet, R.; Prevost, R.; Lesage, D.; Cohen, L.D.; Mory, B.; Ardon, R. Automatic detection and segmentation of kidneys in 3D CT images using random forests. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI*; Springer: Berlin, Germany, 2012; pp. 66–74.
26. Barbu, A.; Suehling, M.; Xu, X.; Liu, D.; Zhou, S.K.; Comaniciu, D. Automatic detection and segmentation of lymph nodes from CT data. *IEEE Trans. Med. Imag.* **2012**, *31*, 240–250.
27. Kass, M.; Witkin, A.; Terzopoulos, D. Snakes: Active contour models. *Int. J. Comput.* **1988**, *1*, 321–331.
28. Blake, A.; Isard, M. *Active Contours*; Springer Verlag: Berlin, Germany, 1998.
29. Malladi, R.; Sethian, J.; Vemuri, B. Shape modelling with front propagation: A level set approach. *IEEE Trans. Pattern Anal. Mach. Intell.* **1995**, *2*, 158–175.
30. Bernard, O.; Friboulet, D.; Thevenaz, P.; Unser, M. Variational B-spline level-set: A linear filtering approach for fast deformable model evolution. *IEEE Trans. Image Process.* **2009**, *18*, 1179–1191.

31. Cootes, T.F.; Beeston, C.; Edwards, G.J.; Taylor, C.J. A unified framework for atlas matching using active appearance models. In *Information Processing in Medical Imaging*; Springer: Berlin, Germany, 1999; pp. 322–333.
32. Xie, J.; Jiang, Y.; Tsui, H. Segmentation of kidney from ultrasound images based on texture and shape priors. *IEEE Trans. Med. Imaging.* **2005**, *24*, 45–57.
33. Viola, P.; Jones, M. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Kauai, HI, USA, 8–14 December 2001.
34. Prinosil, J. Local descriptors based face recognition engine for video surveillance systems. In Proceedings of the 36th IEEE International Conference on Telecommunications and Signal Processing (TSP), Rome, Italy, 2–4 July 2013 .
35. Chu, C.; Bai, J.; Liu, L.; Wu, X.; Zheng, G. Fully automatic segmentation of hip CT images via random forest regression-based Atlas selection and optimal graph search-based surface detection. In *Computer Vision—ACCV*; Springer International Publishing: Berlin, Germany, 2014; pp. 640–654.
36. Kamil, R.; Masek, J.; Burget, R.; Benes, R.; Zavodna, E. Novel method for localization of common carotid artery transverse section in ultrasound images using modified Viola–Jones detector. *Ultrasound Med. Biol.* **2013**, *39*, 1887–1902.
37. Mohan, A.; Papageorgiou, C.; Poggio, T. Example-based object detection in images by components. *PAMI* **2001**, *23*, 349–361.
38. Lienhart, R.; Maydt, J. An extended set of haar-like features for rapid object detection. In Proceedings of the 2002 International Conference on Image Processing, Rochester, NY, USA, 22–25 September 2002; pp. 349–364.
39. Bradley, D.; Roth, G. Adaptive thresholding using the integral image. *J. Graph. Gpu Game Tools.* **2007**, *12*, 13–21.
40. Crow, F.C. Summed-area tables for texture mapping. *ACM Siggraph Comput. Graph.* **1984**, *18*, 207–212.
41. Osuna, E.; Freund, R.; Giros, F. Training support vector machines: An application to face detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, 17–19 June 1997.
42. Papageorgiou, C.P.; Oren, M.; Poggio, T. A general framework for object detection. In Proceedings of the Sixth IEEE International Conference on Computer Vision, Bombay, India, 4–7 January 1998.
43. Lienhart, R.; Kuranov, A.; Pisarevsky, V. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. *Pattern Recognit.* **2003**, *2781*, 297–304.
44. Wang, Y.-Q. An analysis of the Viola–Jones face detection algorithm. *Image Process. Line* **2014**, *4*, 128–148.
45. LM96570 Ultrasound configurable transmit beamformer. Available online: http://www.ti.com.cn/cn/lit/ds/symlink/lm96570.pdf (accessed on 10 March 2013).
46. Fully integrated, eight-channel ultrasound analog front end. Available online: http://www.ti.com/lit/ds/symlink/afe5808.pdf (accessed on 15 January 2013).

47. Vivek, A.; Rajalakshmi, P.; Kumar, P.; Desai, U.B. FPGA based ultrasound backend system with image enhancement technique. In Proceedings of the 2014 IEEE Biosignals and Biorobotics Conference, Salvador, Brazil, 26–28 May 2014; pp. 1–5.
48. Oppenheim, A.V.; Schafer, R.W. *Discrete-Time Signal Processing*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1989.
49. FIR filtering in PSoCTM with application to fast hilbert transform. Available online: www.cypress.com/file/101096 (accessed on 13 March 2013).
50. Hassan, M.A.; Kadah, Y.M. Digital signal processing methodologies for conventional digital medical ultrasound imaging system. *Am. J. Biomed. Eng.* **2013**, *3*, 14–30.
51. The GTK+ project. Available online: http://www.gtk.org/ (accessed on 17 June 2014).
52. NI PXIe-1078. Available online: http://sine.ni.com/nips/cds/view/p/lang/en/nid/209253 (accessed on 10 July 2014).
53. Multiple domain-optimized platforms Spartan-3 generation. Available online: http://www.xilinx.com/products/silicon-devices/fpga/spartan-3.html (accessed on 8 September 2014).
54. Biosono. Available online: http://www.biosono.com/ElctLgd/ElctLgd.php?id=SE1_DSC (accessed on 15 May 2014).
55. Thaddeus, W.; Zagzebski, J.; Varghese, T.; Chen, Q.; Rao, M. The ultrasonix 500RP: A commercial ultrasound research interface. *Trans. Ultrason. Ferroelectr. Freq. Control.* **2006**, *53*, 1772–1782.
56. Lai. H.-C.; Savvides, M.; Chen, T. Proposed FPGA hardware architecture for high frame rate ($>>$ 100 fps) face detection using feature cascade classifiers. In Proceedings of the IEEE International Conference on Biometrics: Theory, Applications, and Systems, Crystal City, VA, USA, 27–29 September 2007; pp. 1–6.
57. Cho, J.; Mirzaei, S.; Oberg, J.; Kastner, R. Fpga-based face detection system using HAAR classifiers. In Proceedings of the 17th ACM/SIGDA International Symposium on Field Programmable Gate Arrays ACM, Monterey, CA, USA, 22–24 February 2009; pp. 103–112.
58. Hefenbrock, D.; Oberg, J.; Thanh, N.T.N.; Kastner, R.; Baden, S.B. Accelerating Viola–Jones face detection to fpga-level using gpus. In Proceedings of the IEEE International Symposium on Field-Programmable Custom Computing Machines, Charlotte, NC, USA, 2–4 May 2010; pp. 11–18.
59. Ren, J.; Kehtarnavaz, N.; Estevez, L. Real-time optimization of Viola -Jones face detection for mobile platforms. In Proceedings of the Circuits and Systems Workshop: System-on-Chip—Design, Applications, Integration, and Software, Dallas, TX, USA, 19–20 October 2008; pp. 1–4.