

Acoustic Real-Time, Low-Power FPGA Based Obstacle Detection For AUVs.

S. Karabchevsky

Department of Electro-Optical
Engineering, Ben Gurion
University of The Negev Israel

D. Kahana

Department of Electro-Optical
Engineering, Ben Gurion
University of The Negev Israel

H. Guterman

Department of Electrical and
Computer Engineering, Ben Gurion
University of The Negev, Israel

Abstract — The underwater robots called Unmanned Underwater Vehicles (UUVs) take over complex and dangerous underwater missions that were previously performed by humans. These vehicles operate in the unknown environments and make their own decisions within the mission based on the readings of the sensors, without any link with a human operator. Independent of the mission, it is critical for the AUVs to be able to avoid submerged obstacles such as cliffs, wrecks, and floating mines. The AUV typically uses underwater imaging sonar that has several drawbacks for obstacle detection purposes, and therefore requires complex image processing algorithms. Due to the imaging sonar limitations, addressing obstacle detection using conventional software algorithms cannot meet an AUV's real-time, low power requirements. A low-power FPGA algorithm for underwater obstacle detection that is based on local image histogram entropy is proposed. The algorithm maintains a real-time reliable performance while meeting the AUV low power budget.

T

I. INTRODUCTION

he most commonly used UUVs are the Remotely Operated Vehicles (ROVs). ROVs are remotely controlled through a wired connection by a remote human operator from a mother vessel. In present times a new class of UUVs is starting to appear on real missions, they are called Autonomous Underwater Vehicles (AUVs). These vehicles operate in the unknown environments and make their own decisions within the mission based on the readings of the sensors, without any link with a human operator. Typical missions for AUVs include inspection of submersed structures, sea-floor exploration, ship wreck discovery, mine detection, and pipeline and cable inspection [1]. Independent of the mission, it is critical for the AUVs to be able to avoid submerged obstacles such as cliffs, nets, wrecks, and floating mines. In ROVs obstacle avoidance is accomplished by a remote operator, the AUV does not have that privilege due to lack of connection between the vehicle and the operator. Due to the lack of an operator, obstacle detection and path planning algorithms must be located onboard the AUV. The algorithms must perform in real-time in order not to miss any obstacle and to leave time for avoidance maneuvering. In addition, the algorithms have to be power efficient due to limited power payload of the vehicle.

The AUVs have to use underwater imaging sonar for

obstacle detection [2] because conventional optical imaging in an underwater environment suffers from serious drawbacks. Typical drawbacks are high light scattering [3], light absorption, and lack of illumination in deep water. The imaging sonar does not require illumination and can obtain larger ranges than standard optical imagers with no dependence on water clarity. Despite the large range advantage, imaging sonar suffers from a low Signal to Noise Ratio (SNR) compared to the conventional optical imagers. The main reason for the low SNR is the speckle noise that is caused by acoustic wave interference [4]. In addition, strong acoustic reflections from the seabed exist when operating close to the seabed, as can be seen in Fig. 1. The imaging sonar drawbacks need to be resolved with on-board algorithms that require a large amount of processing power and cannot be addressed using standard software techniques in real time while fitting the AUV power budget.

A real-time low power obstacle detection hardware method for AUVs hosted on Altera StratixIII FPGA is proposed. The method is based on a hardware modified Frost speckle noise filter and local image histogram entropy segmentation, which is not sensitive to the high speckle noise or to the background returns.

The rest of the paper is organized as follows. In section II, system overview and concept are presented. Section III presents the hardware algorithm implementation. Results are presented in section IV, and finally, conclusions in section V.

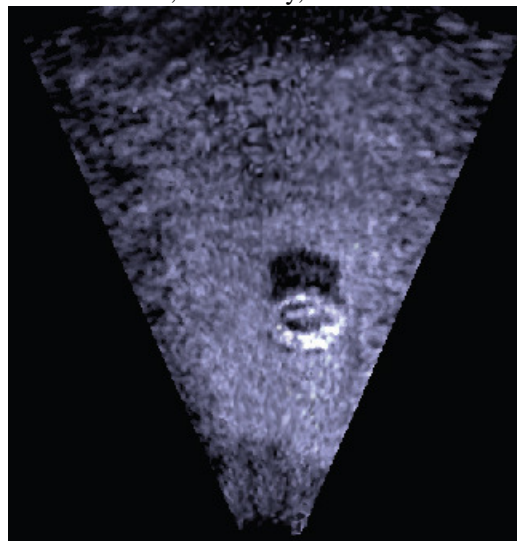


Fig. 1. Sonar image of a tire on the sea floor (from BlueView test data); speckle noise and seabed returns can be easily seen.

A. Target Application

The target of this research is to develop an obstacle detection system for AUVs operating close to the seabed. The movements of the AUV are measured by on-board Gyros and accelerometers to produce an updated position and orientation in world coordinates. The AUV is equipped with horizontally positioned imaging sonar for data acquisition, onboard computer, and an FPGA board. The main missions for the AUV are pipeline/cable inspection and mine detection.

B. Target Sonar

A BlueView P450 Forward Looking Sonar was used for collecting data [5]. The sonar was configured to export data in polar coordinates to reduce computational cost. The sonar has a $45^\circ \times 15^\circ$ field of view, 256 beams with width of $1^\circ \times 15^\circ$ and 0.18° spacing, maximum range of 137 m and range resolution of 2 in, operating frequency of 450 kHz, and update rate of 10 Hz. The images that were acquired are of 256×452 pixels with a resolution of 16 bits per pixel.

II. SYSTEM STRUCTURE

The system is based on two main components, a computer that is responsible for sonar data reception and feature extraction, and an FPGA board that is responsible for all the critical time-consuming parts of the algorithms as described in Fig. 2. The onboard computer receives data from the sonar and forwards it to the FPGA board as a stream of pixels using a PCI Express interface. At the FPGA board in the first stage speckle noise suppression is carried out using an adaptive Frost [6] filter with 17×17 kernel size, then local image histogram entropy is calculated in a 9×9 pixel neighborhood to provide an informational measure. In parallel, a fixed threshold is applied on the filtered image to remove low echo returns that cannot be related to obstacles. At a final stage, the hysteric threshold of the entropy takes place and the result is returned back to the computer for feature extraction. The following sections describe in detail the system algorithms.

A. Frost Filter Speckle Noise Suppression

Speckle noise is a granular noise that exists in coherent imaging such as laser imaging, radar, and ultrasonic imaging. It is a speckle pattern of random intensity produced by the mutual interference of a set of wave fronts. Speckle noise is a multiplicative noise, i.e., it is in direct proportion to the local intensity of the area. The speckle noise has to be removed before calculating local image histogram entropy, because both a speckle noise and an obstacle result in high entropy. Two common approaches exist to deal with that type of noise. The first approach is to average several images acquired from the same scene, which is called multi-look processing or super-resolution [7]. In that way the speckle noise will be reduced due to its random nature, while the observed scene will not be degraded. This method is problematic for the AUV application, due to high computational load and large latency. The second technique is based on filtering the speckle noise based on a single image using a two dimensional filter. Standard filters such as Low-Pass, Gaussian, and Median generally degrade the observed scene. Contrary to the standard filters, adaptive filters take local image information

into consideration while carrying out the filtration process.

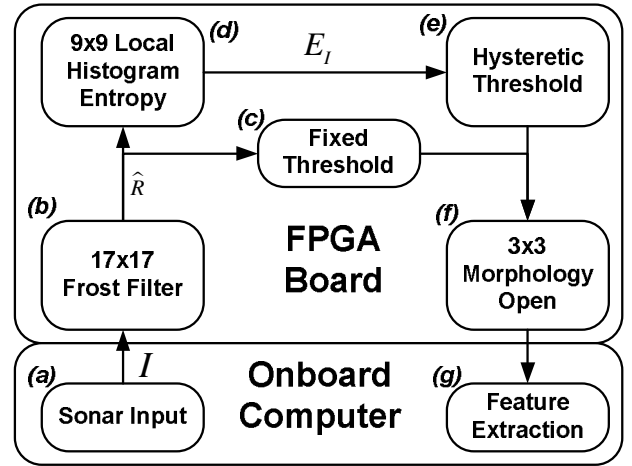


Fig. 2. System architecture. Sonar image is acquired by onboard computer and forwarded to FPGA board (a), then the image is filtered using Frost filter to suppress speckle noise (b). A fixed threshold is applied to remove low echo returns (c); in parallel a local histogram entropy of the image is calculated (d) and hysteric threshold takes place (e). Prior to the feature extraction (g), morphological open operation used (f).

Adaptive filters can reduce speckle noise in homogeneous areas while preserving texture and high frequency information in heterogeneous areas. During our work it became clear that the adaptive Frost [6] filter in a 17×17 pixel neighborhood with $K=1$ is the best choice for this task.

The Frost filter [6] estimates the observed scene by convolving the acquired image with an adaptive kernel, which changes its properties according to the local image statistics. The process can be defined by the following equation:

$$\hat{R}(x, y) = I(x, y) \otimes m(x, y) \quad (1)$$

where (x, y) is the image spatial coordinates, I is the acquired image, \hat{R} is the estimated scene, and $m(t)$ is adaptive kernel defined by:

$$m(x, y) = \frac{e^{-KC_I^2(x, y)|d|}}{\sum e^{-KC_I^2(x, y)|d|}}, C_I(x, y) = \frac{\sigma_I(x, y)}{\bar{I}(x, y)} \quad (2)$$

where K is the dumping factor (between 0.1 and 1), $|d| = |x - x_0| + |y - y_0|$ is the distance of the pixel from the kernel center. $C_I(x, y)$, $\sigma_I(x, y)$ and $\bar{I}(x, y)$ are the image variation, standard deviation and mean in the filter area.

B. Segmentation

Segmentation of sonar images can be very complex, different obstacles with different echo return levels can exist in the single image. In addition, a strong seabed return exists on bottom-following missions. Therefore, standard fixed threshold algorithms [8][9] and even adaptive ones may miss obstacles with a low echo return. During our research it became obvious that local image histogram entropy-based [10] segmentation provides outstanding results both in detection of obstacles and rejecting the seabed returns. The local histogram entropy is based on the Shannon Theorem [11] and was previously used for segmentation of optical images [12]. Entropy is the measure of the information content in a probability distribution, and can be defined by the

following formula:

$$E_l(x, y) = \sum_{i=0}^{N_{bins}} \frac{p(i)}{N_{bins}} \cdot \log\left(\frac{p(i)}{N_{bins}}\right) \quad (3)$$

where $p(i)$ is the value of the particular bin of the N_{bins} local histogram of the image in the $N \times N$ pixel neighborhood at (x, y) coordinates. As our experiments show, obstacles will result in high entropy, while seabed echo returns will result in low entropy. A local histogram of 256 bins and a 9×9 window has shown good quality results.

A hysteric threshold was then applied on the entropy result. A hysteric threshold uses two different thresholds. The first high-level threshold selects the parts of the image that are to be treated as obstacles, while the second low-level threshold selects the areas that will be treated as an obstacle only if they are connected by four-connectivity to the pixels that are above a high-level threshold. 70% and 60% of the maximum entropy level were used as high and low thresholds, respectively as can be seen in. Fig. 3. The first interest of this algorithm is to discard middle value peaks not connected to high entropy regions and ignore the noise.

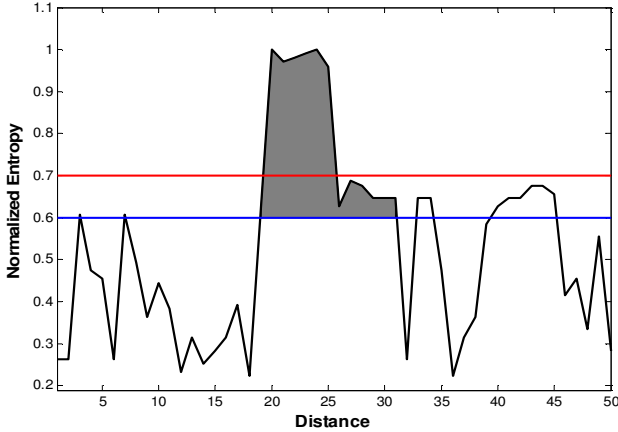


Fig. 3. Hysteric Threshold. The black line indicates the normalized entropy from one of the sonar beams. The red and blue lines indicate high and low level thresholds, while the grey region is the region that was selected.

At a final stage, a 3×3 morphological open operator [13] was applied on the threshold result to remove any remaining noise.

C. Feature extraction

Once segmented, different regions representing the obstacles in the image are labeled using a standard labeling algorithm and the following features for each obstacle region are extracted:

- Position (centroid) in the image;
- Area of the object in pixels;
- Perimeter of the object in pixels.

III. FPGA HARDWARE DESIGN

The algorithm was implemented on a GiDEL PROCe III-110 PCI Express FPGA development board [14] with Altera StratixIII-110E C3 speed grade FPGA [15]. FPGA memory blocks were used to form line delays and Flip-Flops were used to form single pixel delays in the implementation.

The maximal operation frequency of the system is 314

MHz. The power consumption of the system is 2.5 Watts and the maximal frame rate is 2700 FPS.

The FPGA resource utilization is given in Table I.

TABLE I
FPGA RESOURCE USAGE

Parameter	LUTs	Registers	Memory KBits	DSP Blocks
Logic Usage	22,840	21,550	166	176
Usage Percentage	27%	25%	2%	20%

The following sections describe in detail the FPGA design.

A. Frost Filter

The Frost filter implementation requires creation of a new filter kernel for every pixel. As a first step to reduce the computational cost the two-dimensional kernel $m(x, y)$ was separated into two single-dimensional filters, m_x, m_y ; this is possible due to the convolution's algebraic properties:

$$m_x(x, y) = m_y(x, y) = \frac{e^{-KC_l^2(x, y)|x-x_0|}}{\sum e^{-KC_l^2(x, y)|x-x_0|}} \quad (4)$$

The second step is to create in advance a bank of filter kernels instead of calculating the kernel for every pixel. This is done by dividing the range of kernel middle point values $m_x(x, y)|_{x_0}$ (the target space) into N even parts, where N represents the number of filter kernels in the bank. This division implies a non-linear division of the parameter space C_l . In order to calculate the division points of the range of C_l the center point value of the filter's kernel can be written as:

$$m(x, y)|_{x_0} = \frac{1}{\sum e^{-KC_l^2|x-x_0|}} = \frac{1 - e^{-KC_l^2}}{1 - 2e^{-KC_l^2 \frac{s+1}{2}} + e^{-KC_l^2}} \quad (5)$$

where s is the size of the kernel.

The expression $2e^{-KC_l^2 \frac{s+1}{2}}$ is significantly smaller than 1 and can be ignored for kernels bigger than 7. The target space is divided into N linearly spaced regions m_q , the quantized C_l^2 values are then represented by C_q and an equation that links parameter space C_q to the target space m_q is given by:

$$C_q(n) = -\ln\left(\frac{1 - m_q(n)}{1 + m_q(n)}\right) \frac{1}{K} \quad (6)$$

A filter kernel bank is created by substituting C_l in the filter kernel definition with $C_q(n)$, for n between 0 and N-1. The filter kernels are then stored in the FPGA ROM.

In order to select the right filter kernel at run time the following formula is evaluated:

$$n = \arg \min(|C_l^2 - C_q(n)|) = \arg \min(|\bar{I}^2 - \bar{I}^2(C_q(n)+1)|) \quad (7)$$

where n is the index of the kernel in the kernel bank that is used for the pixel (x, y) , \bar{I}^2 is an average of the squared

local image, and \bar{I}^2 is square of average of local image. The kernel selection process is implemented as a set of N comparators with their result encoded into an address of the filter kernel bank. The structure of the filter implementation is shown in Fig. 4.

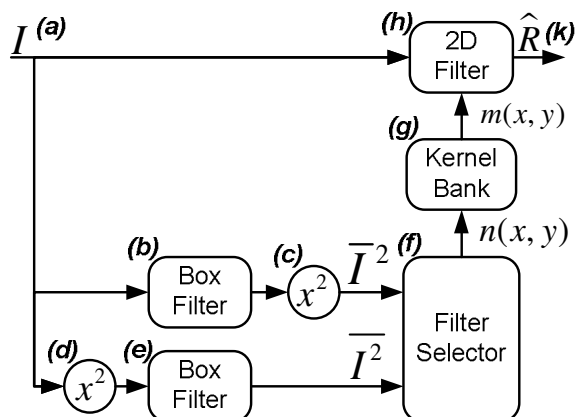


Fig. 4. Block Diagram of the FPGA Frost Filter. The filtering process of the input image (a) begins with calculating the local mean (b) using a box filter and then squaring (c) to obtain a squared mean. In parallel a local mean of squared image is computed by squaring the input image (d) and then applying a box filter (e). Then a filter selection process is carried out (f). The output of the filter selector is used as an address to the filter kernel bank (g), which generates kernels for the filter (h) for scene approximation (k).

B. Local Image Histogram Entropy

The local image histogram entropy algorithm first requires calculation of a histogram in the $N \times N$ neighborhood around each image pixel. When the histogram window slides on the image, for each pixel, N pixels enter the histogram window while N pixels exit the histogram window. We have used a set of chained line delays to form the N entering pixels front, and pixel delays with length of the filter window to form the exiting pixels front. All the pixels are then forwarded to the bin selector, which selects histogram bins that should be updated. At the second stage, the entropy of each bin that is given by the following formula is calculated:

$$E(i) = \frac{p(i)}{N_{bins}} \cdot \log \left(\frac{p(i)}{N_{bins}} \right) \quad (8)$$

The log and the division functions are not suitable for FPGA implementation, due to lack of dedicated FPGA logic for that purpose; therefore we have used ROM-based Look Up Tables (LUT) for that purpose. The output LUT data width was chosen to be 8 bits. As a final stage of the algorithm all the LUT results are summed to form a final 16 bit result. The detailed algorithm structure can be seen in Fig. 5.

C. Morphological Open

A 3×3 pixels morphological open [13] post-threshold noise removal operator was carried out using 3×3 erosion followed by 3×3 dilation. Both the erosion and dilation operators were performed using line and pixel delays to form a filter window and a comparator for decision making. In dilation the comparator is configured to output one when at least one of the inputs is greater than zero. In erosion the comparator is configured to output one only when all the inputs are one. The implementation of the erosion/dilation is shown in Fig. 6.

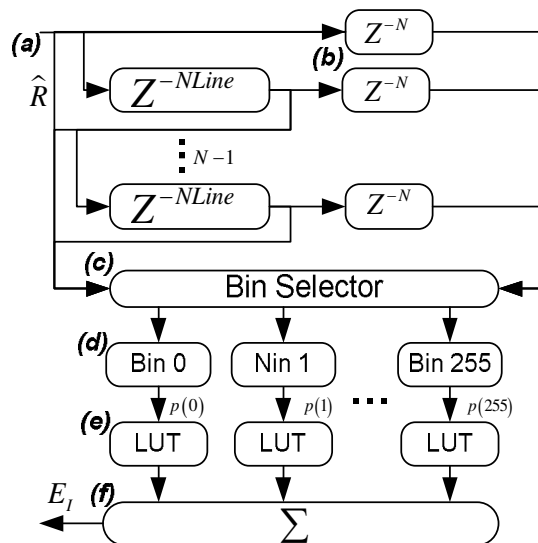


Fig. 5. Block Diagram of the Local Histogram Entropy implementation. The pixels front entering and exiting the filter window are formulated using line delays (a) and N pixel delays (b). The bin selector (c) selects the histogram bins (d) that should be updated. Then the values of the bins $p(i)$ are fed into look-up tables (e) and summed (f) to produce an entropy result.

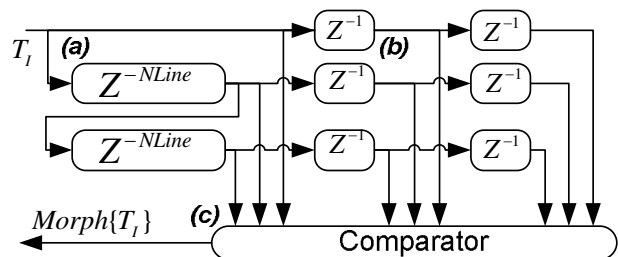


Fig. 6. Block Diagram of erosion/dilation FPGA implementation. A filter window is opened using line delays (a) and single pixels delays (b). The comparator (c) is used for the decision.

IV. RESULTS

The algorithm was tested on several sonar images acquired in different conditions. In all cases the algorithm showed outstanding results. Two of the cases can be seen in Fig. 7, demonstrating detection of a wharf at large echo contrast conditions and Fig. 8, demonstrating detection of objects on a seabed in high noise and seabed echo return conditions.

In order to establish a base-line for the performance test, the Algorithm was executed on an Intel i7-720QM Processor (6M Cache, 1.60 GHz, 4 Cores). The power consumption of the Processor was 45W (as reported by Intel). This CPU was chosen for its low power consumption and high performance. The software implementation frame rate was 0.263 FPS. The hardware architecture provides a speedup of 10,000 and power reduction by 94% compared to the software implementation.

V. CONCLUSIONS

An Adaptive Frost filtering of acoustic images followed by entropic segmentation provides outstanding obstacle detection for AUVs operating close to the seabed in spite of the high speckle noise and strong seabed acoustic returns.

The conventional software-based implementations cannot deal with the high computation complexity of the algorithm both from a real-time aspect and power efficiency. FPGA hardware architecture for the algorithm was proposed in order to reduce the processing time and power consumption. The described hardware implementation shows a significant speedup and higher power efficiency when compared to the traditional software-based implementations.

REFERENCES

- [1] R. M. a. P. M. Elgar Desa, "Potential of autonomous underwater vehicles as new generation ocean data platforms," *Current Science*, vol. 90, May, 2006 2006.
- [2] E. O. Belcher, *et al.*, "Dual-frequency acoustic camera: a candidate for an obstacle avoidance, gap-filler, and identification sensor for untethered underwater vehicles," in *OCEANS '02 MTS/IEEE*, 2002, pp. 2124-2128 vol.4.
- [3] Y. Y. Schechner and N. Karpel, "Recovering scenes by polarization analysis," in *OCEANS '04. MTTSS/IEEE TECHNO-OCEAN '04*, 2004, pp. 1255-1261 Vol.3.
- [4] J. G. Abbott and F. L. Thurstone, "Acoustic speckle: Theory and experimental analysis," *Ultrasonic Imaging*, vol. 1, pp. 303-324, 1979.
- [5] BlueView, "P450-45 Miniature Multibeam Imaging Sonar," www.blueview.com.
- [6] V. S. Frost, *et al.*, "A Model for Radar Images and Its Application to Adaptive Digital Filtering of Multiplicative Noise," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-4, pp. 157-166, 1982.
- [7] K. Kim, *et al.*, "Mosaicing of acoustic camera images," *Radar, Sonar and Navigation, IEE Proceedings -*, vol. 152, pp. 263-270, 2005.
- [8] L. Henriksen, "Real-time underwater object detection based on an electrically scanned high-resolution sonar," in *Autonomous Underwater Vehicle Technology, 1994. AUV '94., Proceedings of the 1994 Symposium on*, 1994, pp. 99-104.
- [9] Y. Petillot, *et al.*, "Underwater vehicle obstacle avoidance and path planning using a multi-beam forward looking sonar," *Oceanic Engineering, IEEE Journal of*, vol. 26, pp. 240-251, 2001.
- [10] C. I. Chang, *et al.*, "Survey and comparative analysis of entropy and relative entropy thresholding techniques," *Vision, Image and Signal Processing, IEE Proceedings -*, vol. 153, pp. 837-850, 2006.
- [11] T. Cover, and Thomas, J., "Elements of information theory," (*John Wiley & Sons, Inc., 1991*).
- [12] A. S. Abutaleb, "Automatic thresholding of gray-level pictures using two-dimensional entropy," *Computer Vision, Graphics, and Image Processing*, vol. 47, pp. 22-32, 1989.
- [13] R. H. a. L. Shapiro, "Computer and Robot Vision," *Addison-Wesley Publishing Company*, vol. 1, pp. 174 - 185, 1992.
- [14] GiDEL, "PROCeIII Product Brief," www.gidel.com.
- [15] Altera, "Stratix III Device Handbook," www.altera.com, 2009.

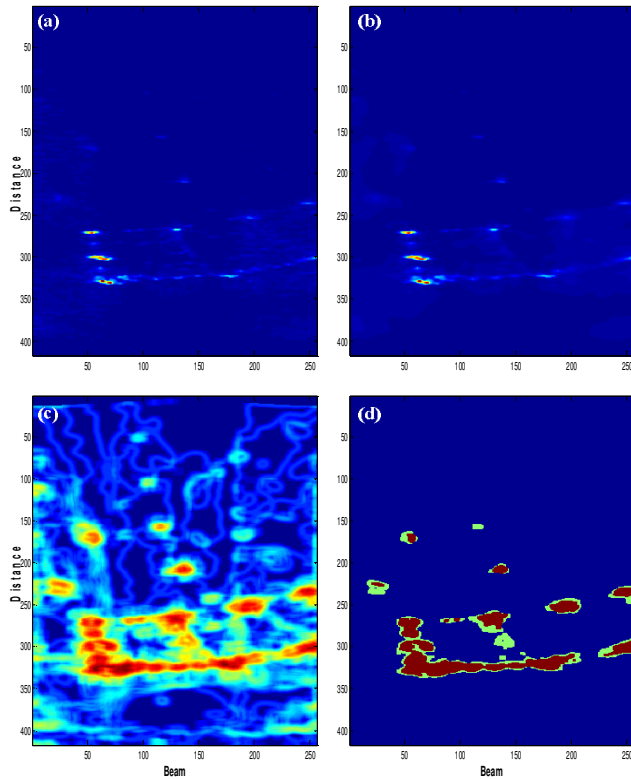


Fig. 7. Detection of wharf on Princess Beach at Eilat. (a) An image acquired from the sonar, (b) image after speckle suppression, (c) Local image histogram entropy, (d) red color for pixels above high-level threshold, and green color for pixels above low-level threshold.

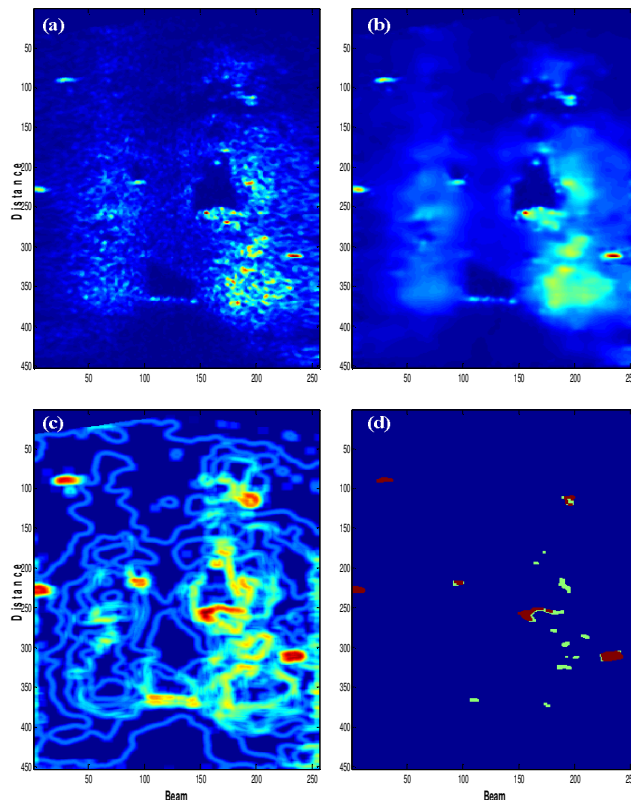


Fig. 8. Detection of objects on seabed (from BlueView Test Data). (a) An image acquired from the sonar, (b) image after speckle suppression, (c) Local image histogram entropy, (d) red color for pixels above high-level threshold