BMSE-D-17-00193 manuscript No.
(will be inserted by the editor)

Underwater Radio Frequency Image Sensor Using Progressive Image Compression and Region of Interest

Eduardo M. Rubino · Diego Centelles · Jorge Sales · José V. Martí · Raúl Marín · Pedro J. Sanz · Alberto J. Álvares

Received: 06/19/2017 (rev. 1) / Accepted: date

Abstract The increasing demand for underwater robotic intervention systems around the world in several application domains requires more versatile and inexpensive systems. By using a wireless communication system, supervised semi-autonomous robots have freedom of movement, however, the limited and varying bandwidth of underwater radio frequency (RF) channels is a major obstacle for the operator to get camera feedback and supervise the intervention. This paper proposes the use of progressive (embedded) image compression and region of interest (ROI) for the design of an underwater image sensor to be installed in an Autonomous Underwater Vehicle (AUV), specially when there are constraints on the available bandwidth, allowing a more agile data exchange between the vehicle and a human operator supervising the underwater intervention. The operator can dynamically decide the size, quality, frame-rate, or resolution of the received images so that the available bandwidth is utilized to its fullest potential and with the required minimum latency. The paper focuses first on the description of the system, which uses a camera, an embedded Linux system and a RF emitter installed in an OpenROV housing cylinder. The RF receiver is connected to a computer on the user side, which controls the camera monitoring parameters, including the compression inputs, such as ROI (Region of Interest), size of the image, and frame rate. The paper focuses on the compression subsystem and does not attempt to improve the communications physical media for better underwater RF links. Instead, it proposes a unified system that uses well integrated modules (compression and transmis-

Eduardo M. Rubino · Diego Centelles · Jorge Sales · José V. Martí · Raúl Marín · Pedro J. Sanz
Computer Science and Engineering Department, University of Jaume-I, 12071 Castellón de la Plana
email: emrubino@gmail.com of Eduardo M. Rubino

Alberto J. Alvares
Department of Mechanical Engineering, University of Brasilia, 70060-900 Brasilia, Brazil

sion) in order to provide the scientific community with a higher level protocol for image compression and transmission in sub-sea robotic interventions.

Keywords Progressive Image compression · Region of Interest (ROI) · Wavelet Transforms · Low Bandwidth Communications · Underwater Vehicles for Intervention

1 Introduction

In the context of the MERBOTS research project (`http://www.irs.uji.es/merbots/`), a three-year coordinated project funded by the Spanish government for the period 2015-2017 under grant DPI2014-57746-C3 [1], one of the objectives is to build a wireless communication system that can provide freedom of movements to the underwater robot and, at the same time, to allow the operator to get feedback and supervise the intervention (Fig. 1). The robotic system under development will assist the archaeologists in the detailed work of monitoring, characterization, study, reconstruction and preservation of archaeological sites, always in accordance with the continuous supervision of the human expert.

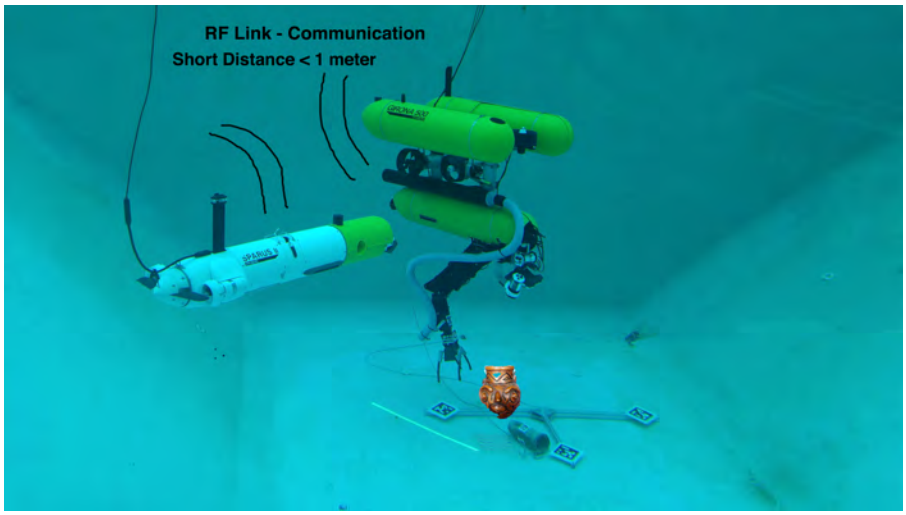

Fig. 1 Search and recovery envisioned concept in the context of archeology. A wireless RF link provides feedback to the user that is supervising the intervention, Autonomous Underwater Vehicle (AUV) Girona 500 and Sparus II

One of the objectives of the MERBOTS project is to provide different communication technologies that can be used to allow the operation of a vehicle without any physical connection to the surface operators, which are supervising and controlling an intervention task, which differentiates this project from

previous national and international research projects in the field of underwater robotic intervention ((i.e. RAUVI [2], TRITON [3], EU FP7 TRIDENT [4]).

The present article describes the current state of the wireless underwater vision system, that is able to transmit telemetry data as well as compressed low-resolution images, allowing further implementation of cooperative intervention missions. The Depth Embedded Block Tree (DEBT) compression system has been designed specifically to maximize the efficiency of the underwater intervention application, taking into account the following facts:

- It uses a progressive compression technique designed to reduce the latency, taking into account that this is an essential part of the whole robotic control system. Also, high quality images, even lossless, can be stored locally and only a prefix of it of any size can be transmitted so that the original images can be retrieved at a later time in order to be studied or archived in more detail;
- The compression algorithm has been implemented in order to be used in very low bandwidth scenarios. It can be applied in underwater radio-frequency communications and other robotic applications;
- An implementation of the compression algorithm has been realized which is capable of compressing more than 30 fps on low power embedded computers (e.g. Raspberry Pi 3 Model B);
- The system is able to send usable images using only a few hundred bytes per frame. The stream can simply be truncated in order to send a lower quality version of the image;
- The user is able to select one or more Regions Of Interest (ROI) in order to get more quality in specific parts of the image. This article explains how this technique has been implemented;
- The compression algorithm was tested with a battery of underwater images in order to better adjust the compression parameters for underwater intervention missions, remembering that JPEG2000 (Joint Photographic Experts Group 2000) is optimized for larger packets that are unrealistic for underwater acoustic or RF transmissions [5].

The use of underwater radio frequency (RF) links is a prime example of a low-bandwidth scenario, but the techniques described here can also be applied to any kind of bandwidth-constrained scenario. The techniques described here can also be used to enhance usability by a great deal in normal bandwidth scenarios, specially when dealing with remote image searching, by using either a low quality or low resolution version of the original image and only requesting more details when the target image has been found.

The RF link was used to test a low-bandwidth communication channel in a real way and not in simulated mode, with less bandwidth than an ultrasonic modem commonly used in underwater applications. RF Modem are less expensive and there was an acoustic modem available for testing. Thus the RF modem allowed to test and validate the DEBT algorithm with progressive image compression and ROI in conditions of lower bandwidth, even having dis-

Table 1 Comparison between the proposed compression algorithm and other methods

	DEBT	JPEG	JPEG2000
Quality and resolution progressive compression	✓	✗	✓
Manual and automatic Region Of Interest (ROI)	✓	✗	✓
Rate Distortion Optimized	✓	✗	✓
Real time (fast) compression	✓	✓	✗
Exact size (truncate) compression	✓	✗	✗
Lossless and Lossy compression	✓	✗	✓
Parallel algorithm	✓	✗	✗
Applicable for underwater acoustic or RF transmissions	✓	✗	✗

tance limitation for RF transmission. That is, if the DEBT algorithm works properly with RF links it will certainly work with acoustic modems.

Table 1 makes a small comparison of the most important differences between the DEBT algorithm [6] and the well known JPEG (Joint Photographic Experts Group) [7] and JPEG2000 [8] algorithms. JPEG is an aging algorithm that, although being quite fast, performs poorly under high compression and does not possess the necessary features. JPEG2000, on the other hand, is a quite complex and sofisticated algorithm that compresses well under almost all conditions and has most of the needed features but is quite slow. A more detailed explanation of the compression features of the DEBT algorithm will be given in section 4.

2 Related Works

Suzuki and Sasaki [9] was the first system to demonstrate image transmission over a vertical path which was developed in Japan. The JPEG standard DCT (Discrete Cosine Transform) was used to encode 256x256 pixel still images with 2 bits per pixel. Transmission of about one frame per 10 seconds was achieved using 4-DPSK (Differential Phase Shift Keying) at 16 kbps. Remarkable results obtained with this system included a video of a slowly moving crab, transmitted acoustically from a 6,500 m deep ocean trench. Another vertical path image transmission system was developed in France and successfully tested in 2,000 m deep water. This system was also based on the JPEG standard and used binary DPSK for transmission at 19 kbps.

An image transmission system has been developed in a Portuguese effort called ASIMOV [10]. In this project, a vertical transmission link is secured by a coordinated operation of an AUV and an ASC (Autonomous Surface Craft). Once the site is chosen and the vehicles are positioned, transmission of a sequence of still images of about 2 frames/sec is accomplished at 30 kbps using an 8PSK (Phase-shift keying) modulation method. Other experimental of underwater video transmission system, developed in Japan [9], employs 4PSK, 8PSK, and 16QAM (Quadrature Amplitude Modulation) signals with 40 KHz bandwidth to achieve transmissions at up to 128 kbps. The system

uses 100 kHz carrier frequency and was tested over a short vertical path of 30 m.

Because underwater images have low contrast, their information is concentrated at low frequencies. Thus, by decomposing the image information into low and high frequency subbands, and encoding the low bands with more precision, it is possible to achieve higher compression ratios. This is the basic motivation behind the work in [11] which used the DWT (Discrete Wavelet Transform) in place of the standard DCT. This algorithm was applied to a sequence of underwater images, taken at 30 frames per second, each having 256x256 8-bit pixels. The achieved compression ratio of 100:1 provided very good quality monochrome video. The resulting bit rate needed to support such high quality is on the order of 160 kbps, which surpasses the capabilities of the current acoustic modem technology.

Another system that exploits wavelet based compression together with motion compensation is proposed in [12]. Although it attains approximately the same compression ratio (100:1) as in [11], it has better visual intelligibility because it employs a generalized dynamic image model (GDIM) that decouples the geometric and photometric variations in an image sequence commonly encountered in deep sea imagery. This approach is in contrast with ordinary terrestrial motion-compensated algorithms, where steady and uniform illumination is the underlying assumption. Using 128x128 pixel frames and 30 frames/sec, the resulting bit rates needed to support real-time video transmission were in the order of 40 kbps.

Pelekanakis [13] presents a high bit rate acoustic link for underwater video transmission. Currently, encoding standards support video transmission at bit rates as low as 64 kbps. While this rate is still above the limit of commercially available acoustic modems, prototype acoustic modems based on phase coherent modulation/detection have demonstrated successful transmission at 30 kbps over a deep water channel.

An experimental system [13], based on DCT and Huffman entropy coding for image compression, and variable rate Mary quadrature amplitude modulation (QAM) was implemented. Phase-coherent equalization is accomplished by joint operation of a decision feedback equalizer (DFE) and a second order phase locked loop (PLL). System performance is demonstrated experimentally, using a transmission rate of 25000 symbols/sec at a carrier frequency of 75 kHz over a 10 m vertical path. Excellent results were obtained, thus demonstrating bit rates as high as 150 kbps, which are sufficient for real-time transmission of compressed video.

Eastwood et al. [14] presents techniques for compression of laser line scan and camera images, as well as format specific data compression for quick-look sonar mapping data. For image compression, both JPEG and a wavelet based technique called Efficient Pyramid Image Coder (EPIC) are examined. JPEG is found to be less efficient than the wavelet transform but has the advantage of being robust with respect to lost data packets. The wavelet based transform is more efficient at high compression rates though above a certain rate both offer similar performance.

Walter et al. [15] presents a new wavelet-based image compression system. The compression system is based on a particular type of compressed encoding of wavelet transforms called Wavelet Difference Reduction (WDR) and describes experimental results in applying a compression algorithm to a suite of underwater camera images. These underwater camera images were required to be compressed at very high compression ratios (400:1, 200:1, 100:1, and 50:1) and the algorithm produced very high-fidelity decompressions. In fact, it performed at a comparable level to a system based on the celebrated Daub CDF-9/7 system (used in JPEG2000 [16]) yet employing 256 times less RAM (Random Access Memory) and a 16-bit dynamic range (with 8-bit images) instead of a 32-bit dynamic range.

Murphy [17] presents an analysis of the unique considerations facing telemetry systems for free-roaming Autonomous Underwater Vehicles used in exploration. These considerations include high-cost vehicle nodes with persistent storage and significant computation capabilities, combined with human surface operators monitoring each node. He then proposes mechanisms for interactive, progressive communications of data across multiple acoustic hops. These mechanisms include wavelet-based embedded coding methods, and a novel image compression scheme based on texture classification and synthesis. The specific characteristics of underwater communication channels, including high latency, intermittent communication, the lack of instantaneous end-to-end connectivity, and a broadcast medium were taken into consideration.

Kaeli in his PHD thesis [5] shows that the fundamental problem in autonomous underwater robotics is the high latency between the capture of image data and the time at which operators are able to gain a visual understanding of the survey environment. Typical missions can generate imagery at rates hundreds of times greater than highly compressed images can be transmitted acoustically, delaying that understanding until after the vehicle has been recovered and the data analyzed. His thesis presents a lightweight framework for processing imagery in real time aboard a robotic vehicle. The work implements a framework on real underwater datasets and demonstrates how it can be used to select summary images for the purpose of creating low-bandwidth semantic maps capable of being transmitted acoustically.

Kaeli [5] compares JPEG, JPEG2000 and SPIHT(Set Partitioning in Hierarchical Trees). JPEG is a common example of a lossy compression format which uses the DCT for each 8x8 block to achieve roughly 10:1 compression without major perceptual changes in the image. JPEG2000 employs variable compression rates using progressive encoding, meaning that a compressed image can be transmitted in pieces or packets that independently add finer detail to the received image. This is particularly well-suited to underwater applications where acoustic channels are noisy and subject to high packet loss, however, it is optimized for larger packets that are unrealistic for underwater acoustic transmissions. Recent work has focused on optimizing similar wavelet decomposition techniques for underwater applications using smaller packet sizes with SPIHT.

Zheng et al. [18] presents a special application of delay tolerant networks (DTNs). Efficient data collection in deep sea poses some unique challenges, due to the need for timely data reporting and the delay of acoustic transmission in the ocean. Autonomous underwater vehicles are deployed in deep sea to surface communications and frequently have to transmit collected data from sensors (in a 2-dimensional or 3-dimensional search space) to the surface stations. However, additional delay occurs at each resurfacing.

Senapati et al. [19] presents a listless implementation of a wavelet based block tree coding (WBTC) algorithm of varying root block sizes. The WBTC algorithm improves the image compression performance of SPIHT at lower rates by efficiently encoding both inter and intra scale correlation using block trees. Though WBTC lowers the memory requirement by using block trees compared to SPIHT, it makes use of three ordered auxiliary lists. The proposed algorithm is combined with DCT and DWT to show its superiority over DCT and DWT based embedded coders, including JPEG2000 at lower rates. The compression performance on most of the the standard test images is nearly the same as WBTC but it outperforms SPIHT by a wide margin particularly at lower bit rates.

Pearlman et al. [20] proposes an embedded, block-based, image wavelet transform coding algorithm of low complexity. It uses a recursive set partitioning procedure to sort subsets of wavelet coefficients by maximum magnitude with respect to thresholds that are integer powers of two. It exploits two fundamental characteristics of an image transform: the well defined hierarchical structure and and energy clustering in frequency and in space. They describe the use of this coding algorithm in several implementations, including reversible (lossless) coding and its adaptation for color images, and show extensive comparisons with other state-of-the-art coders, such as SPIHT and JPEG2000.

Zhang et al [21] presents a new underwater video compression technique based on adaptive hybrid wavelets and directional filter banks to achieve both high coding efficiency and good reconstruction quality at very low-bit rates. A key application is the real-time transmission of video through acoustic channels with limited bandwidth from an autonomous underwater vehicle to a surface station, e.g., for man-in-the-loop monitoring and inspection operations.

According to Esmaiel ([22] and [23]) the SPIHT coder based on the wavelet algorithm is probably the most widely used for image compression, as well as being a basic standard of compression for all subsequent algorithms [24], [25] and [26]. In SPIHT, the information bits are sorted according to the bit information significance. The protection level of transmitted data must take this feature into account and progressive protection is provided to the transmitted bits. This methodology is used to reduce the distortion in the reconstructed image (reduce the difference between the original and the reconstructed images). After image decomposition with CDF-9/7 wavelet, the general SPIHT coding algorithm encodes images by splitting the decomposed image into considerable sections on the basis of the significance classification function [27].

Mohammed and Hamada [28] propose new scheme for efficient rate allocation in conjunction with reducing peak-to-average power ratio (PAPR) in orthogonal frequency-division multiplexing (OFDM) systems. Modification of the SPIHT image coder is proposed to generate four different groups of bitstreams relative to its significances. The significant bits, the sign bits, the set bits and the refinement bits are transmitted in four different groups.

None of the references presented use a solution based on progressive image compression and ROI (Region of Interest) and, together, these are the main contributions of the currently developed algorithm, DEBT. In [6], Rubino et al., presents some initial results for the Raspberry Pi Model 2B platform that allowed for the validation of the proposed approach in a simulated way without using an real RF link. In this work we present real results obtained with an RF link using the Raspberry Pi 3B platform and also describe the system in more depth in experimental tests carried out at the University of Girona using two AUVs, the Girona 500 and Sparus II (`http://cirs.udg.edu/auvs-technology/auvs/`), Fig. 1.

3 The Intervention Domain

Robotic applications and, particularly, Autonomous Underwater Vehicles for Intervention (I-AUV) use images from its built-in camera(s) as one of its main sources of data, among others, in order to control its internal algorithms. In a supervised system, these images should reach the operator with the lowest latency and with the highest quality possible so that he can interact with the system and adjust the task execution in a supervised manner.

As an example, this kind of control has been experimented in the FP7 TRIDENT project, to perform autonomous visually guided grasping in the sea [29].

Besides this, communications is a crucial subsystem in any robotic application, specially the ones that permit the user to interact remotely with the system. Because of that, image compression and transmission is necessary in order to send the required information with the lowest latency and without compromising the network and the whole system.

Although recent studies demonstrate that, using the most efficient modulation methods, it is possible to transmit video through an underwater channel using acoustic signals [30,31] and Blue Light [32], both acoustic and optical signals are not capable to pass through solid objects that could be in the line of sight of the wireless transceivers. Moreover, the performance of these methods depends heavily on the characteristics of the underwater scenario and the type of the channel. On one hand, acoustic systems are greatly affected due to multi-path if the link is horizontal, and also by the acoustic noise originated by human activity or the noise of the sea waves, animals, and other sources. The acoustic noise constrains the range of typical frequencies used in acoustic systems between 8 and 155 KHz [33], which makes it very difficult to achieve high data rates. On the other hand, communication methods based on visible

light only work fine on very clear waters, are greatly affected by scattering, suffer attenuation by absorption, and usually need accurate alignment.

Nevertheless, RF based solutions are not as affected by the typical problems of the acoustic and optical methods, and are much cheaper. Moreover, RF signals can propagate easier from a medium to another, allowing the establishment of a communication link to an underwater transducer from the surface.

The main problem of using RF is the high attenuation that it suffers when the waves go through the water. However, different studies [34,35,36] indicate that, with the necessary antennas, at lower frequencies, and using the best modulation methods, it is possible to set up a communication link up to several tens of meters through the water.

It is worth to mention that the objective of the present work is not to improve the communications physical media for better underwater RF links, but the design of a unified system that uses well integrated modules (i.e. compression and transmission), in order to provide the scientific community with a higher level protocol for image compression and transmission in sub-sea.

The application of the most advanced progressive image compression algorithms, as the ones presented in this document, allows image transmission rates of several frames per second, at the typical latency of the radio-frequency communications.

In the proposed system, a progressive image compression technique and the use of ROI are demonstrated.

3.1 Overall System Description

As can be seen in figure 2 the system has two main parts: (1) the sensor side, and (2) the operator one. At the vehicle side the developed circuitry is installed in the OpenROV (`https://www.openrov.com/`) housing cylinder. It includes a Raspberry PI computer running Linux, the camera acquisition, the compressor and ROI, and the transport layer modules. This is connected to an Arduino board that controls the RF Radiometrix transmitter.

At the user side, the underwater RF receiver is connected to the user computer through an USB (Universal Serial Port) port, and provides the user interface that enables the operator to get the compressed images and select the corresponding Region Of Interest for further inspection.

The RF transmission system is at an early stage and, for evaluation purposes, a low-power radio module has been used. Further work will concentrate on using antennas and transceivers better suited for longer distances. The RF devices used for this experiment are the commercial low power UHF (Ultra High Frequency) modules BiM3B (`http://www.radiometrix.com/content/bim3b`), which work over the 868.3 MHz at 25 mW, and 1/4 wave antennas. On the transmitter side, the electronics involved are the RPi2B and RPi3B, the RaspiCam, an Arduino Pro Micro and a RF module. On the receiver side, an Arduino and a RF module. All the electronics have been encapsulated properly

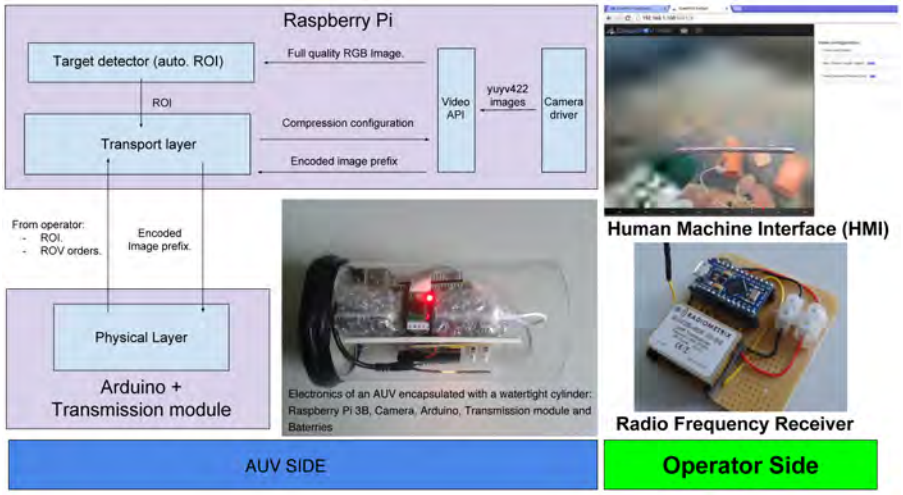

Fig. 2 Overall System Architecture: (left) Sensor Side installed in the OpenROV platform including Raspberry PI for high the compression module, ROI, and network protocol, as well as the physical RF transceiver controlled via an Arduino board. (right) User interface that enables user monitoring of the compressed camera information, as well as the specification of regions of interest by the user

within a watertight container. Transmitter and receiver containers have been attached and fixed to a wooden stick which has been immersed approximately at a depth of 15 cm.

For this experiment, 100 encoded images were transmitted for each distance point at 20, 40, 60, 70, 80 and 100 cm. A prefix of 400 bytes of each encoded image has been transmitted, and each one within a single PDU (Protocol Data Unit). The prefix corresponds to the whole image input for a given quality, that fits in a particular size (e.g. 400 bytes). The number of reception errors (packets lost plus packets with errors) and the RSSI (Received Signal Strength Indicator) were measured for each distance. The receiver (Arduino + RF module) was connected to a PC (Personal Computer) through an USB bus, where a process decoded and displayed each received image and also showed an error counter. The RF link was established at 9600 baud. Each packet has an overhead length of 15 bytes and contains not only the 400 bytes of the encoded image, but also 26 bytes of extra data.

With this preliminary implementation it was possible to transmit each one of the 100 images without errors at a maximum distance of 60 cm in fresh water. Once a distance of 70 cm was reached, only 36 of the 100 images were received properly. For larger ranges ($>$ 70 cm), it was not possible to receive any image using the transceivers and antennas used in this particular experiment. Fig. 3 shows the RSSI sampling at each position and Fig. 4 shows the FPS (Frames per Second) obtained with the same configuration of the protocol used in the water experiments, for different lengths of the encoded

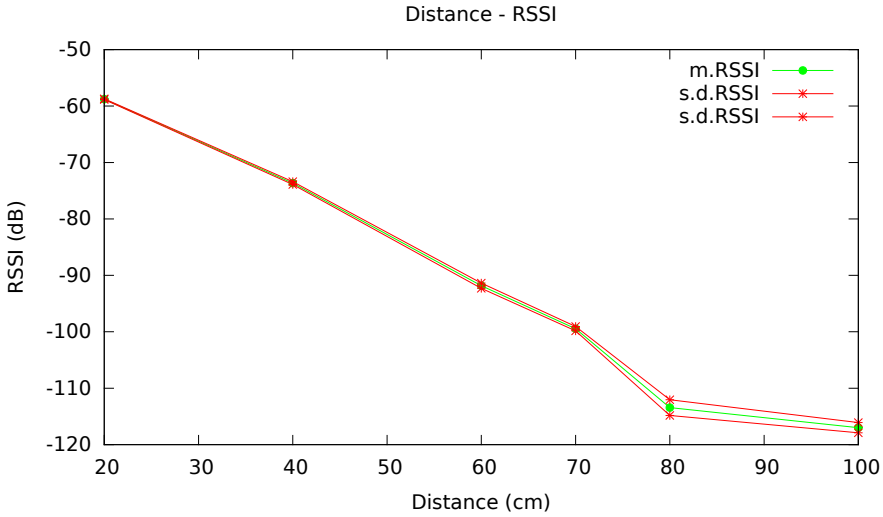

Fig. 3 RSSI in a 8x4 freshwater pool using the BiM3B (868.3 MHz at 25 mW). The RSSI mean is shown with the green line, whereas the red lines represent the standard deviation. Each point is the distance where a sampling has been performed

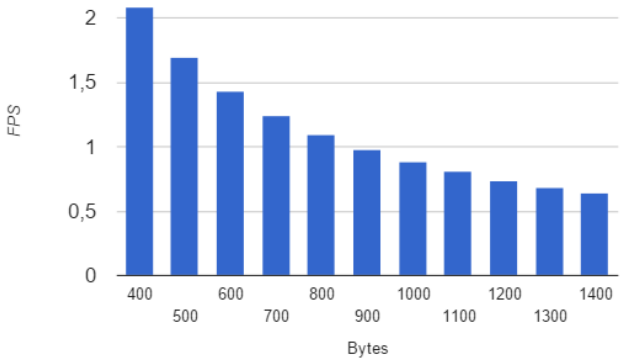

Fig. 4 FPS obtained for each prefix length with the experimental protocol and the RF link established at 9600 baud

image prefix. As it is shown in the image, with a length less than or equal to 800 bytes, a transmission at a frame rate greater than 1 FPS is possible. Usually, a length of 800 bytes allows an operator on the surface to properly monitor the camera sensor input.

3.2 Image Compression

Image compression is a transformation applied to an image in order to reduce its size as much as possible in order to store or transmit it in a more efficient manner. There is a clear distinction between lossless and lossy compression. In lossless compression, the decompressed image will be exactly the same as the original image while, in lossy compression, the decompressed image will be an approximation of the original image. Digital images usually have 3 color components, which means that what we perceive as one color image is (without loss of generality), in fact, composed of a luminance channel (black and white version of the color image) and 2 color difference channels (which can usually be sub sampled without much visual loss).

As opposed to video compression, which takes advantage of the high temporal correlation between adjacent frames in a video sequence and creates inter-frames which are dependent on previous frames, image compression exclusively creates intra-frames, which are independently compressed frames. Intra-frames have the advantage of being able to rapidly adapt to changing conditions in the communications channel as well as increased flexibility in dynamically changing the frame rate and quality parameters, which are of great importance in low bandwidth and low latency communications.

Compression in general and image compression in particular is a very application specific task, with many available trade-offs and many different algorithms that try to maximize (or minimize) some design criteria. Most image compression algorithms are lossy algorithms designed with the sole purpose of minimizing the resulting size of the image with minimal regards to other constraints and usually the whole compressed data is necessary in order to be able to decompress it. A prime example example of this class of algorithms is the JPEG algorithm which is a de facto standard but performs very poorly under high compression.

Progressive or embedded image compression is such that it is trivial and very inexpensive in terms of processing power (there is no need to decompress and recompress the image) to supply an image which is either a lower resolution or a lower quality approximation of the original image. Preferably, the compressed image could be simply truncated at any point, yielding a lower resolution or lower quality version of the original image (in this sense, progressive lossless streams can become lossy by simple truncation). In the case of color images, we could also prepare the image in such a way that a monochrome version of it could be obtained with the same progressive characteristics as before.

4 The Depth Embedded Block Tree (DEBT) algorithm

The main properties sought for a proper implementation of our communications framework are:

- Quality, Resolution, and Color Channel Scalability - Truncating the stream should result in the "best" approximation for the original image or, by rearranging and truncating the stream, in the "best" approximation to a scaled, monochrome or color version of the image;
- ROI (Region of Interest) - Definition of certain areas of the image that should be compressed with lower distortion than the rest of the image, allowing for very high compression ratios while keeping these areas with high quality. The ROI areas could be either chosen automatically by an object recognition mechanism, or by the user, who desires a higher quality on an region that is currently not detailed enough;
- Embedded Lossless - Allow for lossless compression with a stream that yields the "best" possible image at any truncation point and interpreting the truncated stream as a lossy version of the image which was compressed. The image could be stored losslessly for later archival but any desired truncated part of it could be transmitted in real time, only compressing the image once and giving the flexibility of dynamically choosing the transmitted quality or size;
- Fast and Parallelizable - It should perform well on low power, small SBCs (Single Board Computer) with optional hardware floating point arithmetic support and with compression speeds comparable to the JPEG (Joint Photographic Experts Group) algorithm. Also, the algorithm should be parallelizable in order to take advantage of current and future multi-core processors, allowing both lower latency and higher throughput;
- High compression - While this seems to be an obvious property for any image compression algorithm, the goal is to be competitive with current state-of-the-art image compressors.

Scalable compression usually takes advantage of multiresolution signal decomposition, which is natural for dyadic wavelet decomposition but can also be used with DCT [37] or other block transforms by simply rearranging its coefficients.

There are two major classes of transform-based image compression algorithms. The first follows the transform-model-code paradigm with a very distinctive separation of the 3 main steps; the transformation (either a block or wavelet transform), followed by statistical modeling of the coefficients and bit allocation, followed by entropy coding in the form of some sort of context-adaptive arithmetic coding. All JPEG coders are in this category and depend strongly on the final step, which is usually quite slow and needs many operations for each output bit. The JPEG2000 standard, which is the current state-of-the-art image compressor, is an example of this traditional scheme (it is based on the Embedded Block Coding with Optimized Truncation (EBCOT) [38] algorithm).

On the other hand, there are other algorithms which do not have a clear distinction between the model and code steps and do not rely on any sort of final entropy coding, which should make them quite fast as well as good candidates for a parallel implementation. However, most of them rely on or-

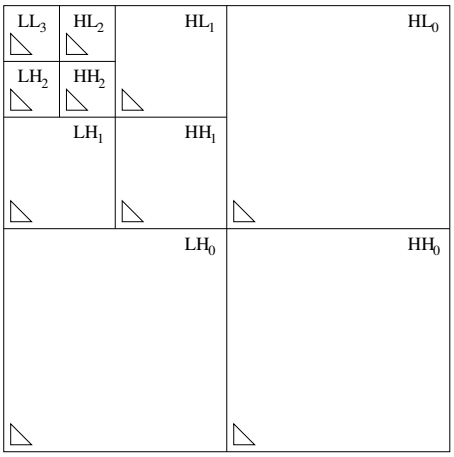

Fig. 5 3 level wavelet decomposition

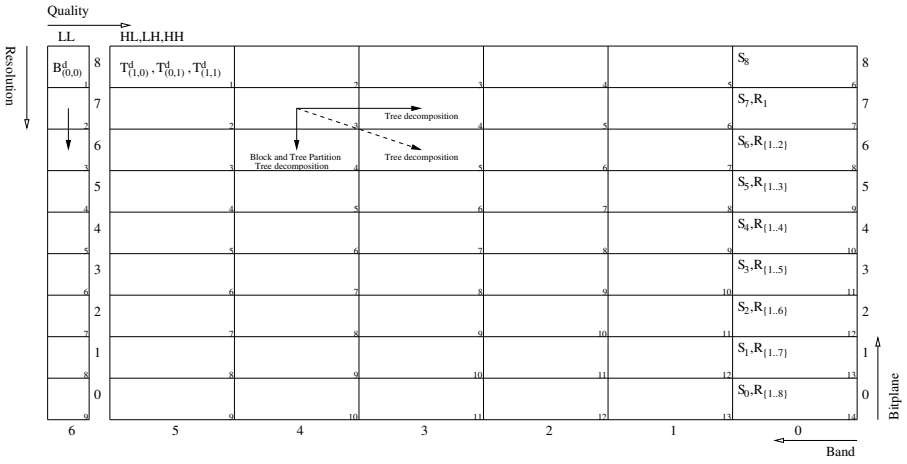

Fig. 6 Bit scan order

thogonal wavelets which makes them unsuitable for lossless compression and most of them have implementation issues dealing with list manipulation and high memory use. Also, none of these algorithms have an efficient implementation available and none of them possess all our requirements simultaneously. Some of the best known algorithms in this class are EZW [39] (Embedded Zerotree Wavelet), SPIHT [40] (Set Partitioning In Hierarchical Trees), SPECK [20] (Set Partition Embedded bloCK), HBC [41] (Hybrid Block Coder), WBTC [42] (Wavelet Block Tree Coder), and GTW [43] (Group Testing for Wavelets), among many others.

The DEBT algorithm has been designed according to this second class of algorithms and possess all the stated properties above. Basically, it consists of:

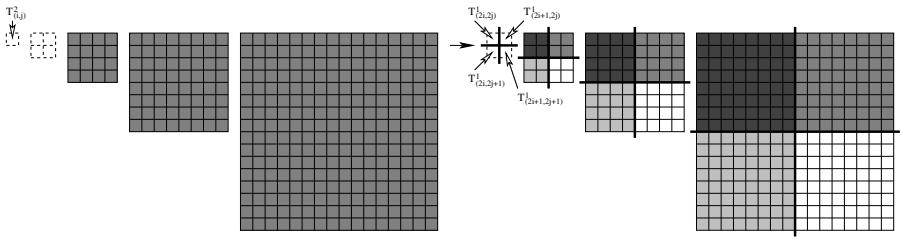

Fig. 7 Variable-depth tree partition

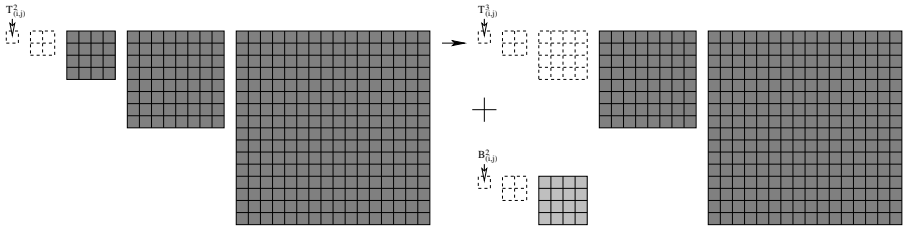

Fig. 8 Variable-depth tree decomposition

1. Wavelet transform - Transform the image using a wavelet transform in N levels (fig. 5 shows a 3 level dyadic wavelet decomposition). Currently, the 5/3, 9/3, 9/7, and 13/7 symmetric biorthogonal integer transforms built from the interpolating Deslauries-Dubuc scaling functions [44] were implemented, along with the the popular real-valued CDF-9/7 symmetric biorthogonal transform [44];
2. The concept of variable-depth trees (simply referred as trees) and variable-depth blocks (simply referred as blocks) is introduced and these are the main data structures used to group similar magnitude coefficients so that the necessary significant coefficient addressing information is conveyed in an efficient manner while exploring the intra and inter band coefficient correlation. Trees and blocks have an associated depth value, which indicates the actual first pyramidal layer where the coefficients are located (the first "depth" layers do not exist). Variable-depth trees make the DEBT algorithm perform extremely well specially in very low bit-rate scenarios. Figs. 7 and 8 both show a depth 2 tree on the left;
3. Group coefficients into trees - A novel method of dynamically subdividing these trees into variable-depth subtrees (simply called subtrees) of a lower depth called "partition" (fig. 7) or into a block of the same depth and a subtree of a higher depth called "decompositon" (fig. 8) allows for the efficient exploration of both intra and inter band correlation while keeping the algorithm simple, effective and fast. In fact, the DEBT algorithm can be viewed as a superset, generalization, unification, and improvement of many of the existing set partition algorithms (SPIHT, SPECK, HBC, WBTC and others) by simply changing a single parameter. Blocks are al-

Table 2 Laplace root distortion per bit ($\sigma^2 = 128^2$)

0.864								
0.500	1.724							
0.500	1.000	3.430						
0.500	1.000	2.000	6.794					
0.500	1.000	2.000	3.996	13.324				
0.500	1.000	2.000	3.996	7.969	25.634			
0.500	1.000	2.000	3.996	7.969	15.753	47.508		
0.500	1.000	2.000	3.996	7.969	15.753	30.099	82.304	
0.500	1.000	2.000	3.996	7.969	15.753	30.099	50.770	128.961
0	1	2	3	4	5	6	7	8
				bitplane				

ways partitioned into lower depth variable-depth subblocks (simply called subblocks);

4. In order to achieve good embeddedness (bit allocation), a modeling of the coefficient distribution must be taken into account so that the instant distortion reductions for significant and refinement coefficients are predicted, which will lead to the desired distortion reduction per bit. Currently, the ordering is done assuming a laplacian distribution for the wavelet coefficients but a more precise modeling, using a generalized power distribution, is under investigation and should yield a better ordering and, therefore, better embeddedness;
   Table 2 shows the square root of the instant distortion decrease per bit for significant (diagonal values) and refinement coefficients (the laplacian distribution, just like the uniform distribution, has the nice property that all refinement distortion decreases for bits in the same bitplane are equal, irrespective of this coefficient's significant bitplane).
   For a laplacian distribution with variance σ^2, the values of the distortion decrease per significant coefficient in an interval $[a, b)$ are given by

$$\Delta D = \left[\frac{1}{\lambda} + a - \frac{b-a}{e^{\lambda(b-a)} - 1}\right]^2 \tag{1}$$

   and the values of the distortion decrease per refinement coefficient which are significant in an interval $[a, b)$ are given by

$$\Delta D_n = \left[\frac{\delta_{n+1}}{\cosh(\lambda\delta_{n+1})}\right]^2, \quad n = 1, 2, \ldots \tag{2}$$

   where

$$\lambda = \sqrt{\frac{2}{\sigma^2}}, \quad \delta_n = \frac{b-a}{2^n}, \quad \cosh(x) = \frac{e^x + e^{-x}}{2} \tag{3}$$

   and n represents the refinement level.
   The number of bits per significant coefficient for coefficients that are significant in interval $[a, b)$ is given by

$$\eta = 1 + \frac{\mathcal{H}(p_s)}{p_s} \tag{4}$$

Table 3 Weights for ibior-13/7 DWT

k	LL_k	$HL_k \mid LH_k$	HH_k
1	1.640625	1.03654814	0.654891968
2	3.20977783	1.73186421	0.934442759
3	6.40856028	3.39070654	1.79398966
4	12.8155956	6.77010059	3.57644415
5	25.630991	13.5386333	7.15128803
6	51.2619553	27.0770607	14.3023653
7	102.523911	54.1540947	28.604702
8	205.047821	108.308182	57.2094002
9	410.095642	216.616364	114.4188
10	820.191284	433.232727	228.837601
11	1640.38257	866.465454	457.675201
12	3280.76514	1732.93091	915.350403
13	6561.53027	3465.86182	1830.70081
14	13123.0605	6931.72363	3661.40161
15	26246.1211	13863.4473	7322.80322
16	52492.2422	27726.8945	14645.6064

where

$$p_s = \frac{e^{-\lambda a} - e^{-\lambda b}}{1 - e^{-\lambda b}} \tag{5}$$

and $\mathcal{H}(x)$ is the binary entropy function (in bits) defined for $0 \leq x \leq 1$ as

$$\mathcal{H}(x) = \mathcal{E}(x) + \mathcal{E}(1 - x) \tag{6}$$

where $\mathcal{E}$ is the entropy function (in bits) defined as $\mathcal{E}(0) = 0$ and, for $0 < x <= 1$, as

$$\mathcal{E}(x) = -x \log_2(x) \tag{7}$$

The distortion decrease per bit for significant coefficients is then calculated by dividing equation 1 by equation 4, i.e., $\Delta D/\eta$. In the case of the DEBT algorithm (and most set partitioning algorithms), the number of bits per refinement coefficient is 1, even though this is not their real entropy;

5. In general, most DWT are not energy preserving so that each subband contributes differently to the total distortion. The weight for each subband can be calculated as a function of its respective reconstruction filter [45] and should be used as a factor for all the values in table 2 for each respective subband. Table 3 shows the weights for each subband for the ibior-13/7 wavelet;
6. Precise distortion decrease assignment to each significant and refinement (subband, bitplane) pair indicates the most important one to send serving as an embedded bit allocation. Fig. 6 shows an example of a 6 level transform resulting in coefficients where the maximum absolute value has 9 bits, ranging from bitplanes 0 to 8. Each cell contains the n-th column of table 2 (where n corresponds to the bitplane level) scaled by the respective subband gain. As there is a single column for the LL subband, and N columns for each of the HL, LH, and HH subbands (N is the number of decompositions) there is a total of $(3N+1)B(B+1)/2$ weighted distortion

reduction per bit values, where B is the number of bitplanes, in the general case. In the case of a laplacian distribution, as all refinement values are the same for each column, the number of weighted distortion reduction per bit values can be reduced to $(3N + 1)(2B - 1)$;

7. Scan all weighted distortion reduction per bit values in decreasing order, output the necessary significance and refinement information, and keep the set partition and decomposition (addressing) information. All housekeeping is done on a fixed size memory pool with size dependent on the image dimensions. Roughly speaking, for an $N \times M$ 8-bit image, the DEBT algorithm needs one $N \times M$ 16-bit array for the transform coefficients, one $N \times M$ 16-bit array for coefficient management and and one $N/2 \times M/2$ 32-bit array for set management.

There are many ways in which different compression algorithms can be evaluated and compared. For quantifying the error between images, two measures are commonly used. They are the MSE (Mean Square Error) and the PSNR (Peak Signal to Noise Ratio). The MSE between an image $\{y_k\}$ and its approximation $\{\hat{y_k}\}$ is given by:

$$\mathrm{MSE} = \frac{\sum_{k=0}^{N-1} (y_k - \hat{y_k})^2}{N}$$

where N is the total number of pixels in each image. The PSNR between two (8 bpp) images, in decibels is given by

$$\mathrm{PSNR} = 10 \log_{10} \left(\frac{255^2}{\mathrm{MSE}} \right)$$

and is used more often since it is a logarithmic measure and the human brain seem to respond logarithmically to changes in intensity. Increasing PSNR means increasing fidelity of compression and, as a rule of thumb, when the PSNR is greater than or equal to 40 dB, it is said that the two images are virtually indistinguishable by human observers.

In order to compare the compression ratio obtained by the DEBT algorithm, standard test images were used. Fig. 9 shows the standard "lena" and "barbara" 512×512, 8-bit grey level images and fig. 10 shows the respective PSNR curves obtained by using the CDF-9/7 wavelet transform and 6 levels of decomposition with the DEBT algorithm. Table 4 compares DEBT versus JPEG2000 for various compression rates. The resulting sizes were obtained by applying the respective rate for the JPEG2000 compressor which were then used to make DEBT compress exactly to those sizes. Even though the current DEBT algorithm is not yet finished and still needs tuning, it does a better job at compressing the test images than the JPEG2000 algorithm by up to 0.5db without using any kind of entropy coding and while being much faster.

Embedded image compression is a very efficient way to cope with varying transmission bandwidth problems in hard real time systems, where it would

Fig. 9 Standard test images "lena" "barbara"

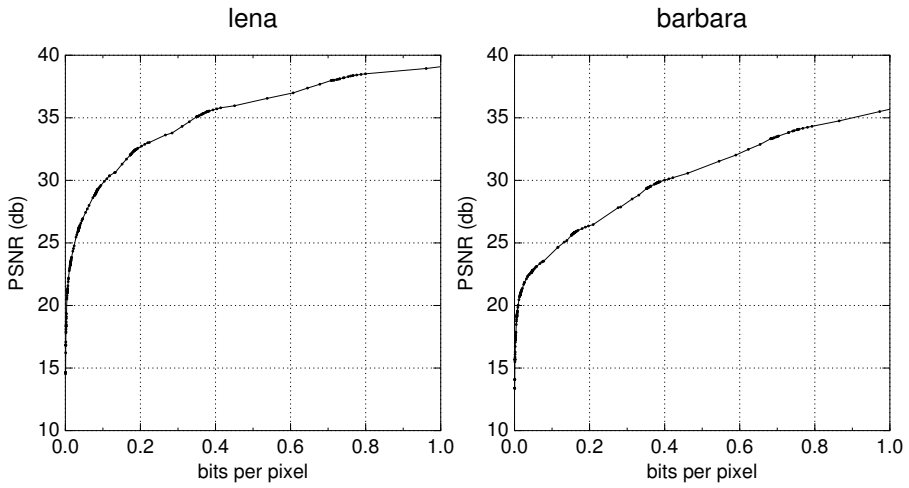

Fig. 10 PSNR for "lena" and "barbara" (6 level CDF-9/7 wavelet - DEBT)

be better to have a low quality version of the current image instead of a high quality version of an old image. The main idea behind using embedded image compression in the current scenario is to group the source of the data (image) with the transmission channel into one manageable whole, increasing the adaptability of the whole system by varying the amount of data transmitted when the channel capacity changes, i.e., increase the image quality when there is bandwidth available and decrease it when there is not, in order to meet a predefined maximum latency or bandwidth. Also, in order to cope with the need of very low latency, the current algorithm has been developed with parallelism in mind, being able to use many threads of execution in order to decrease the latency as much as possible.

Table 4 JPEG2000 comparison

	lena			barbara		
	Size	PSNR (db)		Size	PSNR (db)	
Rate	(bytes)	JPEG2000	DEBT	(bytes)	JPEG2000	DEBT
0.01	2602	28.45	28.87	2572	23.46	23.61
0.02	5202	31.20	31.67	5109	25.45	25.75
0.03	7835	33.07	33.41	7851	27.18	26.95
0.04	10461	34.28	34.67	10479	28.43	28.66
0.05	13090	35.31	35.83	12997	29.56	30.02
0.06	15721	36.10	36.36	15726	30.65	30.89
0.07	18285	36.81	36.89	18177	31.55	31.68
0.08	20842	37.35	37.50	20926	32.37	32.78
0.09	23583	37.95	38.33	23556	33.25	33.79
0.10	26188	38.39	38.85	26152	33.97	34.47

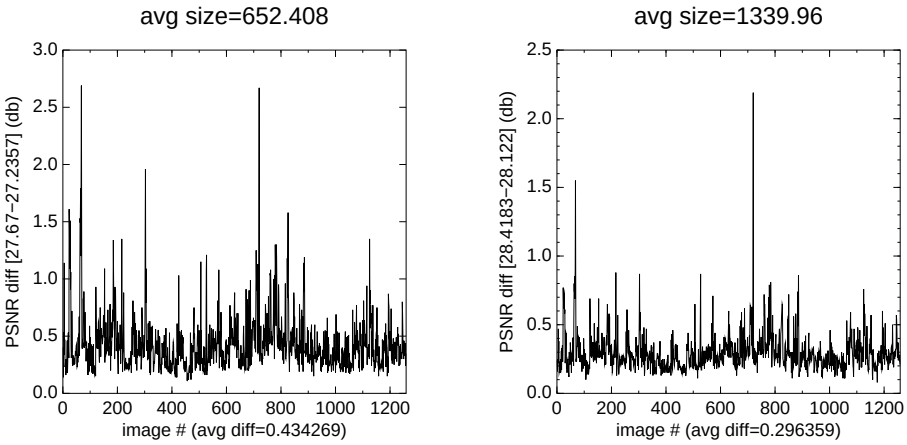

Fig. 11 TasCPC images for rates 0.0005 and 0.001

5 Underwater test images

In order to test and tune some parameters of the algorithm to underwater imagery, we used a set of 1258 underwater gray-scale images from the Australian Center for Field Robotics (`http://marine.acfr.usyd.edu.au/datasets/data/TasCPC/TasCPC_LC.tar.gz`). A low bitrate compression performance comparison against the JPEG2000 algorithm was done using the same parameters we used for the lena and barbara images. The PSNR difference between the DEBT and JPEG2000 algorithm was plotted for 4 different rates (0.0005, 0.001, 0.005, and 0.01). All images are 1360×1024 pixels and were numbered in lexical order from 1 up to 1258.

Fig. 11 shows the compression difference in db for all images for rates 0.0005 and 0.001. It is quite clear that the DEBT algorithm performs better for all images, without exception, for these lower rates by quite a significant margin, reaching a difference of 2.69 db in the 0.0005 case and 2.19 db in the

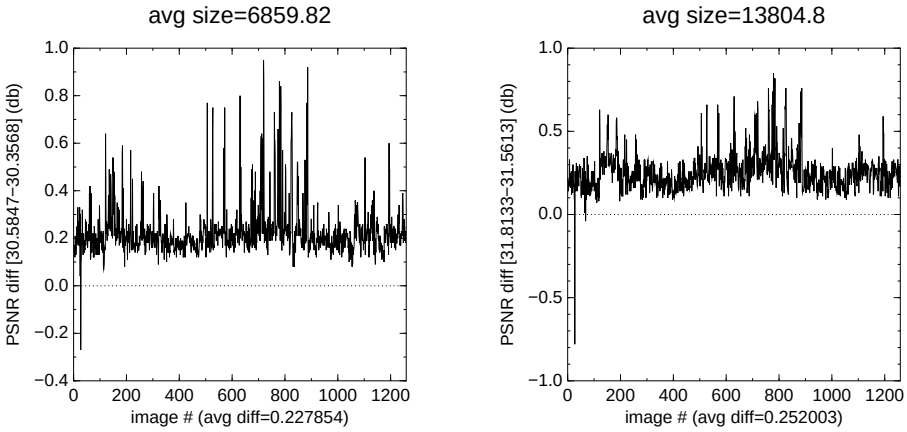

Fig. 12 TasCPC images for rates 0.005 and 0.01

0.001 case. On average, the gain of DEBT over JPEG2000 is 0.43 db and 0.30 db for rates 0.0005 and 0.001, respectively.

As the rate goes up (less compression and more quality), the DEBT algorithm is still superior to the JPEG2000 for all 1258 images in this dataset except for 1 in the 0.005 case and 2 in the 0.01 case and in all these cases the compression quality was very high (the images were very dark), over 40 db or higher for both algorithms. Fig. 12 shows the results for these rates. On average, the gain of DEBT over JPEG2000 is 0.23 db and 0.25 db for rates 0.005 and 0.01, respectively.

A few images were randomly selected from this dataset and are presented in figure 13. Each row shows the original image (1st column) and compressed with the DEBT algorithm at exactly 500, 1000, and 2000 bytes on the 2nd, 3rd, and 4th columns, respectively.

An important point to note is that the DEBT algorithm makes it very easy, by simply changing the weights on each cell shown in fig. 6, to create a stream which is not optimal in the MSE sense but which could be better suited to highlight other characteristics of the image at very low bitrates. These alternate metrics should be the subject of further investigation.

6 Region Of Interest (ROI)

In a low-bandwidth scenario or when the images are highly compressed, there may be circumstances where the image is still not good enough for an operator to distinguish the necessary details. In this case, the use of ROI is an elegant solution to the problem of being able to see the details in part of the image while still maintaining a very high compression level, at the expense of making the other areas in the image less detailed. ROI has been most extensively used in conjunction with medical imaging and are an integral part of the JPEG2000

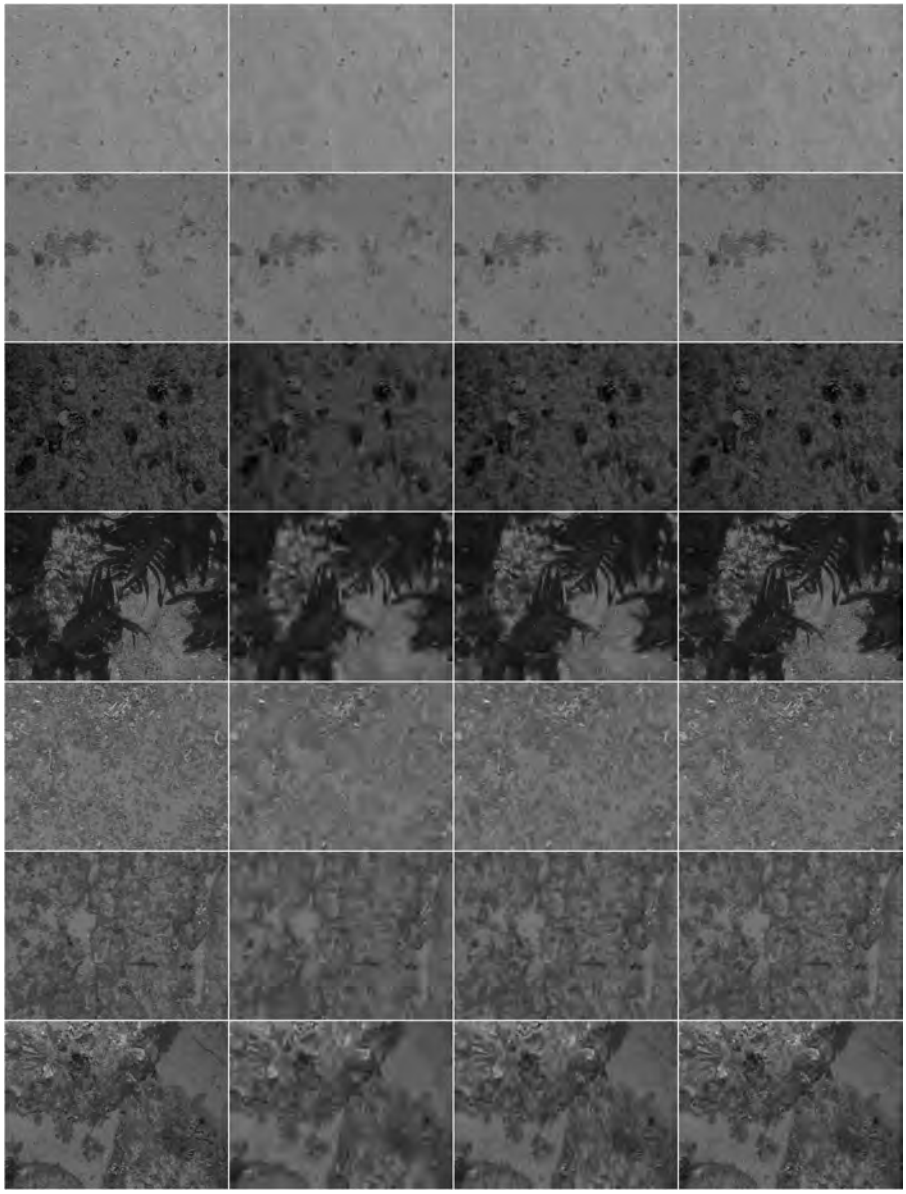

Fig. 13 Underwater images 16, 227, 274, 813, 977, 1082, and 1219 compressed at original, 500, 1000, and 2000 bytes

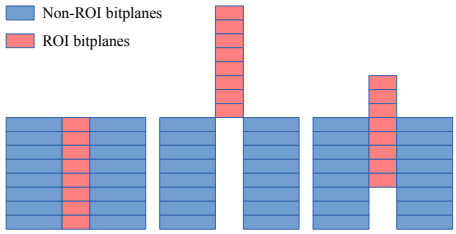

Fig. 14 ROI coding methods: (a) Uniform compression, (b) bitplane shifts for the *maxshift* method, and (c) bitplane shifts for the *scaling*-based method [6]

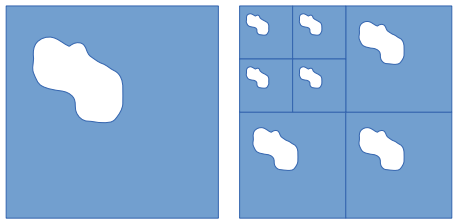

Fig. 15 (a) ROI mask, and (b) Wavelet mask for 2 levels of DWT [6]

standard. Most ROI techniques are usually used in conjunction with wavelet based image coding techniques [46].

For ROI processing, there must be a way for the decoder to know which regions were encoded with higher priority than others. A common method known as *maxshift* [47] [48] is commonly used so that the bitplanes of the ROI region are encoded in its entirety before any bitplanes of the rest of the image (background) (see Fig.14). This has the advantage of almost no overhead (only the number of extra bitplanes are sent so the decoder knows that after reaching this number of bitplanes it should unscale the received coefficients by the amount of bitplanes remaining) but has the disadvantage of having to send the whole ROI, with all its details, before receiving a single bit from the rest of the image.

A more useful method known as *scaling* [47] [48] consists of simply shifting the ROI coefficients by a certain number of bits so that they fool the bit allocation algorithm into thinking that they are more important than they actually are and coding them before other coefficients that became smaller due to the scaling (see Fig. 14). In fact, this effectively blends the ROI coefficients with background coefficients which are also important (same order of magnitude) such that the results are seen with good quality in a lower quality background. The main disadvantage of this method is that a ROI map must be sent as extra information to the decoder (overhead), increasing the minimum amount of bits necessary to recover a suitable approximation to the original image.

This map information can consist of object coordinates (rectangles, ellipses, or arbitrary polygons) or, in case of an arbitrary region, a bitmap of the ROI. In this last case, in order to reduce the amount of overhead, this map could be the resulting map on the last decomposition subband (LL) thereby

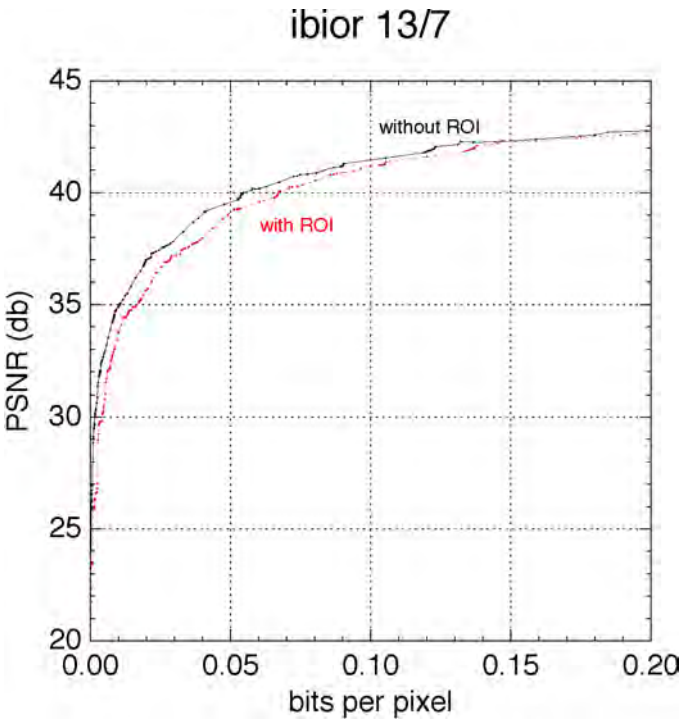

Fig. 16 Effect of ROI in PSNR: without ROI in black and with ROI in red

significantly decreasing the bitmap size but having the drawback of using a coarser scale, depending on the number of dyadic decompositions (for an n-level dyadic decomposition the grid would be $2^n \times 2^n$ pixels). This bitmap usually consists of a small region and, therefore, is a good candidate for some form of run-length encoding (Fig. 15).

Other methods, which interleave the ROI coefficients with the background coefficients, in a predetermined and alternating order, have also been devised in order to minimize the transmission of overhead information but most of them require fundamental algorithmic changes so that both the encoder and the decoder scan the bitplanes in the same order and are more complex than the *maxshift* and *scaling* method. The examples used in this paper were prepared with the *scaling* algorithm with arbitrary coarse regions as described above.

It should be observed that, in general, the use of ROI will impact negatively in the PSNR of the whole reconstructed image but will improve significantly the fidelity in the ROI region itself. In our example, Fig. 16 shows the difference for coding the region around the text with 4 bits of shifting in comparison of the coding of the image without any ROI. The image used for this is the monochrome 1920×1080 image used in fig. 17 and the ROI is the rectangle of dimensions 128×64 with top left corner at $(x_0, y_0) = (1024, 192)$ and bottom right corner at $(x_1, y_1) = (1151, 255)$.

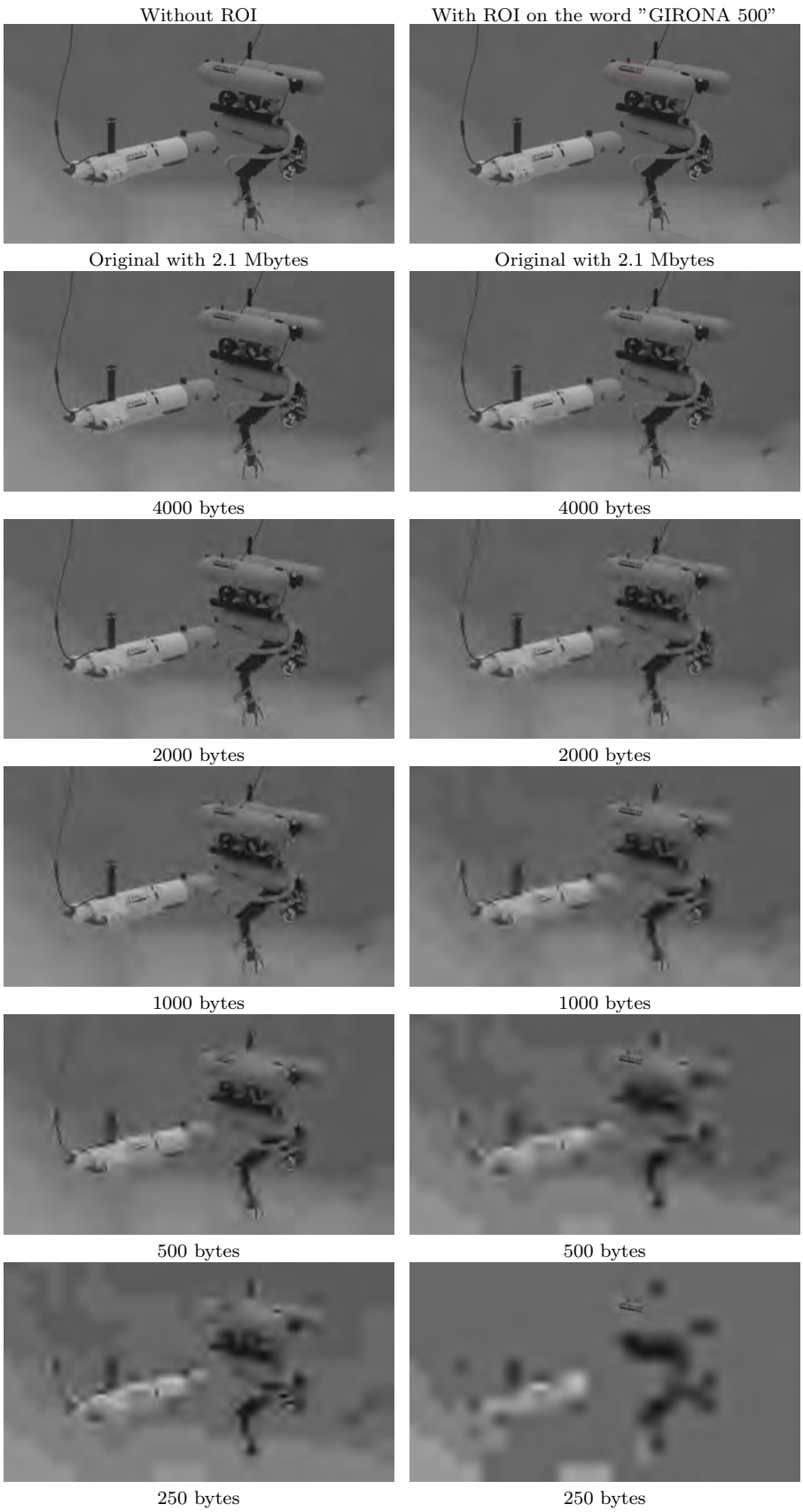

Fig. 17 Comparison of compressed image with and without ROI - DEBT

7 Implementation

A working implementation has been developed in the C programming language with special vector code for both Intel and ARM processors to speed up the inner loop of the wavelet transform routine. A few wavelet transforms were implemented: integer wavelet transforms (interpolating biorthogonal integer transforms 5/3, 9/3, 9/7 and ibior-13/7) and the CDF-9/7 real-valued transform. The wavelet used for the examples in this paper was the ibior-13/7 transform, which has a highpass filter with 7 taps and a lowpass filter with 13 taps. This wavelet allows for lossless compression and can also be truncated at any point yielding performance within 0.5 db of the CDF-9/7 real-valued transform but being much faster due to the all-integer arithmetic and 16-bit coefficients.

For the 1920×1080 pixels, 8 bpp graylevel image used (fig.17) we ran the compressor in both a Raspberry Pi 2 Model B (RPi2B) and Raspberry Pi 3 Model B (RPi3B). The RPi2B is based on a 1.0 GHz quad code ARM processor (quad core ARM Cortex-A7 with 512KB L2 cache) manufactured by Broadcom (BCM2836 SoC) while the RPi3B, the third generation Raspberry Pi, uses a Broadcom BCM2837 SoC (quad core ARM Cortex-A53 with 512KB of L2 cache) operating at 1.3 GHz with 1 GB of DDR2 RAM. Both use a 32-bit memory bus which was operated at 500 MHz.

The timings for each board for compressing the image on the upper left corner of Fig. 17 with and without ROI are displayed on Tables 5 and 6. The algorithm has a parameter that specifies either the max size or a "quality" factor (which bears some inverse relation with the PSNR). The normal usage (if lossless compression is not required) is to use a "good" quality parameter for local storage and transmission of any prefix of this file for lower quality versions of the compressed image. The "quality" used for each line of the tables was 0, 4, 8, 12, 16, and 24. The wavelet used was the ibior-13/7 b-spline interpolating integer transform with 6 decomposition levels. The first line of each table (where the "quality" parameter is 0) is for the lossless case, where MSE = 0 (PSNR = ∞) and all timings were based on the single-threaded version of the algorithm.

The column labeled "pre" (tables 5 and 6) is the time (in milliseconds) for the wavelet transformation and all other tasks needed to actually start running the compression algorithm (mean extraction, significance map, etc). This step is independent of the amount of bits output and is in fact a lower bound for images of this size (it is almost independent of the contents of the image itself and mostly dependent on the image dimensions alone).

The column labeled "code" (Tables 5 and 6) is the time (in milliseconds) for the actual compression algorithm and is directly proportional to the amount of bits output. Therefore, the specification of a quality factor or a maximum size will have a great impact on this part and in the total running time for the image compression.

Table 5 RPi2B and RPi3B timings (without ROI)

			RPi2B			RPi3B		
Q	Size (bytes)	PSNR (db)	Pre (ms)	Code (ms)	Total (ms)	Pre (ms)	Code (ms)	Total (ms)
0	754627	∞	61.9	298.6	360.4	30.3	201.7	232.0
4	59456	43.00	62.5	35.0	97.5	30.4	24.5	54.9
8	23180	41.18	62.0	13.5	75.5	30.4	9.5	39.9
12	16219	40.27	62.6	9.7	72.4	30.3	6.8	37.1
16	10575	39.13	62.6	6.0	68.6	30.4	4.2	34.6
24	6051	37.37	62.0	3.5	65.5	30.4	2.4	32.8

Table 6 RPi2B and RPi3B timings (with ROI)

			RPi2B			RPi3B		
Q	Size (bytes)	PSNR (db)	Pre (ms)	Code (ms)	Total (ms)	Pre (ms)	Code (ms)	Total (ms)
0	758378	∞	62.7	298.8	361.5	30.6	201.7	232.3
4	63516	43.03	62.7	35.7	98.4	30.8	25.2	56.0
8	27023	41.25	62.5	14.7	77.2	30.7	10.3	40.9
12	19648	40.36	63.0	10.9	73.9	30.7	7.6	38.3
16	13710	39.26	63.1	7.3	70.4	30.8	4.9	35.7
24	8768	37.52	63.2	4.6	67.8	30.5	3.1	33.6

Table 7 JPEG2000 X DEBT with ibior-13/7 and CDF-9/7 (no ROI)

Rate	Size	PSNR (db)		
	(bytes)	JPEG2000	DEBT ibior-13/7	DEBT cdf-9/7
0.0001	179	19.15	26.79	27.67
0.0002	400	28.15	29.99	30.08
0.0005	1027	31.87	32.39	32.53
0.001	2055	33.78	34.26	34.52
0.002	4080	35.96	36.11	36.69
0.005	10339	38.60	39.03	39.58
0.01	20702	40.66	40.87	41.69
0.02	41238	42.63	42.39	43.25
0.05	103634	44.60	43.98	44.80
0.1	207315	45.98	45.75	46.00

7.1 Benchmark: DEBT x JPEG2000

Table 7 compares DEBT with JPEG2000. The JPEG2000 implementation used was the "JasPer" program [49] version 2.0.12 with 6 levels of decompositions and the CDF-9/7 wavelet transform. In order to do a "fair" comparison, we have also included a run of our algorithm with the same number of decompostions (6) and the same wavelet (CDF-9/7) along with the previously used 6 levels of decompositions and the ibior-13/7 integer wavelet transform (used for the timing results in the previous tables).

It is important to note that, because JPEG2000 does not has an option of exact output size, the amount of bytes used in the comparison was given by the

resulting size of the JPEG2000 file by using the following: `jasper --input tank.pgm --output tank.jpc --output-format jpc -O rate=X`, where `X` is the rate (first column) for the compression. The resulting file size was then used to compress the same image using our algorithm to this exact size, once with our current parameters (6 decomposition levels and the ibior-13/7 DWT) and another with the same parameters as the ones used in the JPEG2000 case (6 decomposition levels and the CDF-9/7 DWT).

The results show that, for the example image used, our algorithm is quite competitive with the current state-of-the-art JPEG2000 codec, even when using the ibior-13/7 DWT and is vastly superior for very small rates using either DWT. In our example, DEBT constantly outperforms JPEG2000 when using the same number of decompositions (6) and both transforms, while being much faster.

8 Conclusions and Further Work

This paper proposes the use of progressive image compression and region of interest (ROI) for a RF underwater image sensor to be installed in a robotic platform. The operator can dynamically decide the size, quality, frame-rate and resolution of the received images so that the available bandwidth is utilized to its fullest potential and with the required minimum latency.

The system is capable of dynamically and precisely adjust the image compression to either a predefined size or quality and it proved that it was capable of sending good quality images using 400-800 bytes. The frame rate is directly proportional to the channel capacity and can be precisely established by varying the size of the compressed images. Quality is enhanced by letting the operator specify a Region Of Interest in the camera input, which means that more details can be observed in this specific part of the image, maintaining the final image size and the consumed network channel.

Related to the compression algorithm, which is explained in detail in the previous sections, results show that one core of a RPi3B can compress high-resolution full-HD images (1920×1080) in very high quality. It can be seen from table 5 that it can process close to 30 frames per second with a PSNR around 40 db and there are still 3 more cores to be used by other processes.

Currently, a vectorized and parallel implementation of the DEBT algorithm (`https://tinyurl.com/y722ecbf`), including a parallel wavelet transform implementation for the above mentioned transforms, is being developed which should make it able to process very large images in real time on embedded multi-processor single board computers (SBC). Also, the use of a better PDF match for the DWT coefficients instead of the laplace PDF is being implemented (work is being done on the Exponential Power Distribution, also know as the Generalized Gaussian Distribution of Generalized Normal Distribution) which should improve the rate distortion curve of the compression allowing for an optimized stream in case the coefficients are really modeled by such a PDF.

The RPi2B is a less powerful board for images of this dimension but is still able to compress around 15 frames per second at the same 40 db PSNR using a single core. In this case, either the quality, frame size, or frame rate could be tuned so that the desired rate is achieved. Also, once the parallel version of both the DWT and DEBT are implemented, all cores could be used yielding a substantially faster compression rate.

As long as the ROI is a small region, there is not much difference in encoding times for using it, even though there is a small penalty to pay in compression efficiency for the whole image, as expected.

In summary, a specially designed progressive compression algorithm has been implemented so that it possesses both quality and resolution scalability, ROI, and is simple and fast enough with the goal of being usable in limited resource computers while producing compression comparable or better than current state-of-the-art compressors. The results show that it is quite competitive with state-of-the-art compression algorithms like JPEG2000 while being an order of magnitude faster.

Further work will concentrate on improving the communications physical layer in order to obtain communication distances around 5 meters, according to the project needs. Also, the higher level transport protocol will be enhanced by using congestion techniques that obtain a better use of the available bandwidth.

Acknowledgements This work was partly supported by Spanish Ministry under grant DPI2014-57746-C3 (MERBOTS Project), by Universitat Jaume I grants PID2010-12, E-2015-24, PREDOC/2012/47 and PREDOC/2013/46, and by Generalitat Valenciana grant ACIF/2014/298 and in the Brazil CNPQ and FAP/DF.

References

1. P.J. Sanz, A. Peñalver, J. Sales, J.J. Fernández, J. Pérez, D. Fornas, J. García, R. Marin, in *Sixth International Workshop on Marine Technology (MARTECH'15)* (Cartagena, Spain, 2015)
2. P.J. Sanz, M. Prats, P. Ridao, D. Ribas, G. Oliver, A. Orti, in *52-th International Symphosium ELMAR-2010* (Zadar, Croatia, 2010), pp. 471–474
3. P.J. Sanz, A. Peñalver, J. Sales, D. Fornas, J.J. Fernández, J. Perez, J.A. Bernabé, in *2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (IEEE, Manchester, UK, 2013)
4. P.J. Sanz, P. Ridao, G. Oliver, G. Casalino, Y. Petillot, C. Silvestre, C. Melchiorri, A. Turetta, in *OCEANS'13 MTS/IEEE conference* (San Diego, CA, 2013), pp. 1–10
5. J.W. Kaeli, in *Thesis, Massachusetts Institute of Technology* (2013), p. 135
6. E.M. Rubino, J. Sales, D. Centelles, P.J. Sanz, R. Marín, J.V. Marti, in *XXXVI Jornadas de Automática)* (Bilbao, Spain, 2015)
7. W. B. Pennebaker and J. L. Mitchell. JPEG still image data compression standard. New York: Van Nostrand Reinhold, 1992 (1992)
8. D.S. Taubman, M.W. Marcellin, *JPEG 2000: Image Compression Fundamentals, Standards and Practice* (Kluwer Academic Publishers, Norwell, MA, USA, 2001)
9. M.Suzuki, T.Sasaki, in *in Proc. IEEE Oceans'92 Conference* (1992), pp. 567–570. DOI DOI10.1109/DCC.2016.19
10. J. Gomes, V. Barroso, G. Ayela, P. Coince, in *OCEANS 2000 MTS/IEEE Conference and Exhibition. Conference Proceedings (Cat. No.00CH37158)*, vol. 3 (2000), vol. 3, pp. 1633–1637 vol.3. DOI 10.1109/OCEANS.2000.882174

11. D.F. Hoag, V.K. Ingle, R.J. Gaudette, IEEE Journal of Oceanic Engineering **22**(2), 393 (1997). DOI 10.1109/48.585958
12. S. Negahdaripour, A. Khamene, Computer Vision and Image Understanding **79**(1), 162 (2000). DOI http://dx.doi.org/10.1006/cviu.2000.0845. URL `http://www.sciencedirect.com/science/article/pii/S1077314200908452`
13. K. Pelekanakis, in *Thesis, Massachusetts Institute of Technology* (2004), p. 75
14. R. Eastwood, L. Freitag, J. Catipovic, in *OCEANS '96. MTS/IEEE, IEEE Xplore.* (2002), pp. 63–68
15. J.S. Walker, T.Q. Nguyen, Y.J. Chen, Optical Engineering, Research Signposts **5**, 111 (2003)
16. S. Mallat, in *A Wavelet Tour of Signal Processing. The Sparse Way* (2009), p. 805
17. C.A. Murphy, in *Thesis, Massachusetts Institute of Technology and WOODS HOLE OCEANOGRAPHIC INSTITUTION* (2004), p. 75
18. H. Zheng, N. Wang, J. Wu, Parallel and Distributed Computing pp. 1–43 (2017)
19. R.K. Senapati, U.C. Pati, K.K. Mahapatra, {AEU} - International Journal of Electronics and Communications **66**(12), 985 (2012). DOI https://doi.org/10.1016/j.aeue.2012.05.001. URL `http://www.sciencedirect.com/science/article/pii/S1434841112001070`
20. W. Pearlman, A. Islam, N. Nagaraj, A. Said, Circuits and Systems for Video Technology, IEEE Transactions on **14**(11), 1219 (2004). DOI 10.1109/TCSVT.2004.835150
21. Y. Zhang, S. Negahdaripour, Q. Li, Signal Processing: Image Communication **47**, 96 (2016). DOI http://dx.doi.org/10.1016/j.image.2016.06.001. URL `http://www.sciencedirect.com/science/article/pii/S092359651630087X`
22. H.A.H. Esmaiel, *Advanced Multi-Band Modulation Technology for Underwater Communication Systems* (University of Tasmania, Doctoral Thesis, 2015)
23. D.J. Hamada Esmaiel, Journal of Electrical and Electronic Engineering **2**(4), 64 (2014). DOI 10.11648/j.jeee.20140204.12
24. B. Tomasi, L. Toni, P. Casari, J. Preisig, M. Zorzi, in *Proceedings of the Sixth ACM International Workshop on Underwater Networks* (ACM, New York, NY, USA, 2011), WUWNet '11, pp. 9:1–9:8. DOI 10.1145/2076569.2076578. URL `http://doi.acm.org/10.1145/2076569.2076578`
25. T.B. Santoso, Wirawan, G. Hendrantoro, in *2012 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA)* (2012), pp. 37–41. DOI 10.1109/TSSA.2012.6366017
26. R.L. Eastwood, L.E. Freitag, J.A. Catipovic, in *OCEANS 96 MTS/IEEE Conference Proceedings. The Coastal Ocean - Prospects for the 21st Century* (1996), p. 67 suppl. DOI 10.1109/OCEANS.1996.566719
27. Y. Sun, R. ming Li, X. lei Cao, in *Proceedings. 2005 IEEE International Geoscience and Remote Sensing Symposium, 2005. IGARSS '05.*, vol. 2 (2005), vol. 2, pp. 4 pp.–. DOI 10.1109/IGARSS.2005.1525200
28. U.S. Mohammed, H.A. Hamada, International Journal of Video and Network Security **10**(10), 30 (2013)
29. M. Prats, A.P. del Pobil, P.J. Sanz, *Robot physical interaction through the combination of vision, tactile and force feedback. Applications to assistive robotics*. Springer Tracts in Advanced Robotics, Volume 84 (Springer Publishing Company, Incorporated, 2013)
30. C. Pelekanakis, M. Stojanovic, L. Freitag, in *OCEANS 2003. Proceedings*, vol. 2 (2003), vol. 2, pp. 1091–1097 Vol.2. DOI 10.1109/OCEANS.2003.178494
31. J. Ribas, D. Sura, M. Stojanovic, in *OCEANS 2010* (2010), pp. 1–9. DOI 10.1109/OCEANS.2010.5663839
32. N. Farr, A. Bowen, J. Ware, C. Pontbriand, M. Tivey, in *OCEANS 2010 IEEE - Sydney* (2010), pp. 1–6. DOI 10.1109/OCEANSSYD.2010.5603510
33. M. Stojanovic, J. Preisig, Communications Magazine, IEEE **47**(1), 84 (2009). DOI 10.1109/MCOM.2009.4752682
34. H. Zhang, F. Meng, in *Industrial Control and Electronics Engineering (ICICEE), 2012 International Conference on* (2012), pp. 1150–1153. DOI 10.1109/ICICEE.2012.305
35. M. Siegel, R.W.P. King, Antennas and Propagation, IEEE Transactions on **21**(4), 507 (1973). DOI 10.1109/TAP.1973.1140525
36. A. Shaw, A. Al-Shamma'a, S. Wylie, D. Toal, in *Microwave Conference, 2006. 36th European* (2006), pp. 572–575

37. Z. Xiong, O. Guleryuz, M. Orchard, Signal Processing Letters, IEEE **3**(11), 289 (1996). DOI 10.1109/97.542157
38. D. Taubman, E. Ordentlich, M. Weinberger, G. Seroussi, Signal Processing: Image Communication **17**(1), 49 (2002)
39. J. Shapiro, Signal Processing, IEEE Transactions on **41**(12), 3445 (1993). DOI 10.1109/78.258085
40. A. Said, W. Pearlman, Circuits and Systems for Video Technology, IEEE Transactions on **6**(3), 243 (1996). DOI 10.1109/76.499834
41. F.W. Wheeler, W. Pearlman, in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 3 (2000), vol. 3, pp. 861–864 vol.3. DOI 10.1109/ICIP.2000.899592
42. A. Moinuddin, E. Khan, in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 2 (2006), vol. 2, pp. II–II. DOI 10.1109/ICASSP.2006.1660377
43. E. Hong, R. Ladner, Image Processing, IEEE Transactions on **11**(8), 901 (2002). DOI 10.1109/TIP.2002.801124
44. A. Calderbank, I. Daubechies, W. Sweldens, B.L. Yeo, Applied and Computational Harmonic Analysis **5**(3), 332 (1998). DOI http://dx.doi.org/10.1006/acha.1997.0238. URL `http://www.sciencedirect.com/science/article/pii/S1063520397902384`
45. B. Usevitch, in *Data Compression Conference, 1996. DCC '96. Proceedings* (1996), pp. 387–395. DOI 10.1109/DCC.1996.488344
46. V.J. Rehna, M.K.J. Kumar, CoRR **abs/1209.2515** (2012). URL `http://arxiv.org/abs/1209.2515`
47. M. Subedar, L. Karam, G. Abousleman, in *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, vol. 3 (2004), vol. 3, pp. iii–681–4. DOI 10.1109/ICASSP.2004.1326636
48. X. Delaunay, C. Thiebaut, M. Chabert, V. Charvillat, G. Morin, in *On-Board Payload Data Compression Workshop* (Toulouse, France, 2010)
49. M. Adams, F. Kossentini, in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 2 (2000), vol. 2, pp. 53–56